The Stata Journal

Volume 16 Number 2 2016

A Stata Press publication
StataCorp LP
College Station, Texas

The Stata Journal

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

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 800-782-8272, 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.

U.S. and Canada		Elsewhere	
Printed & electronic		**Printed & electronic**	
1-year subscription	$115	1-year subscription	$145
2-year subscription	$210	2-year subscription	$270
3-year subscription	$285	3-year subscription	$375
1-year student subscription	$ 85	1-year student subscription	$115
1-year institutional subscription	$345	1-year institutional subscription	$375
2-year institutional subscription	$625	2-year institutional subscription	$685
3-year institutional subscription	$875	3-year institutional subscription	$965
Electronic only		**Electronic only**	
1-year subscription	$ 85	1-year subscription	$ 85
2-year subscription	$155	2-year subscription	$155
3-year subscription	$215	3-year subscription	$215
1-year student subscription	$ 55	1-year student subscription	$ 55

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.

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

Volume 16 Number 2 2016

The Stata Journal

Articles and Columns 245

Notes and Comments 517

Software Updates 521

The Stata Journal (2016)
16, Number 2, pp. 245–263

Creating LaTeX documents from within Stata using texdoc

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

Abstract. I discuss the use of `texdoc` for creating LaTeX documents from within Stata. Specifically, `texdoc` provides a way to embed LaTeX code directly in a do-file and to automate the integration of results from Stata in the final document. One can use the command, for example, to assemble automatic reports, write a *Stata Journal* article, prepare slides for classes, or put together solutions for homework assignments.

Keywords: pr0062, texdoc, LaTeX, weaving, Stata output, Stata log, reproducible research

1 Introduction

Stata Journal articles and Stata Press books commonly include facsimiles of Stata output. Likewise, Stata output may be part of class notes or presentations. Including Stata output in a LaTeX document is supported by the `sjlatex` package, available from the *Stata Journal* website. For example, the `sjlatex` package provides a LaTeX-style file containing relevant LaTeX commands (`stata.sty`) and a Stata command called `sjlog` to generate LaTeX-formatted log files.

The tools provided by the `sjlatex` package are very helpful, but their usage can be tedious. To simplify that, I created the `texdoc` utility, which is based on `sjlatex` but automizes most of the relevant tasks. Specifically, `texdoc` maintains a do-file that contains Stata commands as well as sections of LaTeX code. The do-file can then be processed by `texdoc` to generate the LaTeX source file including output from the Stata commands. The necessary log files and LaTeX snippets to integrate the Stata output in the final document are produced automatically.[1]

Essentially, `texdoc` is a tool for weaving LaTeX code into a Stata do-file. It differs from other weaving approaches in that it does not rely on external software (for infor-

1. The `texdoc` command was first released in the Statistical Software Components (SSC) archive in 2009. This article describes a heavily revised and expanded version of the command. The most important addition is the `nodo` option, which allows one to work on the text without rerunning the Stata commands. Because of this option and several other additions and improvements, the new version of `texdoc` is much better suited for managing larger projects (such as a book manuscript) than the old version. Furthermore, apart from some extensions related to the inclusion of Stata output (for example, output only or commands only), I have put much effort into improving the robustness of `texdoc` (for example, comments and line breaks are now fully supported, and commands such as `cd` or `clear all` no longer cause problems).

mation on weaving with Stata, see Rising [2008]). Moreover, `texdoc` is not limited to including facsimiles of Stata output in a LaTeX document. It may be generally used for producing dynamic LaTeX documents that combine text sections and results from statistical analysis.

Below I will discuss the features of `texdoc` and provide examples of its usage. To install the required `sjlatex` package on your system, type

```
. net install sjlatex, from(http://www.stata-journal.com/production)
```

Furthermore, to be able to compile a LaTeX document containing Stata output, you need to install the Stata LaTeX files on your system and include `\usepackage{stata}` in the preamble of your LaTeX document. The Stata LaTeX files can be downloaded using `sjlatex install`; see `help sjlatex` after installing `sjlatex`. You may keep the files in the working directory of your LaTeX document or, alternatively, copy the files to the search tree of your LaTeX installation (consult the documentation of your LaTeX installation for information on the search tree).

2 The texdoc command

2.1 Processing a texdoc do-file

The basic procedure is to write a do-file including Stata commands and sections of LaTeX code and then process the do-file by typing `texdoc do`. This command will create the LaTeX source file, possibly including sections of Stata output, which can then be processed by a LaTeX compiler to produce the final document. The syntax of `texdoc do` is

`texdoc do` *filename* [`,` *options*]

filename is the name of the do-file to be processed. *options* are the following:

`init(`*docname*`)` initializes the LaTeX document. Use this option if the do-file does not contain a `texdoc init` command (see section 2.2). *docname* is the name of the LaTeX document, possibly including a path.

init_options are options to be passed through to `texdoc init`. See section 2.2 for details on available options.

`noclose` prevents closing the LaTeX document when `texdoc do` terminates. The default is to close the LaTeX document automatically (also see section 2.6).

`cd` changes the working directory to the directory of the specified do-file for processing the do-file and restores the current working directory after termination. The default is not to change the working directory.

`savecmd(`*newfile*[`, replace append`]`)` saves a copy of the do-file from which all `/*tex tex*/` or `/*** ***/` blocks (see below) have been removed. `replace` overwrites an existing file; `append` appends results to an existing file.

`texdoc do` can be nested. That is, `texdoc do` can be applied in a do-file that is processed by `texdoc do`. Options specified with a nested call to `texdoc do` will be applied to only the nested do-file. This is also true for applications of `texdoc init` or `texdoc close` within the nested do-file: after you terminate a nested do-file, all preexisting `texdoc` settings will be restored. For example, if you use the `init()` option or `texdoc init` to change the LaTeX document in the nested do-file, `texdoc` closes the new LaTeX document and switches back to the previous one when exiting the nested do-file (similarly, if you use `texdoc close` in the nested do-file, the LaTeX document will be reopened after termination). An exception is if you change the LaTeX document in the nested do-file and specify the `noclose` option. In this case, `texdoc` will continue writing to the new LaTeX document.

2.2 Initializing the LaTeX document

At the beginning of a `texdoc` do-file, use `texdoc init` to initialize the LaTeX document unless the `init()` option has been specified with `texdoc do` (see above). The syntax of `texdoc init` is

`texdoc init` [*docname*] [`,` *init_options*]

docname is the name of the LaTeX document, possibly including a path. Alternatively, use `texdoc init` without *docname* to change existing settings after the LaTeX document has been initialized by `texdoc do` or `texdoc init`. *init_options* are the following:

`replace` overwrites an existing LaTeX file.

`append` appends results to an existing LaTeX file.

[`no`]`logdir`[`(`*path*`)`] specifies where to store the Stata output log files. The default is `nologdir`, in which case the log files are stored in the same directory as the LaTeX document, using the name of the LaTeX document as a prefix for the names of the log files; also see the `prefix()` option below. The `logdir` option without argument causes the log files to be stored in a subdirectory with the name of the LaTeX document. The `logdir(`*path*`)` option causes the log files to be stored in subdirectory *path*, where *path* is a relative path starting from the folder of the LaTeX document.

[`no`]`prefix`[`(`*prefix*`)`] specifies the prefix for the automatic names of the Stata output log files and graphs. The names are constructed as *prefix*#, where # is a counter (`1`, `2`, `3`, etc.). The `noprefix` option omits the prefix; the `prefix` option without argument causes *basename*_ to be used as the prefix, where *basename* is the name of the LaTeX document without a path; the `prefix(`*prefix*`)` option causes *prefix* to be used as the prefix. The default prefix is empty if `logdir` or `logdir(`*path*`)` is specified; otherwise, the default prefix is equal to *basename*_. Furthermore, the prefix will be ignored if a custom *name* is provided when calling `texdoc stlog` (see section 2.4). The suffix of the physical log files on disk is always `.log.tex`.

[no]`stpath`[(*path*)] specifies how the path used in the `\input{}` statements to include the Stata output log files in the LaTeX document is to be constructed (`stpath()` has no effect on where the log files are stored in the file system). If `stpath` is specified without an argument, then the path of the LaTeX document (to be precise, the path specified in *docname* when initializing the LaTeX document) is added to the include-path for the log files. Alternatively, specify a custom path as an argument in `stpath()`. The default is `nostpath`, that is, to use no additional path. Specifying a path might be necessary if the LaTeX document is itself an input to a master LaTeX file somewhere else in the file system.

`grdir(`*path*`)` specifies an alternative subdirectory to be used by `texdoc graph` for storing the graph files, where *path* is a relative path starting from the folder of the LaTeX document. The default is to store the graphs in the same directory as the log files.

`gropts(`*graph_options*`)` specifies default options to be passed through to `texdoc graph`. See section 2.5 for details. Updating `gropts()` in repeated calls to `texdoc init` will replace the option as a whole.

stlog_options set the default behavior of `texdoc stlog`. See section 2.4 for details on available options.

2.3 Including LaTeX code

Within a `texdoc` do-file, use

`/*tex` *text* `tex*/`

to include a section of LaTeX code. *text* can contain any text, including multiple lines and paragraphs. It will be passed through to the LaTeX document as is (without expanding Stata macros). The opening tag of a LaTeX section, `/*tex`, must be at the beginning of a line (possibly preceded by white space) and must be followed by a blank or a line break; the closing tag, `tex*/`, must be at the end of a line (possibly followed by white space) and must be preceded by a blank or a line break. As a synonym, for easier typing, you may also use

`/***` *text* `***/`

but note that the two forms may not be mixed (that is, a LaTeX section starting with `/*tex` must be closed by `tex*/`; a section starting with `/***` must be closed by `***/`). A single line of LaTeX code can also be written to the document using

`texdoc write` *textline*

Stata macros in *textline* will be expanded before writing the line to the LaTeX document. Furthermore, to copy the contents of an external file to the LaTeX document, type

`texdoc append` *filename*

filename is the name (and path) of the file to be added. The contents of *filename* will be copied into the LaTeX document as is, at the position where `texdoc append` is specified.

2.4 Including Stata output

The `texdoc stlog` command creates a section in the LaTeX document containing Stata output. The `stata` package providing the `stlog` environment is required to display the output (that is, `\usepackage{stata}` should be included in the preamble of the LaTeX document). The syntax to include a Stata output log is

`texdoc stlog` [*name*] [`,` *stlog_options*]

commands ...

`texdoc stlog close`

`texdoc stlog` opens the log, *commands* are the Stata commands to be logged, and `texdoc stlog close` closes the log. *name* is the name to be used for the log file (possibly including a relative path). If *name* is omitted, an automatic name is used (see the `prefix()` option in section 2.2 for details). Alternatively, you may type

`texdoc stlog` [*name*] `using` *dofile* [`,` *stlog_options*]

dofile is the name (and path) of an external do-file that contains the Stata commands to be logged (`texdoc stlog close` is not needed in this case). Furthermore, among the commands to be logged, you may use

`texdoc stlog oom` *command*

to suppress the output of a specific command and include an "output omitted" message in the log (using the `\oom` command from the `stata` package in LaTeX) and use

`texdoc stlog cnp`

to insert a "continued on the next page" message and a page break (using the `\cnp` command). *stlog_options* are the following:

[`no`]`do` specifies whether to run the Stata commands. The default is `do`; that is, run the commands. Type `nodo` to skip the commands and not write a new log file. `nodo` is useful if the Stata commands have been run before and did not change. For example, specify `nodo` if the Stata output is complete and you want to work on the text without having to rerun the Stata commands. `nodo` works in only noninteractive mode, that is, if the do-file is processed by typing `texdoc do`. Note that the automatic names of Stata output sections change if the order of Stata output sections changes. That is,

`nodo` should be used only if the order did not change or if a fixed name was assigned to the Stata output section.

[`no`]`log` specifies whether the Stata output is to be logged and included in the LaTeX document. The default is `log`; that is, log and include the Stata output. If you type `nolog`, the commands will be run without logging. `nolog` does not appear to be particularly useful because you could simply include the corresponding Stata commands in the do-file without using `texdoc stlog`. However, `nolog` may be helpful in combination with the `nodo` option. It provides a way to include unlogged commands in the do-file that will not be executed if `nodo` is specified.

[`no`]`cmdstrip` specifies whether to strip command lines from the Stata output. The default is `nocmdstrip`; that is, retain the command lines. Specify `cmdstrip` to delete the command lines. Specifically, all lines starting with "`. `" and subsequent lines starting with "`> `" will be removed. `cmdstrip` has no effect if `cmdlog` is specified.

[`no`]`lbstrip` specifies whether to strip line-break comments from command lines in the Stata output. The default is `nolbstrip`; that is, do not strip the line-break comments. Specify `lbstrip` to delete the line-break comments. Specifically, "` ///`" at the end of lines starting with "`. `" or of subsequent lines starting with "`> `" will be removed. `lbstrip` has no effect if `cmdlog` is specified.

[`no`]`ltrim` specifies whether to remove indentation of commands (that is, whether to remove white space on the left of commands) before running the commands and creating the log. This is relevant only in noninteractive mode (that is, if the file is processed by `texdoc do`; furthermore, `ltrim` has no effect on commands called from an external do-file by `texdoc stlog using`). The default is `ltrim`, that is, to remove indentation. The amount of white space to be removed is determined by the minimum indentation in the block of commands.

[`no`]`cmdlog` specifies whether the Stata output includes commands and their output or only a copy of the commands without output. The default is `nocmdlog`; that is, include commands and output. If you type `cmdlog`, then only a copy of the commands without output will be included. `cmdlog` has no effect if `nolog` is specified.

[`no`]`verbatim` specifies whether the command log will be processed by `sjlog`. This is relevant only if `cmdlog` has been specified. The default is `noverbatim`; that is, process the command log with `sjlog`, and use the `stlog` environment in LaTeX to display the output. If you type `verbatim`, then `sjlog` will be skipped, and the `stverbatim` environment will be used. Unless `hardcode` is specified (see below), the log file will be included in the LaTeX document using `\verbatiminput{}`, which requires `\usepackage{verbatim}` in the preamble of the LaTeX document.

[`no`]`hardcode` specifies whether the Stata output is physically copied into the LaTeX document. The default is `nohardcode`; that is, include a link to the log file using an `\input{}` statement in the LaTeX document. If `hardcode` is specified, the log file will be copied directly into the LaTeX document. `hardcode` has no effect if `nolog` or `custom` is specified.

[no]keep specifies whether the external log file will be kept. This is relevant only if hardcode has been specified. The default is keep; that is, keep the log file so that nodo can be applied later. Type nokeep if you want to erase the external log file.

[no]custom specifies whether to use custom code to include the log file in the LaTeX document. The default is nocustom; that is, use standard code to include the log. Specify custom if you want to skip the standard code, and be careful including the log yourself. custom implies nohardcode. custom has no effect if nolog is specified.

2.5 Including graphs

texdoc graph can be used after a Stata output section to export the current graph and include appropriate code in the LaTeX document to display the graph. texdoc graph depends on the information returned by the preceding texdoc stlog close or texdoc stlog using command; the name of the preceding Stata output section will be used to name the graph, and if the nodo option has been specified with texdoc stlog, no graph will be exported, and only the include-code will be written to the LaTeX document. The syntax of texdoc graph is

texdoc graph [, *graph_options*]

graph_options are the following:

as(*fileformats*) sets the output formats. The default is as(pdf). See [G-2] **graph export** for available formats. Multiple formats may be specified as in, for example, as(pdf eps), in which case texdoc graph will create multiple graph files.

name(*name*) specifies the name of the Graph window to be exported. The default is to export the topmost graph.

override_options modify how the graph is converted. See [G-2] **graph export** for details.

optargs(*args*) passes optional arguments through to the LaTeX graph command (as in \includegraphics[*args*]{*filename*} or \epsfig{file=*filename*,*args*}).

[no]suffix specifies whether to type the file suffix in \includegraphics or \epsfig. If only one output format is specified in as(), the default is to type the file suffix. If multiple output formats are specified in as(), the default is to omit the suffix. If the suffix option is specified with multiple output formats, the suffix is determined by the first output format.

[no]epsfig specifies whether to use \epsfig instead of \includegraphics to include the graph in the LaTeX document. The default is noepsfig, that is, to use \includegraphics. The epsfig option implies as(eps) (unless specified otherwise).

[no]center specifies whether to center the graph horizontally in the LaTeX document. The default is center.

[no]figure[(*args*)] specifies whether to include the graph in a (floating) figure environment. The default is `nofigure`. Specify `figure(`*args*`)` to provide arguments to be passed through to the figure environment (as in `\begin{figure}[`*args*`]`).

caption(*string*) provides a caption for the figure. `caption()` implies `figure` (unless `nofigure` is specified).

label(*string*) provides a cross-reference label for the figure. `label()` implies `figure` (unless `nofigure` is specified).

cabove and cbelow specify whether the caption is printed above or below the figure, respectively. Only one of `cabove` and `cbelow` is allowed. `cbelow` is the default.

[no]custom specifies whether to use custom code to include the graph in the LaTeX document. The default is `nocustom`, in which case `texdoc graph` writes code to the LaTeX document to include the graph. Specify `custom` if you want to skip the standard code and include the graph yourself.

2.6 Closing the LaTeX document and exiting the do-file

The syntax to stop writing to the LaTeX document is

```
texdoc close
```

`texdoc do` closes the LaTeX document automatically at the end of the do-file so that `texdoc close` is usually not needed.

Note that the `exit` command (see [R] **exit**) does not cause `texdoc do` to exit the do-file. To exit a `texdoc` do-file, type

```
// texdoc exit
```

(without anything else on the same line and not within a `/*tex tex*/` block).

2.7 Stripping a texdoc do-file

To clear a `texdoc` do-file from all `texdoc` commands, use

`texdoc strip` *filename newname* [, replace append]

filename is the name of the do-file to be stripped, and *newname* is the name of the file to be written to. The `replace` option replaces an existing file; the `append` option appends the results to an existing file. `texdoc strip` removes all `/*tex tex*/` blocks and all `texdoc` commands from the do-file.

2.8 Stored results

`texdoc init` clears `s()`, and `texdoc close` stores the following in `s()`:

Macros

`s(docname)`	name of LaTeX document (including absolute path)	`s(basename)`	base name of LaTeX document (excluding path)
`s(path)`	(absolute) path of LaTeX document	`s(logdir)`	subdirectory used for Stata log files
`s(prefix)`	prefix for automatic Stata log names	`s(stpath)`	include-path to be used for Stata logs in LaTeX document
`s(grdir)`	subdirectory used for graphs (if unequal `s(logdir)`)	`s(gropts)`	default graph export options
`s(nodo)`	`nodo` or empty	`s(nolog)`	`nolog` or empty
`s(cmdstrip)`	`cmdstrip` or empty	`s(lbstrip)`	`lbstrip` or empty
`s(noltrim)`	`noltrim` or empty	`s(cmdlog)`	`cmdlog` or empty
`s(verbatim)`	`verbatim` or empty	`s(hardcode)`	`hardcode` or empty
`s(keep)`	`keep` or empty	`s(custom)`	`custom` or empty

`texdoc stlog close` and `texdoc stlog using` store the following in `s()`:

Macros

`s(name)`	name of the Stata output log (including `logdir()` path)	`s(name0)`	`s(name)` without `logdir()` path
`s(filename)`	name of log file on disk (including path and suffix)	`s(filename0)`	`s(filename)` without suffix
`s(texname)`	name of log file with include-path for use in LaTeX document	`s(texname0)`	`s(texname)` without suffix
`s(indent)`	size of indentation	`s(nodo)`	`nodo` or empty
`s(nolog)`	`nolog` or empty	`s(cmdstrip)`	`cmdstrip` or empty
`s(lbstrip)`	`lbstrip` or empty	`s(noltrim)`	`noltrim` or empty
`s(cmdlog)`	`cmdlog` or empty	`s(verbatim)`	`verbatim` or empty
`s(hardcode)`	`hardcode` or empty	`s(keep)`	`keep` or empty
`s(custom)`	`custom` or empty		

3 Examples

3.1 Basic usage

A typical `texdoc` do-file might look as follows:

begin `example1.texdoc`

```
texdoc init example1.tex, replace

/*tex
\documentclass{article}
\usepackage{stata}
\begin{document}

\section*{Exercise 1}
Open the 1978 Automobile Data and summarize the variables.

tex*/

texdoc stlog
sysuse auto
summarize
texdoc stlog close
```

```
/*tex

\section*{Exercise 2}
Run a regression of price on mileage and weight.

tex*/

texdoc stlog
regress price mpg weight
texdoc stlog close

/*tex

\end{document}
tex*/
```

end `example1.texdoc`

To process the file, type

```
. texdoc do example1.texdoc
```

This command line creates file `example1.tex` and two log files, `example1_1.log.tex` and `example1_2.log.tex`, in the same directory. The contents of `example1.tex` will be

begin `example1.tex`

```
\documentclass{article}
\usepackage{stata}
\begin{document}

\section*{Exercise 1}
Open the 1978 Automobile Data and summarize the variables.

\begin{stlog}
\input{example1_1.log.tex}
\end{stlog}

\section*{Exercise 2}
Run a regression of price on mileage and weight.

\begin{stlog}
\input{example1_2.log.tex}
\end{stlog}

\end{document}
```

end `example1.tex`

You can then use your favorite LaTeX compiler to generate the final document, which will look like what is displayed in figure 1.

Exercise 1

Open the 1978 Automobile Data and summarize the variables.

```
. sysuse auto
(1978 Automobile Data)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
make	0				
price	74	6165.257	2949.496	3291	15906
mpg	74	21.2973	5.785503	12	41
rep78	69	3.405797	.9899323	1	5
headroom	74	2.993243	.8459948	1.5	5
trunk	74	13.75676	4.277404	5	23
weight	74	3019.459	777.1936	1760	4840
length	74	187.9324	22.26634	142	233
turn	74	39.64865	4.399354	31	51
displacement	74	197.2973	91.83722	79	425
gear_ratio	74	3.014865	.4562871	2.19	3.89
foreign	74	.2972973	.4601885	0	1

Exercise 2

Run a regression of price on mileage and weight.

```
. regress price mpg weight
```

Source	SS	df	MS		
				Number of obs	= 74
				F(2, 71)	= 14.74
Model	186321280	2	93160639.9	Prob > F	= 0.0000
Residual	448744116	71	6320339.67	R-squared	= 0.2934
				Adj R-squared	= 0.2735
Total	635065396	73	8699525.97	Root MSE	= 2514

price	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
mpg	-49.51222	86.15604	-0.57	0.567	-221.3025	122.278
weight	1.746559	.6413538	2.72	0.008	.467736	3.025382
_cons	1946.069	3597.05	0.54	0.590	-5226.245	9118.382

Figure 1. Compiled LaTeX document

3.2 Varieties of log files

The default for `texdoc stlog` is to include a log of the commands and their output in the LaTeX document. For example, if you type

```
texdoc stlog
display "2 + 2 = " 2 + 2
display "sqrt(2) = " sqrt(2)
texdoc stlog close
```

the result in the LaTeX document will be as follows:

```
. display "2 + 2 = " 2 + 2
2 + 2 = 4
. display "sqrt(2) = " sqrt(2)
sqrt(2) = 1.4142136
```

To include only a copy of the commands without output, type

```
texdoc stlog, cmdlog
display "2 + 2 = " 2 + 2
display "sqrt(2) = " sqrt(2)
texdoc stlog close
```

which yields the following:

```
display "2 + 2 = " 2 + 2
display "sqrt(2) = " sqrt(2)
```

Conversely, if you want only the output but not the commands, type

```
texdoc stlog, cmdstrip
display "2 + 2 = " 2 + 2
display "sqrt(2) = " sqrt(2)
texdoc stlog close
```

which yields the following:

```
2 + 2 = 4
sqrt(2) = 1.4142136
```

3.3 The hardcode and custom options

By default, `texdoc stlog` writes the log into an external file and then uses an `\input{}` statement in LaTeX to include the file. To embed the log directly into the LaTeX document, specify the `hardcode` option. That is, typing

```
texdoc stlog
display "2 + 2 = " 2 + 2
display "sqrt(2) = " sqrt(2)
texdoc stlog close
```

includes a code snippet such as

```
\begin{stlog}
\input{example5_1.log.tex}
\end{stlog}
```

in the LaTeX file, whereas

```
texdoc stlog, hardcode
display "2 + 2 = " 2 + 2
display "sqrt(2) = " sqrt(2)
texdoc stlog close
```

includes

```
\begin{stlog}
. display "2 + 2 = " 2 + 2
2 + 2 = 4
{\smallskip}
. display "sqrt(2) = " sqrt(2)
sqrt(2) = 1.4142136
{\smallskip}
\end{stlog}
```

Furthermore, if you are not satisfied with the standard code that `texdoc stlog` writes to the LaTeX document, you can specify the `custom` option and create your own variant. For example, `\begin{stlog}` has an `auto` option to pick up the font-size settings (instead of using the default 8-point font). To use this feature, you could apply the `custom` option and type

```
texdoc stlog, custom
display "2 + 2 = " 2 + 2
display "sqrt(2) = " sqrt(2)
texdoc stlog close

texdoc write {\fontsize{10}{11}\selectfont
texdoc write \begin{stlog}[auto]
texdoc write \input{`s(texname)'}
texdoc write \end{stlog}
texdoc write }
```

which would look like this in the compiled LaTeX document:

```
. display "2 + 2 = " 2 + 2
2 + 2 = 4
. display "sqrt(2) = " sqrt(2)
sqrt(2) = 1.4142136
```

`texdoc stlog close` leaves behind some information in `s()` that can be used to build your custom code. As above, if you want to add an include link for the log file in LaTeX, use the filename stored in `s(texname)`; to access the log file in the file system, use the filename stored in `s(filename)`. For example, to copy the log file into the LaTeX document instead of placing an `\input{}` statement in the example above, you could type

```
texdoc write {\fontsize{10}{11}\selectfont
texdoc write \begin{stlog}[auto]
texdoc append `s(filename)'
texdoc write \end{stlog}
texdoc write }
```

3.4 The nodo option

An indispensable option for larger projects is the `nodo` option, which allows you to recompile your document without rerunning the Stata commands. `texdoc` keeps the log files from previous runs so that rerunning the Stata commands would be a waste of time if the Stata commands did not change. Therefore, once the commands in a Stata output section are all set, type

```
texdoc stlog, nodo
commands ...
texdoc stlog close
```

To apply `nodo` to all Stata output sections in the document, specify `nodo` with `texdoc init` or `texdoc do`. To turn the commands back on in a specific section, type

```
texdoc stlog, do
commands ...
texdoc stlog close
```

In fact, using a global `nodo` option and turning the individual sections on and off by specifying the `do` option may be the preferred way of working with `texdoc`. This allows one to rerun all Stata commands at a later point in time (by removing the global `nodo` option) without having to modify the single `texdoc stlog` commands.

Note that `texdoc` uses consecutive numbers to name the log files of the output sections. Thus the name for a specific section will change if other (unnamed) sections are added or deleted in preceding parts of the document. In this case, you may have to rerun all output sections. Hence, if a specific Stata output section contains time-consuming commands, you should assign a fixed name to the output section. For example, typing

```
texdoc stlog bigjob1
commands ...
texdoc stlog close
```

would assign the name `bigjob1` to the output section.

3.5 Including graphs

Graphs can be included in the LaTeX document using the `texdoc graph` command. The basic procedure is to create a graph within a `texdoc stlog` section and then apply `texdoc graph` to export the graph (using the name provided by `texdoc stlog`) and include appropriate code in the LaTeX document to integrate the graph. For example, typing

```
texdoc stlog, nolog
scatter price mpg
texdoc stlog close
texdoc graph
```

would export a PDF graph and include the graph in the LaTeX document using a code snippet such as

```
\begin{center}
    \includegraphics{example9_1.pdf}
\end{center}
```

The `nolog` option has been added in the example to suppress the Stata output in the LaTeX document and display only the graph. The default of `texdoc graph` is to place the graph in a `center` environment. To create a floating figure, use the `figure` option. For example,

```
texdoc stlog, nolog
scatter price mpg
texdoc stlog close
texdoc graph, figure(h!) optargs(scale=0.9) caption(A scatterplot) label(f1)
```

would include the graph as follows:

```
\begin{figure}[h!]
    \centering
    \includegraphics[scale=0.9]{example10_1.pdf}
    \caption{A scatter plot}
    \label{f1}
\end{figure}
```

The `caption()` option has been added to provide a title for the figure; the `label()` option has been added to set a cross-referencing label. Furthermore, note how `figure()` and `optargs()` have been used to pass through optional arguments to the figure environment and the `\includegraphics` command. As illustrated above, `texdoc graph` places the graph either in a `center` environment or in a `figure` environment. To use a different environment, specify `nocenter`, and manually provide the appropriate LaTeX commands using `texdoc write`. For example, to display a right-aligned graph, type

```
texdoc stlog, nolog
scatter price mpg
texdoc stlog close
texdoc write \begin{flushright}
texdoc graph, nocenter
texdoc write \end{flushright}
```

which results in

```
\begin{flushright}
\includegraphics{example11_1.pdf}
\end{flushright}
```

3.6 Including tables

In many cases, the literal Stata output may not be of interest. For example, if running a series of regression models, you may want to display an overall table of the results but not the individual Stata outputs. Using a command such as `esttab` (Jann 2005), you could proceed as follows:

```
texdoc stlog, nolog
sysuse auto
regress price weight
estimates store m1
regress price weight mpg
estimates store m2
regress price weight mpg foreign
estimates store m3
texdoc stlog close

if "`s(nodo)'"=="" {
    esttab m1 m2 m3 using `s(filename0)'.tex, replace se label ///
        nomtitles booktabs align(D{.}{.}{-1}) ///
        title(A regression table\label{table1})
}
texdoc write \input{`s(texname0)'.tex}
```

This would include a table in your document such as the one shown in figure 2. The regression commands have been put into a `texdoc stlog` section in the example above, but `nolog` was specified to turn the log off. Including the commands in a `texdoc stlog` section makes sense to be able to apply the `nodo` option once the commands are complete (note the use of `s(nodo)` to determine whether the `nodo` option has been applied and, hence, whether `esttab` has to be run).

Table 1: A regression table

	(1)	(2)	(3)
Weight (lbs.)	2.044***	1.747**	3.465***
	(0.377)	(0.641)	(0.631)
Mileage (mpg)		−49.51	21.85
		(86.16)	(74.22)
Car type			3673.1***
			(684.0)
Constant	−6.707	1946.1	−5853.7
	(1174.4)	(3597.0)	(3377.0)
Observations	74	74	74

Standard errors in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Figure 2. Compiled LaTeX table

4 Workflow and limitations

`texdoc` do-files may be hard to read because Stata commands and LaTeX code are combined in a single file. To improve clarity, use a text editor that allows you to switch between different settings (syntax highlighting, spell checking, keyboard shortcuts, etc.) depending on whether you work on the Stata commands or the LaTeX code. Some text editors can also be set up so that they automatically apply different settings to different parts of the document (for example, LaTeX settings to `/*tex tex*/` blocks and Stata settings to the rest of the document). Furthermore, define keyboard shortcuts to improve the workflow. For example, define a keyboard shortcut that causes Stata to process the do-file by `texdoc do` in the background if the cursor is within a `/*tex tex*/` block. If a section of Stata commands is selected, the same keyboard shortcut could submit the highlighted commands to a foreground instance of Stata (without using `texdoc do`). You may also want to define a keyboard shortcut that processes the do-file with the `nodo` option turned on so that the LaTeX document can be quickly updated without rerunning the Stata commands.

Furthermore, for larger projects, you may find it helpful to break up the project into several `texdoc` do-files and maintain a master LaTeX file that combines the outputs from the separate files. This allows you to process different parts of the project separately. For example, when working on a book, use a separate do-file for each chapter, and maintain a master do-file such as

```
clear all

texdoc do chapter1.texdoc
texdoc do chapter2.texdoc
texdoc do chapter3.texdoc
...

exit
```

as well as a master LaTeX document such as

```
\documentclass{book}
\usepackage{stata}
\begin{document}

\input{chapter1.tex}
\input{chapter2.tex}
\input{chapter3.tex}
...

\end{document}
```

Finally, although `texdoc` tries to be smart and handle the peculiarities of Stata's language (for example, inline comments and line breaks in commands), there are some limitations and technical issues that you should keep in mind:

- `texdoc` tries to create missing subdirectories using Mata's `mkdir()` function; see [M-5] **chdir( )**. Usually, this works only if all intermediate directories leading to the target subdirectory already exist. If `mkdir()` fails, you will need to create the required directories manually prior to running `texdoc`.
- As mentioned above, `exit` (see [R] **exit**) in a do-file does not cause `texdoc do` to exit the do-file. Type `// texdoc exit` instead.
- `texdoc` commands should always start on a new line with `texdoc` being the first (noncomment) word on the line. For example, do not type

  ```
  . quietly texdoc ...
  ```

  or something similar.
- `texdoc` provides only limited support for the semicolon command delimiter (see [P] **#delimit**). The semicolon command delimiter should work as expected as long as it is turned on and off between `/*tex tex*/` blocks and between `texdoc` commands. However, do not use semicolons to delimit `texdoc` commands.
- `texdoc do` processes the specified do-file piece by piece, from one `/*tex tex*/` block to the next. Therefore, local macros defined in the do-file (see [P] **macro**) will be available only until the next `/*tex tex*/` block (or the next `texdoc init` command or the next `texdoc stlog close` command if the `cmdlog` option has been specified because these commands also cause the do-file to be cut in pieces).
- `texdoc stlog` cannot be nested. Furthermore, do not use `texdoc do` within a `texdoc stlog` section.

- `texdoc do` does not parse the contents of a do-file that is called from the main do-file using the `do` command (see [R] **do**). Thus `/*tex tex*/` blocks in such a file will be ignored (and some options of `texdoc stlog` will not work). Use `texdoc do` to include nested do-files.

- `texdoc` commands can be used interactively by typing them in Stata's Command window or by including them in a regular do-file that is not processed by `texdoc do`. However, `/*tex tex*/` blocks and `// texdoc exit` will be ignored in interactive mode. Furthermore, the `nodo` and `cmdlog` options of `texdoc stlog` do not work in interactive mode (unless applied to `texdoc stlog using`).

- If you apply `texdoc` commands without initializing the LaTeX document, a corresponding message will be displayed, but no error will be returned and execution continues.

- The `$` character is used for global macro expansion in Stata. If you use the `texdoc write` command to write LaTeX code containing `$` math delimiters, type `\$` instead of `$` (no such precautions are required within `/*tex tex*/` blocks). For example, type

```
. texdoc write This is an inline equation: \$ y = x^2 \$
```

- `texdoc stlog` closes the default log if it is on. Use a named log to log a Stata session in which `texdoc stlog` is applied. See the `name()` option in [R] **log**.

5 References

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

Rising, W. 2008. Reproducible Research: Weaving with Stata. 2008 Italian Stata Users Group meeting proceedings.
http://www.stata.com/meeting/italy08/rising_2008.pdf.

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 & Research*, *Stata Journal*, *Public Opinion Quarterly*, and *American Sociological Review*.

The Stata Journal (2016)
16, Number 2, pp. 264–300

Assessing inequality using percentile shares

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

Abstract. At least since Thomas Piketty's best-selling *Capital in the Twenty-First Century* (2014, Cambridge, MA: The Belknap Press), percentile shares have become a popular approach for analyzing distributional inequalities. In their work on the development of top incomes, Piketty and collaborators typically report top-percentage shares, using varying percentages as thresholds (top 10%, top 1%, top 0.1%, etc.). However, analysis of percentile shares at other positions in the distribution may also be of interest. In this article, I present a new command, `pshare`, that estimates percentile shares from individual-level data and displays the results using histograms or stacked bar charts.

Keywords: st0432, pshare, percentile shares, Lorenz curve, concentration curve, inequality, income distribution, wealth distribution, graphics

1 Introduction

Empirical inequality literature heavily relies on the Gini coefficient for the analysis of the development of inequality over time or the analysis of differences in inequality between countries. However, various distributional changes can increase or decrease the Gini coefficient, and it might be important to obtain more detailed knowledge about these processes. Moreover, even if the Gini coefficient remains stable, significant changes in the shape of a distribution may occur. In addition, the specific values of the Gini coefficient, apart from the minimum and the maximum, are difficult to interpret in an absolute sense.

For these reasons, percentile shares have become increasingly popular for analyzing distributional inequality. Percentile shares quantify the proportions of total outcome (for example, of total income) that go to different groups defined in terms of their relative ranks in the distribution. They thus have an intuitive and appealing interpretation and can be used for detailed analysis of distributional changes. The most prominent applications of percentile shares can be found in the works of Thomas Piketty and collaborators (for example, Atkinson, Piketty, and Saez [2011], Piketty and Saez [2014], and Piketty [2014]) and their "World Wealth and Income Database".[1] Piketty and collaborators typically study top-income shares, such as the proportion of income that goes to the top 1% or the top 10%, but the income shares of other percentile groups may be interesting too.

1. See http://topincomes.parisschoolofeconomics.eu/.

In this article, I present a new command, `pshare`, that estimates percentile shares of an outcome variable from individual level data. `pshare` provides standard errors and confidence intervals (CIs) for the estimated percentile shares and supports estimation from complex samples. Furthermore, `pshare` provides subcommands for computing differences in percentile shares across variables or subpopulations and for graphing results as stacked bar charts or histograms.[2]

2 Methods and formulas

2.1 Lorenz ordinates

Let Y be the outcome variable of interest (for example, income). The distribution function of Y is given as $F(y) = \Pr(Y \leq y)$, and the quantile function (the inverse of the distribution function) is given as $Q(p) = F^{-1}(p) = \inf\{y|F(y) \geq p\}$ with $p \in [0, 1]$. Based on these definitions, the ordinates of the Lorenz curve are given as

$$L(p) = \frac{\int_{-\infty}^{Q_p} y\, dF(y)}{\int_{-\infty}^{\infty} y\, dF(y)}$$

Intuitively, a point on the Lorenz curve quantifies the proportion of total outcome of the poorest $p \times 100$ percent of the population. This can easily be seen in the finite population form of $L(p)$, which is given as

$$L(p) = \frac{\sum_{i=1}^{N} Y_i I_{Y_i \leq Q_p}}{\sum_{i=1}^{N} Y_i}$$

with I_A as an indicator function being equal to 1 if A is true and 0 otherwise.

2.2 Percentile shares

Percentile share $S(p_1, p_2)$, with $p_1 \leq p_2$, is equal to the proportion of total outcome that falls into the quantile interval $(Q_{p_1}, Q_{p_2}]$ or, stated differently, the proportion of total outcome pertaining to the population segment from relative rank p_1 to relative rank p_2 in the list of ordered outcomes. This is equal to the difference between the Lorenz ordinates for p_1 and p_2; that is,

$$S(p_1, p_2) = L(p_2) - L(p_1)$$

2. Some of the functionality of `pshare` is also covered by the user-written commands `sumdist` (Jenkins 1999) and `svylorenz` (Jenkins 2006). However, `pshare` specifically focuses on percentile shares and provides a more comprehensive implementation. Furthermore, `sumdist` and `svylorenz` use somewhat different methods to compute the percentile shares (ties are not broken, and flat regions in the distribution function are not interpolated; see below).

or, in the finite population,

$$S(p_1, p_2) = \frac{\sum_{i=1}^{N} Y_i I_{Y_i \leq Q_{p_2}}}{\sum_{i=1}^{N} Y_i} - \frac{\sum_{i=1}^{N} Y_i I_{Y_i \leq Q_{p_1}}}{\sum_{i=1}^{N} Y_i} = \frac{\sum_{i=1}^{N} Y_i \left(I_{Y_i \leq Q_{p_2}} - I_{Y_i \leq Q_{p_1}} \right)}{\sum_{i=1}^{N} Y_i}$$

To simplify notation, we will let $S_\ell^j = S(p_{\ell-1}, p_\ell)$. Furthermore, we will let

$$\mathbf{s}(\mathbf{p}) = \begin{bmatrix} S_1 & S_2 & \cdots & S_k \end{bmatrix}$$

be the $1 \times k$ vector of a disjunctive and exhaustive set of percentile shares across the domain of p using cutoffs $\mathbf{p} = \begin{bmatrix} p_0 & p_1 & \cdots & p_k \end{bmatrix}$ with $p_{\ell-1} < p_\ell$ for all $\ell = 0, \ldots, k$ and $p_0 = 0$ and $p_k = 1$.

Depending on context, it may be sensible to normalize percentile shares by the size of the respective population segment (that is, the proportion of the population covered by the segment, which is equal to $p_\ell - p_{\ell-1}$), which yields percentile share density

$$D_\ell = \frac{S_\ell^j}{p_\ell - p_{\ell-1}}$$

D_ℓ is a density in the sense that $\mathbf{d}(\mathbf{p})$—a disjunctive and exhaustive set of percentile share densities across the domain of p—integrates to 1. Note, however, that D_ℓ may be negative if the outcome variable can take on negative values (for example, debt). A value of $D_\ell = 1$ means that each member in the respective population segment has (on average) an outcome value equal to the average outcome in the population. A value of $D_\ell = 2$ means that each member in the segment has (on average) twice the population average; a value of $D_\ell = -0.5$ means that each member in the segment has (on average) minus half the population average.

Furthermore, percentile shares can be expressed as totals or averages in absolute terms. The finite population form of percentile share totals and averages is given as

$$T_\ell = \sum_{i=1}^{N} Y_i I_{Y_i \leq Q_{p_\ell}} - \sum_{i=1}^{N} Y_i I_{Y_i \leq Q_{p_{\ell-1}}} = \sum_{i=1}^{N} Y_i \left(I_{Y_i \leq Q_{p_\ell}} - I_{Y_i \leq Q_{p_{\ell-1}}} \right) = S_\ell^j \sum_{i=1}^{N} Y_i$$

and

$$A_\ell = \frac{T_\ell}{(p_\ell - p_{\ell-1}) \times N}$$

respectively. T_ℓ is simply the sum of all outcomes in the respective population segment; A_ℓ is the average outcome among the members of the segment.

Finally, with reference to the generalized Lorenz curve, generalized percentile shares can be defined as

$$G_\ell = \text{GL}(p_\ell) - \text{GL}(p_{\ell-1})$$

where the finite-population form of the generalized Lorenz ordinate $\text{GL}(p)$ is

$$\text{GL}(p) = \frac{1}{N}\sum_{i=1}^{N} Y_i I_{Y_i \leq Q_p}$$

so that

$$G_\ell = \frac{1}{N}\sum_{i=1}^{N} Y_i I_{Y_i \leq Q_{p_\ell}} - \frac{1}{N}\sum_{i=1}^{N} Y_i I_{Y_i \leq Q_{p_{\ell-1}}} = \frac{1}{N}\sum_{i=1}^{N} Y_i \left(I_{Y_i \leq Q_{p_\ell}} - I_{Y_i \leq Q_{p_{\ell-1}}} \right)$$

Note that there is an interesting relationship between percentile share averages and generalized percentile shares: percentile share average A_ℓ is equal to $G_\ell/(p_\ell - p_{\ell-1})$; that is, A_ℓ is equal to the difference in the generalized Lorenz ordinates for p_ℓ and $p_{\ell-1}$ divided by the population share $p_\ell - p_{\ell-1}$.

2.3 Point estimation

The above exposition assumes Y to be continuous. Because empirical data are always discrete, the empirical distribution function is nonsmooth, and there may be ties or empty sets at the quantiles of interest. For estimation of percentile shares using empirical data, it makes sense to break ties proportionally and apply linear interpolation in regions where the empirical distribution function is flat.

Let w_i be sampling weights (equal to 1 in unweighted data), and let subscripts in parentheses refer to sorted observations in ascending order of Y. S_ℓ^j can then be estimated from a sample of size n as

$$\widehat{S}_\ell^j = \widehat{L}(p_\ell) - \widehat{L}(p_{\ell-1})$$

with

$$\widehat{L}(p) = (1-\gamma)\widetilde{Y}_{j_p-1} + \gamma\widetilde{Y}_{j_p}$$

where

$$\gamma = \frac{p - \widehat{p}_{j_p-1}}{\widehat{p}_{j_p} - \widehat{p}_{j_p-1}}, \quad \widetilde{Y}_{j_p} = \frac{\sum_{i=1}^{j_p} w_{(i)} Y_{(i)}}{\sum_{i=1}^{n} w_i Y_i}, \quad \text{and} \quad \widehat{p}_{j_p} = \frac{\sum_{i=1}^{j_p} w_{(i)}}{\sum_{i=1}^{n} w_i}$$

and where j_p is set such that $\widehat{p}_{j_p-1} < p \leq \widehat{p}_{j_p}$. This corresponds to estimating Lorenz ordinates by taking quantiles from the running sum of the ordered Y values (divided by the total of Y) according to quantile definition 4 in Hyndman and Fan (1996).

Alternatively, ignoring linear interpolation in flat regions, $L(p)$ can be estimated as

$$\widehat{L}(p) = \widetilde{Y}_{j_p} = \frac{\sum_{i=1}^{j_p} w_{(i)} Y_{(i)}}{\sum_{i=1}^{n} w_i Y_i}$$

which corresponds to quantile definition 1 in Hyndman and Fan (1996).[3]

An estimate for D_ℓ is given as $\widehat{S}^j_\ell/(p_\ell - p_{\ell-1})$. For an estimate of T_ℓ, omit the denominator, $\sum_{i=1}^n w_i Y_i$, in the formula for $\widetilde{Y}_j$. An estimate for A_ℓ can be obtained as $\widehat{T}_\ell/\{(p_\ell - p_{\ell-1})\sum_{i=1}^n w_i\}$. For an estimate of G_ℓ, replace the denominator in the formula for $\widetilde{Y}_j$ by $\sum_{i=1}^n w_i$.

2.4 Variance estimation

An approximate variance matrix for $\widehat{\mathbf{s}}(\mathbf{p})$ can be obtained by using an estimating equations approach as outlined by Binder and Kovacevic (1995) (also see Kovačević and Binder [1997]). Let θ be the parameter of interest (a percentile share in our case), and let $\boldsymbol{\lambda}$ be a vector of nuisance parameters on which θ depends (the two quantiles determining the Lorenz ordinates in our case). According to Kovačević and Binder (1997), the sampling variance of $\widehat{\theta}$ can be approximated by the sampling variance of the total estimator

$$\sum_{i=1}^{n} w_i u_i^*$$

where w_i are sampling weights and u_i^* is the solution of

$$\left\{ -u_i^\theta + \frac{\partial U^\theta}{\partial \boldsymbol{\lambda}} \left(\frac{\partial \mathbf{U}^{\boldsymbol{\lambda}}}{\partial \boldsymbol{\lambda}} \right)^{-1} \mathbf{u}_i^{\boldsymbol{\lambda}} \right\} \left(\frac{\partial U^\theta}{\partial \theta} \right)^{-1}$$

with all unknowns in the final solution replaced by their sample counterparts. u_i^θ and $\mathbf{u}_i^{\boldsymbol{\lambda}}$ are estimating functions such that in the (finite) population, θ and $\boldsymbol{\lambda}$ are the solutions to

$$U^\theta = \sum_{i=1}^{N} u_i^\theta = 0 \quad \text{and} \quad \mathbf{U}^{\boldsymbol{\lambda}} = \sum_{i=1}^{N} \mathbf{u}_i^{\boldsymbol{\lambda}} = \mathbf{0}$$

3. The first approach is the default method in the `pshare` command presented below. The second approach ignoring linear interpolation can be requested by specifying the `step` option. Note that results from the second approach depend on the sort order within ties of Y if there are sampling weights. To enforce stable results in this case, the `pshare` command sorts observations in ascending order of the sampling weights among ties of Y, but this is an arbitrary decision.

In our case, $\theta = S^j_\ell$ and $\boldsymbol{\lambda} = \begin{bmatrix} Q^j_{p_\ell} & Q^j_{p_{\ell-1}} \end{bmatrix}$, where j refers to the analyzed subpopulation. Let $J_i = 1$ if observation i belongs to subpopulation j and $J_i = 0$ otherwise (with $J_i = 1$ for all observations if the entire sample is analyzed). Because

$$S^j_\ell = \frac{\sum_{i=1}^N Y_i \left(I_{y_i \leq Q^j_{p_\ell}} - I_{y_i \leq Q^j_{p_{\ell-1}}} \right) J_i}{\sum_{i=1}^N Y_i J_i} \qquad \text{and} \qquad \sum_{i=1}^N \left(I_{Y_i \leq Q^j_p} - p \right) J_i = 0$$

the estimating functions are

$$u^\theta_i = Y_i \left(I_{Y_i \leq Q^j_{p_\ell}} - I_{Y_i \leq Q^j_{p_{\ell-1}}} \right) J_i - Y_i J_i S^j_\ell \qquad \text{and} \qquad \mathbf{u}^{\boldsymbol{\lambda}}_i = \begin{bmatrix} \left(I_{Y_i \leq Q^j_{p_\ell}} - p_\ell \right) J_i \\ \left(I_{Y_i \leq Q^j_{p_{\ell-1}}} - p_{\ell-1} \right) J_i \end{bmatrix}$$

Furthermore, given these definitions,

$$\frac{\partial U^\theta}{\partial \theta} = -\sum_{i=1}^N Y_i J_i \qquad \text{and} \qquad \frac{\partial U^\theta}{\partial \boldsymbol{\lambda}} \left(\frac{\partial \mathbf{U}^{\boldsymbol{\lambda}}}{\partial \boldsymbol{\lambda}} \right)^{-1} = \begin{bmatrix} E(Y|Y = Q^j_{p_\ell}) \\ -E(Y|Y = Q^j_{p_{\ell-1}}) \end{bmatrix}'$$

Finally, because $E(Y|Y = Q_p) = Q_p$, we get

$$\begin{aligned} u^*_i &= \frac{-\left\{ Y_i \left(I_{Y_i \leq \widehat{Q}^j_{p_\ell}} - I_{Y_i \leq \widehat{Q}^j_{p_{\ell-1}}} \right) J_i - Y_i J_i \widehat{S}^j_\ell \right\}}{-\sum_{k=1}^n w_k Y_k J_k} \\ &\quad + \frac{\widehat{Q}^j_{p_\ell} \left(I_{Y_i \leq \widehat{Q}^j_{p_\ell}} - p_\ell \right) J_i - \widehat{Q}^j_{p_{\ell-1}} \left(I_{Y_i \leq \widehat{Q}^j_{p_{\ell-1}}} - p_{\ell-1} \right) J_i}{-\sum_{k=1}^n w_k Y_k J_k} \\ &= \frac{\left\{ \left(Y_i - \widehat{Q}^j_{p_\ell} \right) I_{Y_i \leq \widehat{Q}^j_{p_\ell}} - \left(Y_i - \widehat{Q}^j_{p_{\ell-1}} \right) I_{Y_i \leq \widehat{Q}^j_{p_{\ell-1}}} + p_\ell \widehat{Q}^j_{p_\ell} - p_{\ell-1} \widehat{Q}^j_{p_{\ell-1}} \right\} J_i}{\sum_{k=1}^n w_k Y_k J_k} \\ &\quad - \frac{Y_i J_i \widehat{S}^j_\ell}{\sum_{k=1}^n w_k Y_k J_k} \end{aligned}$$

The sampling variance of the total of u_i^*, which serves as an approximation of the sampling variance of $\widehat{S}_\ell^j$, can then be estimated using standard techniques as implemented in `total` (see [R] **total**), possibly accounting for complex survey design. The joint variance matrix for all elements of $\widehat{\mathbf{s}}(\mathbf{p})$ can be obtained by applying `total` to a series of appropriate u^* variables. Likewise, for joint estimation across several outcome variables or multiple subpopulations, include multiple series of u^* variables, one series for each outcome variable or subpopulation.[4]

Variance estimators for percentile densities, totals, averages, or generalized shares can be derived analogously. The appropriate u^* variables are obtained by replacing a_i and b in

$$u_i^* = \frac{\left\{\left(Y_i - \widehat{Q}^j_{p\ell}\right) I_{Y_i \leq \widehat{Q}^j_{p\ell}} - \left(Y_i - \widehat{Q}^j_{p\ell-1}\right) I_{Y_i \leq \widehat{Q}^j_{p\ell-1}} + p_\ell \widehat{Q}^j_{p\ell} - p_{\ell-1}\widehat{Q}^j_{p\ell-1}\right\} J_i - a_i}{b}$$

according to the overview in table 1, where n_c is the number of clusters and $\omega_{[i]}$ is the sum of weights in the cluster to which observation i belongs.[5]

4. When computing the u^* variables, the `pshare` command presented below uses definition 4 in Hyndman and Fan (1996) to determine $\widehat{Q}^j_p$ (or definition 1, depending on the method used for estimating the Lorenz ordinates). Furthermore, in analogy to the approach used for point estimation, ties in Y are broken when determining $I(Y_i \leq \widehat{Q}^j_p)$ (based on observations sorted by w_i within ties, which is an arbitrary decision to enforce stable results).
5. Depending on sample design, the denominator in a_i for T may require modification, for example, to take account of stratification. A workaround, followed by the `pshare` command presented below, is to simply set a_i to zero for T. This is a slight deviation from the approach outlined above (because u^* will sum to $\widehat{T}$ instead of zero), but the resulting variance estimates are the same in this case. On a related matter, note that `total` with clusters or weights yields different results than `svy: total` because the former assumes the number of observations or the sum of weights (and not the number of clusters) to be fixed. Likewise, `total` with the `over()` option produces different results than `svy: total`, even in the absence of clusters or weights, because the subgroup sizes are assumed fixed. Despite this disagreement, the `pshare` command presented below, which relies on the `total` command for variance estimation, always yields results that are consistent with `svy: total`, irrespective of whether weights and clusters are specified directly or via the `svy` option.

Table 1. Definitions of a_i and b for different types of percentile shares

	a_i	b
S	$Y_i J_i \widehat{S}^j_\ell$	$\sum_{i=1}^{n} w_i Y_i J_i$
D	$Y_i J_i (p_\ell - p_{\ell-1}) \widehat{D}^j_\ell$	$\sum_{i=1}^{n} w_i Y_i J_i (p_\ell - p_{\ell-1})$
T	$\frac{1}{n_c \omega_{[i]}} \widehat{T}^j_\ell$	1
A	$J_i (p_\ell - p_{\ell-1}) \widehat{A}^j_\ell$	$\sum_{i=1}^{n} w_i J_i (p_\ell - p_{1-\ell})$
G	$J_i \widehat{G}^j_\ell$	$\sum_{i=1}^{n} w_i J_i$

An alternative to the approach outlined above is to estimate the variances using the bootstrap or jackknife method (see [R] **bootstrap** and [R] **jackknife**).

2.5 Extensions

Contrasts

To analyze distributional differences among subpopulations or across time, one can compute contrasts between percentile shares. The most intuitive approach is to compute contrasts as arithmetic differences. For example, given percentile share estimates from two subpopulations (or two variables), A and B, the vector of arithmetic contrasts is

$$\widehat{\mathbf{s}}^A(\mathbf{p}) - \widehat{\mathbf{s}}^B(\mathbf{p})$$

with variance matrix

$$\begin{bmatrix}\mathbf{I}_k & -\mathbf{I}_k\end{bmatrix} \widehat{\mathbf{V}}\left\{\begin{bmatrix}\widehat{\mathbf{s}}^A(\mathbf{p}) & \widehat{\mathbf{s}}^B(\mathbf{p})\end{bmatrix}\right\} \begin{bmatrix}\mathbf{I}_k & -\mathbf{I}_k\end{bmatrix}'$$

where $\mathbf{I}_k$ is the identity matrix of size k and $\widehat{\mathbf{V}}\{\dots\}$ is the joint variance matrix of the percentile shares across both subpopulations (or variables).

Alternatively, contrasts could be computed as ratios or logarithms of ratios. Generally, let

$$\begin{bmatrix} c\left(\widehat{S}^A_1, \widehat{S}^B_1\right) & c\left(\widehat{S}^A_2, \widehat{S}^B_2\right) & \cdots & c\left(\widehat{S}^A_k, \widehat{S}^B_k\right)\end{bmatrix}$$

be the vector of percentile share contrasts between subpopulations (or variables) A and B, with $c(a, b)$ as a function of a and b, such as $c(a, b) = a/b$ (ratio) or $c(a, b) = \ln(a/b)$ (logarithm of ratio). The variance matrix of the vector can then be approximated by the delta method as

$$\mathbf{\Delta}\widehat{\mathbf{V}}\left\{\begin{bmatrix}\widehat{\mathbf{s}}^A(\mathbf{p}) & \widehat{\mathbf{s}}^B(\mathbf{p})\end{bmatrix}\right\}\mathbf{\Delta}'$$

where $\boldsymbol{\Delta}$ is a $k \times 2k$ matrix

$$\begin{bmatrix} \frac{\partial c(\widehat{S}_1^A, \widehat{S}_1^B)}{\partial \widehat{S}_1^A} & 0 & \cdots & 0 & \frac{\partial c(\widehat{S}_1^A, \widehat{S}_1^B)}{\partial \widehat{S}_1^B} & 0 & \cdots & 0 \\ 0 & \frac{\partial c(\widehat{S}_2^A, \widehat{S}_2^B)}{\partial \widehat{S}_2^A} & \cdots & 0 & 0 & \frac{\partial c(\widehat{S}_2^A, \widehat{S}_2^B)}{\partial \widehat{S}_2^B} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial c(\widehat{S}_k^A, \widehat{S}_k^B)}{\partial \widehat{S}_k^A} & 0 & 0 & \cdots & \frac{\partial c(\widehat{S}_k^A, \widehat{S}_k^B)}{\partial \widehat{S}_k^A} \end{bmatrix}$$

In Stata, the `nlcom` command can be used to perform the necessary computations. The derivatives in $\boldsymbol{\Delta}$ are determined numerically by `nlcom` (see [R] **nlcom**).

Renormalization

Percentile shares expressed as proportions or densities are normalized with respect to the total of the analyzed outcome variable in the given (sub)population. Depending on context, it may be sensible to use a different total for normalization. For example, when analyzing different subpopulations, we may want to express results in terms of proportions of the grand total across all subpopulations. Likewise, if analyzing, say, labor income, we may want to express results in terms of total income (labor income plus capital income).

To normalize results to a different total, simply replace denominator $\sum_{i=1}^n w_i Y_i$ in the above percentile share estimators with the appropriate total. For example, to normalize to the total of variable Z instead of the total of variable Y (where Z may be the sum of several variables, possibly including Y), use $\sum_{i=1}^n w_i Z_i$ as the denominator. Similarly, if normalizing percentile shares to the total of a reference (sub)population r instead of subpopulation j, replace the standard denominator $\sum_{i=1}^n w_i Y_i J_i$ with $\sum_{i=1}^n w_i Y_i R_i$, where J_i and R_i are indicators for whether observation i belongs to subpopulation j or r, respectively. When normalizing percentile densities to the total of a reference (sub)population, you need to consider the relative group sizes so that the densities reflect multiples of the average outcome in the reference (sub)population. That is, use

$$\widehat{D}_\ell^{jr} = \frac{\widehat{S}_\ell^{jr}}{(p_\ell - p_{\ell-1})\widehat{P}^{jr}} \qquad \text{with} \qquad \widehat{P}^{jr} = \frac{\sum_{i=1}^n w_i J_i}{\sum_{i=1}^n w_i R_i}$$

to compute the percentile density in subpopulation j with respect to the total of subpopulation r.

For variance estimation, several cases have to be distinguished: 1) normalizing to the total of Z, 2) normalizing to a fixed total τ, 3) normalizing to the total of Y in reference population r, 4) normalizing to the total of Z in reference population r, and 5) normalizing to a fixed total τ in reference population r. In general, when one normalizes densities with respect to a reference population (cases 3 to 5), the relative group size is a further nuisance parameter that has to be considered. Solving the equations for the

different cases leads to the expressions for a_i and b as shown in table 2 (see the section on variance estimation above for background).[6]

Table 2. Definitions of a_i and b for renormalized percentile shares

		a_i	b
(1)	S	$Z_i J_i \widehat{S}^j_\ell$	$\sum_i w_i Z_i J_i$
	D	$Z_i J_i (p_\ell - p_{\ell-1}) \widehat{D}^j_\ell$	$\sum_i w_i Z_i J_i (p_\ell - p_{\ell-1})$
(2)	S	$\frac{\tau}{n_c \omega_{[i]}} \widehat{S}^j_\ell$	τ
	D	$\frac{\tau}{n_c \omega_{[i]}} (p_\ell - p_{\ell-1}) \widehat{D}^j_\ell$	$\tau (p_\ell - p_{\ell-1})$
(3)	S	$Y_i R_i \widehat{S}^{jr}_\ell$	$\sum_i w_i Y_i R_i$
	D	$\left(Y_i R_i - \frac{\sum_k w_k Y_k R_k}{\sum_k w_k R_k} R_i + \frac{\sum_k w_k Y_k R_k}{\sum_k w_k J_k} J_i \right) \times (p_\ell - p_{\ell-1}) \widehat{P}^{jr} \widehat{D}^{jr}_\ell$	$\sum_i w_i Y_i R_i (p_\ell - p_{\ell-1}) \widehat{P}^{jr}$
(4)	S	Like (3), but with all instances of Y replaced by Z	
	D		
(5)	S	$\frac{\tau}{n_c \omega_{[i]}} \widehat{S}^{jr}_\ell$	τ
	D	$\left(\frac{\tau}{n_c \omega_{[i]}} - \frac{\tau}{\sum_k w_k R_k} R_i + \frac{\tau}{\sum_k w_k J_k} J_i \right) \times (p_\ell - p_{\ell-1}) \widehat{P}^{jr} \widehat{D}^{jr}_\ell$	$\tau (p_\ell - p_{\ell-1}) \widehat{P}^{jr}$

(All sums are across the entire sample.)

Concentration shares

A further interesting possibility is to determine the relative ranks of the population members using an alternative outcome variable. By default, observations will be ordered by their Y values. However, we may also order observations by some alternative variable Z. The (finite-population) Lorenz ordinates are then defined as

$$L^Z(p) = \frac{\sum_{i=1}^{N} Y_i I_{Z_i \leq Q^Z_p}}{\sum_{i=1}^{N} Y_i}$$

6. Depending on sample design, expression $\tau/(n_c \omega_{[i]})$ in a_i for cases (2) and (5) may require modification. An alternative, however, is to simply set $\tau/(n_c \omega_{[i]})$ to zero. See footnote 5 above.

and the percentile shares reflect the proportion of total Y that is received by different percentile groups of Z (in this case, the Lorenz curve is called a concentration curve; see Kakwani [1977] and Lambert [2001]). For example, this could be used to analyze how taxes (Y) are distributed across income groups (Z).

For the purpose of estimation, it appears sensible to average Y within ties of Z when computing the concentration curve ordinates so that results are independent of the sort order of the observations. Furthermore, for variance estimation, we need to replace $\widehat{Q}_p$ in the formulas for the u^* variables with $\widehat{E}(Y|Z = Q_p^Z)$, the expected value of Y at the p quantile of Z.[7]

3 The pshare command

Four subcommands are provided. `pshare estimate` computes the percentile shares and their variance matrix; `pshare contrast` computes differences in percentile shares between outcome variables or subpopulations based on the results from `pshare estimate`; `pshare stack` draws a stacked bar chart of the results from `pshare estimate`, and `pshare histogram` draws a histogram of the results from `pshare estimate` or `pshare contrast`.

3.1 Syntax of pshare estimate

The syntax of `pshare estimate` is

```
pshare [estimate] varlist [if] [in] [weight] [,
    {proportion|percent|density|sum|average|generalized} normalize(spec)
    gini {nquantiles(#)|percentiles(numlist)} pvar(pvar) step
    over(varname) total contrast[(spec)] stack[(options)]
    histogram[(options)] vce(vcetype) cluster(clustvar) svy[(subpop)] nose
    level(#) noheader notable nogtable display_options]
```

`pweights`, `iweights`, and `fweights` are allowed; see [U] **11.1.6 weight**. For each specified variable, percentile shares (quintile shares by default) are tabulated along with their standard errors and CIs.[8] If the `over()` option is specified (see below), only one variable is allowed in *varlist*. `pshare` assumes subcommand `estimate` as the default; typing the word "`estimate`" is required only in the case of a name conflict between the first element of *varlist* and the other subcommands of `pshare` (see below). Options are as follows.

7. In the `pshare` command presented below, $E(Y|Z = Q_p^Z)$ is estimated by local linear regression using the Epanechnikov kernel and the default rule-of-thumb bandwidth as described in [R] **lpoly**.
8. Variance estimation is not supported for `iweights` and `fweights`. To compute standard errors and CIs in the case of `fweights`, apply `pshare` to the expanded data (see [D] **expand**).

Main

`proportion`, `percent`, `density`, `sum`, `average`, or `generalized` selects the type of results to be computed. The default is `proportion`, that is, to report percentile shares as proportions. Use the `percent` option to report percentile shares as percentages. Furthermore, use the `density` option to report densities, defined as outcome shares divided by population shares (so that in a bar chart, the areas of the bars are proportional to the outcome shares). Outcome sums (totals) and average outcomes can be requested by the `sum` and `average` options, respectively. Finally, use the `generalized` option to report generalized percentile shares, defined as differences between generalized Lorenz ordinates. Only one of `proportion`, `percent`, `density`, `sum`, `average`, or `generalized` is allowed.

`normalize(`*spec*`)` normalizes results with respect to the specified total (not allowed in combination with `sum`, `average`, or `generalized`). *spec* is

[*over*:] [*total*]

where *over* may be

`.`	the subpopulation at hand (the default)
#	the subpopulation identified by value *#*
`#`*#*	the *#*th subpopulation
`total`	the total across all subpopulations

and *total* may be

`.`	the total of the variable at hand (the default)
`*`	the total of the sum across all analyzed outcome variables
varlist	the total of the sum across the variables in *varlist*
#	a total equal to *#*

total specifies the variables from which the total is to be computed or sets the total to a fixed value. If multiple variables are specified, the total across all specified variables is used (*varlist* may contain external variables that are not among the list of analyzed outcome variables). *over* selects the reference population from which the total is to be computed; *over* is allowed only if the `over()` option has been specified (see below). Subpopulation sizes (sum of weights) are considered for the computation of densities (the `density` option) if *over* is provided so that the densities reflect multiples of the average outcome in the reference population.

`gini` reports the Gini coefficients of the distributions (also known as concentration indices if `pvar()` is specified; see below) to be computed and reported in a separate table. Variance estimation for Gini coefficients is not supported.[9]

9. Following Lerman and Yitzhaki (1989), the concentration index of Y with respect to Z is computed as $C = 2\sum_{i=1}^{n} \widetilde{w}_i(Y_i - \overline{Y})(F_i - \overline{F})/\overline{Y}$, where $\widetilde{w}_i = w_i/\sum_{i=1}^{n} w_i$ are normalized weights, $\overline{Y} = \sum_{i=1}^{n} \widetilde{w}_i Y_i$ is the mean of Y, $\overline{F} = \sum_{i=1}^{n} \widetilde{w}_i F_i$ is the mean of F, and $F_i = \sum_{j=1}^{n} \widetilde{w}_j I_{Z_j \leq Z_i} - \sum_{j=1}^{n} \widetilde{w}_i I_{Z_j = Z_i}/2$ is the midinterval relative rank of Z_i in the empirical distribution of Z. For the Gini coefficient of Y, set $Z = Y$.

Percentiles

`nquantiles(#)` specifies the number of (equally sized) percentile groups to be used or `percentiles(`*numlist*`)` to specify a list of percentile cutoffs. The default is `nquantiles(5)`, which corresponds to `percentiles(20 40 60 80)` or, with shorthand as described in [U] **11.1.8 numlist**, `percentiles(20(20)80)`.

`pvar(`*pvar*`)` causes the percentile groups to be based on variable *pvar* instead of the outcome variable. That is, observations will be sorted in increasing order of *pvar*, and percentiles will be determined from the running sum of the outcome variable across this sort order (using averaged values within ties of *pvar*). Use this option to analyze relations between different variables (for example, how wealth is distributed across different income groups). If `pvar()` is specified, the computed percentile shares correspond to differences between ordinates of the "concentration curve" of the outcome variable with respect to *pvar*.

`step` determines the Lorenz ordinates from the step function of cumulative outcomes. The default is to use linear interpolation in regions where the step function is flat.

Over

`over(`*varname*`)` repeats results for each subpopulation defined by the values of *varname*. Only one outcome variable is allowed if `over()` is specified.

`total` reports additional overall results across all subpopulations. `total` is allowed only if `over()` is specified.

Contrast/Graph

`contrast`[`(`*spec*`)`] computes differences in percentile shares between outcome variables or between subpopulations. *spec* is

[*base*] [`, ratio lnratio`]

where *base* is the name of the outcome variable or the value of the subpopulation to be used as base for the contrasts. If *base* is omitted, adjacent contrasts across outcome variables or subpopulations are computed (or contrasts with respect to the total if total results across subpopulations have been requested).

Use the suboption `ratio` to compute contrasts as ratios or the suboption `lnratio` to compute contrasts as logarithms of ratios. The default is to compute contrasts as differences.

`stack`[`(`*options*`)`] draws a stacked bar chart of the results. *options* are as described for `pshare stack` below.

`histogram`[`(`*options*`)`] draws a histogram of the results. *options* are as described for `pshare histogram` below.

SE/SVY

`vce(`*vcetype*`)` determines how standard errors and CIs are computed. *vcetype* may be

`analytic`
`cluster` *clustvar*
`bootstrap` [`,` *bootstrap_options*]
`jackknife` [`,` *jackknife_options*]

The default is `vce(analytic)`. See [R] **bootstrap** and [R] **jackknife** for *bootstrap_options* and *jackknife_options*, respectively.

`cluster(`*clustvar*`)` is a synonym for `vce(cluster` *clustvar*`)`.

`svy`[`(`*subpop*`)`] causes the survey design to be taken into account for variance estimation; see [SVY] **svyset**. Specify *subpop* to restrict survey estimation to a subpopulation, where *subpop* is

[*varname*] [*if*]

The subpopulation is defined by observations for which *varname* $\neq 0$ and for which the `if` condition is met. See [SVY] **subpopulation estimation** for more information on subpopulation estimation.

The `svy` option is allowed only if the variance estimation method set by `svyset` is Taylor linearization (the default). For other variance estimation methods, the usual `svy` prefix command may be used; see [SVY] **svy**. For example, type "`svy brr: pshare` ..." to use balanced repeated-replication variance estimation. `pshare` does not allow the `svy` prefix for Taylor linearization because of technical reasons. This is why the `svy` option is provided.

`nose` suppresses the computation of standard errors and CIs. Use the `nose` option to speed up computations when analyzing census data. The `nose` option may also be useful to speed up computations when using a prefix command that uses replication techniques for variance estimation, such as [SVY] **svy jackknife**. The `vce(bootstrap)` and `vce(jackknife)` options imply `nose`.

Reporting

`level(#)` specifies the confidence level, as a percentage, for CIs. The default is `level(95)` or as set by `set level`.

`noheader` suppresses the output header; only the coefficient table is displayed.

`notable` suppresses the coefficient table.

`nogtable` suppresses the table containing Gini coefficients.

display_options are standard reporting options such as `cformat()`, `pformat()`, `sformat()`, or `coeflegend`; see [R] **estimation options**.

3.2 Syntax of pshare contrast

`pshare contrast` computes differences in percentile shares between outcome variables or subpopulations. It requires results from `pshare estimate` to be in memory, which will be replaced by the results from `pshare contrast`.[10] The syntax is

`pshare contrast` [*base*] [`, ratio lnratio stack`[`(`*options*`)`] `histogram`[`(`*options*`)`] *display_options*]

where *base* is the name of the outcome variable or the value of the subpopulation to be used as base for the contrasts. If *base* is omitted, `pshare contrast` computes adjacent contrasts across outcome variables or subpopulations (or contrasts with respect to the total if total results across subpopulations have been requested). Options are as follows:

`ratio` causes contrasts to be reported as ratios. The default is to report contrasts as differences.

`lnratio` causes contrasts to be reported as logarithms of ratios. The default is to report contrasts as differences.

`stack`[`(`*options*`)`] draws a stacked bar chart of the results. *options* are as described for `pshare stack` below.

`histogram`[`(`*options*`)`] draws a histogram of the results. *options* are as described for `pshare histogram` below.

display_options are standard reporting options such as `cformat()`, `pformat()`, `sformat()`, or `coeflegend`; see [R] **estimation options**.

10. Alternatively, to compute the contrasts directly, you may apply the `contrast()` option to `pshare estimate` (see above).

3.3 Syntax of pshare stack

`pshare stack` draws a stacked bar chart of percentile shares. It requires results from `pshare estimate` to be in memory.[11] The syntax is

`pshare stack` [`,` {`vertical`|`horizontal`} `proportion` `reverse` `keep(`*list*`)`
`sort`[`(`*options*`)`] `gini(%`*fmt*`)` `nogini` `labels("`*label1*`" "`*label2*`" ...)`
`plabels("`*label1*`" "`*label2*`" ...)` `barwidth(`*#*`)` *barlook_options*
`p`*#*`(`*barlook_options*`)` `values`[`(%`*fmt*`)`] *marker_label_options* `addplot(`*plot*`)`
twoway_options]

Options are as follows.

Main

`vertical` or `horizontal` specifies whether a vertical or a horizontal bar plot is drawn; the default is `horizontal`.

`proportion` scales the population axis as a proportion (0 to 1). The default is to scale the axis as a percentage (0 to 100).

`reverse` orders percentile groups from top to bottom (the richest are leftmost, the poorest are rightmost). The default is to order percentile groups from bottom to top (the poorest are leftmost, the richest are rightmost).

`keep(`*list*`)` selects and orders the results to be included as separate bars. Use `keep()` with multiple outcome variables or subpopulations. For multiple outcome variables, *list* is a list of the names of the outcome variables to be included. When `over()` was specified in `pshare estimate`, *list* is a list of the values of the subpopulations to be included. *list* may also contain `total` for the overall results (if overall results were requested). Furthermore, *list* may also contain elements such as `#1`, `#2`, `#3`, etc., to refer to the 1st, 2nd, 3rd, etc., outcome variable or subpopulation.

`sort`[`(`*options*`)`] orders the bars for the different outcome variables or subpopulations by the level of inequality, where *options* are `gini` to sort by Gini coefficients (if Gini coefficients have been computed), `descending` to sort in descending order, and `tfirst` or `tlast` to place the total across subpopulations first or last, respectively. The default is to sort in ascending order of the shares of the top percentile group.

`gini(%`*fmt*`)` sets the format for the Gini coefficients included in the graph as secondary axis labels; see [D] **format**. The default is `gini(%9.3g)`. Gini coefficients will be included only if information on Gini coefficients is available in the provided results (that is, if the `gini` option has been applied to `pshare estimate`).

11. You may also draw a chart directly by applying the `stack()` option to `pshare estimate` or `pshare contrast` (see above).

`nogini` suppresses the Gini coefficients. This is relevant only if the `gini` option has been specified when calling `pshare estimate`.

Labels/rendering

`labels("`*label1*`" "`*label2*`" ...)` specifies custom axis labels for the outcome variables or subpopulations.

`plabels("`*label1*`" "`*label2*`" ...)` specifies custom legend labels for the bar segments (that is, the percentile groups).

`barwidth(`#`)` sets the width of the bars as proportion of the spacing between bar positions. The default is `barwidth(0.75)`, leaving white space of 1/3 barwidth between the bars.

barlook_options and `p`#`(`*barlook_options*`)` affect the rendition of the plotted bars, where `p`#`()` applies to the #th segment (the #th percentile group) of the stacked bars; see [G-3] ***barlook_options***.

`values`[`(%`*fmt*`)`] prints the values of the percentile shares as marker labels at the center of the bar segments. The default is `values(%9.3g)`; see [D] **format**.

marker_label_options affect the rendition of the values included as marker labels using the `values()` option; see [G-3] ***marker_label_options***. Do not use `mlabel()` or `mlabvposition()`.

Add plots

`addplot(`*plot*`)` adds other plots to the generated graph; see [G-3] ***addplot_option***.

Y axis, X axis, Title, Caption, Legend, Overall

twoway_options are general twoway options, other than `by()`, as documented in [G-3] ***twoway_options***.

3.4 Syntax of pshare histogram

`pshare histogram` draws a histogram of percentile shares or percentile share contrasts. It requires results from `pshare estimate` or `pshare contrast` to be in memory.[12] The syntax is

`pshare histogram` [`,` {`vertical`|`horizontal`} `proportion` `keep(`*list*`)`
`max(`#[`,` *options*]`)` `min(`#[`,` *options*]`)` `prange(`*min max*`)` `gini(%`*fmt*`)` `nogini`
barlook_options `step` `spikes`[`(`#`)`] `labels("`*label1*`" "`*label2*`" ...)`
`byopts(`*byopts*`)` `overlay` `o`#`(`*options*`)` `psep`[`("`*label1*`" "`*label2*`" ...)`]
`p`#`(`*options*`)` `level(`#`)` `ci(`*citype*`)` `ciopts(`*options*`)` `cibelow` `noci`
`addplot(`*plot*`)` *twoway_options*]

Options are as follows.

Main

`vertical` or `horizontal` specifies whether a vertical or a horizontal plot is drawn. The default is to draw a vertical bar plot.

`proportion` scales the population axis as a proportion (0 to 1). The default is to scale the axis as a percentage (0 to 100).

`keep(`*list*`)` selects and orders the results to be included as separate subgraphs, where *list* is a list of the names of the outcome variables or the values of the subpopulations to be included. *list* may also contain `total` for the overall results if overall results were requested. Furthermore, you may use elements such as `#1`, `#2`, `#3`, ... to refer to the 1st, 2nd, 3rd, ... outcome variable or subpopulation.

`max(`#[`,` *options*]`)` top-codes results at # and `min(`#[`,` *options*]`)` bottom-codes results at #. This is useful if there are large differences in the plotted values and you want to restrict the axis range. The truncated values will be included in the graph as marker labels. *options* are `format(%`*fmt*`)` to set the format for the marker labels (default is `format(%9.3g)`; see [D] **format**), *marker_label_options* to affect the rendition of the marker labels (see [G-3] ***marker_label_options***), and `nolabels` to omit the marker labels.

`prange(`*min max*`)` restricts the range of percentile groups to be included in the graph. Only results for percentile groups whose lower and upper cumulative population bounds (in percent) are within *min* and *max* will be plotted. *min* and *max* must be within $[0, 100]$. For example, to include only the lower half of the distribution, type `prange(0 50)`.

12. You may also draw the histogram directly by applying the `histogram()` option to `pshare estimate` or `pshare contrast` (see above).

`gini(%`*fmt*`)` sets the format for the Gini coefficients included in the subgraph labels; see [D] **format**. The default is `gini(%9.3g)`. Gini coefficients will be included only if information on Gini coefficients is available in the provided results (that is, if the `gini` option has been applied to `pshare estimate`).

`nogini` suppresses the Gini coefficients. This is relevant only if the `gini` option has been specified when calling `pshare estimate`.

Labels/rendering

barlook_options affect the rendition of the plotted bars; see [G-3] ***barlook_options***.

`step` uses a step function (line plot) instead of a histogram to draw the results. Use *line_options* instead of *barlook_options* to affect the rendition of the plotted line; see [G-3] ***line_options***. `step` may be included in `o#()`, if `overlay` has been specified, to apply `step` to selected outcome variables or subpopulations (see below).

`spikes`[`(#)`] uses (equally spaced) spikes instead of histogram bars to draw the results. # specifies the number of spikes, and the default is `spikes(100)`. Use *line_options* instead of *barlook_options* to affect the rendition of the plotted spikes; see [G-3] ***line_options***. Confidence intervals will be omitted.

`labels("`*label1*`" "`*label2*`" ...)` specifies custom labels for the subgraphs of the outcome variables or subpopulations.

`byopts(`*byopts*`)` determines how subgraphs are combined; see [G-3] ***by_option***.

`overlay` includes results from multiple outcome variables or subpopulations in the same plot instead of creating subgraphs. `overlay` and `psep()` are not both allowed. Specifying `overlay` implies `noci`.

`o#(`*options*`)` affects the rendition of the bars of the #th outcome variable or subpopulation if `overlay` has been specified. *options* are `step` (draw step function instead of bars) and *barlook_options* (affect rendition of the plotted bars).

`psep`[`("`*label1*`" "`*label2*`" ...)`] causes different rendering to be used for each percentile group and includes a corresponding legend in the graph. The default is to draw all bars in the same style. `psep()` and `overlay` are not both allowed.

`p#(`*options*`)` affects the rendition of the bars of the #th percentile group if `psep()` has been specified. *options* are *barlook_options* (affect rendition of the plotted bars) and `ciopts(`*options*`)` (affect rendition of the confidence spikes).

Confidence intervals

`level(#)` specifies the confidence level, as a percentage, for CIs. The default is the level that has been used for computing the `pshare` results. `level()` cannot be used together with `ci(bc)`, `ci(bca)`, or `ci(percentile)`. To change the level for these CIs, you need to specify `level()` when computing the results.

`ci(`*citype*`)` chooses the type of CIs to be plotted for results that have been computed using the bootstrap technique. *citype* may be `normal` (normal-based CIs, the default), `bc` (bias-corrected [BC] CIs), `bca` (BC and accelerated CIs), or `percentile` (percentile CIs). `bca` is available only if BC_a CIs have been requested when running `pshare estimate` (see [R] **bootstrap**).

`ciopts(`*options*`)` affects the rendition of the plotted confidence spikes. *options* depend on the plot type used for the confidence spikes. The default plot type is capped spikes; see [G-2] **graph twoway rcap**. To use uncapped spikes, for example, type `ciopts(recast(rspike))`; see [G-2] **graph twoway rspike**. `ciopts()` may be included in `p#()`, if `psep` has been specified, to affect the rendition of the confidence spikes for selected percentile groups.

`cibelow` places CI spikes behind the plotted bars. The default is to draw the spikes in front of the bars.

`noci` omits CI spikes from the plot.

Add plots

`addplot(`*plot*`)` provides a way to add other plots to the generated graph; see [G-3] ***addplot_option***.

Y axis, X axis, Title, Caption, Legend, Overall

twoway_options are general twoway options, other than `by()`, as documented in [G-3] ***twoway_options***.

4 Examples

4.1 Basic application

By default, `pshare` computes outcome shares of quintile groups. The following example shows the results for wages in the 1988 extract of the U.S. National Longitudinal Study of Young Women data shipped with Stata:

```
. sysuse nlsw88
(NLSW, 1988 extract)
. pshare estimate wage, percent
Percentile shares (percent)       Number of obs   =      2,246
```

wage	Coef.	Std. Err.	[95% Conf.	Interval]
0-20	8.018458	.1403194	7.743288	8.293627
20-40	12.03655	.1723244	11.69862	12.37448
40-60	16.2757	.2068139	15.87013	16.68127
60-80	22.47824	.2485367	21.99085	22.96562
80-100	41.19106	.6246426	39.96612	42.41599

The `percent` option was specified to express results as percentages. We can see, for example, that the 20% best-earning women in the data receive 41% of the total of wages, whereas the 20% poorest-earning women receive only 8%. If wages were distributed evenly, then all quintile groups would receive 20%.

To compute decile shares, we could type

```
. pshare estimate wage, percent nquantiles(10)
Percentile shares (percent)       Number of obs   =      2,246
```

wage	Coef.	Std. Err.	[95% Conf.	Interval]
0-10	3.426509	.0702149	3.288816	3.564202
10-20	4.591949	.0813845	4.432352	4.751546
20-30	5.544608	.0842676	5.379357	5.709858
30-40	6.491941	.0934605	6.308663	6.675219
40-50	7.542334	.1023013	7.341719	7.742948
50-60	8.733366	.1131891	8.5114	8.955333
60-70	10.24571	.1284118	9.993888	10.49752
70-80	12.23253	.1367424	11.96438	12.50069
80-90	14.65518	.1493718	14.36226	14.9481
90-100	26.53588	.682887	25.19672	27.87503

The results indicate that the 10% best-earning women get 26.5% of the wages, whereas the 10% poorest-earning women get only 3.4%.

`pshare` does not require the percentile groups to be of equal size. To compute the shares of, say, the bottom 50%, the mid 40%, and the top 10%, we could type

```
. pshare estimate wage, percent percentiles(50 90)
Percentile shares (percent)       Number of obs   =      2,246
```

wage	Coef.	Std. Err.	[95% Conf.	Interval]
0-50	27.59734	.3742279	26.86347	28.33121
50-90	45.86678	.4217771	45.03967	46.6939
90-100	26.53588	.682887	25.19672	27.87503

The `percentiles()` option specifies the cutoffs defining the percentile groups. That is, `percentiles(50 90)` indicates to use three groups, 0–50, 50–90, and 90–100. We see that the lower-paid half of women gets about the same share of total wages as the best-paid 10%.

4.2 Stacked bar charts

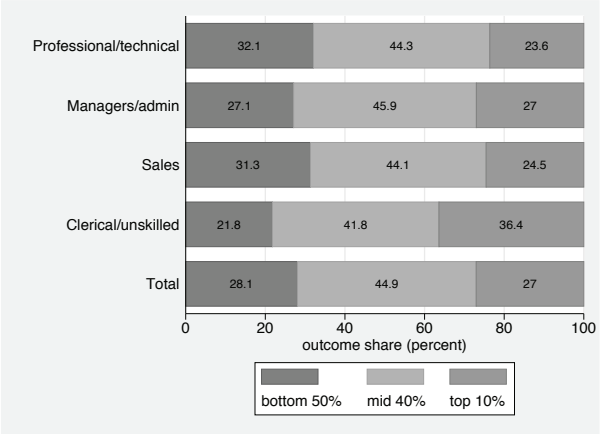
`pshare` supports two types of graphical displays of percentile shares. The first type is a stacked bar chart. For example, to compare wage distributions by some occupational groups, we could type

```
. pshare estimate wage if occupation<=4, percent percentiles(50 90)
> over(occupation) total gini
Percentile shares (percent)       Number of obs   =      1,409
            1: occupation = Professional/technical
            2: occupation = Managers/admin
            3: occupation = Sales
            4: occupation = Clerical/unskilled
```

wage	Coef.	Std. Err.	[95% Conf.	Interval]
1				
0-50	32.08652	.9560224	30.21114	33.9619
50-90	44.30132	.8461561	42.64146	45.96118
90-100	23.61216	1.468329	20.73181	26.49251
2				
0-50	27.11145	1.015934	25.11854	29.10436
50-90	45.90042	.8232238	44.28555	47.5153
90-100	26.98812	1.337874	24.36368	29.61256
3				
0-50	31.34111	.730376	29.90836	32.77385
50-90	44.1261	.7914729	42.57351	45.6787
90-100	24.53279	1.378169	21.8293	27.23627
4				
0-50	21.78931	1.909258	18.04401	25.53461
50-90	41.83106	2.046101	37.81733	45.84479
90-100	36.37963	2.898928	30.69295	42.06631
total				
0-50	28.06045	.4731704	27.13226	28.98865
50-90	44.91512	.4944292	43.94522	45.88502
90-100	27.02443	.8354367	25.38559	28.66326

	Gini
1	.273825
2	.3373482
3	.2833736
4	.4357447
total	.3279324

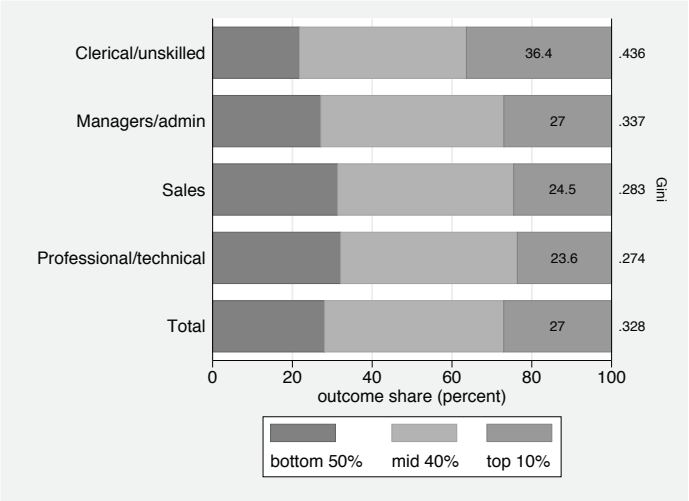
```
. pshare stack, plabels("bottom 50%" "mid 40%" "top 10%") values nogini
```



The `over(occupation)` option causes results to be computed by the subpopulations defined by the values of `occupation`, the `total` option requests total results across subpopulations to be included, and the `gini` option causes Gini coefficients to be computed. The `plabels()` option of `pshare stack` provides custom labels for the legend keys, the `values` option causes the values of the shares to be included as marker labels in the graph, and the `nogini` option suppresses the Gini coefficients that would be included in the graph as secondary axis labels (see next example).

To sort the bars by level of inequality, we could type

```
. pshare stack, plabels("bottom 50%" "mid 40%" "top 10%") values
> sort(gini tlast descending) mlabsize(zero) p3(mlabsize(small))
```

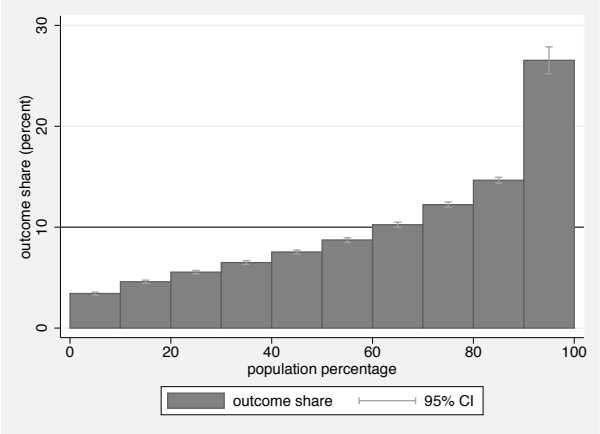


The `gini` argument in `sort()` causes bars to be sorted by Gini coefficients, `tlast` specifies placing the overall results last, and `descending` requests sorting from highest inequality to lowest inequality. The example also illustrates how to print marker labels only for specific percentile groups. The global option `mlabsize(zero)` sets the size of the marker labels to zero so that they are invisible, but `p3(mlabsize(small))` resets the marker label size for the third percentile group to `small`.

4.3 Histograms

The second type of graphical display supported by `pshare` is a percentile share histogram. The basic idea is to display a bar chart in which the area of each bar is proportional to the outcome share of the respective percentile group. An example with decile shares is as follows:

```
. pshare estimate wage, percent nquantiles(10)
  (output omitted)
. pshare histogram, yline(10)
```



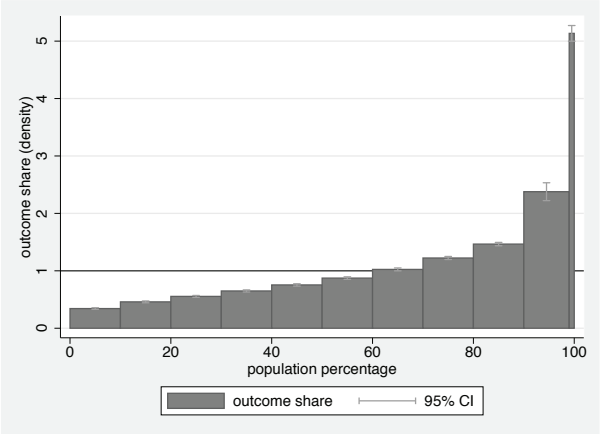
The `yline(10)` option was added to print a reference line at 10%. This would be the share each group would receive in an equal distribution.

If percentile groups are of unequal size, then densities instead of percentages or proportions should be used to construct the histogram (otherwise, the areas of the bars would no longer be proportional to the shares). Here is an example in which the top 1% is a separate group:

```
. pshare estimate wage, density percentiles(10(10)90 99)
Percentile shares (density)       Number of obs   =      2,246
```

wage	Coef.	Std. Err.	[95% Conf.	Interval]
0-10	.3426509	.0070215	.3288816	.3564202
10-20	.4591949	.0081384	.4432352	.4751546
20-30	.5544608	.0084268	.5379357	.5709858
30-40	.6491941	.009346	.6308663	.6675219
40-50	.7542334	.0102301	.7341719	.7742948
50-60	.8733366	.0113189	.85114	.8955333
60-70	1.024571	.0128412	.9993888	1.049752
70-80	1.223253	.0136742	1.196438	1.250069
80-90	1.465518	.0149372	1.436226	1.49481
90-99	2.377868	.0794248	2.222114	2.533622
99-100	5.135065	.0696951	4.998392	5.271739

```
. pshare histogram, yline(1)
```



Percentile share densities have an intuitive interpretation. They indicate how much each member in a group gets (on average) in relation to the overall average. In the example, we see that the average pay of the lowest 10% is only about 35% of the overall average. On the other hand, the members in the top percentage group earn wages that are more than five times the average wage. An alternative interpretation is as follows: Think of 100 representative dollars to be distributed among 100 people. In an equal distribution everyone would get one dollar. If, however, you divide the 100 dollars according to the observed distribution, then the density of a particular group indicates how many representative dollars a person in that group would get. In the example above, we see that the 10 women at the bottom would only get 35 cents each, whereas the top women would get more than 5 dollars (about 15 times as much). We also see that about 60% of the women are below the equal distribution line (that is, they receive below-average wages).

Note that the percentile density histogram is closely related to the so-called quantile plot (see [R] **diagnostic plots** and Cox [1999]), also known as Pen's "Parade of Dwarfs (and a few Giants)" (Pen 1971, 48–59). The difference is that a quantile plot usually displays individual observations using the original scale of the outcome variable. In the percentile density histogram, the values are averaged within bins and normalized by the population average.

4.4 Contrasts

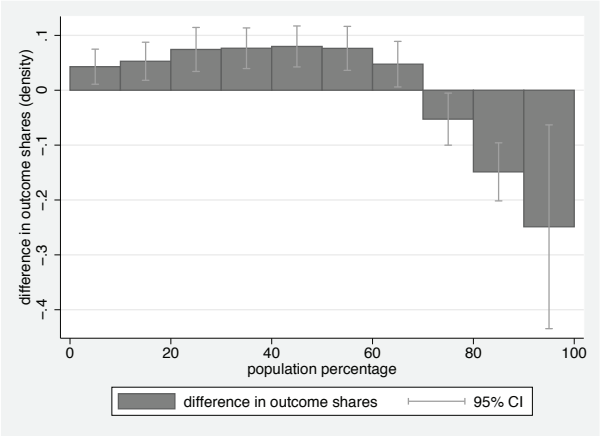
Differences between subpopulations

A useful feature of `pshare` is that contrasts between distributions can be computed. For example, the difference in the wage distribution between unionized and nonunionized women could be analyzed as follows:

```
. pshare estimate wage, density over(union) n(10)
  (output omitted)
. pshare contrast 0
Differences in percentile shares (density)      Number of obs     =      1,878
           0: union = nonunion
           1: union = union
```

wage	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
1						
0-10	.0429197	.016305	2.63	0.009	.0109419	.0748975
10-20	.0528084	.0177041	2.98	0.003	.0180866	.0875301
20-30	.0743417	.0204516	3.64	0.000	.0342315	.1144519
30-40	.0765406	.018892	4.05	0.000	.0394891	.1135922
40-50	.0798209	.0190538	4.19	0.000	.0424521	.1171897
50-60	.0763097	.0204552	3.73	0.000	.0361924	.116427
60-70	.0475279	.0211824	2.24	0.025	.0059843	.0890715
70-80	-.0526677	.0242038	-2.18	0.030	-.1001369	-.0051984
80-90	-.1487654	.0269943	-5.51	0.000	-.2017074	-.0958234
90-100	-.2488358	.094742	-2.63	0.009	-.4346464	-.0630251

```
(contrasts with respect to union = 0)
. pshare histogram
```

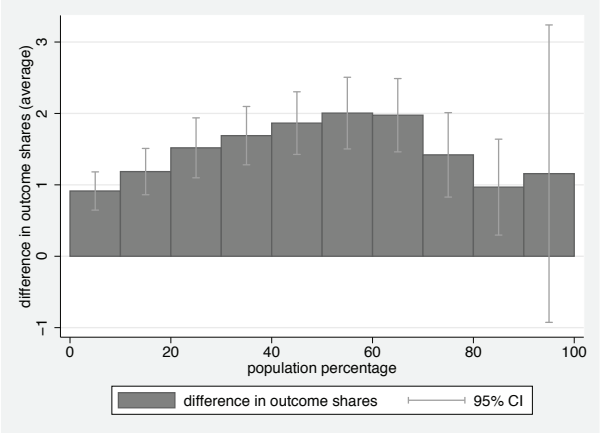


From the results, we see that the bottom 70% are relatively better off if unionized; the top 30% are relatively worse off. The differences are expressed in representative dollars; that is, the bottom 70% gain around 5 representative cents, and the top 10%

lose about a quarter of a representative dollar. However, note that these differences reflect only differences in the distributional shape; they are net of a possible overall difference in the wage levels between unionized and nonunionized workers.

To take the different wage levels of unionized and nonunionized workers into account, specify the `average` option so that the results are expressed as average wages. Furthermore, note that instead of using the `pshare contrast` command, you can also compute contrasts directly by applying the `contrast()` option to `pshare estimate`:

```
. pshare estimate wage, average over(union) n(10) contrast(0) histogram
```
(*output omitted*)



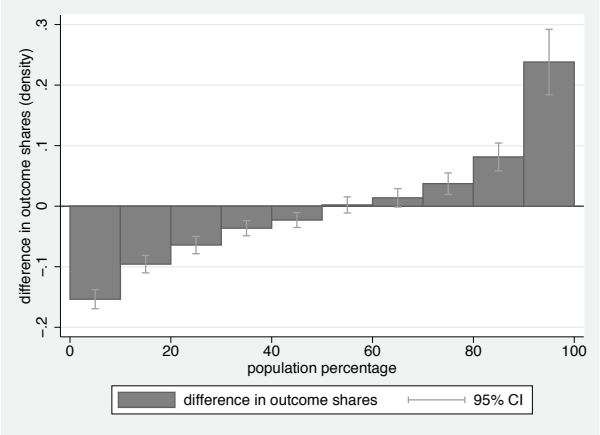
From these results, we see that unionized workers are better off across the board (by about one to two dollars per hour). Hence, from a welfare perspective, one could argue that the wage distribution of unionized women is strictly preferable over the wage distribution of nonunionized women (the wage distribution of unionized women generalized Lorenz dominants the wage distribution of nonunionized women; see, for example, Lambert [2001]). We also see that the (absolute) gains are somewhat larger in the middle of the distribution than at the top and at the bottom.

Differences between outcome variables

Instead of comparing subpopulations, `pshare` can also be used to compare distributions of different variables. For example, we could be interested in how the distribution changes once we move from hourly wages to weekly earnings:

```
. generate weekly = hours * wage
(4 missing values generated)
. label variable weekly "weekly earnings"
. pshare estimate wage weekly, density n(10) contrast(wage)
```
(*output omitted*)

```
. pshare histogram, yline(0)
```



We see that weekly earnings are considerably more unequal than hourly wages. Apparently, and as expected by economic theory, women with higher wages do supply more labor, so they get a larger share of weekly earnings than of hourly wages.

4.5 Concentration shares

The relation between two continuous variables can be analyzed by the `pshare` command using the `pvar()` option (in this case, percentile shares correspond to differences in concentration curve ordinates). In the last example, we saw that weekly earnings are distributed more unequally than hourly wages, which implies that women with higher wages work longer hours. Hence, it might be interesting to see how labor supply is distributed across wage groups:

```
. pshare estimate hours, pvar(wage) average n(10)
Percentile shares (average)       Number of obs   =      2,242
```

hours	Coef.	Std. Err.	[95% Conf.	Interval]
0-10	33.05259	.889763	31.30775	34.79744
10-20	33.6382	.8199639	32.03023	35.24616
20-30	34.78557	.7480189	33.31869	36.25245
30-40	37.14429	.6222536	35.92404	38.36454
40-50	37.73974	.6375459	36.4895	38.98998
50-60	38.6289	.670502	37.31403	39.94377
60-70	39.17663	.5903086	38.01902	40.33424
70-80	38.59946	.5712248	37.47928	39.71965
80-90	40.03568	.5799854	38.89832	41.17305
90-100	39.38002	.660688	38.08439	40.67564

```
(percentile groups with respect to wage)
```

The results indicate that average labor supply by women in the bottom 30% of the wage distribution is only about 33 to 35 hours per week, whereas in the upper half of the wage distribution, it is about 40 hours per week. To obtain results expressed in relation to the overall average, use the `density` option:

```
. pshare estimate hours, pvar(wage) density n(10)
Percentile shares (density)       Number of obs   =      2,242
```

hours	Coef.	Std. Err.	[95% Conf.	Interval]
0-10	.8880782	.0222773	.8443919	.9317646
10-20	.9038126	.0205245	.8635637	.9440616
20-30	.934641	.0188478	.8976801	.971602
30-40	.9980166	.0159431	.9667519	1.029281
40-50	1.014016	.0162895	.9820715	1.04596
50-60	1.037906	.0170757	1.00442	1.071392
60-70	1.052623	.0153487	1.022524	1.082722
70-80	1.037115	.0149871	1.007725	1.066505
80-90	1.075704	.0151754	1.045945	1.105464
90-100	1.058088	.0169731	1.024803	1.091372

```
(percentile groups with respect to wage)
```

We see, for example, that the weekly labor supply of women in the top 10% of the wage distribution is about 6% higher than average labor supply. The weekly labor supply of women in the bottom 10% of the wage distribution is 11% below the average.

The same technique could also be used, for example, to study the relation between income and wealth or between received bequests and existing income or wealth (for example, how much of the sum of all bequests in a given year goes to the wealthiest 10% of the population). Furthermore, it could be used to study the composition of income by sources or to study the effects of redistribution (for example, how much the different income percentiles contribute to overall taxes and how the empirical tax progression looks).

4.6 Processing results from pshare

`pshare estimate` and `pshare contrast` post their results in the `e()` return (see [P] **ereturn**; also see [U] **13.5 Accessing coefficients and standard errors**), so they can be processed by postestimation commands such as `test` (see [R] **test**), `lincom` (see [R] **lincom**), and `nlcom` (see [R] **nlcom**) or tabulated and graphed by programs such as `estout` (Jann 2005, 2007) and `coefplot` (Jann 2014).

For example, to compute the Palma ratio of wages—top 10% share divided by bottom 40% share (see, for example, Cobham, Schlogl, and Sumner [2015])—we could type

```
. pshare estimate wage, percentiles(40 90)
Percentile shares (proportion)    Number of obs   =      2,246
```

wage	Coef.	Std. Err.	[95% Conf.	Interval]
0-40	.2005501	.0029161	.1948315	.2062687
40-90	.5340912	.0048778	.5245258	.5436566
90-100	.2653588	.0068289	.2519672	.2787503

```
. nlcom (Palma: _b[90-100] / _b[0-40])
      Palma:  _b[90-100] / _b[0-40]
```

wage	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Palma	1.323155	.0506042	26.15	0.000	1.223972	1.422337

Furthermore, the Lorenz ordinates used to compute the percentile shares are stored by `pshare` in `e(L_ll)` (lower bounds) and `e(L_ul)` (upper bounds). To tabulate the Lorenz ordinates together with the percentile shares, we could type

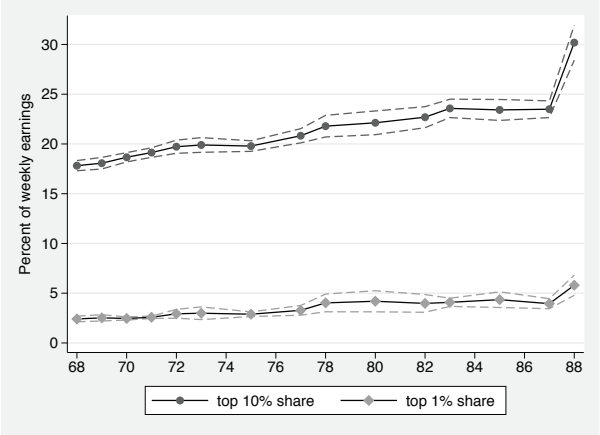
```
. pshare estimate wage
  (output omitted)
. estout, cell((b(label(share)) L_ll L_ul)) mlabels(none)
```

	share	L_ll	L_ul
0-20	.0801846	0	.0801846
20-40	.1203655	.0801846	.2005501
40-60	.162757	.2005501	.3633071
60-80	.2247824	.3633071	.5880894
80-100	.4119106	.5880894	1

Finally, `estimates store` (see [R] **estimates store**) can be used to make copies of results from different calls to `pshare` for later usage by commands such as `estout` or `coefplot`. In the following example, `coefplot` is used to plot the top decile share and the top centile share of weekly earnings against time:

```
. use http://www.stata-press.com/data/r14/nlswork.dta, clear
(National Longitudinal Survey.  Young Women 14-26 years of age in 1968)
. gen weekly = exp(ln_wage) * hours
(67 missing values generated)
. pshare estimate weekly, percent percentile(90) over(year) vce(cluster idcode)
  (output omitted)
. estimates store p90
. pshare estimate weekly, percent percentile(99) over(year) vce(cluster idcode)
  (output omitted)
```

```
. estimates store p99
. coefplot (p90, keep(*:90-100) label("top 10% share"))
>          (p99, keep(*:99-100) label("top 1% share")) ,
>          at(_eq) recast(connected) ciopts(recast(rline) lpattern(dash))
>          xlabel(68(2)88) ylabel(0(5)30, angle(horizontal))
>          ytitle("Percent of weekly earnings")
```



Through the years, as the respondents grew older, the share of the top decile increased from about 18% to 30%. The share of the top centile increased from 2.5% to about 5%.[13]

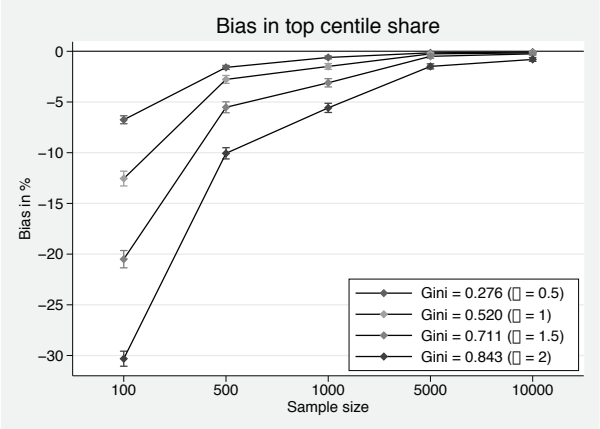
5 Small-sample bias

Estimates of percentile shares are affected by small-sample bias, especially at the top of the distribution. The bias can be substantial if the distribution is highly skewed and the number of observations is small. Thus, to obtain reliable estimates for shares of small top groups such as, say, the top 0.1% share, one must use large samples.

The simulation below provides some results for the relative bias in the estimate of the top 1% share for different sample sizes using a lognormal distribution. The scale parameter of the lognormal distribution is varied between $\sigma = 0.5$ (corresponding to a Gini coefficient of 0.276) and $\sigma = 2$ (corresponding to a Gini coefficient of 0.843).

13. The `vce(cluster idcode)` option has been added because the data are from a panel study where `idcode` identifies individuals. Adding the option in the example is not strictly necessary because the variances of the yearly estimates are not affected much by the clustering. However, it will be relevant once differences between years are analyzed.

```
. set seed 3230982
. program mysim, rclass
  1.     syntax [, n(integer 1000) Sigma(real 1) ]
  2.     drop _all
  3.     qui set obs `n´
  4.     tempvar y
  5.     gen `y´ = exp(rnormal(0, `sigma´))
  6.     pshare estimate `y´, nose percentile(99)
  7.     local b = 1 - normal(invnorm(0.99) - `sigma´)
  8.     return scalar bias = (_b[99-100] - `b´) / `b´
  9. end
. local i 0
. capture matrix drop R
. foreach sigma in 0.5 1 1.5 2 {
  2.     local ++i
  3.     local gini = 2*normal(`sigma´/sqrt(2)) - 1
  4.     foreach n in 100 500 1000 5000 10000 {
  5.         quietly simulate r(bias), reps(10000): mysim, n(`n´) sigma(`sigma´)
  6.         quietly ci means _sim_1
  7.         matrix tmp = r(mean), r(lb), r(ub)
  8.         matrix rownames tmp = s`i´:`n´
  9.         matrix R = nullmat(R) , tmp´
 10.     }
 11. }
. local i 0
. local plots
. foreach sigma in 0.5 1 1.5 2 {
  2.     local ++i
  3.     local lbl `: di %9.3f 2*normal(`sigma´/sqrt(2)) - 1´
  4.     local lbl Gini = `lbl´ ({&sigma} = `sigma´)
  5.     local plots `plots´ (matrix(R), keep(s`i´:) label("`lbl´"))
  6. }
. coefplot `plots´, ci((R[2] R[3])) vertical nooffset rescale(100)
>     msymbol(d) xtitle(Sample size) recast(connected) ciopts(recast(rcap))
>     ytitle(Bias in %) ylabel(#10, angle(horizontal)) yline(0)
>     title(Bias in top centile share) legend(cols(1) position(0) bplace(se))
```



For example, in a sample of 100 observations, the top centile share is underestimated by about 30% for a lognormal distribution with a Gini coefficient of 0.843. For lower levels of inequality, the underestimation is less severe but still substantial. This is not much of a surprise because in a sample of 100 observations, the top centile group contains only a single observation. However, also with a sample size of 1,000, the top centile share is underestimated by about 5% for the distribution with a Gini coefficient of 0.843.

The simulation results suggest that for moderately skewed distributions (such as the income distribution with a typical Gini coefficient between about 0.3 and 0.6), there should be a minimum of about 10 observations in the top group to keep the error within acceptable bounds of just a few percent. Estimating the top 0.1% share, for example, requires a sample size of at least 10,000 observations. However, for accurate estimation of top shares in extremely skewed distributions (such as the wealth distribution with Gini coefficients as high as 0.8 or event 0.9), minimum sample-size requirements may be considerably higher (such as 50 or even 100 observations in the top group).

6 Discussion

In this article, I presented only a selection of the features of the `pshare` command. It has been designed in such a way that it offers a wide variety of possible applications and can be used in many different situations. For example, much effort has been put into the support for complex survey data, a topic that has not been touched on in the examples. Nonetheless, a number of limitations and remaining issues need to be mentioned.

First, `pshare` is designed to be applied to individual-level data. Often, however, data on the distribution of income or wealth are available in the form of aggregate tables (typically from tax statistics). In such tables, individual-level units are grouped into outcome brackets, and for each bracket, the number of units and the outcome total are reported. `pshare` can be applied to such grouped data by computing the average outcome per bracket and weighting the data by the number of units. However, such a procedure assumes perfect equality within brackets and thus provides only a lower bound of the true inequality in the distribution (see, for example, Cowell [2011]). It would be worthwhile to develop a companion command for grouped data that also offers upper-bound estimates and intermediate estimates.

Second, analytic variance estimation implemented in `pshare` is only approximate and, possibly, more accurate estimation procedures could be developed. For example, variance estimation for percentile shares based on the concentration curve (that is, if the `pvar()` option is specified) requires the estimation of the expectation of the outcome variable at specific quantiles of the auxiliary variable. In the current implementation of `pshare`, this is accomplished by local linear regression using a constant bandwidth (see footnote 7). Some preliminary simulations indicate that this procedure generates consistent estimates of standard errors. However, the accuracy and stability of the standard error estimates could possibly be improved by using a variable bandwidth depending on the local density of the data. Furthermore, `pshare` reports symmetric,

normal-based CIs that may not be very accurate in small samples. A topic for future research could thus be to develop refined estimation of CIs.

Third, as discussed above, percentile shares are affected by small-sample bias. Future research will have to show whether a suitable correction procedure can be designed. A main challenge is to ensure that the correction does not increase the mean squared error (MSE) of the estimates. The problem can be illustrated by a simple bootstrap correction procedure. Let $\widehat{S}$ be the uncorrected estimate in the original sample and $\overline{S}$ be the mean of the estimates from a number of bootstrap samples. The bias in the bootstrap samples with respect to the original sample is then given as $\overline{S} - \widehat{S}$. The idea is to use the bootstrap bias as an approximation of the bias of the sample with respect to the population. Hence, a corrected estimate of S can be obtained as $\widehat{S}^{\text{corr}} = \widehat{S} - (\overline{S} - \widehat{S}) = 2\widehat{S} - \overline{S}$. Alternatively, the correction could also be based on ratios or on odds ratios between $\overline{S}$ and $\widehat{S}$. Findings from simulations with such procedures are that the bootstrap correction mostly removes the bias, unless the distribution is extremely skewed. At the same time, however, MSE increases. The reason is obvious: the larger the top share in a given sample turns out to be, the larger will be the bootstrap correction. This inflates sampling variance. Possibly, however, parametric extreme-value estimation may be used to design a correction procedure that does not increase the MSE.

7 Acknowledgments

The histogram and stacked bar plots produced by `pshare` have been inspired by the graphs shown in a video posted by Evan Klassen on YouTube[14] and a TED talk by Dan Ariely.[15] The `max()` and `min()` options of `pshare histogram` have been independently suggested by Hans-Jürgen Andreß and Jonas Meier. Furthermore, I would like to thank Stephen P. Jenkins for his helpful advice.

8 References

Atkinson, A. B., T. Piketty, and E. Saez. 2011. Top incomes in the long run of history. *Journal of Economic Literature* 49: 3–71.

Binder, D. A., and M. S. Kovacevic. 1995. Estimating some measures of income inequality from survey data: An application of the estimating equations approach. *Survey Methodology* 21: 137–145.

Cobham, A., L. Schlogl, and A. Sumner. 2015. Inequality and the tails: The Palma proposition and ratio revisited. United Nations DESA Working Paper No. 143. http://www.un.org/esa/desa/papers/2015/wp143_2015.pdf.

Cowell, F. A. 2011. *Measuring Inequality*. 3rd ed. Oxford: Oxford University Press.

14. See http://www.youtube.com/watch?v=slTF_XXoKAQ.
15. See https://www.ted.com/talks/dan_ariely_how_equal_do_we_want_the_world_to_be_you_d_be_surprised.

Cox, N. J. 1999. gr42: Quantile plots, generalized. *Stata Technical Bulletin* 51: 16–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 113–116. College Station, TX: Stata Press.

Hyndman, R. J., and Y. Fan. 1996. Sample quantiles in statistical packages. *American Statistician* 50: 361–365.

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

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

———. 2014. Plotting regression coefficients and other estimates. *Stata Journal* 14: 708–737.

Jenkins, S. P. 1999. sg104: Analysis of income distributions. *Stata Technical Bulletin* 48: 4–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 243–260. College Station, TX: Stata Press.

———. 2006. svylorenz: Stata module to derive distribution-free variance estimates from complex survey data, of quantile group shares of a total, cumulative quantile group shares. Statistical Software Components S456602, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s456602.html.

Kakwani, N. C. 1977. Measurement of tax progressivity: An international comparison. *Economic Journal* 87: 71–80.

Kovačević, M. S., and D. A. Binder. 1997. Variance estimation for measures of income inequality and polarization—The estimating equations approach. *Journal of Official Statistics* 13: 41–58.

Lambert, P. J. 2001. *The Distribution and Redistribution of Income*. 3rd ed. Manchester: Manchester University Press.

Lerman, R. I., and S. Yitzhaki. 1989. Improving the accuracy of estimates of Gini coefficients. *Journal of Econometrics* 42: 43–47.

Pen, J. 1971. *Income Distribution*. London: Allen Lane.

Piketty, T. 2014. *Capital in the Twenty-First Century*. Cambridge, MA: Belknap Press.

Piketty, T., and E. Saez. 2014. Inequality in the long run. *Science* 344: 838–843.

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*, the *Stata Journal*, *Public Opinion Quarterly*, and the *American Sociological Review*.

The Stata Journal (2016)
16, Number 2, pp. 301–315

Regression models for bivariate count outcomes

Xinling Xu
Department of Epidemiology and Biostatistics
University of South Carolina
Columbia, SC
xxu@email.sc.edu

James W. Hardin
Department of Epidemiology and Biostatistics
University of South Carolina
Columbia, SC
jhardin@sc.edu

Abstract. We present a new command, `bivcnto`, for fitting regression models suitable for analyzing correlated count outcomes. `bivcnto` allows specification of two correlated count outcomes with either two outcome-specific covariate lists or one common covariate list and fits models using a copula function approach in the general case or using specific parameterizations by Marshall and Olkin (1985, *Journal of the American Statistical Association* 80: 332–338) or Famoye (2010a, *Journal of Applied Statistics* 37: 969–981; 2010b, *Statistica Neerlandica* 64: 112–124). `bivcnto` also calculates a likelihood-ratio test comparing the joint model with estimation of two independent outcome-specific models.

Keywords: st0433, bivcnto, copula function, correlated count data, Poisson, negative binomial, Famoye bivariate Poisson regression, Marshall–Olkin bivariate negative binomial regression, Famoye bivariate negative binomial regression, Famoye bivariate generalized Poisson regression, general bivariate count regression

1 Introduction

While there are several official commands for analyzing a single-count outcome (`nbreg`, `poisson`, etc.) as well as user-written additions (for example, `nbregf` and `nbregw` from Harris, Hilbe, and Hardin [2014] and `nbregp` from Hardin and Hilbe [2014]), regression modeling of correlated count outcomes is not currently supported in Stata.

We present a new estimation command, `bivcnto`, to evaluate the Famoye bivariate Poisson regression model, the Marshall–Olkin bivariate negative binomial regression model, the Famoye bivariate negative binomial regression model, the Famoye bivariate generalized Poisson regression model, and a general bivariate count regression model (computed from copula functions) that allows the user to specify the marginal distribution of each outcome.

This article is organized as follows. In section 2, we introduce the general concepts of the copula approach and the various supported regression models. In section 3, syntax is presented for `bivcnto`, followed by examples in section 4.

2 Bivariate count-data models

2.1 Copula approach for bivariate data

While the distribution of univariate discrete data has been well developed, it is sometimes necessary to introduce a bivariate distribution for correlated discrete data. In many cases, when the marginal distributions are not independent, there is no explicit form for the bivariate distribution. Copula functions were originally introduced in Sklar (1959) and then revisited in Sklar (1973).

In general, if we define q random uniform variables, $U_1, \ldots, U_q$ on the $[0, 1]$ interval, we may define a function

$$C(u_1, \ldots, u_q) = P(U_1 \leq u_1, \ldots, U_q \leq u_q)$$

where the function $C(\cdot, \cdots, \cdot)$ is a copula and u_j is a particular realization of U_j for $j = 1, 2, \ldots, q$ for $q \geq 2$. To be considered a copula, the function C must have a domain on the q-dimensional unit hypercube, must be grounded, and must be increasing over its domain. If so, then we may write

$$\begin{aligned} C\{F_1(x_1), \ldots, F_q(x_q)\} &= P\left\{F_1^{-1}(U_1) \leq x_1, \ldots, F_q^{-1}(U_q) \leq x_q)\right\} \\ &= F(x_1, \ldots, x_q) \end{aligned}$$

That is, the copula as a function of the marginal distributions can be used to calculate the joint distribution; see Frees and Valdez (1998) for a complete review.

Using this approach, we can define

$$C(u, v; \theta) = F(y_1, y_2)$$

where $u = F_1(y_1)$, $v = F_2(y_2)$ are the cumulative distribution functions of the two marginal distributions, respectively, and the θ parameter measures the dependence between the two outcomes y_1 and y_2. In the related software, we have built-in support for calculations based on the following copula functions, where $\eta = 1 - \exp(-\theta)$. $\Phi^{-1}(\cdot)$ is the quantile function of the standard normal distribution, and $\Phi_2(\cdot, \cdot, \cdot)$ is the cumulative distribution function of the bivariate normal distribution:

$$\begin{aligned} C_1(u, v; \theta) &= -\frac{1}{\theta} \log\left[\left\{\eta - (1 - e^{-\theta u})(1 - e^{-\theta v})\right\}/\eta\right] \\ C_2(u, v; \theta) &= \Phi_2\left\{\Phi^{-1}(u), \Phi^{-1}(v), \theta\right\} \\ C_3(u, v; \theta) &= \left(u^{-\theta} + v^{-\theta} - 1\right)^{-1/\theta} \end{aligned}$$

We note that in each copula function, the dependence parameter θ has a different range and so is parameterized in the software in a function-specific manner:

$$\begin{aligned} C_1(u, v, \theta) &: -\infty < \theta < \infty \\ C_2(u, v, \theta) &: -1 \leq \theta \leq 1 \\ C_3(u, v, \theta) &: -1 < \theta < \infty, \theta \neq 0 \end{aligned}$$

In each case, an unrestricted parameter θ_u is used in the software, where θ is then calculated. The copula function C_1, known as Frank's copula and specified in the software as `copula(frank)`, is parameterized as $\theta = \theta_u \in \Re$ (without restrictions). The copula function C_2, known as the normal copula and specified in the software as `copula(normal)`, is parameterized using the inverse hyperbolic tangent function $\theta = \{\exp(2\theta_u) - 1\}/\{\exp(2\theta_u) + 1\} \in (-1, 1)$. The copula function C_3, known as the Kimeldorf and Sampson (KS) copula and specified in the software as `copula(kimeldorf)`, is parameterized as $\theta = \exp(\theta_u) - 1 \geq -1$; the software ensures the value of θ_u is never equal to 0. See Joe (1997) for additional properties of these copula functions.

In general, the normal copula function is popular in financial modeling. However, because it imposes a linear correlation structure on the two variables, it may not be suitable for every situation. As such, two well-known copula functions are introduced as alternatives.

Both Frank's copula and the KS copula belong to the family of Archimedean copulas. Frank's copula is a symmetric copula function, while the KS copula is asymmetric. One advantage of Archimedean copulas is they have an explicit form and are easily generated. Depending on whether the correlation is believed to be more positive or negative, the user can select an asymmetric copula.

Using the copula approach to calculating the joint distribution of the count outcomes, we can specify each marginal distribution, and the two distributions are not required to be the same. The `bivcnto` command will allow specification of each marginal distribution as Poisson, negative binomial, or generalized Poisson distribution; see Harris, Yang, and Hardin (2012), and download associated software for individual regression models based on this last distribution.

The Fréchet–Hoeffding theorem states that the following bounds hold for any copula:

$$\max\{F_1(y_1) + F_2(y_2) - 1, 0\} \leq F(y_1, y_2) \leq \min\{F_1(y_1), F_2(y_2)\}$$

The software imposes these bounds on the marginal probabilities to ensure that bivariate probabilities stay within the range.

There are many articles about the applications of copula functions for bivariate count data. Lee (1999) studied the application of the Frank copula in the Australian Rugby League dataset. The two marginals followed the negative binomial distributions. Even though the term "copula" was not explicitly mentioned, van Ophem (1999) used the normal copula with specified Poisson marginal distributions. McHale and Scarf (2007) described how to apply Archimedian copulas for Poisson–Poisson and negative-binomial–negative-binomial scenarios and presented detailed examples using the Frank copula and KS copula. The command we develop and describe, `bivcnto`, can be used for carrying out all the analyses described in these articles. The main inspiration for the work described here is Cameron et al. (2004). However, they were actually more focused on the induced distribution of the difference of two correlated count distributions.

2.2 The Famoye bivariate Poisson regression model

The Famoye bivariate Poisson regression model allows negative, zero, or positive correlation.

Famoye (2010a) presents a bivariate Poisson distribution given by

$$P(y_1, y_2) = \mu_1^{y_1}\mu_2^{y_2}e^{-\mu_1-\mu_2}\left\{1+\lambda(e^{-y_1}-e^{-d\mu_1})(e^{-y_2}-e^{-d\mu_2})\right\}/y_1!y_2!$$

where μ_1 and μ_2 are the mean parameters for the two marginal Poisson distributions, respectively, and $d = 1 - \exp(-1)$. In this presentation, the two outcomes are independent when $\lambda = 0$. Also, note that the sign of λ indicates the sign of the correlation between the outcomes. The correlations can be calculated per observation as

$$\rho_i = \lambda d^2\sqrt{\mu_{1i}\mu_{2i}}\exp\{-d(\mu_{1i}+\mu_{2i})\}$$

Subsequently, the average of the ρ_i can be used to summarize the correlation of the outcome variables.

2.3 Two parameterizations of the bivariate negative binomial regression model

Many approaches to bivariate count-data modeling have been suggested and researched. The new `bivcnto` command includes support for two parameterizations of bivariate negative binomial regression.

The Marshall–Olkin model

The Marshall–Olkin model is a "shared frailty" model for which the dispersion parameters of the marginal distributions are set to be equal, and there is no additional parameter directly measuring correlation between the two outcomes; see Marshall and Olkin (1985). The probability mass function if given by

$$f(y_1, y_2|\lambda_1, \lambda_2, \alpha) = \frac{\Gamma(y_1+y_2+\alpha)}{y_1!y_2!\Gamma(\alpha)}\left(\frac{\lambda_1}{\lambda_1+\lambda_2+1}\right)^{y_1}\left(\frac{\lambda_2}{\lambda_1+\lambda_2+1}\right)^{y_2}\left(\frac{1}{\lambda_1+\lambda_2+1}\right)^{\alpha}$$

where λ_1 and λ_2 are the two marginal means and α is the (common) overdispersion parameter.

While this approach defines a specific distribution for which estimates can easily be computed, the marginals are required to be negative binomial, the correlation between the outcomes is positive, and the heterogeneity in each marginal is assumed to be equal. The correlation between the outcomes for this model is given by

$$\mathrm{Corr}(y_1, y_2) = \frac{\lambda_1\lambda_2}{\sqrt{(\lambda_1^2+\alpha\lambda_1)(\lambda_2^2+\alpha\lambda_2)}}$$

The Famoye model

In comparison with the Marshall–Olkin bivariate negative binomial regression model, Famoye's presentation is more general. The Famoye model allows negative, zero, or positive correlation, and allows separate dispersion parameters for the marginal distributions.

Famoye (2010a) presents a bivariate negative binomial distribution given by

$$
\begin{aligned}
P(y_1, y_2) &= \left\{ \prod_{k=1}^{2} \binom{m_k^{-1} + y_k - 1}{y_k} \left(\frac{\mu_k}{m_k^{-1} + \mu_k} \right)^{y_k} \left(\frac{m_k^{-1}}{m_k^{-1} + \mu_k} \right)^{m_k^{-1}} \right\} \\
&\quad \times \left\{ 1 + \lambda \left(e^{-y_1} - c_1 \right) \left(e^{-y_2} - c_2 \right) \right\}
\end{aligned}
$$

where μ_k is the mean of the marginal negative binomial distribution with $k = 1, 2$, and $c_k = \{(1-\theta_k)/(1-\theta_k e^{-1})\}^{m_k^{-1}}$ with $\theta_k = \mu_k/(m_k^{-1} + \mu_k)$. In this presentation, the two outcomes are independent when $\lambda = 0$. Also, note that the sign of λ indicates the sign of the correlation between the outcomes. The parameter m_k is the dispersion parameter for the negative binomial distribution for $k = 1, 2$. As $m_k \to 0$, the negative binomial marginal model reduces to that of a Poisson marginal model. The correlations can be calculated per observation as

$$
\rho_i = \lambda d^2 \sqrt{\mu_{1i}\mu_{2i}(1 + m_1\mu_{1i})(1 + m_2\mu_{2i})} \;\; (1 + dm_1\mu_{1i})^{-1-1/m_1} \, (1 + dm_2\mu_{2i})^{-1-1/m_2}
$$

Subsequently, the average of the ρ_i can be used to summarize the correlation of the outcome variables.

2.4 The Famoye bivariate generalized Poisson regression model

Famoye (2010b) presents the following bivariate generalized Poisson distribution, which allows negative, zero, or positive correlation.

$$
\begin{aligned}
P(y_1, y_2) &= \left(\prod_{k=1}^{2} \left[\frac{\theta_k^{y_k} (1 + \alpha_k y_k)^{y_k - 1}}{y_k!} \exp\left\{ -\theta_k (1 + \alpha_k y_k) \right\} \right] \right. \\
&\qquad \left. \left\{ 1 + \lambda (e^{-y_1} - c_1)(e^{-y_2} - c_2) \right\} \right) \\
c_k &= \exp\left\{ \theta - t(s_k - 1) \right\} \\
0 &= \ln(s_k) - \alpha_k \theta_k (s_k - 1) + 1
\end{aligned}
$$

The mean parameters of the marginal generalized Poisson distributions are given by $\mu_k = \theta_k/(1 - \alpha_k\theta_k)$. In this presentation, the two outcomes are independent when $\lambda = 0$. Also, note that the sign of λ indicates the sign of the correlation between the outcomes.

3 Syntax

Software accompanying this article includes the command files as well as supporting files for prediction and help. In the following syntax diagrams, unspecified options include the usual collection of maximization and display options available to all estimation commands.

Equivalent in syntax to the `biprobit` command, the basic syntax for bivariate count regression models is presented as a bivariate syntax (common covariates for the two outcomes),

`bivcnto` *depvar*$_1$ *depvar*$_2$ [*indepvars*] [*if*] [*in*] [*weight*] [`, pfamoye gfamoye famoye molkin offset1(`*varname*`) offset2(`*varname*`) dist1(`*distribution*`) dist2(`*distribution*`) copula(`*function*`)`]

and as an outcome-specific covariate list syntax,

`bivcnto` *equation*$_1$ *equation*$_2$ [*if*] [*in*] [*weight*] [`, pfamoye gfamoye famoye molkin dist1(`*distribution*`) dist2(`*distribution*`) copula(`*function*`)`]

where *equation*$_1$ and *equation*$_2$ are specified as

`(`[*eqname*`:`] *depvar* [`=`] [*indepvars*] [`, noconstant offset(`*varname*`)`]`)`

In either case, the user specifies one of 1) `pfamoye` to fit Famoye's parameterization of a bivariate Poisson model, 2) `famoye` to fit Famoye's parameterization of a bivariate negative binomial model, 3) `molkin` to fit Marshall–Olkin's parameterization of a bivariate negative binomial model, 4) `gfamoye` to fit Famoye's parameterization of a bivariate generalized Poisson model, or 5) a combination of `dist1()`, `dist2()`, and `copula()` to fit a bivariate count model with given marginal distributions. *distribution* can be specified as `poisson`, `nbinomial`, or `gpoisson`; `copula()` can be specified as `frank`, `normal`, or `kimeldorf`.

Help files are included for the estimation and postestimation specifications of these models. The help files include example specifications of many possible models.

4 Example

In this example, we use data from the German health registry for the year 1984; see Hardin and Hilbe (2012). The data include responses by persons as to the number of doctor visits and number of hospital visits occurring in the previous year, as well as independent variables, including marital status, sex, whether the person has children, etc. Initially, we load the data and generate indicator variables of interest to assess the interaction of sex and marital status.

Because we believe that the number of doctor visits and hospital visits should be correlated for each person, we fit bivariate count-data models. Model 1 is a bivariate generalized Poisson model for which joint probabilities are estimated using the normal copula function. Relative to Poisson marginal distributions, this model will accommodate underdispersion or overdispersion.

```
. use rwm1984
(German health data for 1984; Hardin & Hilbe, GLM and Extensions, 3rd ed)
. correlate docvis hospvis
(obs=3,874)
             |   docvis  hospvis
-------------+------------------
      docvis |   1.0000
     hospvis |   0.1458   1.0000
. generate byte postHS = edlevel1==0
. generate byte MarM = (married==1 & female==0) // married males
. generate byte MarF = (married==1 & female==1) // married females (reference
> group)
. generate byte SinM = (married==0 & female==0) // single males
. generate byte SinF = (married==0 & female==1) // single females
```

```
. bivcnto docvis hospvis MarM SinM SinF kids outwork postHS, copula(normal) irr
> dist1(gp) dist2(gp) nolog
Bivariate count regression                      Number of obs   =       3874
Distribution 1 : gpoisson                       LR chi2(12)     =     279.76
Distribution 2 : gpoisson                       Prob > chi2     =     0.0000
Copula function: Normal
Log likelihood = -9528.074                      Pseudo R2       =     0.0145
```

	IRR	Std. Err.	z	P>|z|	[95% Conf.	Interval]
docvis						
MarM	.7794394	.040415	-4.81	0.000	.7041195	.8628163
SinM	.5928822	.0482416	-6.42	0.000	.5054845	.6953909
SinF	1.012853	.0635227	0.20	0.839	.8956983	1.14533
kids	.7019465	.0292074	-8.51	0.000	.646973	.761591
outwork	1.364784	.063108	6.73	0.000	1.246534	1.494251
postHS	.9114119	.0486906	-1.74	0.083	.8208064	1.012019
_cons	3.686083	.1952554	24.63	0.000	3.322585	4.089348
hospvis						
MarM	1.332045	.1976339	1.93	0.053	.9959257	1.781601
SinM	.8377243	.2073132	-0.72	0.474	.515766	1.36066
SinF	1.283144	.2433393	1.31	0.189	.8848118	1.860801
kids	.820675	.0997965	-1.63	0.104	.64664	1.041549
outwork	1.32767	.1787069	2.11	0.035	1.019805	1.728477
postHS	.7786416	.1272684	-1.53	0.126	.5652093	1.07267
_cons	.0929525	.0143513	-15.39	0.000	.0686816	.1258003
/atanhdelta1	.8420648	.0159916	52.66	0.000	.8107218	.8734077
/atanhdelta2	.262389	.02783	9.43	0.000	.2078433	.3169348
/atanhtheta	.3643823	.0280477	12.99	0.000	.3094099	.4193548
delta1	.6869011	.0084462			.6699882	.7031014
delta2	.2565287	.0259986			.2049013	.3067327
theta	.3490683	.0246301			.2999001	.3963867

```
LR test of independence: chi2(1) = 166.729                  Prob > chi2 = 0.0000
```

Model 2 is a bivariate negative binomial model for which joint probabilities are estimated using the normal copula function. Because the negative binomial marginal distributions allow for overdispersion relative to the Poisson model, the results should be comparable to model 1.

```
. bivcnto docvis hospvis MarM SinM SinF kids outwork postHS, copula(normal) irr
> dist1(nbinomial) dist2(nbinomial) nolog
Bivariate count regression                      Number of obs   =       3874
Distribution 1 : nbinomial                      LR chi2(12)     =     204.00
Distribution 2 : nbinomial                      Prob > chi2     =     0.0000
Copula function: Normal
Log likelihood = -9563.0986                     Pseudo R2       =     0.0106
```

	IRR	Std. Err.	z	P>|z|	[95% Conf.	Interval]
docvis						
MarM	.7553233	.0482379	-4.39	0.000	.6664565	.8560398
SinM	.5428058	.0539187	-6.15	0.000	.446778	.6594733
SinF	1.094595	.0964304	1.03	0.305	.9210115	1.300893
kids	.6699012	.0361331	-7.43	0.000	.6026964	.7445997
outwork	1.367538	.080724	5.30	0.000	1.218131	1.53527
postHS	.7767982	.0525775	-3.73	0.000	.680291	.8869961
_cons	3.815242	.2435281	20.98	0.000	3.366585	4.32369
hospvis						
MarM	1.339777	.2106125	1.86	0.063	.9845185	1.823229
SinM	.8486239	.2191493	-0.64	0.525	.5115643	1.407765
SinF	1.267091	.265061	1.13	0.258	.8409039	1.90928
kids	.8669514	.1132898	-1.09	0.275	.6710628	1.120022
outwork	1.318504	.1887649	1.93	0.053	.9959056	1.745599
postHS	.7299941	.1267115	-1.81	0.070	.5194799	1.025817
_cons	.0821457	.0128806	-15.94	0.000	.0604109	.1117003
/lnalpha1	.801881	.0312111	25.69	0.000	.7407083	.8630536
/lnalpha2	1.517031	.1581257	9.59	0.000	1.20711	1.826951
/atanhtheta	.3524641	.027713	12.72	0.000	.2981476	.4067807
alpha1	2.229731	.0695924			2.097421	2.370388
alpha2	4.558669	.7208429			3.343807	6.214911
theta	.3385591	.0245365			.2896165	.3857358

```
LR test of independence: chi2(1) = 194.597                  Prob > chi2 = 0.0000
```

Model 3 is a simplification wherein we illustrate whether the marginal distributions could be modeled as Poisson. This is for illustration more than anything because the first two models already established significant overdispersion. This model is a specific parameterization of the joint distribution that is not estimated via copula functions.

```
. bivcnto docvis hospvis MarM SinM SinF kids outwork postHS, irr pfamoye nolog
Famoye bivariate Poisson regression              Number of obs   =       3874
Distribution 1 : poisson                         LR chi2(12)     =    1525.70
Distribution 2 : poisson                         Prob > chi2     =     0.0000
Parameteriz.   : Famoye
Log likelihood = -17292.442                      Pseudo R2       =     0.0423
```

	IRR	Std. Err.	z	P>|z|	[95% Conf.	Interval]
docvis						
MarM	.8256116	.0194582	-8.13	0.000	.7883417	.8646434
SinM	.7184228	.0265246	-8.96	0.000	.6682719	.7723373
SinF	1.09806	.0306338	3.35	0.001	1.039631	1.159773
kids	.7111962	.0137636	-17.61	0.000	.6847252	.7386906
outwork	1.496067	.0309472	19.47	0.000	1.436625	1.557969
postHS	.7867019	.0205356	-9.19	0.000	.7474652	.8279982
_cons	3.475433	.0773372	55.98	0.000	3.327113	3.630365
hospvis						
MarM	1.506575	.1606378	3.84	0.000	1.222452	1.856734
SinM	.8145506	.1690816	-0.99	0.323	.5422863	1.22351
SinF	.9681015	.1582902	-0.20	0.843	.7026589	1.33382
kids	.9758035	.0906394	-0.26	0.792	.8133863	1.170652
outwork	1.685831	.1670805	5.27	0.000	1.388201	2.047271
postHS	.7455801	.0964673	-2.27	0.023	.5785769	.9607879
_cons	.1135148	.013544	-18.24	0.000	.0898446	.1434211
/lambda	1.239064	.0311917	39.72	0.000	1.17793	1.300199

```
LR test of independence: chi2(1) = 150.989                Prob > chi2 = 0.0000
```

Model 4 is a specific parameterization of a bivariate generalized Poisson regression model. It is similar to model 1, except that joint probabilities are not estimated using a copula function approach.

```
. bivcnto docvis hospvis MarM SinM SinF kids outwork postHS, irr gfamoye nolog
Famoye bivariate gen. Poisson regression          Number of obs   =       3874
Distribution 1 : gpoisson                         LR chi2(12)     =     274.35
Distribution 2 : gpoisson                         Prob > chi2     =     0.0000
Parameteriz.   : Famoye
Log likelihood = -9565.9571                       Pseudo R2       =     0.0141
```

	IRR	Std. Err.	z	P>|z|	[95% Conf.	Interval]
docvis						
MarM	.7763308	.039299	-5.00	0.000	.703004	.857306
SinM	.6034113	.0488873	-6.24	0.000	.5148142	.7072555
SinF	1.013709	.0634444	0.22	0.828	.8966841	1.146006
kids	.711594	.0296011	-8.18	0.000	.6558789	.7720418
outwork	1.363803	.0616085	6.87	0.000	1.248244	1.490061
postHS	.9097265	.0485966	-1.77	0.077	.8192956	1.010139
_cons	3.716485	.1944046	25.10	0.000	3.35434	4.117728
hospvis						
MarM	1.329661	.1971473	1.92	0.055	.9943383	1.778065
SinM	.8904528	.2160404	-0.48	0.632	.5534694	1.432611
SinF	1.251321	.2385659	1.18	0.240	.8611644	1.818241
kids	.8117869	.0988215	-1.71	0.087	.6394734	1.030532
outwork	1.430081	.1917152	2.67	0.008	1.099637	1.859824
postHS	.7621674	.1252311	-1.65	0.098	.5523192	1.051745
_cons	.1052071	.016458	-14.39	0.000	.0774263	.1429559
/atanhdelta1	.850666	.0161206	52.77	0.000	.8190702	.8822619
/atanhdelta2	.3611645	.0316393	11.42	0.000	.2991526	.4231764
/theta	1.61282	.1117518	14.43	0.000	1.393791	1.83185
delta1	.6914173	.008414			.6745635	.7075506
delta2	.3462394	.0278463			.2905369	.399603

```
LR test of independence: chi2(1) = 90.9624                Prob > chi2 = 0.0000
```

Model 5 is a specific parameterization of a bivariate negative binomial regression model. It is similar to model 2 (and model 6), except that joint probabilities are not estimated using a copula function approach.

```
. bivcnto docvis hospvis MarM SinM SinF kids outwork postHS, irr famoye nolog
Famoye bivariate neg bin regression              Number of obs   =       3874
Distribution 1 : nbinomial                       LR chi2(12)     =     196.14
Distribution 2 : nbinomial                       Prob > chi2     =     0.0000
Parameteriz.   : Famoye
Log likelihood = -9613.7926                      Pseudo R2       =     0.0101
```

	IRR	Std. Err.	z	P>|z|	[95% Conf.	Interval]
docvis						
MarM	.7850615	.0504826	-3.76	0.000	.6920987	.890511
SinM	.7066418	.0680447	-3.61	0.000	.585106	.8534226
SinF	1.114674	.0996621	1.21	0.225	.9354971	1.328168
kids	.7344114	.0395042	-5.74	0.000	.6609262	.816067
outwork	1.499644	.0882375	6.89	0.000	1.336301	1.682953
postHS	.7512271	.0515644	-4.17	0.000	.6566661	.859405
_cons	3.513729	.2202173	20.05	0.000	3.107567	3.972976
hospvis						
MarM	1.50087	.2589676	2.35	0.019	1.070218	2.104813
SinM	.8452774	.2431846	-0.58	0.559	.4809615	1.485553
SinF	.9809291	.2345484	-0.08	0.936	.6139149	1.567354
kids	1.042998	.1494766	0.29	0.769	.7875786	1.381252
outwork	1.630445	.2583429	3.09	0.002	1.195181	2.224223
postHS	.7499424	.138282	-1.56	0.119	.5224861	1.076418
_cons	.0858934	.0148254	-14.22	0.000	.0612407	.12047
/lnalpha1	.8320476	.0308378	26.98	0.000	.7716065	.8924886
/lnalpha2	2.259601	.1077669	20.97	0.000	2.048381	2.47082
/lambda	1.690835	.1250408	13.52	0.000	1.44576	1.935911
alpha1	2.298019	.0708659			2.163239	2.441197
alpha2	9.579261	1.032327			7.755337	11.83214

```
LR test of independence: chi2(1) = 93.209                  Prob > chi2 = 0.0000
```

Model 6 is a specific parameterization of a bivariate negative binomial regression model. It is similar to model 2 (and model 5), except that joint probabilities are not estimated using a copula function approach. Also, the Marshall–Olkin model (model 6) assumes that dependency of the two marginal distributions is captured through the constraint that the two marginal distributions have equal dispersion parameters; this is the reason there is only one dispersion parameter in the model output.

```
. bivcnto docvis hospvis MarM SinM SinF kids outwork postHS, irr molkin nolog
Marshall-Olkin bivariate neg bin regression       Number of obs   =       3874
Distribution 1 : nbinomial                        LR chi2(12)     =     231.33
Distribution 2 : nbinomial                        Prob > chi2     =     0.0000
Parameteriz.   : Marshall-Olkin
Log likelihood = -9814.5153                       Pseudo R2       =     0.0116
```

	IRR	Std. Err.	z	P>|z|	[95% Conf.	Interval]
docvis						
MarM	.790513	.0505189	-3.68	0.000	.6974478	.8959964
SinM	.7052426	.0671038	-3.67	0.000	.5852572	.8498267
SinF	1.127196	.0998536	1.35	0.177	.9475349	1.340923
kids	.7256723	.0385847	-6.03	0.000	.6538549	.8053781
outwork	1.49835	.0876521	6.91	0.000	1.336038	1.680381
postHS	.7449296	.0504891	-4.34	0.000	.6522639	.8507601
_cons	3.51977	.2189786	20.23	0.000	3.115715	3.976224
hospvis						
MarM	1.369553	.1743942	2.47	0.014	1.067062	1.757794
SinM	.6829585	.1530451	-1.70	0.089	.4401974	1.059598
SinF	.9624444	.1805015	-0.20	0.838	.6664035	1.389998
kids	1.019477	.1084942	0.18	0.856	.8275446	1.255925
outwork	1.662235	.1954399	4.32	0.000	1.320113	2.093022
postHS	.7681872	.1119307	-1.81	0.070	.5773519	1.0221
_cons	.0915673	.0120894	-18.11	0.000	.0706901	.1186102
/lnalpha	.8042573	.0304566	26.41	0.000	.7445636	.8639511
alpha	2.235036	.0680715			2.105522	2.372516

```
LR test of independence: chi2(1) = 308.236                Prob > chi2 = 0.0000
```

In a bivariate model of doctor visits and hospital visits (across all the highlighted examples), it can be seen that married men have a lower rate of doctor visits than married women. Single men have an even lower rate, but single women do not differ from married women in terms of their rates of doctor visits.

The correlation from the bivariate generalized Poisson model indicates that the outcomes are positively correlated at about 35%. This model has a better fit than two independent generalized Poisson regression models, with a χ^2 value of 166.7 and associated p-value less than 0.0001, thus rejecting the assumption of independence. The `delta1` and `delta2` from the output are the dispersion parameters for the marginal Poisson distribution. Relative to Poisson distribution, a positive value of this dispersion parameter indicates overdispersion and a negative value indicates underdispersion. From the output of the first model, we conclude that (relative to marginal Poisson distributions) the overdispersion of the number of doctor visits is about 2.5 times the overdispersion of the number of hospital visits (`delta1`/`delta2` = 2.7). The Famoye bivariate generalized poisson model result is similar, as shown in the fourth model output.

For the second model output using negative binomial distributions with the normal copula, the `alpha1` and `alpha2` output lines are for the dispersion parameters of the marginal negative binomial distributions, with larger values indicative of greater disper-

sion (relative to a marginal Poisson distribution). When $\alpha = 0$, the dispersion of the distribution is 0, and it reduces to a Poisson distribution. In both outputs using copula functions, the two estimated θ correlation parameters are close. The likelihood-ratio test of independence again rejects the independence assumption.

When we compare the last two models with the copula method, assuming the marginal distributions are negative binomial distributions, the significance of predictors for doctor visits do not change. The significant dependence parameters indicate a positive correlation between doctor visits and hospital visits.

5 References

Cameron, A. C., T. Li, P. K. Trivedi, and D. M. Zimmer. 2004. Modelling the differences in counted outcomes using bivariate copula models with application to mismeasured counts. *Econometrics Journal* 7: 566–584.

Famoye, F. 2010a. On the bivariate negative binomial regression model. *Journal of Applied Statistics* 37: 969–981.

———. 2010b. A new bivariate generalized Poisson distribution. *Statistica Neerlandica* 64: 112–124.

Frees, E. W., and E. A. Valdez. 1998. Understanding relationships using copulas. *North American Actuarial Journal* 2: 1–25.

Hardin, J. W., and J. M. Hilbe. 2012. *Generalized Linear Models and Extensions*. 3rd ed. College Station, TX: Stata Press.

———. 2014. Regression models for count data based on the negative binomial(p) distribution. *Stata Journal* 14: 280–291.

Harris, T., J. M. Hilbe, and J. W. Hardin. 2014. Modeling count data with generalized distributions. *Stata Journal* 14: 562–579.

Harris, T., Z. Yang, and J. W. Hardin. 2012. Modeling underdispersed count data with generalized Poisson regression. *Stata Journal* 12: 736–747.

Joe, H. 1997. *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

Lee, A. 1999. Applications: Modelling rugby league data via bivariate negative binomial regression. *Australian & New Zealand Journal of Statistics* 41: 141–152.

Marshall, A. W., and I. Olkin. 1985. A family of bivariate distributions generated by the bivariate Bernoulli distribution. *Journal of the American Statistical Association* 80: 332–338.

McHale, I., and P. Scarf. 2007. Modelling soccer matches using bivariate discrete distributions with general dependence structure. *Statistica Neerlandica* 61: 432–445.

Sklar, A. 1959. Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris* 8: 229–231.

———. 1973. Random variables, distribution functions, and copulas. *Kybernetica* 9: 449–460.

van Ophem, H. 1999. A general method to estimate correlated discrete random variables. *Econometric Theory* 15: 228–237.

About the authors

Xinling Xu is a PhD candidate in the Department of Epidemiology and Biostatistics at the University of South Carolina in Columbia, SC.

James W. Hardin is an associate professor in the Department of Epidemiology and Biostatistics and an affiliated faculty in the South Carolina Rural Health Research Center at the University of South Carolina in Columbia, SC.

The Stata Journal (2016)
16, Number 2, pp. 316–330

Implementing weighted-average estimation of substance concentration using multiple dilutions

Ying Xu
Center for Quantitative Medicine
Duke–NUS Graduate Medical School
Singapore, Singapore
tinayxu@gmail.com

Paul Milligan
Department of Infectious Disease Epidemiology
London School of Hygiene and Tropical Medicine
London, UK
paul.milligan@lshtm.ac.uk

Edmond J. Remarque
Department of Parasitology
Biomedical Primate Research Centre
Rijswijk, The Netherlands
remarque@gmail.com

Yin Bun Cheung
Center for Quantitative Medicine
Duke–NUS Graduate Medical School
Singapore, Singapore
and Department for International Health
University of Tampere
Tampere, Finland
yinbun.cheung@duke-nus.edu.sg

Abstract. In medicine and chemistry, immunoassays are often used to measure substance concentration. These tests use an S-shaped standard curve to map the observed optical responses to the underlying concentration. The enzyme-linked immunosorbent assay is one such test that is commonly used to measure antibody concentration in vaccine and infectious disease research. The enzyme-linked immunosorbent assay and other immunoassays usually involve a series of doubling or tripling dilutions of the test samples so that some of the diluted samples fall within the near-linear range in the center of the standard curve. The dilution that falls within or is nearest to the center of the near-linear range may then be selected for statistical analysis. This common practice of using one dilution does not fully use the information from multiple dilutions and reduces accuracy. We describe a recently proposed weighted-average estimation approach for analyzing multiple-dilution data (Cheung et al. 2015, *Journal of Immunological Methods* 417: 115–123), and we present the new `wavemid` command, which carries out the approach. We also present the new command `midreshape`, which processes raw data in text format exported from some microplate readers into analyzable data format. We use data from an experimental study of malaria vaccine candidates to demonstrate use of the two commands.

Keywords: st0434, wavemid, midreshape, immunoassay, multiple dilutions, weighted-average estimation

1 Introduction

In medicine and chemistry, immunoassays are important tools for detecting and quantifying substance concentration. For example, in vaccine and infectious disease research, the concentration of an antibody or antigen is often measured using the enzyme-linked immunosorbent assay (ELISA). This assay measures the concentration in each sample indirectly with an optical signal generated by an enzyme. For each test sample, the concentration is determined by comparing the observed optical density (OD) with an S-shaped standard curve (see figure 1 for an example). For example, if the observed OD is 2, then the estimated log(concentration) would be 2.56. Typically, the standard curve is a sigmoid curve described by a four-parameter logistic model on the logarithm scale (Ratkowsky and Reedy 1986; O'Connell, Belanger, and Haaland 1993) that characterizes the relationship between the OD and the concentration of the target substance in a set of standard solutions with known concentrations.

A common limitation of immunoassays is that they can obtain an accurate estimate of concentration only if the sample concentration falls within the "optimal" range (that is, the near-linear part in the center of the standard sigmoid curve). When the sample concentration falls outside this range, the assay lacks accuracy. To solve this, one can conduct a series of dilutions of each original test sample, as illustrated in figure 1. For each test sample, the observed response from one dilution that is within or nearest to the center of the optimal range is chosen for subsequent statistical analysis, while data from the other dilutions are ignored. The concentration of the substance in the original sample is then estimated using the inversion of a standard curve to obtain the concentration level of the chosen diluted sample and then multiplying the estimate by the corresponding dilution factor.

One problem with this approach is that selecting one diluted sample may involve some arbitrariness, thus limiting reproducibility. More importantly, using only one data point per original sample for statistical analysis does not fully use the available information, thus reducing accuracy. Remarque (2007) proposed calculating a weighted average of all data points by assigning the weights in a 100:1 ratio for data points in the optimal range versus those outside. This approach uses all the data. However, there is no formal justification to the weights and the ranges to which the weights are applicable, other than knowledge on the technical limits of the laboratory apparatus.

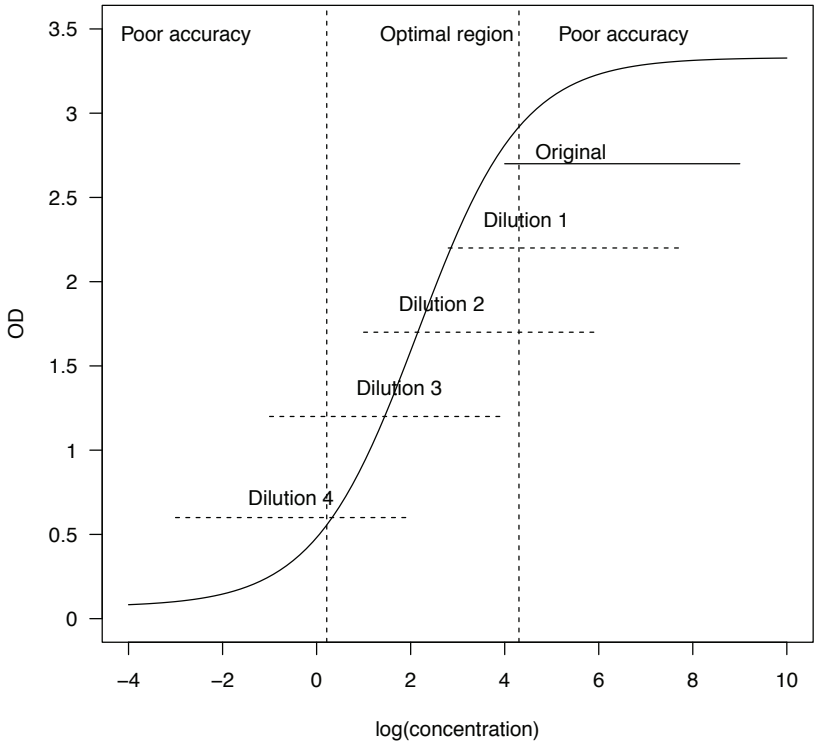


Figure 1. Illustration of a standard curve relating OD to sample concentration with four dilution levels of a set of original samples

To circumvent these problems, Cheung et al. (2015) proposed a weighted-average estimator for using data from multiple dilutions, where the weights are proportional to the inverse of the variance of the individual estimates. They showed by simulation that the proposed estimator yields more-accurate estimates. Using data from a vaccine study, they demonstrated that this method could lead to different practical conclusions. They also proposed a simplified version of the estimator, which is useful if the analyst cannot implement the inverse-variance method. This method is applicable to similar types of assays, not just ELISA.

In this article, we describe the inverse-variance weighted-average estimator proposed by Cheung et al. (2015), and we present a new command, `wavemid`, for implementing this procedure. We also present the new command `midreshape`, which processes raw data in text format exported from some microplate readers into analyzable data format. We use an experimental study of malaria vaccine candidates to illustrate the use of the two commands.

2 Weighted-average estimation approach

A typical microplate has 96 wells for an immunoassay measurement of up to 96 samples at the same time. Some of the wells hold standard samples whose true concentration levels are known. The remaining wells hold a set of (diluted) test samples with unknown concentrations. The outputs of the immunoassay include up to 96 observed OD values. To estimate unknown concentrations in the test samples, one must establish the standard curve that maps the observed response to the known concentrations by using the standard samples.

2.1 Estimation of the standard curve

Suppose one microplate holds n_s unique standard samples with known concentrations denoted by $x_{\text{Standard},1}, x_{\text{Standard},2}, \ldots, x_{\text{Standard},n_s}$. For the ith ($i = 1, \ldots, n_s$) standard sample, there are m_i replicate measurements. Typical laboratory practice is to have $m_i = 2$ or 3 for all i. The total number of replicated standard concentrations is $N_s = \sum_{i=1}^{n_s} m_i$. Let $y_{\text{Standard},i,j}$ denote the observed response for the jth replicate of a standard sample with concentration $x_{\text{Standard},i}$.

A common parameterization of the standard curve is a four-parameter logistic model,

$$E(Y) = Q(x|\mathbf{b}) = b_2 + \frac{b_1 - b_2}{1 + \left(\frac{x}{b_3}\right)^{b_4}} \tag{1}$$

where $E(Y)$ denotes the mean of response Y (for example, OD) at concentration x (O'Connell, Belanger, and Haaland 1993). In the parameter vector $\mathbf{b} = (b_1, b_2, b_3, b_4)^T$, b_1 and b_2 are, respectively, the lower and upper asymptotes of the standard curve as the concentration $x \to 0$ and $x \to \infty$. The parameter b_3 corresponds to the concentration at the midpoint of the two asymptotes, and b_4 is related to the slope of the standard curve. The variance of response Y is often formulated as a power-of-the-mean function, that is,

$$\text{Var}(Y) = \sigma^2 \{Q(x|\mathbf{b})\}^{2\theta} \tag{2}$$

where σ is the scale parameter and θ represents the degree of heteroskedasticity (Carroll and Ruppert 1988; Davidian and Haaland 1990). Expressions (1) and (2) together define the response model.

Given data for the standard samples $\{(y_{\text{Standard},i,j}, x_{\text{Standard},i}) : j = 1, \ldots, m_i; i = 1, \ldots, n_s\}$, a generalized least-squares (GLS) estimation algorithm may be used to fit the model parameters $\mathbf{b}$, σ, and θ. Further details of the GLS estimation algorithm can be found in Carroll and Ruppert (1988), Davidian and Haaland (1990), and O'Connell, Belanger, and Haaland (1993). Briefly, steps for this method are as follows:

1. Initialize the parameter $\mathbf{b}$ by an ordinary least-squares regression of $y_{\text{Standard},i,j}$ on $x_{\text{Standard},i}$ in the dataset. Denote the resultant estimate as $\widehat{\mathbf{b}}^{(s)}$ and $s = 0$.

2. Estimate (θ, σ) using the pseudolikelihood method, that is, by minimizing the following log-likelihood function in (θ, σ):

$$PL(\theta,\sigma) = N_s \log(\sigma) + \theta \sum_{i=1}^{n_s}\sum_{j=1}^{m_i} Q\left(x_{\text{Standard},i}|\widehat{\mathbf{b}}^{(s)}\right) + \frac{1}{2\sigma^2}\sum_{i=1}^{n_s}\sum_{j=1}^{m_i}\left[\frac{y_{\text{Standard},i,j} - Q\left(x_{\text{Standard},i}|\widehat{\mathbf{b}}^{(s)}\right)}{\left\{Q\left(x_{\text{Standard},i}|\widehat{\mathbf{b}}^{(s)}\right)\right\}^{\theta}}\right]^2$$

   Denote the resultant estimate as $(\widehat{\theta}^{(s)}, \widehat{\sigma}^{(s)})$. Form the estimated weights:

$$\widehat{v}_{ij} = \left\{Q\left(x_{\text{Standard},i}|\widehat{\mathbf{b}}^{(s)}\right)\right\}^{-2\widehat{\theta}^{(s)}}$$

3. Use $\widehat{v}_{ij}$ from step 2 to update the estimate $\widehat{\mathbf{b}}^{(s)}$ that minimizes

$$\sum_{i=1}^{n_s}\sum_{j=1}^{m_i}\widehat{v}_{ij}\left\{y_{\text{Standard},i,j} - Q(x_{\text{Standard},i}|\mathbf{b})\right\}^2$$

   Set $s = s + 1$ and return to step 2.

   Iterate between steps 2 and 3 until the parameter estimate $\widehat{\theta}$ converges. Denote the resultant estimates for the model parameters as $\widehat{\mathbf{b}} = (\widehat{b}_1, \widehat{b}_2, \widehat{b}_3, \widehat{b}_4)^T$, $\widehat{\sigma}$, and $\widehat{\theta}$.

Upon convergence, the variance of estimate $\widehat{\mathbf{b}}$ can be readily derived as

$$\text{Var}\left(\widehat{\mathbf{b}}\right) = \widehat{\sigma}^2(\mathbf{X}^T\mathbf{G}^{-1}\mathbf{X})^{-1}$$

where the matrix $\mathbf{X}$ is an $N_s \times 4$ gradient matrix. For this matrix, the columns are the partial derivative of the four-parameter logistic function $Q(x|b)$ with respect to each of the parameters in $\mathbf{b}$ evaluated at $\mathbf{b} = \widehat{\mathbf{b}}$, denoted by $\{\delta Q(x|\widehat{\mathbf{b}})/\delta\mathbf{b}\}^T$. The rows correspond to each of the N_s standard concentrations. The elements of row vector $\{\delta Q(x|\widehat{\mathbf{b}})/\delta\mathbf{b}\}^T$ are

$$\left\{\delta Q\left(x|\widehat{\mathbf{b}}\right)/\delta\mathbf{b}\right\}^T = \left\{\frac{1}{1+\left(\frac{x}{\widehat{b}_3}\right)^{\widehat{b}_4}}, \frac{\left(\frac{x}{\widehat{b}_3}\right)^{\widehat{b}_4}}{1+\left(\frac{x}{\widehat{b}_3}\right)^{\widehat{b}_4}}, \frac{\left(\frac{x}{\widehat{b}_3}\right)^{\widehat{b}_4}\left(\widehat{b}_1-\widehat{b}_2\right)\frac{\widehat{b}_4}{\widehat{b}_3}}{1+\left(\frac{x}{\widehat{b}_3}\right)^{\widehat{b}_4}}, -\frac{\left(\frac{x}{\widehat{b}_3}\right)^{\widehat{b}_4}\left(\widehat{b}_1-\widehat{b}_2\right)\log\frac{x}{\widehat{b}_3}}{1+\left(\frac{x}{\widehat{b}_3}\right)^{\widehat{b}_4}}\right\}$$

The first m_1 rows of the gradient matrix $\mathbf{X}$ are each $\{\delta Q(x_{\text{Standard},1}|\widehat{\mathbf{b}})/\delta\mathbf{b}\}^T$. The rows $(m_1 + 1)$ to $(m_1 + m_2)$ are each $\{\delta Q(x_{\text{Standard},2}|\widehat{\mathbf{b}})/\delta\mathbf{b}\}^T$, and so on. The matrix $\mathbf{G}$ is an $N_s \times N_s$ diagonal matrix, with the first m_1 diagonal elements equal to $\{Q(x_{\text{Standard},1}|\widehat{\mathbf{b}})\}^{2\widehat{\theta}}$, the $(m_1 + 1)$th to $(m_1 + m_2)$th diagonal elements equal to $\{Q(x_{\text{Standard},2}|\widehat{\mathbf{b}})\}^{2\widehat{\theta}}$, and so on.

2.2 Estimating concentration in test samples

Having established the standard curve, we now estimate the concentration in the test samples on the same microplate. Each original test sample with unknown concentration x is to be diluted in K steps. Typically, in immunoassays, $K = 4$. For the kth $(k = 1, \ldots, K)$ diluted sample, the underlying concentration is x/dil_k, where dil_k denotes the dilution factor. Often, $\text{dil}_{k+1}/\text{dil}_k = 2$ or 3, representing doubling or tripling of the dilutions. Let y_k denote the observed response, which can be the mean of r replicates. On the log scale, the estimate for the unknown concentration x based on its kth diluted sample (that is, the dilution-specific concentration estimate), denoted by $\widehat{x}_k$, is

$$\log(\widehat{x}_k) = \log\left(\widehat{b}_3\right) + \log(\text{dil}_k) + \frac{1}{\widehat{b}_4}\log\left(\frac{\widehat{b}_1 - y_k}{y_k - \widehat{b}_2}\right) \tag{3}$$

Cheung et al. (2015) proposed an inverse-variance weighted-average estimator of the log-transformed concentration, $\log(x)$, for the original serum sample, denoted as $\widehat{\log(x)}$,

$$\widehat{\log(x)} = \frac{1}{\sum_{k=1}^{K} w_k} \sum_{k=1}^{K} w_k \log(\widehat{x}_k) \tag{4}$$

where w_k is the weight assigned to the log-transformed, dilution-specific concentration estimate as follows:

$$w_k = \left[\left(\frac{1}{\widehat{b}_1 - y_k} + \frac{1}{y_k - \widehat{b}_2}\right)^2 y_k^{2\widehat{\theta}}/r + \widehat{b}_4^2 \left\{\frac{\delta \log(\widehat{x}_k)}{\delta \mathbf{b}}\right\}^T \left(\mathbf{X}^T \mathbf{G}^{-1} \mathbf{X}\right)^{-1} \left\{\frac{\delta \log(\widehat{x}_k)}{\delta \mathbf{b}}\right\}\right]^{-1} \tag{5}$$

Details of the derivation and interpretation of (2) are in Cheung et al. (2015). Briefly, (2) contains two additive terms. The first additive term involves a product of two elements. The first element contains the lower and upper asymptotes ($\widehat{b}_1$ and $\widehat{b}_2$). With other factors remaining the same, the closer the OD is to the midpoint of ($\widehat{b}_1$ and $\widehat{b}_2$) (the center of the optimal range), the heavier the weight it receives. The second element contains $\widehat{\theta}$, which reflects the degree of heteroskedasticity in the variance of the response. With other remaining factors the same, an OD value with higher variance receives a lower weight. The second additive term accounts for the uncertainty in estimating parameter vector $\mathbf{b}$ for the standard curve. It involves a column vector, $\{\delta \log(\widehat{x}_k)\}/\delta \mathbf{b}$, and two matrices, $\mathbf{X}$ and $\mathbf{G}$. The matrices $\mathbf{X}$ and $\mathbf{G}$ are both based on the standard samples and were previously defined in section 2.1. The column vector $\{\delta \log(\widehat{x}_k)\}/\delta \mathbf{b}$ is the partial derivative of $\log(\widehat{x}_k)$ for test sample (4) with respect to the parameter vector $\mathbf{b}$ evaluated at $\mathbf{b} = \widehat{\mathbf{b}}$.

$$\frac{\delta \log(\widehat{x}_k)}{\delta \mathbf{b}} = \frac{1}{\widehat{b}_4}\left\{\frac{1}{\widehat{b}_1 - y_k}, \frac{1}{y_k - \widehat{b}_2}, \frac{\widehat{b}_4}{\widehat{b}_3}, -\frac{1}{\widehat{b}_4}\log\left(\frac{\widehat{b}_1 - y_k}{y_k - \widehat{b}_2}\right)\right\}^T$$

The variance for the proposed inverse-variance weighted-average estimate is

$$\begin{aligned}\text{Var}\left\{\widehat{\log(x)}\right\} = &\frac{1}{\sum_{k=1}^{K} 1/\text{Var}\left\{\log\left(\widehat{x}_k\right)\right\}} \\ &+ \frac{2\widehat{\sigma}^2}{\left[\sum_{k=1}^{K} 1/\text{Var}\left\{\log\left(\widehat{x}_k\right)\right\}\right]^2} \sum\sum_{1\le j_1<j_2\le K} \frac{1}{\text{Var}\left\{\log\left(\widehat{x}_{j_1}\right)\right\}} \frac{1}{\text{Var}\left\{\log\left(\widehat{x}_{j_2}\right)\right\}} \\ &\left\{\frac{\delta\log\left(\widehat{x}_{j_1}\right)}{\delta\widehat{\mathbf{b}}}\right\}^T \left(\mathbf{X}^T\mathbf{G}^{-1}\mathbf{X}\right)^{-1} \left\{\frac{\delta\log\left(\widehat{x}_{j_2}\right)}{\delta\widehat{\mathbf{b}}}\right\} \end{aligned} \tag{6}$$

In (4), the term Var{log($\widehat{x}_k$)} is the variance of the dilution-specific estimate and can be estimated from a bivariate Taylor-series expansion of (4) with respect to y_k and $\widehat{\mathbf{b}}$, which results in

$$\begin{aligned}\text{Var}\left\{\log\left(\widehat{x}_k\right)\right\} = &\left\{\frac{1}{\widehat{b}_4}\left(\frac{1}{\widehat{b}_1 - y_k} + \frac{1}{y_k - \widehat{b}_2}\right)\right\}^2 \widehat{\sigma}^2 y_k^{2\widehat{\theta}}/r \\ &+ \left\{\frac{\delta\log\left(\widehat{x}_k\right)}{\delta\widehat{\mathbf{b}}}\right\}^T \text{Var}\left(\widehat{\mathbf{b}}\right) \left\{\frac{\delta\log(\widehat{x}_k)}{\delta\widehat{\mathbf{b}}}\right\}\end{aligned}$$

3 The wavemid command

3.1 Syntax

`wavemid` [*if*] [*in*], `testsample(`*varname*`)` `od(`*varname*`)` `standard(`*varname*`)`
`dilutionfactor(`*varname*`)` `id(`*varname*`)` `saving(`*filename*[, `replace`]`)`
[`iterate(`#`)` `tolerance(`#`)` `plot`
`graphexport(`*graphfilename*.*suffix*[, `replace`]`)`]

3.2 Options

`testsample(`*varname*`)` specifies a binary variable in the dataset that takes the value `1` for test samples and `0` for standard samples. `testsample()` is required.

`od(`*varname*`)` specifies a variable corresponding to the OD in the dataset. `od()` is required.

`standard(`*varname*`)` specifies a variable in the dataset that contains the concentration of the standard sample and the missing value for the test sample. `standard()` is required.

`dilutionfactor(`*varname*`)` specifies a variable in the dataset that contains the dilution factor of the test sample and the missing value for the standard sample. `dilutionfactor()` is required.

`id(`*varname*`)` specifies a variable in the dataset that identifies the samples. `id()` is required.

`saving(`*filename*[`, replace`]`)` creates an output data file (*filename*`.dta`) that contains the sample identifier, the concentration estimate, the variance of the log-transformed concentration estimate, the dilution-specific mean ODs among the replicates, and the dilution-specific weight. Use `replace` to overwrite the existing *filename*`.dta`. `saving()` is required.

`iterate(`#`)` specifies the maximum number of iterations for GLS estimation. When the number of iterations equals `iterate()`, estimation stops and presents the current results. If convergence is declared before this threshold is reached, estimation will stop when convergence is declared. The default is `iterate(10)`.

`tolerance(`#`)` specifies the tolerance for the parameter theta in the power-of-the-mean variance function. When the relative change in the parameter theta from one iteration to the next is less than or equal to `tolerance()`, the `tolerance()` convergence criterion is satisfied. The default is `tolerance(10^(-4))`.

`plot` plots the standard curve with the observed data for the standard samples on the logarithm scale for the concentration.

`graphexport(`*graphfilename*`.`*suffix*[`, replace`]`)` exports the graph that is generated after `plot` to *graphfilename*`.`*suffix*. Use `replace` to overwrite the existing *graphfilename*`.`*suffix*.

4 The midreshape command

Immunoassay data can be directly entered into Stata without using `midreshape`. However, some microplate readers export raw data and sample labels as a text file in a popular format. A typical layout of the text file includes the following three panels of data: 1) sample identifiers, 2) ODs, and 3) concentrations for standard samples and dilution factors for test samples. A name is shown in the line above each panel to describe what that panel is. Within each panel, the data values are exhibited as an 8×12 matrix corresponding to the 96-well microplate layout. See section 5 for an example. The `midreshape` command converts the text file into Stata and reshapes the data to the format required by `wavemid`. Users of `midreshape` should carefully examine whether their raw data file is in the format described here.

`midreshape using` *filename*`, template(`*string*`) od(`*string*`) concentration(`*string*`)`
`test(`*string*`) standard(`*string*`)` [`separator(`*list_separator*`)`]

4.1 Options

`template(`*string*`)` specifies the name of the panel corresponding to the sample identifiers. Space characters in the panel name may be safely ignored. This option is not case

sensitive. For instance, if the panel name is `Template Name`, then you may specify `template(Template Name)`, `template(TemplateName)`, `template(Templatename)`, `template(templateName)`, or `template(templatename)`. `template()` is required.

`od(`*string*`)` specifies the name of the panel corresponding to the ODs. Space characters in the panel name may be safely ignored. This option is not case sensitive. `od()` is required.

`concentration(`*string*`)` specifies the name of the panel that corresponds to the concentrations (for the standard samples) and dilution factors (for the test samples). Space characters in the panel name may be safely ignored. This option is not case sensitive. `concentration()` is required.

`test(`*string*`)` specifies the prefix of the sample identifiers to indicate which are the test samples. Multiple prefixes, separated by space characters, may be specified. This option is case sensitive. `test()` is required.

`standard(`*string*`)` specifies the prefix of the sample identifiers to indicate which are the standard samples. Multiple prefixes, separated by space characters, may be specified. This option is case sensitive. `standard()` is required.

`separator(`*list_separator*`)` specifies the character to be used to separate the values. `separator(comma)` specifies that values be comma-separated. `separator(tab)` specifies that values be tab-separated. Users may also specify other separation characters. For instance, if values in the file are separated by a semicolon, the user may specify `separator(";")`. The default is `separator(tab)`.

5 Example

5.1 Preparing an analysis dataset from a raw text file exported from an optical reader

The example dataset we use here is part of an experimental study that immunized rabbits with one of four malaria vaccine candidates. Here we used the data on antibodies to Apical Membrane Antigen 1 from one ELISA microplate. Apical Membrane Antigen 1 is an antigen that plays an important role in the invasion of red blood cells and hepatocytes by the parasites.

The 96-well microplate follows the typical 8-row by 12-column format. Each plate includes 2 blank samples (negative controls), 14 standard samples (seven concentrations with two replicates each), and 80 test samples. Each test sample begins with a starting dilution of 1:24000, followed by tripling dilutions, to obtain four diluted samples per serum sample. The data were stored in a text file, `example_12_tab.txt`, exported by a microplate reader. In the text file, the sample identifiers are shown in the following format:

```
Template
BLK   STD04 SMP05 SMP20 SMP29 SMP13 SMP22 SMP05 SMP20 SMP29 SMP13 SMP22
BLK   STD04 SMP05 SMP20 SMP29 SMP13 SMP22 SMP05 SMP20 SMP29 SMP13 SMP22
STD07 STD03 SMP06 SMP21 SMP30 SMP14 SMP28 SMP06 SMP21 SMP30 SMP14 SMP28
STD07 STD03 SMP06 SMP21 SMP30 SMP14 SMP28 SMP06 SMP21 SMP30 SMP14 SMP28
STD06 STD02 SMP13 SMP22 SMP05 SMP20 SMP29 SMP13 SMP22 SMP05 SMP20 SMP29
STD06 STD02 SMP13 SMP22 SMP05 SMP20 SMP29 SMP13 SMP22 SMP05 SMP20 SMP29
STD05 STD01 SMP14 SMP28 SMP06 SMP21 SMP30 SMP14 SMP28 SMP06 SMP21 SMP30
STD05 STD01 SMP14 SMP28 SMP06 SMP21 SMP30 SMP14 SMP28 SMP06 SMP21 SMP30
```

`BLK` indicates blank samples, and the prefixes `STD` and `SMP` differentiate the standard samples from the test samples, respectively. The duplicated numeric suffixes indicate two replicates per standard sample and per diluted test sample. All values are tab-delimited. The raw ODs corresponding to these samples are displayed in the text file following the same matrix format:

```
Raw Data
0.070 0.740 2.976 2.362 2.229 2.339 1.731 1.103 0.656 0.527 0.825 0.460
0.068 0.824 2.728 2.712 1.792 2.652 1.682 1.279 0.727 0.480 0.816 0.400
0.112 1.389 2.427 2.619 3.165 1.934 1.871 0.982 0.816 1.028 0.570 0.568
0.125 1.360 2.617 2.158 2.942 2.085 1.986 0.931 0.703 1.143 0.458 0.564
0.185 2.176 2.882 2.497 2.130 1.453 1.177 1.809 0.908 0.519 0.359 0.241
0.193 2.277 3.126 2.410 1.958 1.402 1.274 1.400 0.971 0.526 0.334 0.276
0.387 2.798 2.640 2.853 1.719 1.318 1.961 1.251 1.007 0.456 0.321 0.509
0.304 2.775 2.937 2.806 1.826 1.596 2.153 1.166 1.039 0.405 0.341 0.568
```

This is followed by a third data matrix that shows concentrations for the standard samples and dilution factors for the test samples:

```
Concentrations / Dilutions
       1.9235  24000 24000 24000 72000 72000 216000 216000 216000 648000 648000
       1.9235  24000 24000 24000 72000 72000 216000 216000 216000 648000 648000
0.0712 5.7704  24000 24000 24000 72000 72000 216000 216000 216000 648000 648000
0.0712 5.7704  24000 24000 24000 72000 72000 216000 216000 216000 648000 648000
0.2137 17.3111 24000 24000 72000 72000 72000 216000 216000 648000 648000 648000
0.2137 17.3111 24000 24000 72000 72000 72000 216000 216000 648000 648000 648000
0.6411 51.9333 24000 24000 72000 72000 72000 216000 216000 648000 648000 648000
0.6411 51.9333 24000 24000 72000 72000 72000 216000 216000 648000 648000 648000
```

The first two elements in the first column of the above matrix were blank, corresponding to the two blank samples as seen in the `Template` block. Each test sample (for example, `SMP05`) occupied eight wells, that is, two replicates in each of the four dilution levels.

The blank samples on microplates are usually used for quality control purposes, and thus we do not include the two blank samples in our statistical analysis. We now use the `midreshape` command to prepare the dataset for further analysis.

```
. midreshape using example_12_tab.txt, template(Template) od(Raw Data)
> concentration(Concentrations / Dilutions) test(SMP) standard(STD)
```

The command creates five variables: `testSample` (`1` if a test sample and `0` if a standard sample), `ID`, `OD`, `standard` (missing if `testSample` = `1`), and `dilution_factor` (missing if `testSample` = `0`). Data are sorted and shown in ascending order for

`testSample`, `ID`, and `dilution_factor`. The table below shows the actual data for the first 20 rows.

```
. list in 1/20, abbreviate(20) separator(20)
```

	testSample	ID	OD	standard	dilution_factor
1.	0	STD01	2.798	51.9333	.
2.	0	STD01	2.775	51.9333	.
3.	0	STD02	2.176	17.3111	.
4.	0	STD02	2.277	17.3111	.
5.	0	STD03	1.389	5.7704	.
6.	0	STD03	1.36	5.7704	.
7.	0	STD04	.824	1.9235	.
8.	0	STD04	.74	1.9235	.
9.	0	STD05	.387	.6411	.
10.	0	STD05	.304	.6411	.
11.	0	STD06	.185	.2137	.
12.	0	STD06	.193	.2137	.
13.	0	STD07	.125	.0712	.
14.	0	STD07	.112	.0712	.
15.	1	SMP05	2.976	.	24000
16.	1	SMP05	2.728	.	24000
17.	1	SMP05	1.958	.	72000
18.	1	SMP05	2.13	.	72000
19.	1	SMP05	1.279	.	216000
20.	1	SMP05	1.103	.	216000

5.2 Implementing the weighted-average estimation approach using the wavemid command

Now we apply the `wavemid` command to the data described previously.

```
. wavemid, testsample(testSample) od(OD) standard(standard)
> dilutionfactor(dilution_factor) id(ID) saving(plate_estimate, replace)
> plot graphexport(standard_curve.png,replace)
file plate_estimate.dta saved
(file standard_curve.png written in PNG format)
Estimation of Standard Curve Based on Standard Samples.................
```

LOF: Source	SS	df	MS
Pure Error	.012874484	7	.001839212
Lack of Fit	.007700135	3	.002566712
Error	.020574615	10	.002057462

```
Number of obs  =        14
F( 3,    7)    =    1.3955
Prob   >   F   =    0.3215
R-squared      =  .9984546
Root MSE       =  .0014696
```

	Est	Std Err
b1	0.0742918	0.1485931
b2	3.3345455	0.0771453
b3	8.6078653	2.3250908
b4	0.8994575	0.0428863
theta	0.4502218	
sigma	0.0418301	

The regression table above gives the parameter estimates for $\mathbf{b}$, σ, and θ. A lack-of-fit test is also conducted to assess how well this estimated standard curve fit to the observations from standard samples (O'Connell, Belanger, and Haaland 1993). This is a common practice in immunoassays. The residual sum of squares error is partitioned into two components: the sum of squares due to pure error (SSPE) and the sum of squares due to lack of fit (SSLOF). Further details on how to calculate the sum of squares error, SSPE, and SSLOF can be found elsewhere (see, for example, Brook and Arnold [1985, 48–49]).

Then, a variance-ratio F test can be conducted using the test statistic SSLOF/SSPE, which follows an F distribution with degrees of freedom $d_1 = n_s - p$ and $d_2 = N_s - n_s$, where n_s is the number of unique concentrations in the standard samples, p is the number of parameters in the standard curve mean response model, and N_s is the total number of standard samples on the microplate. In this example, $n_s = 7$, $p = 4$ (that is, a four-parameter logistic model), and $N_s = 14$. Thus, $d_1 = 3$ and $d_2 = 7$, hence, SSLOF/SSPE $\sim F_{3,7}$. The lack-of-fit test shown in the above table gave an insignificant p-value (0.3215), which suggests that the null hypothesis of lack of fit could not be rejected at the 5% significance level. The R-squared value being close to 1 also indicated that the estimated standard curve fit the data well on the standard samples. This is further demonstrated graphically in figure 2, which is produced by the `plot` option and depicts the estimated standard curve on the log(concentration) scale together with the scatterplot of the observed data on the standard samples. By specifying the `graphexport()` option, the generated graph is saved to an external file.

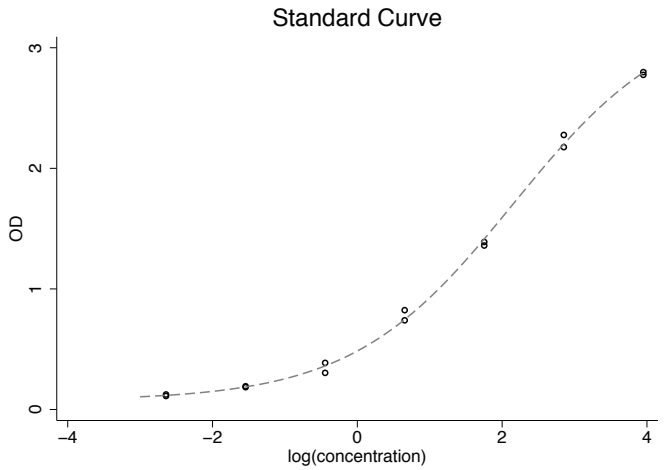


Figure 2. Plot of standard curve and observed data generated by `wavemid`

The results on concentration estimates for the test samples are saved in `plate_estimate.dta`.

```
. use plate_estimate, clear
. describe
Contains data from plate_estimate.dta
  obs:            10
 vars:            15                          16 Nov 2015 11:33
 size:           610
```

variable name	storage type	display format	value label	variable label
ID	str5	%9s		Sample ID
CONC	float	%9.0g		Weighted average estimate
VAR_logCONC	float	%9.0g		Var(ln(CONC))
dilution_fact~1	float	%9.0g		Dilution factor
OD_avg1	float	%9.0g		Mean of responses
Wt_log1	float	%9.0g		Dilution-specific weight
dilution_fact~2	float	%9.0g		Dilution factor
OD_avg2	float	%9.0g		Mean of responses
Wt_log2	float	%9.0g		Dilution-specific weight
dilution_fact~3	float	%9.0g		Dilution factor
OD_avg3	float	%9.0g		Mean of responses
Wt_log3	float	%9.0g		Dilution-specific weight
dilution_fact~4	float	%9.0g		Dilution factor
OD_avg4	float	%9.0g		Mean of responses
Wt_log4	float	%9.0g		Dilution-specific weight

```
Sorted by: ID  CONC  VAR_logCONC
```

The variables `CONC` and `VAR_logCONC` are calculated based on (5) and (4), respectively. The variables `Wt_log1` to `Wt_log4` are the weights at each of the four dilution levels, each calculated based on (2). Below is a list of these data for each of the 10 test samples.

```
. list ID CONC VAR_logCONC Wt_log*, separator(10)
```

	ID	CONC	VAR_lo~C	Wt_log1	Wt_log2	Wt_log3	Wt_log4
1.	SMP05	949363.4	.0056274	.1191221	.1641234	.6275659	.0891886
2.	SMP06	630065.2	.0033245	.0823517	.1337856	.7578192	.0260436
3.	SMP13	1578578	.0030945	.0703939	.0967548	.1928905	.6399609
4.	SMP14	939857.7	.0060724	.1258611	.1737726	.6145999	.0857664
5.	SMP20	433232.4	.0026274	.1522356	.4140148	.411734	.0220156
6.	SMP21	445023.4	.0024643	.1270483	.3108616	.5471681	.0149219
7.	SMP22	602859.1	.0031466	.082595	.1418577	.7500857	.0254616
8.	SMP28	758736.4	.0034011	.0812947	.1216422	.7110945	.0859686
9.	SMP29	305864.5	.0045231	.2056204	.6929231	.0930961	.0083604
10.	SMP30	861425.4	.0037226	.0629233	.1337282	.7196823	.0836661

6 Conclusion

In immunoassays, accurate measurement of concentration using a standard curve requires the use of a dilution that ensures the OD is in the near-linear part of the curve. Analyzing only one dilution wastes useful information and decreases measurement accuracy. We have, therefore, developed an approach that allows data from several dilutions of each sample to be used in a weighted analysis, which gives improved estimates of the substance concentration. The superiority of this approach versus the conventional approach was previously demonstrated by Cheung et al. (2015).

In this article, we demonstrated how to implement inverse-variance weighted-average estimation to analyze data on multiple dilutions by using the `wavemid` command. We also discussed how the `midreshape` command can be used to convert data from the format commonly provided by a microplate reader. These commands are useful when using Stata in laboratory-based biomedical research.

7 Acknowledgments

This work was supported by the National Research Foundation in Singapore, under its Clinician Scientist Award (Award No. NMRC/CSA/039/2012) administered by the Singapore Ministry of Health's National Medical Research Council.

8 References

Brook, R. J., and G. C. Arnold. 1985. *Applied Regression Analysis and Experimental Design*. New York: CRC Press.

Carroll, R. J., and D. Ruppert. 1988. *Transformation and Weighting in Regression*. New York: Chapman & Hall.

Cheung, Y. B., Y. Xu, E. J. Remarque, and P. Milligan. 2015. Statistical estimation of antibody concentration using multiple dilutions. *Journal of Immunological Methods* 417: 115–123.

Davidian, M., and P. D. Haaland. 1990. Regression and calibration with nonconstant error variance. *Chemometrics and Intelligent Laboratory Systems* 9: 231–248.

O'Connell, M. A., B. A. Belanger, and P. D. Haaland. 1993. Calibration and assay development using the four-parameter logistic model. *Chemometrics and Intelligent Laboratory Systems* 20: 97–114.

Ratkowsky, D. A., and T. J. Reedy. 1986. Choosing near-linear parameters in the four-parameter logistic model for radioligand and related assays. *Biometrics* 42: 575–582.

Remarque, E. J. 2007. Auditable data analysis and management system for ELISA (ADAMSEL-v1.1). http://www.transvac.org/sites/default/files/uploads/docs/Projects/OPTIMALVAC/ADAMSEL_30_Manual.pdf.

About the authors

Ying Xu is an assistant professor in the Center for Quantitative Medicine at Duke–NUS Medical School in Singapore. Her research interests include statistical methodology related to infectious disease and human growth as well as design and analysis of clinical trials.

Paul Milligan is a reader in epidemiology and medical statistics at the London School of Hygiene and Tropical Medicine. His research focuses on the epidemiology and control of malaria and statistical methods with application to infectious disease research.

Edmond J. Remarque is a principal investigator at the Biomedical Primate Research Center in the Netherlands. His research interests include immunology, vaccines, and statistics. He is the developer of the Auditable Data Analysis and Management System for ELISA.

Yin Bun Cheung is a professor in the Center for Quantitative Medicine at Duke–NUS Medical School in Singapore, and he is an adjunct professor in the Department for International Health at the University of Tampere in Finland. His research areas include statistical methods for analysis of vaccine efficacy and immunogenicity as well as the impact and interplay of infection and undernutrition on child health in developing countries.

The Stata Journal (2016)
16, Number 2, pp. 331–367

Inference in regression discontinuity designs under local randomization

Matias D. Cattaneo
University of Michigan
Ann Arbor, MI
cattaneo@umich.edu

Rocío Titiunik
University of Michigan
Ann Arbor, MI
titiunik@umich.edu

Gonzalo Vazquez-Bare
University of Michigan
Ann Arbor, MI
gvazquez@umich.edu

Abstract. We introduce the `rdlocrand` package, which contains four commands to conduct finite-sample inference in regression discontinuity (RD) designs under a local randomization assumption, following the framework and methods proposed in Cattaneo, Frandsen, and Titiunik (2015, *Journal of Causal Inference* 3: 1–24) and Cattaneo, Titiunik, and Vazquez-Bare (2016, Working Paper, University of Michigan, http://www-personal.umich.edu/~titiunik/papers/CattaneoTitiunikVazquezBare2015_wp.pdf). Assuming a known assignment mechanism for units close to the RD cutoff, these functions implement a variety of procedures based on randomization inference techniques. First, the `rdrandinf` command uses randomization methods to conduct point estimation, hypothesis testing, and confidence interval estimation under different assumptions. Second, the `rdwinselect` command uses finite-sample methods to select a window near the cutoff where the assumption of randomized treatment assignment is most plausible. Third, the `rdsensitivity` command uses randomization techniques to conduct a sequence of hypothesis tests for different windows around the RD cutoff, which can be used to assess the sensitivity of the methods and to construct confidence intervals by inversion. Finally, the `rdrbounds` command implements Rosenbaum (2002, *Observational Studies* [Springer]) sensitivity bounds for the context of RD designs under local randomization. Companion R functions with the same syntax and capabilities are also provided.

Keywords: st0435, rdrandinf, rdwinselect, rdsensitivity, rdrbounds, regression discontinuity designs, quasi-experimental techniques, causal inference, randomization inference, finite-sample methods, Fisher's exact p-values, Neyman's repeated sampling approach

1 Introduction

Conventional inference methods for regression discontinuity (RD) designs use nonparametric local polynomial techniques, rely on large-sample approximations, and provide estimators and inference procedures for the (super) population under repeated sampling assumptions. For example, see Hahn, Todd, and van der Klaauw (2001) and Calonico, Cattaneo, and Titiunik (2014b) for treatment-effect estimation and inference and McCrary (2008) and Cattaneo, Jansson, and Ma (2016b) for manipulation testing. Reviews and further references on flexible parametric and nonparametric approaches to analyzing RD designs are given by Imbens and Lemieux (2008), Lee and Lemieux (2010), Calonico, Cattaneo, and Titiunik (2015a), Keele and Titiunik (2015), and Skovron and

Titiunik (2015). These approaches rely on smoothness assumptions leading to nonparametric identification of, and valid inference procedures for, RD treatment effects at the cutoff in the population.

An alternative approach to analyzing RD designs relies on the idea of local randomization, which postulates that treatment assignment may be regarded as (or at least approximated by) a known randomization mechanism near the RD cutoff (Lee 2008). In other words, in this approach, the units closest to the cutoff are viewed as being part of a local randomized experiment. Building on this intuitive idea, Cattaneo, Frandsen, and Titiunik (2015) and Cattaneo, Titiunik, and Vazquez-Bare (2016) develop a framework and methodological tools to analyze RD designs under a local randomization assumption. The approach is based on methods from the classical literature on the analysis of experiments (Rosenbaum 2002, 2010; Imbens and Rubin 2015), where the (possibly transformed) potential outcomes are regarded as fixed, and inference is based on the randomization distribution of the treatment assignment. This alternative inference approach is useful in many cases, including when the dataset of interest is small, the running variable is not continuous, matching techniques are used, or multiple RD cutoffs are analyzed. For a geographic RD example applying some of these ideas, see Keele, Titiunik, and Zubizarreta (2015).

In this article, we introduce the `rdlocrand` package to conduct finite-sample inference in RD designs using randomization inference and related techniques under a local randomization assumption. This package contains the following four commands:

1. `rdrandinf`. This command implements several methods from the literature on the analysis of experiments, including Fisher's exact p-values and tests, Neyman's repeated sampling inference, and related methods. These methods rely on finite-sample exact techniques (that is, randomization inference methods) and on particular large-sample approximations. The command offers point estimation, hypothesis testing, and confidence interval procedures under different assumptions, allowing in particular for regression-based outcome transformations.

2. `rdwinselect`. This command uses finite-sample (but also large-sample) methods to select a window near the cutoff where the local randomization assumption is most plausible, under some intuitive conditions. This command is called by the `rdrandinf` command to select a data-driven window around the cutoff where inference is conducted, but `rdwinselect` can also be used as a stand-alone command for analysis and falsification of RD designs under a local randomization assumption.

3. `rdsensitivity`. This command uses randomization inference methods to conduct a sequence of hypothesis tests for different windows around the RD cutoff, which can be used to assess the sensitivity of the methods and to construct confidence intervals by inversion.

4. `rdrbounds`. This command implements randomization-based sensitivity analysis (Rosenbaum 2002) in the context of RD designs under local randomization.

Altogether, the `rdlocrand` package offers a complete and systematic analysis of RD designs under a local randomization assumption and some related conditions. These finite-sample methods can be used for empirical analysis of RD designs under a set of assumptions that are different from the conventional assumptions in the nonparametric literature (for example, randomization-based methods allow for discrete running variables). In addition, the procedures discussed here can be used as a robustness check on the conventional inference methods that use large-sample approximations.

In the remainder of this article, we provide methodological, practical, and empirical introductions to these commands. First, we briefly review the main theoretical concepts underlying the methods implemented (section 2). Second, we provide syntax and a brief explanation of the functionalities of each of the four commands (sections 3, 4, 5, and 6). Finally, we present a detailed empirical analysis replicating the empirical results originally reported in Cattaneo, Frandsen, and Titiunik (2015), showing some of the main options implemented by these commands (section 7). We briefly conclude in section 8.

For related Stata (and R) commands (`rdrobust`, `rdbwselect`, and `rdplot`) providing graphical presentation, estimation, and inference in RD designs using local polynomial techniques, see Calonico, Cattaneo, and Titiunik (2014a, 2015b). In addition, for Stata (and R) commands (`rddensity` and `rdbwdensity`) implementing manipulation testing based on discontinuity in density using local polynomial techniques, see Cattaneo, Jansson, and Ma (2016a).

2 Overview of methods

This section briefly describes the statistical framework and methods for RD designs under local randomization. The framework and methods considered here were introduced in Cattaneo, Frandsen, and Titiunik (2015) and Cattaneo, Titiunik, and Vazquez-Bare (2016)—we refer the reader to these references for further details. For a review on randomization inference and related experimental and quasi-experimental methods, see Rosenbaum (2002, 2010) and Imbens and Rubin (2015).

2.1 Statistical framework

We start by introducing some notation. The sample consists of n units indexed by $i = 1, \ldots, n$. The running variable, score, or index is denoted by R_i, and the treatment indicator is D_i. In a sharp RD design, $D_i = \mathbb{1}(R_i \geq \overline{r})$, where $\overline{r}$ is a fixed and known cutoff and $\mathbb{1}(\cdot)$ is the indicator function. Let $\mathbf{D}$ be the $n \times 1$ vector collecting the observed treatment assignment for the n observations, and analogously for $\mathbf{R}$; that is, $\mathbf{D} = (D_1, D_2, \ldots, D_n)'$ and $\mathbf{R} = (R_1, R_2, \ldots, R_n)'$. The potential outcome of unit i under a vector of scores $\mathbf{r}$ and a vector of treatment assignments $\mathbf{d}$ is $y_i(\mathbf{d}, \mathbf{r})$ taking values in a set $\mathcal{Y}$. For example, $\mathcal{Y} = \{0, 1\}$ if the potential outcomes are binary, $\mathcal{Y} = [0, \infty)$ if they are nonnegative, and so on. The support of $\mathbf{D}$ is denoted by $\mathsf{supp}(\mathbf{D}) := \Omega$ and, similarly, $\mathsf{supp}(\mathbf{R}) := \mathcal{R}$. We collect these potential outcomes in a

vector $\mathbf{y}(\mathbf{d},\mathbf{r}) = \{y_1(\mathbf{d},\mathbf{r}), y_2(\mathbf{d},\mathbf{r}), \ldots, y_n(\mathbf{d},\mathbf{r})\}'$. Finally, the vector of observed outcomes is $\mathbf{Y} = \mathbf{y}(\mathbf{D},\mathbf{R})$, which is a random variable by virtue of $\mathbf{R}$ and, as a consequence, $\mathbf{D}$ being random. Throughout, we use lowercase to denote fixed, nonrandom variables and uppercase to denote random variables.

The local randomization framework for RD designs is characterized by two features: i) a known randomization (or treatment assignment) mechanism in some region or window around the cutoff and ii) an exclusion restriction on the (transformed) potential outcomes. More precisely, the first condition is related to the existence of a window $W_0 = [\overline{r} - w, \overline{r} + w]$, $w > 0$, around the cutoff in which, as in a randomized controlled experiment, treatment assignment of units with $R_i \in W_0$ follows a known distribution: the probability law $\mathbb{P}(\mathbf{D}_{W_0} = \mathbf{d})$ is known, where $\mathbf{D}_{W_0}$ is the subvector of $\mathbf{D}$ corresponding to the observations with $R_i \in W_0$ and $\mathbf{d} \in \{0,1\}^{n_{W_0}}$ with $n_{W_0} = \sum_{i=1}^{n} \mathbb{1}(R_i \in W_0)$ denoting the number of units in W_0.

For implementation purposes, the `rdlocrand` command considers two types of randomization mechanisms. In the first one, called fixed-margins randomization or complete randomization, each treatment assignment chooses a given number of treated units among the n_{W_0} units in W_0. Thus, in applications, the probability distribution of $\mathbf{D}_{W_0}$ is given by

$$\mathbb{P}(\mathbf{D}_{W_0} = \mathbf{d}) = \binom{n_{W_0}}{m_{W_0}}^{-1} \qquad \forall\, \mathbf{d} \in \Omega_0$$

where $m_{W_0} = \sum_{i=1}^{n} \mathbb{1}(R_i \in W_0) D_i$ is the number of treated units and Ω_0 is the set of n_{W_0}-dimensional vectors with exactly m_{W_0} ones. In the second assignment mechanism, units are assigned to treatment following independent Bernoulli trials with some probability $q \in (0,1)$, so that

$$\mathbb{P}(\mathbf{D}_{W_0} = \mathbf{d}) = \prod_{i=1}^{n_{W_0}} q^{d_i}(1-q)^{1-d_i} \qquad \forall\, \mathbf{d} \in \Omega_0$$

where now $\Omega_0 = \{\mathbf{d} \in \{0,1\}^{n_{W_0}}\}$. The parameter q is often unknown but can be easily estimated, for example, as the proportion of treated units in the window; that is, $\widehat{q} = m_{W_0}/n_{W_0}$.

The second feature of the local randomization framework is a model or, more precisely, an exclusion restriction for the (transformed) potential outcomes. The idea is that the potential outcomes can be transformed to eliminate the direct dependence between potential outcomes and the running variable, that is, that there exists some transformation $\phi(\cdot)$ of the potential outcomes such that

$$\phi(\mathbf{y}(\mathbf{d},\mathbf{r}),\mathbf{d},\mathbf{r}) = \widetilde{\mathbf{y}}(\mathbf{d}_{W_0}) \quad \forall\, \mathbf{r} \in \mathcal{R}$$

In practice, this assumption is likely to be more plausible for units within the window W_0. In the end, the goal is to transform the (potential) outcomes so that $\widetilde{y}_i(\mathbf{d},\mathbf{r}) = \widetilde{y}_i(\mathbf{d}_{W_0})$ for all units with score in W_0, so their potential outcomes are not a function of the score except through the treatment indicator. In some applications, the transformation is not needed [that is, $\phi(\cdot)$ is the identity function and $y_i(\mathbf{d},\mathbf{r}) = y_i(\mathbf{d})$ directly], but

in other cases, the researcher may want to incorporate some parametric relationship. This restriction, for example, could reflect the standard practice in the RD literature of fitting a lower-order polynomial regression at each side of the cutoff.

For the outcome transformation, `rdlocrand` considers polynomial transformations with different centerings. Specifically, all the commands in the `rdlocrand` package allow for a transformation of (potential) outcomes based on a polynomial model of order p of the form

$$y_i(\mathbf{d},\mathbf{r}) = \begin{cases} \alpha_i(\mathbf{d}_{W_0}) + \beta_{01}(r_i - r_0) + \beta_{02}(r_i - r_0)^2 + \cdots + \beta_{0p}(r_i - r_0)^p & \text{if } d_i = 0 \\ \alpha_i(\mathbf{d}_{W_0}) + \beta_{11}(r_i - r_1) + \beta_{12}(r_i - r_1)^2 + \cdots + \beta_{1p}(r_i - r_1)^p & \text{if } d_i = 1 \end{cases}$$

for all i such that $R_i \in W_0$, where r_1 and r_0 are two evaluation points set in advance; natural choices are i) the cutoff point ($r_1 = r_0 = \overline{r}$) and ii) the center of the control ($r_0 = \overline{r} - w/2$) and treatment ($r_1 = \overline{r} + w/2$) regions within W_0. Of course, in the absence of a polynomial transformation (that is, $\beta_{0j} = 0 = \beta_{1j}$ for $j = 1, 2, \ldots, p$), we obtain the natural exclusion restriction

$$y_i(\mathbf{d},\mathbf{r}) = \alpha_i(\mathbf{d}_{W_0}) =: \widetilde{y}_i(\mathbf{d}_{W_0})$$

for all units with their score inside W_0. The polynomial transformation assumption allows for the potential outcomes to have some direct dependence on the score, although this assumption is of course strong and very specific. In this case, the transformed potential outcomes are

$$\widetilde{y}_i(\mathbf{d}_{W_0}) := \begin{cases} y_i(\mathbf{d},\mathbf{r}) - \beta_{01}(r_i - r_0) - \cdots - \beta_{0p}(r_i - r_0)^p = \alpha_i(\mathbf{d}_{W_0}) & \text{if } d_i = 0 \\ y_i(\mathbf{d},\mathbf{r}) - \beta_{11}(r_i - r_1) - \cdots - \beta_{1p}(r_i - r_1)^p = \alpha_i(\mathbf{d}_{W_0}) & \text{if } d_i = 1 \end{cases}$$

for all i such that $R_i \in W_0$. Thus the transformed potential outcomes isolate the portion of the potential outcome that is related to the treatment but unrelated to the particular value taken by the running variable. The coefficients β_{dj} ($d = 0, 1$ and $j = 1, 2, \ldots, p$) are analytically computed by least squares.

In the model above, the transformed potential outcomes depend only on $\mathbf{d}_{W_0}$, and the probability law of $\mathbf{D}_{W_0}$ is known for all units with score in W_0. Therefore, among other possibilities, randomization inference can be used for hypothesis testing and confidence interval construction. As discussed below, we will need to further restrict the model (that is, impose the so-called stable unit treatment value assumption, or SUTVA). Before discussing how to perform inference in this setting, we describe our window-selection method.

2.2 Window selection

Cattaneo, Frandsen, and Titiunik (2015) propose a data-driven method to find the window W_0 where the local randomization assumption is assumed to hold. The idea is that because treatment assignment is "as if random" inside the window, the distribution of preintervention covariates and postintervention unaffected-by-treatment outcomes

should be the same between treated and control units. This observation is closely related to the ideas underlying balance tests in the experimental and nonexperimental literature. For the procedure to be useful, the distribution of these covariates for control and treatment units should be unaffected by the treatment within W_0 but should be affected by the treatment outside the window.

To describe the approach, we will let $\mathbf{x}$ be the $n \times k$ matrix collecting the k covariates and, for an arbitrary window W, $\mathbf{x}_W$ be the subvector corresponding to units with scores inside the window W. Then, the window-selection algorithm is the following:

1. Set an initial "small" window, $\widehat{W}_1$.
2. For each of the k covariates, conduct a test of the null hypothesis of no effect of the treatment on the covariate using some test-statistic $\mathcal{T}(\mathbf{x}_{\widehat{W}_1}, \mathbf{R}_{\widehat{W}_1})$. Take the minimum p-value from the k tests.
3. If the minimum p-value obtained in step 2, p_1, is less than some prespecified level—for example, 0.15—the initial window was too large;[1] decrease the initial window and start over. If the initial window cannot be decreased (for example, because a smaller window would contain too few data points), conclude that W_0 cannot be found. Otherwise, if $p_1 \geq 0.15$, choose a larger window $\widehat{W}_2 \supset \widehat{W}_1$, and go back to step 2 to calculate p_2. Repeat the process until the minimum p-value is less than 0.15. The selected window is the largest window such that the minimum p-value is larger than or equal to 0.15 in that window and in all windows contained in it.

The resulting window, $\widehat{W}$, is the estimate of W_0. The idea of this algorithm is to choose the largest window (or one of the largest windows) in which all the covariates are balanced. This window-selection procedure is implemented by `rdwinselect` using both randomization inference and large-sample methods discussed next. We emphasize that although this procedure performs multiple hypothesis tests, we do not use multiple testing adjustments because the goal is to be conservative and reject the null often so that the chosen window is small.

2.3 Randomization inference

The framework in section 2.1 provides all the necessary information to test the sharp null hypothesis of no treatment effect using randomization inference methods. More precisely, consider the following hypothesis:

$$\mathrm{H}_0: \quad \alpha_i(\mathbf{d}_{W_0}) = \alpha_i(\mathbf{d}'_{W_0}) \qquad \forall i : R_i \in W_0 \quad \text{and} \quad \forall \mathbf{d}_{W_0}, \mathbf{d}'_{W_0}$$

This hypothesis is usually known as the sharp null hypothesis of no treatment effect or Fisher's null hypothesis. In the randomization inference literature, a hypothesis is said

1. In some cases, this could also happen if the initial window was too small because balance tests may be unreliable in these cases. We recommend starting with initial windows of at least 10 observations at each side of the cutoff.

to be "sharp" if it allows the researcher to impute all missing potential outcomes. Under this hypothesis, we have $\widetilde{y}_i(\mathbf{d}_{W_0}) = \alpha_i(\mathbf{d}_{W_0})$ for all units inside the window W_0; thus all the potential outcomes can be observed. Collecting all observed transformed outcomes for units in W_0 in the vector $\widetilde{\mathbf{Y}}_{W_0}(\mathbf{D}_{W_0})$ and all $\alpha_i(\mathbf{d}_{W_0})$ in W_0 in the vector $\boldsymbol{\alpha}^0_{W_0}$, under H_0, we have that $\widetilde{\mathbf{Y}}_{W_0} = \boldsymbol{\alpha}^0_{W_0}$. But $\boldsymbol{\alpha}^0_{W_0}$ is constant under all realizations of $\mathbf{D}$. Therefore, any test-statistic $\mathcal{T}(\mathbf{D}_{W_0}, \widetilde{\mathbf{Y}}_{W_0})$ satisfies $\mathcal{T}(\mathbf{D}_{W_0}, \widetilde{\mathbf{Y}}_{W_0}) = \mathcal{T}(\mathbf{D}_{W_0}, \boldsymbol{\alpha}^0_{W_0})$ under the sharp null hypothesis, implying that its null distribution is known—because the only randomness in $\mathcal{T}(\mathbf{D}_{W_0}, \boldsymbol{\alpha}^0_{W_0})$ originates in the treatment assignment mechanism, $\mathbf{D}$, whose distribution is assumed to be known.

An exact p-value for an observed value of the statistic $\mathcal{T}_{\text{obs}}$ can be easily computed as

$$\mathbb{P}\left\{\mathcal{T}\left(\mathbf{D}_{W_0}, \widetilde{\mathbf{Y}}_{W_0}\right) \geq \mathcal{T}_{\text{obs}}\right\} = \sum_{\mathbf{d}\in\Omega_0} \mathbb{1}\left\{\mathcal{T}\left(\mathbf{D}_{W_0}, \widetilde{\mathbf{Y}}_{W_0}\right) \geq \mathcal{T}_{\text{obs}}\right\} \mathbb{P}\left(\mathbf{D}_{W_0} = \mathbf{d}\right)$$

where $\mathbb{P}(\mathbf{D}_{W_0} = \mathbf{d})$ follows a known distribution, as mentioned above. In practice, the cardinality of Ω_0 will be very large even for relatively small sample sizes; thus the p-value is approximated by drawing treatment assignment vectors from Ω_0 at random. The larger the number of repetitions, the more precise the approximation will be.

A common simplifying assumption in this context is the SUTVA; that is, unit i's (transformed) potential outcome depends only on unit i's score and treatment assignment; that is, $\widetilde{y}_i(\mathbf{d}, \mathbf{r}) = \widetilde{y}_i(d_i, r_i)$ for all i such that $R_i \in W_0$. Although the main ideas and results discussed here do not require this condition, in practice, the outcome transformation model may be hard or impossible to implement without restrictions on the degree of interference between units. Thus our implementations effectively use SUTVA whenever a transformation is used. Section 2.5 discusses ways to relax this assumption when interference between units is suspected. SUTVA implies that each unit has exactly two transformed potential outcomes, $\widetilde{y}_i(1)$ and $\widetilde{y}_i(0)$.

After imposing SUTVA, we can apply the same reasoning above to test any sharp null hypothesis. For example, if one is willing to assume that the treatment effect is an additive constant τ, one can test whether $\tau = \tau_0$:

$$\mathrm{H}_0\colon \quad \alpha_i(1) = \alpha_i(0) + \tau_0 \quad \forall i : R_i \in W_0$$

The idea is that in this case, $\widetilde{y}_i = \widetilde{y}_i(0) + \tau_0 D_i$; hence, $\widetilde{y}_i - \tau_0 D_i$ does not vary with D_i. The null hypothesis is then tested using $\mathcal{T}(\mathbf{D}_{W_0}, \widetilde{\mathbf{Y}}_{W_0} - \tau\mathbf{D}_{W_0})$.

For the rest of this article, we impose SUTVA to simplify the discussion and presentation, except where we note explicitly otherwise.

2.4 Statistics and confidence intervals

As we will illustrate in section 7, the commands presented implement up to four different statistics: difference in means (DM), Kolmogorov–Smirnov (KS), Wilcoxon rank

sum (RS), and Hotelling's T^2 statistics. The last statistic is available only for window selection (`rdwinselect`) but not for RD inference (`rdrandinf`) or sensitivity analysis (`rdsensitivity`, `rdrbounds`). The formulas below are written in terms of the outcome variable (adjusted by the null value τ_0 when required); when implementing `rdwinselect`, one applies the same formulas using each covariate x_{ij} $(j = 1, \ldots, k)$ as an outcome instead of $\widetilde{y}_i - \tau_0 D_i$.

In addition, under appropriate assumptions, different types of confidence intervals can be constructed using randomization inference methods under the local randomization assumption. In this section, we review two methods implemented in the commands `rdsensitivity` (under a treatment-effect model) and `rdrandinf`, allowing for the presence of interference.

DM

The DM statistic is calculated as

$$\mathcal{T}_{\text{DM}} = \frac{1}{m_{W_0}} \sum_{i:R_i \in W_0} \left(\widetilde{Y}_i - \tau_0 D_i\right) D_i - \frac{1}{n_{W_0} - m_{W_0}} \sum_{i:R_i \in W_0} \left(\widetilde{Y}_i - \tau_0 D_i\right) (1 - D_i)$$

with $m_{W_0} = \sum_{i:R_i \in W_0} D_i$ and $n_{W_0} - m_{W_0} = \sum_{i:R_i \in W_0} (1 - D_i)$ as defined above.

Unlike the other estimators that will be discussed, the DM has the appealing feature that it provides a point estimate of the average treatment effect (ATE) without imposing any restrictions on how the treatment effect varies across units. Moreover, one can easily show that this statistic is unbiased for the ATE when the treatment assignment follows a fixed-margin randomization scheme or Bernoulli trials. Hence, $\mathcal{T}_{\text{DM}}$ plays the double role of test statistic and point estimate of the ATE. One should keep in mind, however, that the null hypothesis being tested is that the treatment effect is zero for all units, which is much stronger than the hypothesis that the ATE is zero.

In terms of implementation, `rdlocrand` calculates $\mathcal{T}_{\text{DM}}$ as the estimate of a fully interacted pth-order polynomial regression using observations inside the window, which is simply a regression on the indicator of being above the cutoff D_i and a polynomial on R_i allowed to differ for units above and below the cutoff. Note that this is equivalent to running two separate regressions of the outcome on a polynomial of R_i, one above and the other below the cutoff, and taking the difference in the intercepts. In this case, $p = 0$ would correspond to a simple DM (that is, no model transformation).

KS

The KS statistic is obtained as

$$\mathcal{T}_{\text{KS}} = \sup_{y \in \mathcal{Y}} \left| \widehat{F}(y|D = 1) - \widehat{F}(y|D = 0) \right|$$

where $\widehat{F}(y|D = 1)$ and $\widehat{F}(y|D = 0)$ are estimates of the distribution function of the transformed outcome for the treated and control units, respectively. When $p = 0$, that

is, $Y_i = \widetilde{Y}_i$ (that is, there is no transformation), this statistic is calculated using the `ksmirnov` Stata command directly. Otherwise, when $p > 0$, the statistic is obtained in two steps. First, $Y_i - \tau_0 D_i$ is regressed on a pth-degree polynomial separately for treated and control units; for each regression, the estimate of the intercept is added to the residuals. The resulting values are the transformed outcomes for treated and controls, $\widetilde{Y}_i$. Second, the KS statistic is constructed using $\widetilde{Y}_i - \tau_0 D_i$.

Wilcoxon RS

This statistic is the studentized version of the Wilcoxon RS statistic, computed as

$$\mathcal{T}_{\text{RS}} = \frac{T - \mathbb{E}(T)}{\sqrt{\mathbb{V}(T)}}$$

where T is the sum of ranks for the control group (Wilcoxon's statistic) and $\mathbb{E}(T)$ and $\mathbb{V}(T)$ are the expectation and variance of the Wilcoxon RS statistic. This is calculated using the `ranksum` command. As for the KS statistic, when $p > 0$, the statistic is calculated using the transformed outcomes, $\widetilde{Y}_i$, as mentioned above.

Hotelling's $\mathbf{T^2}$

The window-selection procedure described above involves performing k hypothesis tests, one for each covariate, and choosing the minimum between the k resulting p-values. The `rdwinselect` command also allows the user to perform a joint test using Hotelling's T^2 statistic, which is based on the DM between the whole vector of covariates, instead of each mean individually. The statistic is obtained as

$$\mathcal{T}_{\text{H}} = (\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_0)\mathbf{S}^{-1}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_0)'$$

where $\overline{\mathbf{x}}_0$ and $\overline{\mathbf{x}}_1$ correspond to the sample means of covariates for control and treatment units, respectively, and $\mathbf{S}$ is the estimated covariance matrix—where all estimates are computed using the units inside the given window. This statistic is obtained using the `hotelling` command. Note that in this case, only one p-value is obtained for each window.

Confidence intervals under parametric model

Once a parametric model for treatment effects is specified, a finite-sample valid confidence interval can be constructed using randomization inference methods. The idea is to invert a sequence of hypothesis tests and to report all the values in the parameter space for which the (sharp) null hypothesis of no treatment effect is not rejected at a given level of significance. This method, which is standard in the literature, can be easily obtained using the `rdrandinf` command or the sensitivity analysis command `rdsensitivity` with the option `ci(·)`, as we discuss and illustrate below.

Confidence intervals under arbitrary interference

The presence of interference is immaterial when testing the sharp null of no effect, and the methods described in section 2.3 can still be applied without modification. Point estimation, however, becomes infeasible because when arbitrary interference is allowed, the definition of a treatment effect is not straightforward. In this case, Rosenbaum (2007) suggests a way to construct confidence intervals for a particular measure of the benefits of the treatment. The `rdrandinf` command implements this procedure via the `interfci()` option, which we briefly summarize here.

Define a placebo trial (or uniformity trial) as a trial in which units are randomly divided into two groups, but then treatment is withheld from all units. Because, by construction, nobody receives treatment, the transformed outcomes in the uniformity trial, $\widetilde{\mathbf{Y}}_{\mathtt{U},W_0}$, satisfy $\widetilde{\mathbf{Y}}_{\mathtt{U},W_0} = \boldsymbol{\alpha}_{W_0}$. If $\mathcal{T}_{\mathtt{U}} := \mathcal{T}(\mathbf{D}_{W_0}, \widetilde{\mathbf{Y}}_{\mathtt{U},W_0})$ is the value of the statistic in this placebo trial, then even though the value of $\mathcal{T}_{\mathtt{U}}$ is unobservable, its distribution is known because in a placebo trial, the sharp null hypothesis holds. Consider the magnitude $\Delta = \mathcal{T} - \mathcal{T}_{\mathtt{U}}$, which measures the difference between the statistic under the experiment under consideration and under the placebo experiment. If the treatment has no effect, $\Delta = 0$. For instance, if $\mathcal{T}$ is the DM, then

$$\Delta = \frac{\sum\limits_{i\in\mathcal{I}_0}\left(\widetilde{Y}_i - \widetilde{Y}_{\mathtt{U}i}\right) D_i}{\sum\limits_{i\in\mathcal{I}_0} D_i} - \frac{\sum\limits_{i\in\mathcal{I}_0}\left(\widetilde{Y}_i - \widetilde{Y}_{\mathtt{U}i}\right) (1-D_i)}{\sum\limits_{i\in\mathcal{I}_0} (1-D_i)}$$

where $\mathcal{I}_0 = \{i : R_i \in W_0\}$. For this choice of $\mathcal{T}$, Δ is the difference between how much the treated and control groups deviate (on average) from the zero-effect case. In other words, Δ measures how much bigger the treatment effect is for the treated group compared with the control group.

A confidence interval for Δ is constructed in the following way. Let κ_1 and κ_2 be some constants satisfying $\kappa_2 < \kappa_1$. Then, it is easy to see that

$$\mathbb{P}\left(\mathcal{T} - \kappa_1 \leq \Delta \leq \mathcal{T} - \kappa_2\right) = \mathbb{P}(\kappa_2 \leq \mathcal{T}_{\mathtt{u}} \leq \kappa_1)$$

Hence, if κ_1 and κ_2 are chosen to be, say, the $\alpha/2$ and $1-\alpha/2$ quantiles of $\mathcal{T}_{\mathtt{u}}$ for some level α, it follows that $\Delta \in [\mathcal{T} - \kappa_1, \mathcal{T} - \kappa_2]$ with probability $1-\alpha$. In practice, the values for κ_1 and κ_2 can be recovered from the randomization distribution of $\mathcal{T}_{\mathtt{u}}$, and $\mathcal{T}$ is replaced by its observed value. (Of course, the level of the confidence interval may not be exactly $1-\alpha$ because the distribution of $\mathcal{T}_{\mathtt{u}}$ is discrete, but the quantiles can be chosen so that the level is at least $1-\alpha$.)

2.5 Sensitivity analysis

We also provide two alternative approaches to sensitivity analysis using randomization inference methods under the local randomization assumption. These methods are implemented in the `rdsensitivity` command for sensitivity to model specification and window length and in the `rdrbounds` command for sensitivity to randomization mechanism and selection bias.

Sensitivity to parametric model and window length

Although the randomization inference framework used in this article suggests an intuitively and appealing window selector, in practice, the user may want to explore how robust the results are to different window lengths and parametric models. The `rdsensitivity` command calculates the p-values for the desired hypothesis test over a range of window lengths, given the parametric model chosen by the researcher. For each window, this command considers a range of treatment effects under the null hypothesis, a feature that offers a way of analyzing the sensitivity of the "point estimates" by carrying out a family of hypothesis tests within and across windows. These methods may also be used to analyze the sensitivity to the specific parametric model by simply using this command with different models and then comparing the results.

In addition, for a given window, the above procedure can be used directly to obtain confidence intervals via inversion of the corresponding hypothesis test, as we mentioned previously. We illustrate empirically all of these features below.

Misspecification of the randomization mechanism: Sensitivity to unobserved confounders

This approach is quite standard in the randomization inference literature (Rosenbaum 2002); thus we include it here as an additional robustness check. The only substantive modification relative to the conventional approach is the explicit use of the different windows around the cutoff, which provides an additional dimension for sensitivity analysis.

Assume that, inside the window, the randomization mechanism follows a Bernoulli experiment where the individual probability of treatment is

$$\mathbb{P}(D_i = 1) = q_i = \frac{\exp(\gamma U_i)}{1 + \exp(\gamma)} \tag{1}$$

and U_i is an unobserved binary variable. This gives

$$q_i = q_L = \frac{1}{1+\exp(\gamma)} \equiv \frac{1}{1+\Gamma} \qquad \text{and} \qquad q_i = q_H = \frac{\exp(\gamma)}{1+\exp(\gamma)} \equiv \frac{\Gamma}{1+\Gamma}$$

for units with $U_i = 0$ and $U_i = 1$, respectively, where $\Gamma \equiv \exp(\gamma)$. It follows that the odds ratio for units i and j satisfies

$$\frac{1}{\Gamma} \leq \frac{q_i/(1-q_i)}{q_j/(1-q_j)} \leq \Gamma$$

or, equivalently,

$$\left| \log\left(\frac{q_i}{1-q_i}\right) - \log\left(\frac{q_j}{1-q_j}\right) \right| \leq \gamma$$

In a completely randomized experiment, $\gamma = 0$, so all units have the same probability of treatment. Thus γ measures the degree of departure from a randomized experiment

(under the assumptions imposed). The idea of Rosenbaum's sensitivity analysis is to use (1) to see how the randomization p-value varies over a range of values for γ (or Γ).

To make this method operational, one must assign the unobservable U_i to units in the sample in a way that ensures that the distribution of the statistic of interest can be bounded by two distributions that can be obtained from the data. To this end, following Rosenbaum (2002), the `rdrbounds` command sorts the outcome in decreasing order and performs a search over the sets $\mathcal{U}^+$ and $\mathcal{U}^-$, where $\mathcal{U}^+$ contains all vectors with ones in the first ℓ positions and zeros in the remaining positions, and $\mathcal{U}^-$ contains vectors with zeros in the first ℓ positions and ones in the remaining ones.

2.6 Extension to fuzzy RD designs

In the so-called fuzzy RD design, treatment is no longer deterministically assigned based on the running variable; hence, compliance is imperfect among units below and above the cutoff. This implies that treatment assignment and treatment status can in principle be different. To adapt the previous notation to this context, let $\mathbf{d}(\mathbf{r})$ be the vector of potential treatment assignments, so that the observed treatment status is given by $\mathbf{D} = \mathbf{d}(\mathbf{R})$. Treatment assignment for units i is now given by $Z_i = \mathbb{1}(R_i \geq \overline{r})$, and the vector of treatment assignments is $\mathbf{Z}_{W_0}$ restricted to the units within the window W_0 (using the same notation introduced previously). The usual way to handle fuzzy designs is to use the vector $\mathbf{Z}$ as an instrument for $\mathbf{D}$. We assume that the distribution of $\mathbf{Z}$ is known. Imbens and Rosenbaum (2005) discuss randomization inference methods in the context of IV models.

Under the null hypothesis that the treatment effect is $\tau = \tau_0$, the adjusted responses $\widetilde{y}_i - \tau_0 D_i$ are fixed and unrelated to the instrument. Therefore, the distribution of any statistic $\mathcal{T}(\mathbf{Z}_{W_0}, \widetilde{\mathbf{Y}}_{W_0} - \tau_0 \mathbf{D}_{W_0})$ is known and can be used to perform inference in the same way as it was for sharp designs. The statistic used in this case, which we denote $\mathcal{T}_{\text{AR}}$,[2] is the difference in the adjusted transformed responses between units assigned to treatment and control

$$\mathcal{T}_{\text{AR}} = \frac{\sum\limits_{i \in \mathcal{I}_0} \left(\widetilde{Y}_i - \tau_0 D_i\right) Z_i}{\sum\limits_{i \in \mathcal{I}_0} Z_i} - \frac{\sum\limits_{i \in \mathcal{I}_0} \left(\widetilde{Y}_i - \tau_0 D_i\right) (1 - Z_i)}{\sum\limits_{i \in \mathcal{I}_0} (1 - Z_i)}$$

where as before $\mathcal{I}_0 = \{i : R_i \in W_0\}$. When the outcomes are transformed using a linear model, this statistic is equivalent to the difference in the intercepts between two reduced form regressions of Y_i on r_i, one at each side of the cutoff, then adjusted by τ_0.

For comparison, we provide an implementation of a Wald-type two-stage least-squares statistic (TSLS). Inference for the TSLS statistic relies on asymptotic approximations and is sensitive to weak instruments.

2. AR stands for Anderson–Rubin.

3 The rdrandinf command

This section describes the syntax of `rdrandinf`, which calculates randomization p-values on a specified window under different randomization mechanism and potential-outcomes models.

3.1 Description

`rdrandinf` implements randomization inference and related methods for RD designs, using observations in a specified or data-driven selected window around the cutoff where local randomization is assumed to hold.

3.2 Syntax

`rdrandinf` *outvar runvar* [*if*] [*in*] [`, cutoff(#) wl(#) wr(#)`
`statistic(`*stat*`) p(#) evall(#) evalr(#) kernel(`*kerneltype*`) nulltau(#)`
`ci(`*level* [*tlist*]`) interfci(#) fuzzy(`*fuzzy_var* [*fuzzy_stat*]`) d(#) dscale(#)`
`bernoulli(`*varname*`) reps(#) seed(#) covariates(`*varlist*`) obsmin(#)`
`obsstep(#) wmin(#) wstep(#) nwindows(#) rdwstat(`*stat*`) approximate`
`rdwreps(#) level(#) plot graph_options(`*graphopts*`) quietly`]

outvar is the outcome variable. *runvar* is the running variable (also known as the score or forcing variable).

3.3 Options

`cutoff(#)` specifies the RD cutoff for the running variable *runvar*. The default is `cutoff(0)`.

`wl(#)` specifies the left limit of the window. The default is the minimum of the running variable.

`wr(#)` specifies the right limit of the window. The default is the maximum of the running variable.

`statistic(`*stat*`)` specifies the statistic to be used for randomization inference. *stat* may be `ttest` (DM), `ksmirnov` (KS statistic), `ranksum` (Wilcoxon–Mann–Whitney studentized statistic), or `all`, which specifies all three statistics. The default is `statistic(ttest)`.

`p(#)` specifies the order of the polynomial for the outcome transformation model. The default is `p(0)`.

`evall(#)` specifies the point at the left of the cutoff at which the transformed outcome is evaluated. The default is the cutoff value.

`evalr(#)` specifies the point at the right of the cutoff at which the transformed outcome is evaluated. The default is the cutoff value.

`kernel(`*kerneltype*`)` specifies the type of kernel to use as the weighting scheme. *kerneltype* may be `uniform` (uniform kernel), `triangular` (triangular kernel), or `epan` (Epanechnikov kernel). The default is `kernel(uniform)`.

`nulltau(#)` sets the value of the treatment effect under the null hypothesis. The default is `nulltau(0)`.

`ci(`*level* [*tlist*]`)` calculates a confidence interval for the treatment effect by test inversion, where *level* specifies the level of the confidence interval and *tlist* indicates the grid of treatment effects to be evaluated. This option uses `rdsensitivity` to calculate the confidence interval; type `help rdsensitivity` for details.

`interfci(#)` sets the level for Rosenbaum's confidence interval under arbitrary interference between units (Rosenbaum 2007).

`fuzzy(`*fuzzy_var* [*fuzzy_stat*]`)` specifies the name of the endogenous treatment variable in the fuzzy design. The options for statistics in fuzzy designs are `ar` for an Anderson–Rubin-type statistic and `tsls` for a TSLS statistic. The default *fuzzy_stat* is `ar`.

`d(#)` specifies the effect size for asymptotic power calculation. The default is 0.5 times the standard deviation of the outcome variable for the control group.

`dscale(#)` specifies the fraction of the standard deviation of the outcome variable for the control group used as an alternative hypothesis for asymptotic power calculation. The default is `dscale(.5)`.

`bernoulli(`*varname*`)` specifies that the randomization mechanism follow Bernoulli trials instead of fixed margins randomization. The values of the probability of treatment for each unit are indicated in *varname*.

`reps(#)` specifies the number of replications. The default is `reps(1000)`.

`seed(#)` sets the seed for the randomization test. With this option, the user can manually set the desired seed or can enter the value -1 to use the system seed. The default is `seed(666)`.

Note: When the window around the cutoff is not specified, `rdrandinf` can select the window automatically using the `rdwinselect` command. The following options are available:

`covariates(`*varlist*`)` specifies the covariates used by the `rdwinselect` command.

`obsmin(#)` specifies the minimum number of observations above and below the cutoff in the smallest window used by the `rdwinselect` command. The default is `obsmin(10)`.

`obsstep(#)` specifies the minimum number of observations to be added on each side of the cutoff for the sequence of nested windows constructed by the `rdwinselect` command. The default is `obsstep(2)`.

`wmin(#)` specifies the smallest window to be used (if `minobs()` is not specified) by the `rdwinselect` command. Specifying both `wmin()` and `obsmin()` returns an error.

`wstep(#)` specifies the increment in window length (if `obsstep()` is not specified) by the `rdwinselect` command. Specifying both `obsstep()` and `wstep()` returns an error.

`nwindows(#)` specifies the number of windows to be used by the `rdwinselect` command. The default is `nwindows(10)`.

`rdwstat(`*stat*`)` specifies the statistic to be used by the `rdwinselect` command (see help file for options). The default is `rdwstat(ttest)`.

`approximate` forces the `rdwinselect` command to conduct the covariate balance tests using a large-sample approximation instead of finite-sample exact randomization inference methods.

`rdwreps(#)` specifies the number of replications to be used by the `rdwinselect` command. The default is `rdwreps(1000)`.

`level(#)` specifies the minimum accepted value of the p-value from the covariate balance tests to be used by the `rdwinselect` command. The default is `level(.15)`.

`plot` draws a scatterplot of the minimum p-value from the covariate balance test against window length implemented by the `rdwinselect` command.

`graph_options(`*graphopts*`)` passes the *graphopts* options to the plot. Options such as titles should be written without double quotes.

`quietly` suppresses output from the `rdwinselect` command.

4 The rdwinselect command

This section describes the syntax of the `rdwinselect` command. This command finds the largest window in which a set of covariates is found to be balanced between treated and control groups.

4.1 Description

`rdwinselect` implements the window-selection procedure based on balance tests for RD designs under local randomization. Specifically, it constructs a sequence of nested windows around the RD cutoff and reports binomial tests for the running variable *runvar* and covariate balance tests for covariates *covariates* (if specified). The recommended window is the largest window around the cutoff such that the minimum p-value of the balance test is larger than a prespecified level for all nested (smaller) windows. By default, the p-values are calculated using randomization inference methods.

4.2 Syntax

`rdwinselect` *runvar* [*covariates*] [*if*] [*in*] [`, cutoff(#) obsmin(#) obsstep(#) wmin(#) wstep(#) nwindows(#) statistic(`*stat*`) approximate p(#) evalat(`*point*`) kernel(`*kerneltype*`) reps(#) seed(#) level(#) plot graph_options(`*graphopts*`)`]

runvar is the running variable (also known as the score or forcing variable). *covariates* is the list of covariates to be used in the balancing tests. This list is optional, but the recommended window is provided only when at least one covariate is specified.

4.3 Options

`cutoff(#)` specifies the RD cutoff for the running variable *runvar*. The default is `cutoff(0)`.

`obsmin(#)` specifies the minimum number of observations above and below the cutoff in the smallest window. The default is `obsmin(10)`.

`obsstep(#)` specifies the minimum number of observations to be added on each side of the cutoff in all but the first window. The default is `obsstep(2)`.

`wmin(#)` specifies the smallest window to be used (if `minobs()` is not specified). Specifying `wmin()` and `obsmin()` returns an error.

`wstep(#)` specifies the increment in window length (if `obsstep()` is not specified). Specifying both `obsstep()` and `wstep()` returns an error.

`nwindows(#)` specifies the number of windows to be used. The default is `nwindows(10)`.

`statistic(`*stat*`)` specifies the statistic to be used. *stat* may be one of the following: `ttest` (DM), `ksmirnov` (KS statistic), `ranksum` (Wilcoxon–Mann–Whitney studentized statistic), or `hotelling` (Hotelling's T-squared statistic). The default is `statistic(ttest)`.

`approximate` performs the covariate balance test using a large-sample approximation instead of randomization inference.

`p(#)` specifies the order of the polynomial for the outcome adjustment model. The default is `p(0)`.

`evalat(`*point*`)` specifies the point at which the adjusted variable is evaluated. *point* may be `cutoff` or `means`. The default is `evalat(cutoff)`.

`kernel(`*kerneltype*`)` specifies the type of kernel to use as the weighting scheme. *kerneltype* may be `uniform` (uniform kernel), `triangular` (triangular kernel), or `epan` (Epanechnikov kernel). The default is `kernel(uniform)`.

`reps(#)` sets the number of replications for the randomization test. The default is `reps(1000)`.

`seed(#)` sets the initial seed for the randomization test. With this option, the user can manually set the desired seed or can enter the value -1 to use the system seed. The default is `seed(666)`.

`level(#)` specifies the minimum accepted value of the p-value from the covariate balance tests to be used. The default is `level(.15)`.

`plot` draws a scatterplot of the minimum p-value from the covariate balance test against window length.

`graph_options(`*graphopts*`)` passes the *graphopts* options to the plot. Options such as titles should be written without double quotes.

5 The rdsensitivity command

This section describes the syntax of the `rdsensitivity` command, which is used to analyze the sensitivity of randomization p-values and confidence intervals to different window lengths.

5.1 Description

`rdsensitivity` performs sensitivity analysis for RD designs under local randomization.

5.2 Syntax

`rdsensitivity` *outvar runvar* [*if*] [*in*] [`, cutoff(#) wlist(`*numlist*`)`
`tlist(`*numlist*`) saving(`*filename*`) nodots nodraw verbose`
`ci(`*window* [*level*]`) statistic(`*stat*`) p(#) evalat(`*point*`) kernel(`*kerneltype*`)`
`fuzzy(`*fuzzy_var*`) reps(#) seed(#)`]

outvar is the outcome variable. *runvar* is the running variable (also known as the score or forcing variable).

5.3 Options

`cutoff(#)` specifies the RD cutoff for the running variable *runvar*. The default is `cutoff(0)`.

`wlist(`*numlist*`)` specifies the list of window lengths to be evaluated. By default, the program constructs 10 windows around the cutoff, the first one including 10 treated and control observations and adding 5 observations to each group in subsequent windows.

`tlist(`*numlist*`)` specifies the list of values of the treatment effect under the null to be evaluated. By default, the program uses 10 evenly spaced points within the asymptotic confidence interval for a constant treatment effect in the smallest window to be used.

`saving(`*filename*`)` saves the dataset containing the data for the contour plot in *filename*. This allows the user to replicate and modify the appearance of the plot and conduct further sensitivity analysis.

`nodots` suppresses replication dots.

`nodraw` suppresses the contour plot.

`verbose` displays the matrix of results.

`ci(`*window* [*level*]`)` returns the confidence interval corresponding to the window length indicated in *window*. The value in `ci()` needs to be one of the values in `wlist()`. The level of the confidence interval can be specified with the `level()` option. The default level is 0.05, corresponding to a 95% confidence interval.

`statistic(`*stat*`)` specifies the statistic to be used in randomization inference. *stat* may be `ttest` (DM), `ksmirnov` (KS statistic), `ranksum` (Wilcoxon–Mann–Whitney studentized statistic), or `all`, which specifies all three statistics. The default is `statistic(ttest)`.

`p(#)` specifies the order of the polynomial for the outcome transformation model. The default is `p(0)`.

`evalat(`*point*`)` specifies the point at which the adjusted variable is evaluated. *point* may be `cutoff` or `means`. The default is `evalat(cutoff)`.

`kernel(`*kerneltype*`)` specifies the type of kernel to use as the weighting scheme. *kerneltype* may be `uniform` (uniform kernel), `triangular` (triangular kernel), or `epan` (Epanechnikov kernel). The default is `kernel(uniform)`.

`fuzzy(`*fuzzy_var*`)` specifies the name of the endogenous treatment variable in the fuzzy design. This option uses an Anderson–Rubin-type statistic.

`reps(#)` specifies the number of replications for the randomization test. The default is `reps(1000)`.

`seed(#)` sets the initial seed for the randomization test. With this option, the user can manually set the desired seed or can enter the value −1 to use the system seed. The default is `seed(666)`.

6 The rdrbounds command

This section describes the syntax of the `rdrbounds` command, which calculates lower and upper bounds for the randomization p-value under different degrees of departure from a local randomized experiment, as suggested by Rosenbaum (2002).

6.1 Description

`rdrbounds` calculates Rosenbaum bounds for p-values in RD designs under local randomization.

6.2 Syntax

`rdrbounds` *outvar runvar* [*if*] [*in*] [`, cutoff(#) prob(`*varname*`)`
`gammalist(`*numlist*`) expgamma(`*numlist*`) wlist(`*numlist*`) ulist(`*numlist*`)`
`bound(`*bounds*`) fmpval statistic(`*stat*`) p(#) evalat(`*point*`)`
`kernel(`*kerneltype*`) nulltau(#) fuzzy(`*fuzzy_var*`) reps(#) seed(#)`]

outvar is the outcome variable. *runvar* is the running variable (also known as the score or forcing variable).

6.3 Options

`cutoff(#)` specifies the RD cutoff for the running variable *runvar*. The default is `cutoff(0)`.

`prob(`*varname*`)` specifies the name of the variable containing individual probabilities of treatment in a Bernoulli trial when the selection factor gamma is zero. The default is the proportion of treated units in each window (assumed equal for all units).

`gammalist(`*numlist*`)` specifies the list of values of gamma to be evaluated.

`expgamma(`*numlist*`)` specifies the list of values of exp(gamma) to be evaluated. The default is `expgamma(1.5 2 2.5 3)`.

`wlist(`*numlist*`)` specifies the list of window lengths to be evaluated. By default, the program constructs 10 windows around the cutoff, the first one including 10 treated and control observations and then adding 5 observations to each group in subsequent windows.

`ulist(`*numlist*`)` specifies the list of vectors of the unobserved confounder to be evaluated. The default takes all vectors with ones in the first k positions and zeros in the remaining position; see Rosenbaum (2002).

`bound(`*bounds*`)` specifies which bounds the command calculates. *bounds* may be `upper` (upper bound), `lower` (lower bound), or `both` (upper and lower bounds). The default is `bound(both)`.

`fmpval` calculates the p-value under fixed margins randomization in addition to the p-value under Bernoulli trials.

`statistic(`*stat*`)` specifies the statistic to be used in randomization inference. *stat* may be `ttest` (DM), `ksmirnov` (KS statistic), or `ranksum` (Wilcoxon–Mann–Whitney studentized statistic). The default is `statistic(ranksum)`.

`p(`#`)` specifies the order of the polynomial for the outcome transformation model. The default is `p(0)`.

`evalat(`*point*`)` specifies the point at which the transformed variable is evaluated. *point* may be `cutoff` or `means`. The default is `evalat(cutoff)`.

`kernel(`*kerneltype*`)` specifies the type of kernel to use as the weighting scheme. *kerneltype* may be `uniform` (uniform kernel), `triangular` (triangular kernel), or `epan` (Epanechnikov kernel). The default is `kernel(uniform)`.

`nulltau(`#`)` sets the value of the treatment effect under the null hypothesis. The default is `nulltau(0)`.

`fuzzy(`*fuzzy_var*`)` specifies the name of the endogenous treatment variable in the fuzzy design. This option uses an Anderson–Rubin-type statistic.

`reps(`#`)` sets the number of replications for the randomization test. The default is `reps(500)`.

`seed(`#`)` sets the initial seed for the randomization test. With this option, the user can manually set the desired seed or can enter the value -1 to use the system seed. The default is `seed(666)`.

7 Illustration of methods

We illustrate our commands using the dataset from Cattaneo, Frandsen, and Titiunik (2015), which has also been used to illustrate the nonparametric local polynomial methods in Calonico, Cattaneo, and Titiunik (2014a, 2015b) and Cattaneo, Jansson, and Ma (2016a). `rdlocrand_senate.dta` contains information on 1,390 U.S. Senate elections between 1914 and 2010 and was used before to analyze the effect of the incumbent status of a political party on the probability of winning future elections. The running variable in this dataset is `demmv`, the Democratic margin of victory in a statewide Senate election at t, defined as the difference in vote share between the Democratic party and its strongest opponent. A positive value of the running variable indicates that the Democratic party won the election, and the cutoff is therefore $\overline{r} = 0$.

We start by loading the dataset and providing some descriptive statistics:

```
. use rdlocrand_senate
. global covariates presdemvoteshlag1 population demvoteshlag1 demvoteshlag2
> demwinprv1 demwinprv2 dopen dmidterm
. describe $covariates
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
presdemvotesh~1 float   %9.0g                 Democratic presidential vote share
                                                at t-1
population      long    %10.0g                Population
demvoteshlag1   float   %9.0g                 Democratic vote share at t-1
demvoteshlag2   float   %9.0g                 Democratic vote share at t-2
demwinprv1      float   %9.0g                 =1 if Democratic won at t-1
demwinprv2      float   %9.0g                 =1 if Democratic won at t-2
dopen           float   %9.0g                 =1 if open seat
dmidterm        float   %9.0g                 =1 if midterm election at t
. summarize demmv $covariates
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       demmv |      1,390    7.171159    34.32488       -100        100
presdemvot~1 |      1,387    46.11975    14.31701          0   97.03408
  population |      1,390     3827919     4436950      78000   3.73e+07
demvoteshl~1 |      1,349    52.69048     18.2706          0        100
demvoteshl~2 |      1,308    52.86918    18.23913          0        100
-------------+---------------------------------------------------------
  demwinprv1 |      1,349    .5441067    .4982355          0          1
  demwinprv2 |      1,308     .543578    .4982879          0          1
       dopen |      1,380    .2471014    .4314826          0          1
    dmidterm |      1,390    .5136691     .499993          0          1
```

The running variable ranges from -100 to 100 with an average of 7 percentage points. The outcome of interest is `demvoteshfor2`, the Democratic vote share in the following election for the same Senate seat—which, given the staggered nature of Senate elections in the United States, occurs two elections later, at $t+2$. The set of covariates, described in the output above is exactly the one used in Cattaneo, Frandsen, and Titiunik (2015); thus we can replicate their empirical findings using the new commands introduced here.

The most basic syntax for `rdwinselect` is the following:

```
. rdwinselect demmv $covariates, cutoff(0)

Window selection for RD under local randomization

Cutoff c = 0.00 | Left of c   Right of c        Number of obs  =          1390
----------------+----------------------        Order of poly  =             0
    Number of obs |       640          750       Kernel type    =       uniform
   1st percentile |         6            7       Reps           =          1000
   5th percentile |        32           37       Testing method =     rdrandinf
  10th percentile |        64           75       Balance test   =         ttest
  20th percentile |       128          150

                  |   Bal. test         Var. name    Bin. test
 Window length /2 |    p-value        (min p-value)   p-value     Obs<c   Obs>=c
-----------------+------------------------------------------------------------
            0.529 |      0.210       demvoteshlag2      0.327        10       16
            0.733 |      0.262            dopen         0.200        15       24
            0.937 |      0.132            dopen         0.126        16       27
            1.141 |      0.044            dopen         0.161        20       31
            1.346 |      0.229         dmidterm         0.382        28       36
            1.550 |      0.102         dmidterm         0.728        35       39
            1.754 |      0.075         dmidterm         0.747        41       45
            1.958 |      0.046         dmidterm         0.602        43       49
            2.163 |      0.075         dmidterm         0.480        45       53
            2.367 |      0.132            dopen         0.637        53       59
Variable used in binomial test (running variable): demmv
Covariates used in balance test: presdemvoteshlag1 population demvoteshlag1
> demvoteshlag2 demwinprv1 demwinprv2 dopen dmidterm
Recommended window is [-.733; .733] with 39 observations (15 below, 24 above).
```

Because in this particular application the cutoff is zero, which is the default value, the `cutoff()` option can be omitted. Thus all the remaining examples will not specify this option. In practice, when the cutoff is not zero, the user can simply specify `cutoff()`. Alternatively, it may be easier to simply redefine the running variable by recentering it at the cutoff. By default, `rdwinselect` uses the difference-in-means statistic to perform hypothesis tests—but this can be changed with the `statistic()` option.

The output of `rdwinselect` is divided in three panels. The upper-left panel provides information on sample sizes. The first row gives the total number of observations to the left and to the right of the cutoff and also the total sample size. The following four rows provide the same information but around small neighborhoods around the cutoffs defined by the 1st, 5th, 10th, and 20th percentiles of the running variable.

The upper-right panel indicates the total sample size, the degree of the polynomial used by `rdrandinf`, the type of kernel used for the weighting scheme (`uniform`, `triangular`, or `epan`), the number of replications in the permutation test (whenever this test is performed), the method used to perform the covariate balance tests (`approximate` or `rdrandinf`), and the test statistic used (`test`, `ksmirnov`, or `ranksum`).

Finally, the main panel gives the result of the two balance tests performed at each of the windows considered. The first column provides the window length of each window considered, divided by two. For example, a value of 0.529 in this column refers to the window $[\overline{r} - 0.529; \overline{r} + 0.529]$, where $\overline{r}$ is the cutoff (equal to 0 in this case) and the

window length is $\overline{r}+0.529-(\overline{r}-0.529)=1.058$. The second column, labeled `Bal. test p-value`, provides the minimum p-value of the balancing test; the name of the corresponding variable associated with this p-value is given in column 3, `Var. name (min p-value)`. The p-value is obtained by either randomization inference or an asymptotic approximation, depending on the option specified. The fourth column gives the p-value from a binomial probability test of the hypothesis that the probability of treatment is 0.5. Type `help bitest` for further details. Columns 5 and 6 give the number of observations to the left and right of the cutoff inside each window. As indicated in the last line of the output, the largest recommended window (the largest window for which the second column is equal to or above 0.15) in this case is $[-0.733; 0.733]$ and contains 15 observations below the cutoff and 24 observations above.

By default, `rdwinselect` starts with a window that contains at least 10 observations at each side of the cutoff and increases the length, ensuring that at least 2 observations are added in each successive window. The user can choose these two values using the `obsmin()` and `obsstep()` options, respectively, or can define the windows in terms of their length instead of the number of observations. For instance, Cattaneo, Frandsen, and Titiunik (2015) start from the window $[-0.5; 0.5]$ and increase the width by 0.125 using 10,000 replications in the permutation test. To replicate their results, we can type

```
. rdwinselect demmv $covariates, wmin(.5) wstep(.125) reps(10000)

Window selection for RD under local randomization

Cutoff c = 0.00 | Left of c   Right of c        Number of obs  =          1390
----------------+----------------------        Order of poly  =             0
    Number of obs |       640          750       Kernel type    =       uniform
  1st percentile |         6            7       Reps           =         10000
  5th percentile |        32           37       Testing method =     rdrandinf
 10th percentile |        64           75       Balance test   =         ttest
 20th percentile |       128          150

                |   Bal. test         Var. name    Bin. test
 Window length /2 |   p-value       (min p-value)   p-value     Obs<c   Obs>=c
-----------------+-----------------------------------------------------------
           0.500 |       0.268   demvoteshlag2       0.230         9       16
           0.625 |       0.423         dopen         0.377        13       19
           0.750 |       0.265         dopen         0.200        15       24
           0.875 |       0.153         dopen         0.211        16       25
           1.000 |       0.074         dopen         0.135        17       28
           1.125 |       0.039         dopen         0.119        19       31
           1.250 |       0.063         dopen         0.105        21       34
           1.375 |       0.140      dmidterm         0.539        30       36
           1.500 |       0.092      dmidterm         0.640        34       39
           1.625 |       0.113      dmidterm         0.734        37       41
Variable used in binomial test (running variable): demmv
Covariates used in balance test: presdemvoteshlag1 population demvoteshlag1
> demvoteshlag2 demwinprv1 demwinprv2 dopen dmidterm

Recommended window is [-.875; .875] with 41 observations (16 below, 25 above).
```

Note that the minimum p-value for the largest recommended window, which is $[-0.875; 0.875]$, is slightly above the cutoff value 0.15. To compare our results with the ones in Cattaneo, Frandsen, and Titiunik (2015), we will set the window $[-0.75; 0.75]$ as our preferred choice.[3]

Additionally, observe that the minimum p-value is not necessarily monotonic on the length of the window. The `plot` option allows the user to depict graphically how these values change for different lengths. We will set the number of windows to 80 to have more observations in the plot, and we will specify the `approximate` option to speed up the calculations. With this option, the command uses the large-sample approximation instead of randomization inference. It is useful for illustration because it is much faster, but it can be misleading because the approximation may be poor when the sample is small. The output from `rdwinselect` with 80 windows is a long table and will be omitted. The resulting graph is shown in figure 1.

```
. quietly rdwinselect demmv $covariates, wmin(.5) wstep(.125)
> nwin(80) approximate plot
```

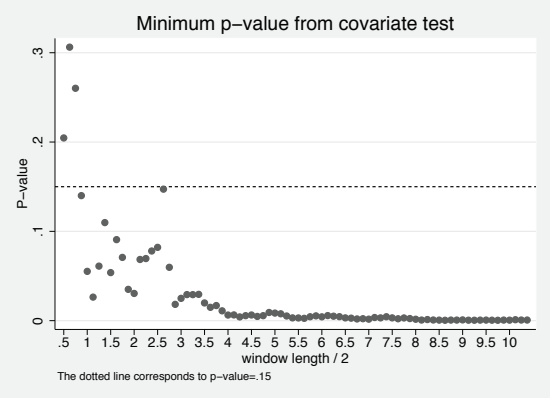


Figure 1. Plot of p-values

The figure shows that the p-values vary widely for very short windows, but the sequence stabilizes once the window length is large enough (around the value 3 in this case).

Once the window has been selected, randomization inference to test the sharp null hypothesis of no treatment effect can be performed using `rdrandinf`. For example, take the window $[-0.75; 0.75]$, which is the one selected by Cattaneo, Frandsen, and Titiunik (2015) and replicated above. The basic syntax for `rdrandinf` is

3. The user trying to replicate this code should be aware that these calculations involve permuting the treatment variable many times for a large set of covariates. Running this command may take about 40 minutes.

```
. rdrandinf demvoteshfor2 demmv, wl(-.75) wr(.75)
Selected window = [-.75 ; .75]
Running randomization-based test...
Randomization-based test complete.

Inference for sharp design

Cutoff c = 0.00 |  Left of c  Right of c        Number of obs =           1390
                |                              Order of poly =              0
     Number of obs |        595         702     Kernel type   =        uniform
Eff. Number of obs |         15          22     Reps          =           1000
   Mean of outcome |     42.808      52.497     Window        =    set by user
   S.D. of outcome |      7.042       7.742     H0:       tau =          0.000
            Window |     -0.750       0.750     Randomization =  fixed margins
Outcome: demvoteshfor2. Running variable: demmv.

                |                Finite sample          Large sample
        Statistic |       T          P>|T|          P>|T|   Power vs d =      3.52
------------------+-------------------------------------------------------------
    Diff. in means |     9.689       0.000          0.000                   0.300
```

Like the output of `rdwinselect`, that of `rdrandinf` is divided in three panels. The upper-left panel provides the number of observations at each side of the cutoff, sample size below and above the cutoff inside the specified window, some descriptive statistics for the outcome inside the window, and the selected window. Note that the first line in this panel displays the number of observations with nonmissing values of the outcome and running variable, so the sample sizes shown can differ from the total sample size. The upper-right panel gives the total sample size, the order of the polynomial, the type of kernel used for the weighting scheme (`uniform`, `triangular`, or `epan`), the number of replications in the permutation test, and whether the window was specified by the user by setting `wl()` and `wr()` or calculated using `rdwinselect`, as will be illustrated shortly.

Finally, the main panel gives the results from the randomization test. The first column, labeled `Statistic`, indicates the statistic used in the randomization test. The second column gives the observed value of the selected statistic, and the third column shows its finite-sample p-value obtained from the randomization test. The fourth column gives the asymptotic p-value, that is, the p-value obtained from the corresponding asymptotic distribution of the chosen statistic. Finally, the fifth column gives the asymptotic power against an alternative value that can be specified using the `d()` or `dscale()` option. The default is `dscale(.5)`, that is, an effect size equal to half the standard deviation of the outcome for the control group inside the window (the critical value for the power calculation is set to 1.96).

As mentioned above, `rdrandinf` uses the DM as the default statistic, but it can also use the KS and the RS statistics. By adding `statistic(all)` as an option, we can obtain the result for all three statistics. These last two statistics are obtained using the `ksmirnov` and `ranksum` Stata commands, respectively. See the corresponding Stata help files for further details. The output is

```
. rdrandinf demvoteshfor2 demmv, wl(-.75) wr(.75) statistic(all)
Selected window = [-.75 ; .75]
Running randomization-based test...
Randomization-based test complete.

Inference for sharp design

Cutoff c = 0.00 |  Left of c  Right of c        Number of obs =           1390
                |                             Order of poly =              0
     Number of obs |        595         702    Kernel type   =        uniform
Eff. Number of obs |         15          22    Reps          =           1000
   Mean of outcome |     42.808      52.497    Window        =    set by user
   S.D. of outcome |      7.042       7.742    H0:       tau =          0.000
            Window |     -0.750       0.750    Randomization =  fixed margins
Outcome: demvoteshfor2. Running variable: demmv.

                |                Finite sample          Large sample
        Statistic |       T           P>|T|          P>|T|   Power vs d =     3.52
------------------+-------------------------------------------------------------
    Diff. in means |     9.689         0.000          0.000                  0.300
Kolmogorov-Smirnov |     0.552         0.002          0.005                      .
   Rank sum z-stat |    -3.217         0.000          0.001                  0.209
```

We can see that the three statistics provide basically the same result in terms of inference; the randomization test rejects the sharp null of no treatment effect at one percent significance level in all three cases. Also note that the `rdrandinf` command does not provide the asymptotic power for the KS statistic.

The window in which to perform the randomization-based tests can be set manually using `wl()` and `wr()`. These options specify the lower and upper limits of the chosen window. Importantly, these are window limits and not lengths, so for instance, if the cutoff is 100 and the user wants a window of ± 5, the correct syntax is `wl(95) wr(105)`. We advise the user to always normalize the cutoff to zero by centering the running variable to avoid confusion.

Alternatively, the user can specify the list of covariates to have `rdrandinf` select the window automatically using `rdwinselect`. All the options allowed in `rdwinselect` can be passed through `rdrandinf`. By default, the output for `rdwinselect` is displayed before the output for `rdrandinf`, but it can be suppressed using the `quietly` option. For example,

```
. rdrandinf demvoteshfor2 demmv, statistic(all) covariates($covariates)
> wmin(.5) wstep(.125) level(0.16) quietly rdwreps(10000)
Calculating window...

Selected window = [-.75 ; .75]

Running randomization-based test...
Randomization-based test complete.

Inference for sharp design

Cutoff c = 0.00 |  Left of c  Right of c        Number of obs =           1390
----------------+----------------------        Order of poly =              0
     Number of obs |        595         702     Kernel type   =        uniform
Eff. Number of obs |         15          22     Reps          =           1000
   Mean of outcome |     42.808      52.497     Window        =    rdwinselect
   S.D. of outcome |      7.042       7.742     H0:       tau =          0.000
            Window |     -0.750       0.750     Randomization =  fixed margins

Outcome: demvoteshfor2. Running variable: demmv.
```

		Finite sample	Large sample		
Statistic	T	P>|T|	P>|T|	Power vs d =	3.52
Diff. in means	9.689	0.000	0.000		0.300
Kolmogorov-Smirnov	0.552	0.002	0.005		.
Rank sum z-stat	-3.217	0.000	0.001		0.209

Note that the reported p-values are slightly different. As explained above, the reason is that the two commands are performing the randomization test starting from different seeds. The user can obtain the exact same results for the two syntaxes by setting the same seed—for example, `seed(9876)`—in both commands.

The `rdrandinf` command allows the user to specify a polynomial transformation model for the outcomes using the `p()` option. By default, the command sets `p(0)`, which means no transformation. When `p()` is set to an integer larger than zero, the slopes (and possibly higher-order terms) are subtracted from the outcomes, leaving a residualized version of the outcome that differs only above and below the cutoff in the intercept. For instance, to perform a linear transformation, we type

```
. rdrandinf demvoteshfor2 demmv, statistic(all) wl(-.75) wr(.75) p(1)
Selected window = [-.75 ; .75]
Running randomization-based test...
Randomization-based test complete.

Inference for sharp design

Cutoff c = 0.00 |  Left of c  Right of c        Number of obs =           1390
                |                             Order of poly =              1
     Number of obs |        595         702    Kernel type   =        uniform
Eff. Number of obs |         15          22    Reps          =           1000
   Mean of outcome |     42.808      52.497    Window        =    set by user
   S.D. of outcome |      7.042       7.742    H0:       tau =          0.000
            Window |     -0.750       0.750    Randomization =  fixed margins
Outcome: demvoteshfor2. Running variable: demmv.

                |                Finite sample          Large sample
        Statistic |       T           P>|T|        P>|T|   Power vs d =     3.52
------------------+-------------------------------------------------------------
    Diff. in means |    15.297         0.000        0.066                  0.071
Kolmogorov-Smirnov |     0.797         0.000            .                      .
   Rank sum z-stat |    -4.455         0.000            .                      .
```

When a model for the outcomes is specified—that is, when `p()` is set to a number greater than zero—with the option `statistic(ttest)`, `rdrandinf` fits a regression of the outcome on the treatment dummy interacted with a polynomial of the running variable, and uses the difference in intercepts as the test-statistic. The other test statistics use the residuals described above as outcomes. Note that the command does not provide the asymptotic p-value or the asymptotic power of the KS and RS statistics, because the asymptotic distribution does not account for the model transformation and hence can be misleading.

In the presence of arbitrary interference, a confidence interval for a particular measure of the effects of the program (described in section 2.5) can be obtained with the `interfci()` option. For example, to obtain a 95% confidence interval, we type

```
. rdrandinf demvoteshfor2 demmv, wl(-.75) wr(.75) interfci(.05)
Selected window = [-.75 ; .75]
Running randomization-based test...
Randomization-based test complete.

Inference for sharp design

Cutoff c = 0.00 |  Left of c  Right of c        Number of obs =           1390
                |                             Order of poly =              0
     Number of obs |        595         702    Kernel type   =        uniform
Eff. Number of obs |         15          22    Reps          =           1000
   Mean of outcome |     42.808      52.497    Window        =    set by user
   S.D. of outcome |      7.042       7.742    H0:       tau =          0.000
            Window |     -0.750       0.750    Randomization =  fixed margins
Outcome: demvoteshfor2. Running variable: demmv.

                |                Finite sample          Large sample
        Statistic |       T          P>|T|          P>|T|   Power vs d =     3.52
------------------+-------------------------------------------------------------
    Diff. in means |     9.689       0.000          0.000                  0.300

Confidence interval under interference

        Statistic |  [95% Conf. Interval]
------------------+----------------------
    Diff. in means |     3.963      15.525
```

In terms of interpretation, one should remember that the confidence interval under interference is not a confidence interval for the point estimate (and in fact, it may not even contain the point estimate). As explained above, the interference confidence interval is constructed based on the difference between the observed statistic and the statistic that would be observed if the treatment was withheld from all units. In our application, allowing for arbitrary interference, we can say with 95% confidence that the "excess" benefit of the treated group compared with the control group is roughly between 4 and 15.5. Again, in this particular example, the point estimate under SUTVA happens to be contained in the confidence interval under interference, but this need not be the case and has no clear interpretation.

The `rdlocrand` package provides two types of sensitivity analyses to assess how p-values change with window length. The first one, `rdsensitivity`, calculates and plots a matrix of p-values over a range of values for the treatment effect under the null hypothesis (rows) and window lengths (columns). For instance, we can see how the p-values change by starting from the selected window, increasing the window length by 0.25 and over a range of treatment effects that is roughly the point estimate plus and minus 10:

```
. rdsensitivity demvoteshfor2 demmv, wlist(.75(.25)2) tlist(0(1)20) nodots verbose
Running randomization-based test...
Randomization-based test complete.
```

	.75	1	1.25	1.5	1.75	2
0	0	.001	0	0	0	0
1	0	.001	0	0	0	0
2	.001	.001	.002	0	0	0
3	.013	.004	.003	0	0	0
4	.03	.009	.007	.002	.001	0
5	.068	.023	.018	.013	.003	.005
6	.147	.062	.042	.037	.039	.029
7	.309	.144	.093	.088	.106	.092
8	.518	.306	.173	.239	.233	.262
9	.788	.534	.299	.427	.484	.569
10	.918	.869	.497	.731	.83	.939
11	.608	.844	.756	.907	.839	.668
12	.378	.514	.969	.574	.496	.36
13	.201	.268	.665	.323	.231	.134
14	.102	.139	.428	.154	.09	.036
15	.04	.051	.254	.064	.035	.007
16	.019	.016	.13	.022	.009	.002
17	.008	.006	.073	.004	.003	0
18	.003	0	.032	.001	0	0
19	.001	0	.01	.001	0	0
20	0	0	.002	0	0	0

In the above syntax, `nodots` suppresses the replication dots, and `verbose` indicates that the matrix of p-values will be displayed as part of the output (otherwise, the only output of the command is the plot, unless `nodraw` is specified).

One way to interpret these results is to see the range of values for which the p-value is above, say, 0.05, as a 95% confidence interval for the point estimate (assuming a constant additive treatment effect). Thus the above table shows how the confidence interval for the treatment effect changes as the window length increases. For instance, the 95% confidence interval for the window $[-.75; .75]$ is roughly [5; 14], whereas for the window $[-2; 2]$, it becomes [7; 13]. In this case, the point estimate seems to be relatively stable over the range of windows considered. A more thorough analysis should consider a finer grid of values in `wlist` and `tlist` and a larger number of replications, although these changes can increase computing time considerably. The options `wlist()` and `tlist()` admit the usual Stata *numlist* syntax, such as `wlist(.75 1 1.5 2)`, `wlist(.75(.25)2)`, `wlist(.75(.5)1 2 3)`, etc. The confidence interval for the window $[-.75; .75]$ can be obtained using the `ci()` option:

```
. rdsensitivity demvoteshfor2 demmv, wlist(.75(.25)2) tlist(0(1)20) nodots ci(.75)
Running randomization-based test...
Randomization-based test complete.
Confidence interval for w = .75
```

Statistic	[95% Conf.	Interval]
Diff. in means	5.000	14.000

The `saving()` option saves the dataset with the contour plot data so that the user can replicate and also modify the contour plot. The code to reproduce the default plot is

```
. rdsensitivity demvoteshfor2 demmv, wlist(.75(.25)10) tlist(0(1)20) nodots
> saving(graphdata)
Running randomization-based test...
Randomization-based test complete.
. preserve
. use graphdata, clear
. twoway contour pvalue t w, ccuts(0(0.05)1)
. restore
```

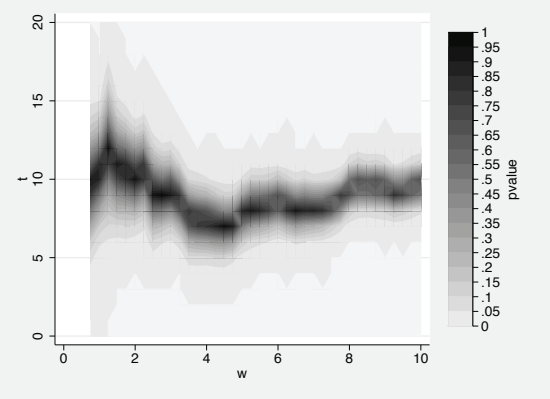


Figure 2. Sensitivity analysis

The resulting graph is shown in figure 2. The plot depicts the grid of window lengths in the horizontal axis and the grid of treatment effects under the null. The color represents the p-value for each pair of window length and treatment effect, where light gray corresponds to zero and black corresponds to one. This is simply a graphical display of the results given by `rdsensitivity`. With the data obtained, and using the `saving()` option, the user can also modify the appearance of the graph or construct different types of plots. For instance, the following command uses a gray scale instead of the default colors, with the resulting plot displayed in figure 3.

```
. preserve
. use graphdata, clear
. twoway contour pvalue t w, ccuts(0(0.05)1) ccolors(gray*0.01 gray*0.05
> gray*0.1 gray*0.15 gray*0.2 gray*0.25 gray*0.3 gray*0.35
> gray*0.4 gray*0.5 gray*0.6 gray*0.7 gray*0.8 gray*0.9 gray
> black*0.5  black*0.6 black*0.7 black*0.8 black*0.9 black)
> xlabel(.75(1.25)10) ylabel(0(2)20, nogrid) graphregion(fcolor(none))
. restore
```

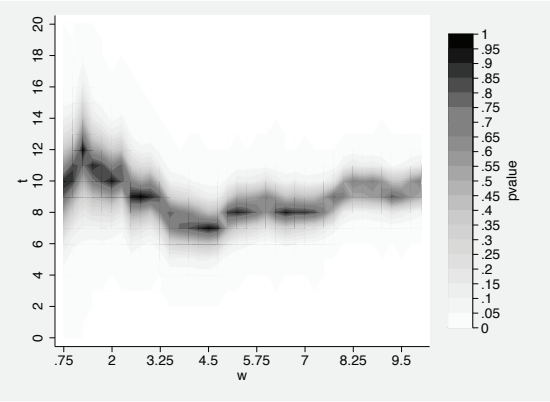


Figure 3. Manually modifying sensitivity plot

Additionally, `rdsensitivity` can be called from within `rdrandinf` to obtain confidence intervals for the point estimates obtained using the `ci()` option. The syntax is the following:

```
. rdrandinf demvoteshfor2 demmv, wl(-.75) wr(.75) ci(.05 3(1)20)
Selected window = [-.75 ; .75]
Running randomization-based test...
Randomization-based test complete.

Inference for sharp design

Cutoff c = 0.00 |  Left of c  Right of c       Number of obs =           1390
----------------+----------------------       Order of poly =              0
     Number of obs |       595         702      Kernel type   =        uniform
Eff. Number of obs |        15          22      Reps          =           1000
   Mean of outcome |    42.808      52.497      Window        =    set by user
   S.D. of outcome |     7.042       7.742      H0:       tau =          0.000
            Window |    -0.750       0.750      Randomization =  fixed margins

Outcome: demvoteshfor2. Running variable: demmv.

                |                Finite sample          Large sample
        Statistic |       T          P>|T|        P>|T|   Power vs d =      3.52
------------------+-------------------------------------------------------------
    Diff. in means |     9.689        0.000         0.000                  0.300
------------------------------------------------------------------------------

Calculating confidence interval...
Confidence interval obtained.
```

```
Confidence interval for w = .75
```

Statistic	[95% Conf.	Interval]
Diff. in means	5.000	14.000

The second type of sensitivity analysis is performed with `rdrbounds`. As explained above, this command calculates upper and lower bounds for the randomization p-value under Bernoulli trials for a range of values of a parameter $\Gamma \equiv \exp(\gamma)$ that captures the strength with which an unobservable binary variable U_i affects the probability of selection into treatment $\mathbb{P}[D_i = 1]$. The basic syntax is

```
. rdrbounds demvoteshfor2 demmv, expgamma(1.5 2 3) wlist(.5 .75 1) reps(1000)
Calculating randomization p-values...
```

w =	0.500	0.750	1.000
Bernoulli p-value	0.005	0.000	0.000

```
Running sensitivity analysis...
```

gamma	exp(gamma)	w =	0.500	0.750	1.000
0.41	1.50	lower bound	0.004	0.000	0.000
		upper bound	0.024	0.005	0.002
0.69	2.00	lower bound	0.005	0.000	0.000
		upper bound	0.052	0.025	0.008
1.10	3.00	lower bound	0.005	0.000	0.000
		upper bound	0.194	0.145	0.058

The output from `rdrbounds` is divided in two parts. The first one shows the randomization p-value based on Bernoulli trials for each window. The second panel presents the lower and upper bounds for the p-values for different values of Γ and windows. The wider the distance between the lower and upper bounds, the more sensitive the inference to deviations from a randomized experiment.

There are two directions in which one can look at the second panel. Fixing a column (that is, for a fixed window length) and moving across rows, the table indicates how inference is affected by different degrees of departure from a randomized experiment. This is the type of sensitivity analysis described in Rosenbaum (2002). On the other hand, fixing a row (that is, for a fixed value of Γ) and moving across columns, the matrix shows how sensitive the upper and lower bounds for the p-value are to the choice of the window.

In our application, when the window length is 0.5, the Bernoulli p-value is 0.005. We can see that when $\Gamma = 1.5$, the upper bound is 0.024, so the effect remains significant at 5%. When $\Gamma = 2$, the effect is still significant at the 10% level. On the other hand, when $\Gamma = 3$, the bounds become 0.005 and 0.194, so the evidence for a statistically significant effect is weaker. For the window length selected using `rdwinselect`, $[-0.75; 0.75]$, the p-values look fairly robust to misspecification of the selection mechanism: even if the

unobservable confounder increased the odds ratio of a treated unit compared with a control unit by a factor of 2, the effect would remain significant at the 10% level.

The `fmpval` option adds the fixed margins randomization p-value to the first panel of the output. This allows the user to compare the p-values obtained using each method.

```
. rdrbounds demvoteshfor2 demmv, expgamma(1.5 2 3) wlist(.5 .75 1) reps(1000)
> fmpval
Calculating randomization p-values...
                            w =     0.500       0.750       1.000
------------------------------------------------------------------
              Bernoulli p-value     0.005       0.000       0.000
          Fixed margins p-value     0.007       0.000       0.001
------------------------------------------------------------------

Running sensitivity analysis...
gamma  exp(gamma)             w =     0.500       0.750       1.000
------------------------------------------------------------------
 0.41      1.50     lower bound     0.004       0.000       0.000
                    upper bound     0.024       0.005       0.002
------------------------------------------------------------------
 0.69      2.00     lower bound     0.005       0.000       0.000
                    upper bound     0.052       0.025       0.008
------------------------------------------------------------------
 1.10      3.00     lower bound     0.005       0.000       0.000
                    upper bound     0.194       0.145       0.058
------------------------------------------------------------------
```

We can see that the p-values obtained by both methods are very similar, which we found to be usually true in applications as long as the number of replications is large enough.

Finally, when we use outcome transformation, the `rdrandinf` command allows the user to choose in which point to evaluate the transformed outcomes. By default, the evaluation point is the cutoff, which emulates the idea used in the local polynomial approach of estimating the effect at the cutoff. However, whenever the local randomization assumption is plausible, the cutoff need not be the point of interest. For instance, to set the evaluation points at the means of the running variable inside the window below and above the cutoff, we can type

```
. quietly summarize demmv if abs(demmv)<=.75 & demmv>=0 & demmv!=. &
> demvoteshfor2!=.
. local mt=r(mean)
. quietly summarize demmv if abs(demmv)<=.75 & demmv<0  & demmv!=. &
> demvoteshfor2!=.
. local mc=r(mean)
. rdrandinf demvoteshfor2 demmv, wl(-.75) wr(.75) p(1) evall(`mc´) evalr(`mt´)
Selected window = [-.75 ; .75]
Running randomization-based test...
Randomization-based test complete.
Inference for sharp design
```

Cutoff c = 0.00	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	15	22
Mean of outcome	42.808	52.497
S.D. of outcome	7.042	7.742
Window	-0.750	0.750

Number of obs =	1390
Order of poly =	1
Kernel type =	uniform
Reps =	1000
Window =	set by user
H0: tau =	0.000
Randomization =	fixed margins

Outcome: demvoteshfor2. Running variable: demmv.

		Finite sample	Large sample		
Statistic	T	P>|T|	P>|T|	Power vs d =	3.52
Diff. in means	9.689	0.000	0.000		0.293

In this case, the user can verify that the point estimate is the same as when no transformation is used: this is because the transformation comes from a linear regression that by construction passes through the means. The p-values, however, can differ. Incidentally, note that the means are taken over the sample inside the window with nonmissing values for the outcome and the running variable. The reason is that `rdrandinf` drops the observations inside the window with missing outcomes or running variables. Similarly, `rdwinselect` drops, at each evaluated window, the observations with missing values of the covariates and running variables.

8 Conclusion

We introduced and discussed the commands `rdrandinf`, `rdwinselect`, `rdsensitivity`, and `rdrbounds`, which together offer a comprehensive and systematic set of tools to analyze RD designs under a local randomization assumption. These methods complement existing procedures based on nonparametric asymptotic approximations by providing an alternative based on exact finite-sample results. These methods should be used only when the local randomization assumption, possibly after transformation of outcomes, is warranted. In these cases, our implementation can be used for both conducting inference in RD designs and offering a robustness check on more conventional methods based on nonparametric local polynomial techniques. Companion R functions are also available from the authors.

9 Acknowledgments

We thank Jake Bowers and David Drukker for useful comments on this project. The authors gratefully acknowledge financial support from the National Science Foundation through grant SES-1357561.

10 References

Calonico, S., M. D. Cattaneo, and R. Titiunik. 2014a. Robust data-driven inference in the regression-discontinuity design. *Stata Journal* 14: 909–946.

———. 2014b. Robust nonparametric confidence intervals for regression-discontinuity designs. *Econometrica* 82: 2295–2326.

———. 2015a. Optimal data-driven regression discontinuity plots. *Journal of the American Statistical Association* 110: 1753–1769.

———. 2015b. rdrobust: An R package for robust nonparametric inference in regression-discontinuity designs. *R Journal* 7: 38–51.

Cattaneo, M. D., B. R. Frandsen, and R. Titiunik. 2015. Randomization inference in the regression discontinuity design: An application to party advantages in the U.S. Senate. *Journal of Causal Inference* 3: 1–24.

Cattaneo, M. D., M. Jansson, and X. Ma. 2016a. rddensity: Manipulation testing based on density discontinuity in R. Working Paper, University of Michigan. http://www-personal.umich.edu/∼cattaneo/software/rddensity/R/rddensity-manual.pdf.

———. 2016b. Simple local regression distribution estimators with an application to manipulation testing. Working Paper, University of Michigan. http://www-personal.umich.edu/∼cattaneo/papers/Cattaneo-Jansson-Ma_2016_LocPolDensity.pdf.

Cattaneo, M. D., R. Titiunik, and G. Vazquez-Bare. 2016. Comparing inference approaches for RD designs: A reexamination of the effect of head start on child mortality. Working Paper, University of Michigan. http://www-personal.umich.edu/∼titiunik/papers/CattaneoTitiunikVazquezBare2015_wp.pdf.

Hahn, J., P. Todd, and W. van der Klaauw. 2001. Identification and estimation of treatment effects with a regression-discontinuity design. *Econometrica* 69: 201–209.

Imbens, G. W., and T. Lemieux. 2008. Regression discontinuity designs: A guide to practice. *Journal of Econometrics* 142: 615–635.

Imbens, G. W., and P. R. Rosenbaum. 2005. Robust, accurate confidence intervals with a weak instrument: Quarter of birth and education. *Journal of the Royal Statistical Society: Series A* 168: 109–126.

Imbens, G. W., and D. B. Rubin. 2015. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. New York: Cambridge University Press.

Keele, L., R. Titiunik, and J. R. Zubizarreta. 2015. Enhancing a geographic regression discontinuity design through matching to estimate the effect of ballot initiatives on voter turnout. *Journal of the Royal Statistical Society: Series A* 178: 223–239.

Keele, L. J., and R. Titiunik. 2015. Geographic boundaries as regression discontinuities. *Political Analysis* 23: 127–155.

Lee, D. S. 2008. Randomized experiments from non-random selection in U.S. House elections. *Journal of Econometrics* 142: 675–697.

Lee, D. S., and T. Lemieux. 2010. Regression discontinuity designs in economics. *Journal of Economic Literature* 48: 281–355.

McCrary, J. 2008. Manipulation of the running variable in the regression discontinuity design: A density test. *Journal of Econometrics* 142: 698–714.

Rosenbaum, P. R. 2002. *Observational Studies*. 2nd ed. New York: Springer.

———. 2007. Interference between units in randomized experiments. *Journal of the American Statistical Association* 102: 191–200.

———. 2010. *Design of Observational Studies*. New York: Springer.

Skovron, C., and R. Titiunik. 2015. A practical guide to regression discontinuity designs in political science. Working Paper, University of Michigan. http://www-personal.umich.edu/~titiunik/papers/SkovronTitiunik2015.pdf.

About the authors

Matias D. Cattaneo is an associate professor of economics and an associate professor of statistics at the University of Michigan.

Rocío Titiunik is an associate professor of political science at the University of Michigan.

Gonzalo Vazquez-Bare is a PhD candidate in economics at the University of Michigan.

The Stata Journal (2016)
16, Number 2, pp. 368–385

Simpler standard errors for two-stage optimization estimators

Joseph V. Terza
Department of Economics
Indiana University Purdue University Indianapolis
Indianapolis, IN
jvterza@iupui.edu

Abstract. Aiming to lessen the analytic and computational burden faced by practitioners seeking to correct the standard errors of two-stage estimators, I offer a heretofore unexploited simplification of the conventional formulation for the most commonly encountered cases in empirical application—two-stage estimators that involve maximum likelihood or pseudomaximum likelihood estimation. With the applied researcher in mind, I focus on the two-stage residual inclusion estimator designed for nonlinear regression models involving endogeneity. I demonstrate the analytics and Stata and Mata code for implementing my simplified standard-error formula by applying the two-stage residual inclusion method to the birthweight model of Mullahy (1997, *Review of Economics and Statistics* 79: 586–593) using his original data.

Keywords: st0436, two-stage optimization estimators, standard errors, asymptotic theory, endogeneity, two-stage residual inclusion, sandwich estimator

1 Introduction

Asymptotic theory for the two-stage optimization estimator (2SOE) (in particular, correct formulation of the asymptotic standard errors) has been available to applied researchers for decades (see Murphy and Topel [1985] for cases in which each of the stages involves a maximum likelihood estimator [MLE] and Newey and McFadden [1994] and White [1994] for more general classes of 2SOE). Despite textbook treatments of the subject (Cameron and Trivedi 2005, Greene 2012, and Wooldridge 2010) and articles offering requisite computer code (in Stata) (Hardin 2002 and Hole 2006), when conducting statistical inference based on two-stage estimates, applied researchers often implement bootstrapping methods or ignore the two-stage nature of the estimator and report the uncorrected second-stage outputs from packaged statistical software. In this article, aiming toward easy software implementation (in Stata), I offer the practitioner a heretofore largely unexploited simplification of the textbook asymptotic covariance matrix formulations (and their estimators—standard errors) for the most commonly encountered versions of the 2SOE—those involving an MLE or the nonlinear least squares (NLS) method in either stage. In addition, and perhaps more importantly from a practitioner's standpoint, I focus on regression models involving endogeneity—a sampling problem whose solution often requires a 2SOE. In particular, to fix ideas, I detail my simplified covariance specifications for the two-stage residual inclusion (2SRI) estimator

suggested by Terza, Basu, and Rathouz (2008). I illustrate the analytics and Stata code for my approach by applying 2SRI to a model of birthweight as a function of endogenous smoking during pregnancy (see Mullahy [1997]).

The remainder of the article is organized as follows. In section 2, I review the asymptotic theory of 2SOE and give the conventional textbook formulation of the corresponding correct asymptotic covariance matrix. I also show how this formulation can be simplified when the estimator implements either NLS or MLE. In section 3, I detail the 2SRI estimator and, in light of the discussion in section 2, derive its correct (and simplified) asymptotic standard errors. In section 4, I discuss the birthweight illustration in full analytic and Stata and Mata coding detail. Section 5 summarizes. Technical details are given in appendixes.

2 2SOEs and their asymptotic standard errors

The most commonly applied parametric estimators reside in the class of optimization estimators (OEs)—statistical methods that produce estimates as optimizers of well-specified objective functions (sometimes called M-estimators). The most prominent OE examples are the MLE and the NLS estimators. Model design or computational convenience often dictates that an OE be implemented in two stages. In such cases, the parameter vector of interest is partitioned as $\boldsymbol{\omega}' = [\boldsymbol{\alpha}' \ \boldsymbol{\beta}']$ and conformably estimated in two stages. First, an estimate of $\boldsymbol{\alpha}$ is obtained as the optimizer of an appropriately specified first-stage objective function,

$$\sum_{i=1}^{n} q_1(\boldsymbol{\alpha}, \mathbf{V}_{1i}) \tag{1}$$

where $\mathbf{V}_{1i}$ denotes the relevant subvector of the observable data for the ith sample individual ($i = 1, \ldots, n$). Next, an estimate of $\boldsymbol{\beta}$ is obtained as the optimizer of

$$\sum_{i=1}^{n} q_2\left(\widehat{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \mathbf{V}_{2i}\right) \tag{2}$$

where $\mathbf{V}_{2i}$ denotes the second-stage analog to $\mathbf{V}_{1i}$ and $\widehat{\boldsymbol{\alpha}}$ is the first-stage estimate of $\boldsymbol{\alpha}$.

It is well established that under general conditions, this 2SOE is consistent and asymptotically normal.[1] My interest here is in simplifying the typical textbook formulation of the corresponding asymptotic covariance matrix of $\widehat{\boldsymbol{\omega}}' = [\widehat{\boldsymbol{\alpha}}' \ \widehat{\boldsymbol{\beta}}']$, where $\widehat{\boldsymbol{\beta}}$ denotes the second-stage estimator obtained from (2).[2] Before proceeding, I establish the following notational conventions:

1. See Newey and McFadden (1994) or White (1994) for details.
2. For textbook discussions of this issue, see Wooldridge (2010, sec. 12.4.2); Greene (2012, sec. 14.7); and Cameron and Trivedi (2005, sec. 6.6).

1. q_1 is shorthand notation for $q_1(\boldsymbol{\alpha}, \mathbf{V}_1)$ as defined in (1), where $\mathbf{V}_1$ is the random vector representing the population from which the observed data in $\mathbf{V}_{1i}$ was sampled.

2. q_2 is shorthand notation for $q_2(\widehat{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \mathbf{V}_2)$ as defined in (2), where $\mathbf{V}_2$ is the random vector representing the population from which the observed data in $\mathbf{V}_{2i}$ was sampled.

3. $\boldsymbol{\nabla}_s \mathbf{q}$ denotes the gradient of q with respect to parameter subvector $\mathbf{s}$—a row vector.

4. $\boldsymbol{\nabla}_{st} \mathbf{q}$ denotes the matrix whose typical element is $\partial^2 q / \partial s_j \partial t_m$—its row dimension corresponds to that of its first subscript, and the column dimension corresponds to that of its second subscript.

From Newey and McFadden (1994) or White (1994), we have that the correct asymptotic covariance matrix of $\widehat{\boldsymbol{\omega}}' = [\widehat{\boldsymbol{\alpha}}' \ \widehat{\boldsymbol{\beta}}']$ is

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_{11} & \mathbf{D}_{12} \\ \mathbf{D}_{12}' & \mathbf{D}_{22} \end{bmatrix}$$

where

$$\mathbf{D}_{11} = \textsc{avar}\left(\widehat{\boldsymbol{\alpha}}\right) = \mathbf{E}\left(\boldsymbol{\nabla}_{\alpha\alpha}\mathbf{q}_1\right)^{-1} \mathbf{E}\left(\boldsymbol{\nabla}_{\alpha}\mathbf{q}_1'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_1\right) \mathbf{E}\left(\boldsymbol{\nabla}_{\alpha\alpha}\mathbf{q}_1\right)^{-1} \tag{3}$$

$$\begin{aligned} \mathbf{D}_{12} = {} & \mathbf{E}\left(\boldsymbol{\nabla}_{\alpha\alpha}\mathbf{q}_1\right)^{-1} \mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_1\right)' \mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right)^{-1} \\ & - \textsc{avar}\left(\widehat{\boldsymbol{\alpha}}\right) \mathbf{E}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right)' \mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right)^{-1} \end{aligned} \tag{4}$$

$$\begin{aligned} \mathbf{D}_{22} = {} & \textsc{avar}\left(\widehat{\boldsymbol{\beta}}\right) \\ = {} & \mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right)^{-1} \left\{ \mathbf{E}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right) \textsc{avar}\left(\widehat{\boldsymbol{\alpha}}\right) \mathbf{E}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right)' \right. \\ & - \mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_1\right) \mathbf{E}\left(\boldsymbol{\nabla}_{\alpha\alpha}\mathbf{q}_1\right)^{-1} \mathbf{E}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right)' \\ & \left. - \mathbf{E}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right) \mathbf{E}\left(\boldsymbol{\nabla}_{\alpha\alpha\mathbf{q}_1}\right)^{-1} \mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_1\right)' \right\} \mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right)^{-1} \\ & + \mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right)^{-1} \mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\beta}\mathbf{q}_2\right) \mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right)^{-1} \end{aligned} \tag{5}$$

and $\textsc{avar}(\mathbf{c})$ denotes the asymptotic covariance matrix of the estimator $\mathbf{c}$. For cases in which the ultimate estimation objective is $\boldsymbol{\beta}$, only $\mathbf{D}_{22}$ is of interest. In most cases, however, an estimate of $\mathbf{D}$ will be needed. Hence, my interest is in simplifying the details of the full formulation of $\mathbf{D}$.

Hardin (2002) rediscovers the general formulation in (2) through (5) but focuses on the special case in which MLEs are applied in both stages of the 2SOE estimator (henceforth MLE-MLE-2SOE). He refers to (2) through (5) as the "sandwich" formulation of the asymptotic covariance matrix. In this case, based on the well-known asymptotic theory for the MLE combined with results given in Murphy and Topel (1985), I show in *Appendix A* that the following legitimate substitutions can be made in (2) through (5),[3]

$$\mathbf{D}_{11} = \mathbf{AVAR}\left(\widehat{\boldsymbol{\alpha}}\right) = -\mathbf{E}\left(\boldsymbol{\nabla}_{\alpha\alpha}\mathbf{q}_1\right)^{-1} \tag{6}$$

$$\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right) = -\mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\beta}\mathbf{q}_2\right) \tag{7}$$

$$\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right) = -\mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_2\right) \tag{8}$$

I refer to (2) through (5) with substitutions (2) through (4) as the MT (Murphy–Topel) formulation of the asymptotic covariance matrix. Clearly, (2) through (4) are simplifying in that they substantially reduce both analytic and computational demands. Nevertheless, Hardin (2002) advocates using the sandwich formulation on the grounds that it 1) is robust, 2) is easy to calculate, and 3) admits Wald tests of hypotheses across the two stages of the model. It is clear, however, that 1) the sandwich and MT formulations are equally robust given the assumptions underlying the MLE-MLE-2SOE; 2) the MT formulation is easier to calculate; and 3) the full MT formulation also facilitates tests of hypotheses involving elements of both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Thus, for the MLE-MLE-2SOE, I see no apparent justification for incurring the higher analytic and computational costs of the sandwich formulation in practice. Hole (2006) apparently agrees and shows how calculation of the MT formulation of the asymptotic covariance matrix of the MLE-MLE-2SOE can be simplified using the `scores` option of the Stata `predict` command.

I extend this line of research by including a broader class of 2SOE estimators that subsumes the MLE-MLE-2SOE as a special case. I first note that in general, $\mathbf{D}_{11}$ warrants no discussion, because neither its formulation nor its estimation is affected by the two-stage nature of the estimator—$\boldsymbol{\beta}$ does not appear in (1). Therefore, the correct standard errors of $\widehat{\boldsymbol{\alpha}}$, and other inferential statistics pertaining to $\widehat{\boldsymbol{\alpha}}$, can be obtained from the "packaged" output of the software used for first-stage estimation. In other words, we can rewrite (2) as

$$\mathbf{D}_{11} = \mathbf{AVAR}^{*}\left(\widehat{\boldsymbol{\alpha}}\right) \tag{9}$$

where $\mathbf{AVAR}^{*}(\mathbf{c})$ denotes the matrix to which the packaged estimate of the asymptotic covariance matrix of the estimator $\mathbf{c}$ converges in probability (by "packaged", I refer to what would be obtained from any econometrics computer package for the estimator $\mathbf{c}$, ignoring the fact that it is a component of a 2SOE). By the same token, I need only consider how the choice of method for the second-stage determines the formulation and estimation of $\mathbf{D}_{12}$ and $\mathbf{D}_{22}$ in (4) and (5), respectively. For these reasons, I extend the discussion to the class of 2SOEs whose first stage is any OE and whose second stage is either MLE or NLS (the two most commonly implemented OEs). This class of estimators

3. In fact, Hardin (2002) imposes (4) in his specification of the sandwich formulation of the asymptotic covariance matrix of the MLE-MLE-2SOE. Strictly speaking, this is not in keeping with his formulation of the matrix A in (5)—the "bread" matrix in his sandwich formulation.

will henceforth be denoted 2SOE*. I will show that for 2SOE*, heretofore unexploited simplifications in the general asymptotic covariance formulation [(2) through (5)] are possible. First, as I show in *Appendix B*, for 2SOE*,

$$\mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_1\right)=\mathbf{0} \tag{10}$$

Until now, this condition was not recognized and therefore was not incorporated in either the textbook treatments of 2SOE or the studies by Hardin (2002) and Hole (2006). Applying (3) and (5) and noting that

$$\mathbf{AVAR}^{*}\left(\widehat{\boldsymbol{\beta}}\right)=\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right)^{-1}\mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\beta}\mathbf{q}_2\right)\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right)^{-1}$$

we see that (4) and (5) can be substantially simplified as, respectively,

$$\mathbf{D}_{12}=-\mathbf{AVAR}^{*}\left(\widehat{\boldsymbol{\alpha}}\right)\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right)'\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right)^{-1} \tag{11}$$

and

$$\begin{aligned}\mathbf{D}_{22}&=\mathbf{AVAR}\left(\widehat{\boldsymbol{\beta}}\right)\\&=\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right)^{-1}\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right)\mathbf{AVAR}^{*}\left(\widehat{\boldsymbol{\alpha}}\right)\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right)'\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right)^{-1}\\&\quad+\mathbf{AVAR}^{*}\left(\widehat{\boldsymbol{\beta}}\right)\end{aligned} \tag{12}$$

Equations (6) and (7) highlight the covariance matrix components that can be obtained directly from packaged econometric software versus those that require special programming. If the second stage ($\widehat{\boldsymbol{\beta}}$) is MLE, using (3) and (4) and noting that $\mathbf{AVAR}^{*}(\widehat{\boldsymbol{\beta}})=-\mathbf{E}(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2)^{-1}$, we can rewrite (6) and (7) as

$$\mathbf{D}_{12}=\mathbf{AVAR}^{*}\left(\widehat{\boldsymbol{\alpha}}\right)\mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_2\right)'\mathbf{AVAR}^{*}\left(\widehat{\boldsymbol{\beta}}\right) \tag{13}$$

and

$$\begin{aligned}\mathbf{D}_{22}&=\mathbf{AVAR}\left(\widehat{\boldsymbol{\beta}}\right)\\&=\mathbf{AVAR}^{*}\left(\widehat{\boldsymbol{\beta}}\right)\mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_2\right)\mathbf{AVAR}^{*}\left(\widehat{\boldsymbol{\alpha}}\right)\mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_2\right)'\mathbf{AVAR}^{*}\left(\widehat{\boldsymbol{\beta}}\right)\\&\quad+\mathbf{AVAR}^{*}\left(\widehat{\boldsymbol{\beta}}\right)\end{aligned} \tag{14}$$

In this case, the only component that must be analytically derived and coded is $\mathbf{E}(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_2)$. A consistent estimator of this component is

$$\widehat{\mathbf{E}}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_2\right)=\frac{\sum_{i=1}^{n}\boldsymbol{\nabla}_{\beta}\widehat{\mathbf{q}}_{2i}'\boldsymbol{\nabla}_{\alpha}\widehat{\mathbf{q}}_{2i}}{n} \tag{15}$$

where $\widehat{q}_{2i}$ is shorthand notation for $q_2(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\beta}}, \mathbf{V}_{2i})$. Equation (3) can be coded in Mata. Here consistent estimators of the components of the asymptotic covariance matrix given in (2), (3), and (8) are

$$\widehat{D}_{11} = n\widehat{\mathbf{AVAR}}^{*}(\widehat{\boldsymbol{\alpha}}) \tag{16}$$

$$\widehat{\mathbf{D}}_{12} = \left\{n\widehat{\mathbf{AVAR}}^{*}(\widehat{\boldsymbol{\alpha}})\right\}\widehat{\mathbf{E}}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_2\right)'\left\{n\widehat{\mathbf{AVAR}}^{*}\left(\widehat{\boldsymbol{\beta}}\right)\right\} \tag{17}$$

$$\widehat{\mathbf{D}}_{22} = \left\{n\widehat{\mathbf{AVAR}}^{*}\left(\widehat{\boldsymbol{\beta}}\right)\right\}\widehat{\mathbf{E}}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_2\right)\left\{n\widehat{\mathbf{AVAR}}^{*}(\widehat{\boldsymbol{\alpha}})\right\}\widehat{\mathbf{E}}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_2\right)' \left\{n\widehat{\mathbf{AVAR}}^{*}\left(\widehat{\boldsymbol{\beta}}\right)\right\} + \left\{n\widehat{\mathbf{AVAR}}^{*}\left(\widehat{\boldsymbol{\beta}}\right)\right\} \tag{18}$$

$\widehat{\mathbf{AVAR}}^{*}(\widehat{\boldsymbol{\alpha}})$ and $\widehat{\mathbf{AVAR}}^{*}(\widehat{\boldsymbol{\beta}})$ are the estimated covariance matrices obtained from the first- and second-stage packaged regression outputs, respectively.

When the second stage is NLS, the main component of (2) can be written as

$$q_2\left(\widehat{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \mathbf{V}_{2i}\right) = -\left\{Y_i - J\left(\widehat{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \mathbf{Z}_{2i}\right)\right\}^2$$

where $J(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{Z}_2)$ denotes the relevant nonlinear regression function and $\mathbf{V}_{2i} = [Y_i \;\; \mathbf{Z}_{2i}]$. In this case, two components of (6) and (7) must be analytically derived—namely, $\mathbf{E}(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2)$ and $\mathbf{E}(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2)$. In *Appendix C*, I show that these components can be consistently estimated using

$$\widehat{\mathbf{E}}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right) = \frac{\sum_{i=1}^{n}\boldsymbol{\nabla}_{\beta}\widehat{\mathbf{J}}_i'\boldsymbol{\nabla}_{\alpha}\widehat{\mathbf{J}}_i}{n} \tag{19}$$

and

$$\widehat{\mathbf{E}}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right) = \frac{\sum_{i=1}^{n}\boldsymbol{\nabla}_{\beta}\widehat{\mathbf{J}}_i'\boldsymbol{\nabla}_{\beta}\widehat{\mathbf{J}}_i}{n} \tag{20}$$

respectively, where $\widehat{J}_i$ is shorthand notation for $J(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\beta}}, \mathbf{Z}_{2i})$.[4] Expressions (19) and (20) appear neither in textbook treatments of the subject nor in the studies by Hardin (2002) and Hole (2006) and constitute a useful simplification in 2SOE with second-stage NLS in that they require the derivation and coding of only first-order partial derivatives. Using these results, we have that when the second stage is NLS, consistent estimators of the components of the asymptotic covariance matrix are

4. Strictly speaking, both (19) and (20) should be multiplied by -2, but because the 2s cancel in the formulations of (22) and (23), we suppress them here.

$$\widehat{\mathbf{D}}_{11} = n\widehat{\mathbf{AVAR}}^{*}(\widehat{\boldsymbol{\alpha}}) \tag{21}$$

$$\widehat{\mathbf{D}}_{12} = \left\{n\widehat{\mathbf{AVAR}}^{*}(\widehat{\boldsymbol{\alpha}})\right\}\widehat{\mathbf{E}}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right)'\widehat{\mathbf{E}}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right)^{-1} \tag{22}$$

$$\widehat{\mathbf{D}}_{22} = \widehat{\mathbf{E}}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right)^{-1}\widehat{\mathbf{E}}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right)\left\{n\widehat{\mathbf{AVAR}}^{*}(\widehat{\boldsymbol{\alpha}})\right\}\times\widehat{\mathbf{E}}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right)'\widehat{\mathbf{E}}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right)^{-1} + n\widehat{\mathbf{AVAR}}^{*}\left(\widehat{\boldsymbol{\beta}}\right) \tag{23}$$

and $\widehat{\mathbf{AVAR}}^{*}(\widehat{\boldsymbol{\alpha}})$ and $\widehat{\mathbf{AVAR}}^{*}(\widehat{\boldsymbol{\beta}})$ are defined as in (3) through (18).

So, for example, in either case (second-stage MLE or NLS), the t statistic

$$\frac{\sqrt{n}\left(\widehat{\beta}_k - \beta_k\right)}{\sqrt{\widehat{D}_{22(k)}}} \tag{24}$$

for the kth element of β is asymptotically standard normally distributed (where $\widehat{D}_{22(k)}$ denotes the kth diagonal element of $\widehat{\mathbf{D}}_{22}$) and can be used to test the hypothesis that $\beta_k = \beta_k^0$ for β_k^0, a given null value of β_k. In practice, the following version of (24) is used for this purpose,

$$\frac{\widehat{\beta}_k - \beta_k}{\sqrt{\widehat{D}^{\dagger}_{22(k)}}} \tag{25}$$

where $\widehat{D}^{\dagger}_{22(k)}$ denotes the kth diagonal element of $\widehat{\mathbf{D}}^{\dagger}_{22}$, which is the same as $\widehat{\mathbf{D}}_{22}$ except that all the "n's" are suppressed, including those in the denominators of (3), (19), and (20). $\widehat{\mathbf{D}}^{\dagger}_{22}$ takes the following forms for the MLE and NLS cases, respectively,

$$\widehat{\mathbf{D}}^{\dagger}_{22} = \widehat{\mathbf{AVAR}}^{*}\left(\widehat{\boldsymbol{\beta}}\right)\left(\widehat{\mathbf{A}}_{\beta\alpha}\right)\widehat{\mathbf{AVAR}}^{*}(\widehat{\boldsymbol{\alpha}})\left(\widehat{\mathbf{A}}_{\beta\alpha}\right)'\widehat{\mathbf{AVAR}}^{*}\left(\widehat{\boldsymbol{\beta}}\right) + \widehat{\mathbf{AVAR}}^{*}\left(\widehat{\boldsymbol{\beta}}\right) \tag{26}$$

and

$$\widehat{\mathbf{D}}^{\dagger}_{22} = \left(\widehat{\mathbf{B}}_{\beta\beta}\right)^{-1}\left(\widehat{\mathbf{B}}_{\beta\alpha}\right)\widehat{\mathbf{AVAR}}^{*}(\widehat{\boldsymbol{\alpha}})\left(\widehat{\mathbf{B}}_{\beta\alpha}\right)'\left(\widehat{\mathbf{B}}_{\beta\beta}\right)^{-1} + \mathbf{AVAR}^{*}\left(\widehat{\boldsymbol{\beta}}\right) \tag{27}$$

where

$$\widehat{\mathbf{A}}_{\beta\alpha} = \sum_{i=1}^{n}\boldsymbol{\nabla}_{\beta}\widehat{\mathbf{q}}'_{2i}\boldsymbol{\nabla}_{\alpha}\widehat{\mathbf{q}}_{2i}$$

$$\widehat{\mathbf{B}}_{\beta\alpha} = \sum_{i=1}^{n}\boldsymbol{\nabla}_{\beta}\widehat{\mathbf{J}}'_{i}\boldsymbol{\nabla}_{\alpha}\widehat{\mathbf{J}}_{i}$$

and

$$\widehat{\mathbf{B}}_{\beta\beta} = \sum_{i=1}^{n}\boldsymbol{\nabla}_{\beta}\widehat{\mathbf{J}}'_{i}\boldsymbol{\nabla}_{\alpha}\widehat{\mathbf{J}}_{i}$$

Expressions in (26) and (27), along with the analogous formulations corresponding to (3), (17), (21), and (22), constitute substantial simplification of a) the textbook formulations given in (2), (4), and (5) and b) the covariance formulations offered for the special case (MLE-MLE-2SOE) considered by Hardin (2002) and Hole (2006).

3 2SRI

Consider a nonlinear regression model with dependent variable Y and a particular regressor of policy interest X_p. Suppose that the data on X_p are sampled endogenously—that is, it is correlated with an unobservable variable, X_u, that is also correlated with Y. To formalize this, we follow Terza, Basu, and Rathouz (2008) and assume that

$$E\left(Y|X_p, \mathbf{X}_o, X_u\right) = \mu\left(X_p, \mathbf{X}_o, X_u\left(\mathbf{W}, \boldsymbol{\alpha}\right); \boldsymbol{\beta}\right) \quad \text{[outcome regression]} \tag{28}$$

and

$$X_p = r(\mathbf{W}, \boldsymbol{\alpha}) + X_u \quad \text{[auxiliary regression]} \tag{29}$$

where $\mathbf{X}_o$ denotes a vector of observable confounders (observable variables that are possibly correlated with both Y and X_p), $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ are parameter vectors, $\mathbf{W} = [\mathbf{X}_o \;\; \mathbf{W}^+]$, $\mathbf{W}^+$ is a vector of identifying instrumental variables (possibly comprising a single element), and $\mu()$ and $r()$ are known functions. Note that (29) implies that X_u can be written as the following function of W and α:

$$X_u(\mathbf{W}, \boldsymbol{\alpha}) = X_p - r(\mathbf{W}, \boldsymbol{\alpha}) \tag{30}$$

Hence, its parametric representation in (28). Stated succinctly, the relevant outcome regression is

$$Y = \mu\left(X_p, \mathbf{X}_o, X_u\left(\mathbf{W}, \boldsymbol{\alpha}\right); \boldsymbol{\beta}\right) + e \tag{31}$$

where e is the random-error term, tautologically defined as $e = Y - \mu(X_p, \mathbf{X}_o, X_u; \boldsymbol{\beta})$ so that $E(e|X_p, \mathbf{X}_o, X_u) = 0$. The β parameters in (31) are not directly estimable (for example, via NLS) because of the presence of the unobservable confounder X_u, which embodies the endogeneity of X_p. However, Terza, Basu, and Rathouz (2008) show that the following two-stage estimator of $\boldsymbol{\omega} = [\boldsymbol{\alpha}' \;\; \boldsymbol{\beta}']$ is feasible and consistent.

First stage: Obtain a consistent estimate of α by applying NLS to (29) and compute the residual as

$$\widehat{X}_u = X_p - r(\mathbf{W}, \widehat{\boldsymbol{\alpha}})$$

where $\widehat{\boldsymbol{\alpha}}$ is the first-stage estimate of $\boldsymbol{\alpha}$.

Second stage: Estimate $\boldsymbol{\beta}$ by applying NLS to

$$Y = \mu\left(X_p, \mathbf{X}_o, \widehat{X}_u; \boldsymbol{\beta}\right) + e^{\text{2SRI}}$$

where e^{2SRI} denotes the regression-error term. Terza, Basu, and Rathouz (2008) call this method 2SRI.

To detail the asymptotic covariance matrix of this 2SRI estimator, I cast it in the framework of the generic 2SOE discussed above. In this case, the main components of the first- and second-stage objective functions are, respectively,

$$q_1(\boldsymbol{\alpha}, \mathbf{V}_{1i}) = -\left\{X_{pi} - r\left(\mathbf{W}_i, \boldsymbol{\alpha}\right)\right\}^2$$

and

$$q_2\left(\widehat{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \mathbf{V}_{2i}\right) = -\left\{Y - \mu\left(X_p, \mathbf{X}_o, \widehat{X}_u; \boldsymbol{\beta}\right)\right\}^2$$

where $\mathbf{V}_{1i} = [X_{pi} \;\; \mathbf{W}_i]$ and $\mathbf{V}_{2i} = [Y_i \;\; X_{pi} \;\; \mathbf{W}_i]$. This version of the 2SRI estimator implements NLS in its second stage. Therefore, expressions (19) through (23) are relevant for calculating the correct asymptotic standard errors of $\widehat{\boldsymbol{\omega}} = [\widehat{\boldsymbol{\alpha}}' \;\; \widehat{\boldsymbol{\beta}}']$ (the 2SRI estimator of $\boldsymbol{\omega} = [\boldsymbol{\alpha}' \;\; \boldsymbol{\beta}']$). In particular, if conventional t testing of the elements of $\boldsymbol{\beta}$ is the objective, then one may focus on the requisite analytics and coding of (25).

Note that MLE can be implemented in either of the stages of the 2SRI method. For MLE to be implemented in the first stage, the auxiliary regression in (29) must be replaced by an assumption that specifies a known form for the conditional density of $(X_p|\mathbf{W})$, say, $g(X_p|\mathbf{W}; \boldsymbol{\alpha})$. Of course, such an assumption would imply a formulation for the relevant conditional mean $E(X_p|\mathbf{W})$, say, $r(\mathbf{W}, \boldsymbol{\alpha})$. Therefore, in this case, the first stage of the estimator would consist of maximizing (1) with $q_1(\boldsymbol{\alpha}, \mathbf{V}_{1i})$ replaced by $\ln\{g(X_{pi}|\mathbf{W}_i; \boldsymbol{\alpha})\}$ and computing the residuals as in (30). For MLE to be implemented in the second stage, the outcome regression in (28) must be replaced by an assumption that specifies a known form for the conditional density of $(Y|X_p, \mathbf{X}_o, X_u)$, say, $f(Y|X_p, \mathbf{X}_o, X_u; \boldsymbol{\beta})$. The second stage of the estimator would then consist of maximizing (2) with $q_2(\widehat{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \mathbf{V}_{2i})$ replaced by $\ln[f(Y_i|X_{pi}, \mathbf{X}_{oi}, \widehat{X}_{ui}; \boldsymbol{\beta})]$. To obtain the correct asymptotic covariance matrix, we would use the expressions in (3) through (18).

4 An example: Smoking and infant birthweight—Testing for endogeneity

Here we revisit the regression model of Mullahy (1997) in which

$$\begin{aligned} Y &= \text{infant birthweight in lbs.} \\ X_p &= \text{number of cigarettes smoked per day during pregnancy} \\ \mathbf{X}_o &= [\texttt{PARITY WHITE MALE}] \\ \mathbf{W}^+ &= [\texttt{EDFATHER EDMOTHER FAMINCOME CIGTAX88}] \end{aligned}$$

where

Variable	Description
PARITY	birth order
WHITE	1 if white, 0 otherwise
MALE	1 if male, 0 otherwise
EDFATHER	paternal schooling in years
EDMOTHER	maternal schooling in years
FAMINCOME	family income
CIGTAX88	cigarette tax

The relevant outcome regression in Mullahy's (1997) model can be written as[5]

$$E\left(Y|X_p,\mathbf{X}_o,X_u\right)=\exp\left(X_p\beta_p+\mathbf{X}_o\boldsymbol{\beta}_o+X_u\beta_u\right) \tag{32}$$

In the original study, the model was fit via a generalized method of moment (GMM) procedure that does not require specification of an auxiliary regression for X_p. However, this GMM method does not permit identification and estimation of β_u. Note that this precludes a direct test of endogeneity because under the assumed regression specification in (32), the null hypothesis that X_p is exogenous is tantamount to $\beta_u=0$. However, such a test is supported in the 2SRI estimation framework. To this end, we specify the following version of (29),

$$X_p=\exp\left(\mathbf{W}\boldsymbol{\alpha}\right)+X_u$$

and, using Mullahy's data, apply the 2SRI method discussed in the previous section with (32) as the relevant version of (28). In this illustration, we applied NLS in both of the stages of 2SRI. The first- and second-stage 2SRI parameter estimates ($\widehat{\boldsymbol{\alpha}}$ and $\widehat{\boldsymbol{\beta}}=[\widehat{\beta}_p\quad\widehat{\boldsymbol{\beta}}_o'\quad\widehat{\beta}_u]$, respectively) were obtained via the Stata `glm` command with the `family(gaussian)`, `link(log)`, and `vce(robust)` options. We focus here on asymptotic t testing of the conventional null hypotheses for the individual elements of $\boldsymbol{\beta}$ (that is, $H_o\colon\beta_k=0$, where β_k denotes the kth element of $\boldsymbol{\beta}$). Therefore, the expression for $\widehat{\mathbf{D}}_{22}^{\dagger}$ in (27) is relevant with

$$\begin{aligned}J(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{Z}_2)&=\mu\left(X_p,\mathbf{X}_o,X_p-r\left(\mathbf{W},\boldsymbol{\alpha}\right);\boldsymbol{\beta}\right)\\&=\exp\left[X_p\beta_p+\mathbf{X}_o\boldsymbol{\beta}_o+\left\{X_p-\exp\left(\mathbf{W}\boldsymbol{\alpha}\right)\right\}\beta_u\right]\end{aligned} \tag{33}$$

where $\mathbf{Z}_2=[X_p\quad\mathbf{X}_o\quad\mathbf{W}^{+}]$. After each of the 2SRI estimation stages, we saved the parameter vectors ($\widehat{\boldsymbol{\alpha}}$ and $\widehat{\boldsymbol{\beta}}$) and their corresponding packaged covariance matrix estimators [$\widehat{\mathbf{AVAR}}^{*}(\widehat{\boldsymbol{\alpha}})$ and $\widehat{\mathbf{AVAR}}^{*}(\widehat{\boldsymbol{\beta}})$] in Mata. Moreover, using (33), we get

$$\boldsymbol{\nabla}_{\alpha}\mathbf{J}\left(\widehat{\boldsymbol{\alpha}},\widehat{\boldsymbol{\beta}},\mathbf{Z}_{2i}\right)=-2\widehat{\beta}_u\exp\left(\mathbf{X}_i\widehat{\boldsymbol{\beta}}\right)\exp\left(\mathbf{W}_i\widehat{\boldsymbol{\alpha}}\right)\mathbf{W}_i$$

5. Equation (32) is written in my notation, not Mullahy's. He does not explicitly specify the model in terms of the unobservable confounder X_u. Nevertheless, (32) is substantively identical to Mullahy's model (see Terza [2006]).

and

$$\boldsymbol{\nabla}_{\beta}\mathbf{J}\left(\widehat{\boldsymbol{\alpha}},\widehat{\boldsymbol{\beta}},\mathbf{Z}_{2i}\right)=2\exp\left(\mathbf{X}_{i}\widehat{\boldsymbol{\beta}}\right)\mathbf{X}_{i}$$

where $\mathbf{X}_i = [X_{pi} \;\; \mathbf{X}_{oi} \;\; \widehat{X}_{ui}]$. The requisite Mata code for calculating the elements of $\widehat{\mathbf{D}}_{22}^{\dagger}$ in (27) is[6]

$$\begin{aligned}
\exp\left(\mathbf{W}_{i}\widehat{\boldsymbol{\alpha}}\right) &: \texttt{expWalpha}\ \text{(saved using } \texttt{predict} \text{ for 1st-stage glm)}\\
\exp\left(\mathbf{X}_{i}\widehat{\boldsymbol{\beta}}\right) &: \texttt{expXbeta}\ \text{(saved using } \texttt{predict} \text{ for 2nd-stage glm)}\\
\boldsymbol{\nabla}_{\alpha}\mathbf{J}\left(\widehat{\boldsymbol{\alpha}},\widehat{\boldsymbol{\beta}},\mathbf{Z}_{2i}\right) &: \texttt{paJ=-bxu:*expXbeta:*expWalpha:*W}\\
\boldsymbol{\nabla}_{\beta}\mathbf{J}\left(\widehat{\boldsymbol{\alpha}},\widehat{\boldsymbol{\beta}},\mathbf{Z}_{2i}\right) &: \texttt{pbJ=expXbeta:*X}\\
\widehat{\mathbf{B}}_{\beta\alpha} &: \texttt{Bba=pbJ'*paJ}\\
\widehat{\mathbf{B}}_{\beta\beta} &: \texttt{Bbb=pbJ'*pbJ}\\
\widehat{\mathbf{D}}_{22}^{\dagger} &: \texttt{D22= invsym(Bbb)*Bba*covalpha*Bba'*invsym(Bbb)+covbeta}
\end{aligned}$$

The first- and second-stage 2SRI results are given in tables 1 and 2, respectively. Table 2 also displays Mullahy's GMM estimates and, as a baseline, reports the simple NLS estimates that ignore potential endogeneity. To indicate the strength of the instrumental variables (that is, the elements of $\mathbf{W}^{+}$), I conducted a Wald test of their joint significance. The value of the chi-squared test statistic is 49.33, and the null is roundly rejected at any reasonable level of significance. The second-stage 2SRI estimates shown in table 2 are virtually identical to the GMM estimates, but the former, unlike the latter, provide a direct test of the endogeneity of the cigarette consumption variable via the asymptotic t statistic for the coefficient of X_u. According to this test, the exogeneity null hypothesis is rejected at nearly the 1% significance level. To get a sense of the bias from neglecting to consider the two-stage nature of the estimator in calculating the asymptotic standard errors, I also display in table 2 the packaged second-stage `glm` t statistics as reported in the Stata output. The mean absolute bias across the uncorrected asymptotic t statistics given in column 3 of table 2 for the four regressors and X_u is nearly 9%.

6. See *Appendix D* for the complete do-file.

Table 1. 2SRI first-stage estimates

Variable	Estimate	Asymptotic t statistic
PARITY	0.04	1.14
WHITE	0.28	0.86
MALE	0.15	−1.84
EDFATHER	−0.03	−3.34
EDMOTHER	−0.10	−2.65
FAMINCOM	−0.02	1.44
CIGTAX	0.02	5.60
Constant	2.04	0.56

$n = 1388$

Table 2. 2SRI second-stage, GMM, and NLS estimates

		2SRI		GMM		NLS	
Variable	Estimate	Correct asymptotic t statistic	Uncorrected asymptotic t statistic	Estimate	Asymptotic t statistic	Estimate	Asymptotic t statistic
CIGS	−0.01	−3.68	−4.08	−0.01	−3.46	0.00	−5.62
PARITY	0.02	3.18	3.41	0.02	3.33	0.01	2.99
WHITE	0.05	4.22	4.55	0.05	4.44	0.06	4.75
MALE	0.03	3.13	3.35	0.03	2.95	0.03	2.90
X_u	0.01	2.56	2.83	–	–	–	–
Constant	1.95	117.64	123.74	1.94	121.71	1.93	133.70

$n = 1388$

5 Summary

To aid applied researchers seeking to implement 2SOE, I offer simplified versions of the textbook formulations of the correct asymptotic standard errors for cases in which the second-stage method is MLE or NLS. My results apply to, and thus simplify, the standard-error formulations offered by Hardin (2002) and Hole (2006) for the special case in which both stages of the 2SOE are MLEs (MLE-MLE-2SOE). As an illustration, I detail 2SRI estimation of a nonlinear model with an endogenous regressor in which the second-stage estimator is NLS.

6 Acknowledgments

This research was supported by a grant from the Agency for Healthcare Research and Quality (R01 HS017434-01). This article was presented at the 2012 Stata Conference in San Diego, CA, July 26–27, 2012, and as part of the keynote presentation at the Mexican Stata Users Group Meeting in Mexico City, November 14, 2014.

7 References

Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.

Greene, W. H. 2012. *Econometric Analysis*. 7th ed. Upper Saddle River, NJ: Prentice Hall.

Hardin, J. W. 2002. The robust variance estimator for two-stage models. *Stata Journal* 2: 253–266.

Hole, A. R. 2006. Calculating Murphy–Topel variance estimates in Stata: A simplified procedure. *Stata Journal* 6: 521–529.

Mullahy, J. 1997. Instrumental-variable estimation of count data models: Applications to models of cigarette smoking behavior. *Review of Economics and Statistics* 79: 586–593.

Murphy, K. M., and R. H. Topel. 1985. Estimation and inference in two-step econometric models. *Journal of Business and Economic Statistics* 3: 370–379.

Newey, W. K., and D. McFadden. 1994. Large sample estimation and hypothesis testing. In *Handbook of Econometrics*, ed. R. F. Engle and D. L. McFadden, vol. 4, 2111–2245. Amsterdam: Elsevier.

Terza, J. V. 2006. Estimation of policy effects using parametric nonlinear models: A contextual critique of the generalized method of moments. *Health Services and Outcomes Research Methodology* 6: 177–198.

Terza, J. V., A. Basu, and P. J. Rathouz. 2008. Two-stage residual inclusion estimation: Addressing endogeneity in health econometric modeling. *Journal of Health Economics* 27: 531–543.

White, H. 1994. *Estimation, Inference and Specification Analysis*. New York: Cambridge University Press.

Wooldridge, J. M. 2010. *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press.

About the author

Joseph Terza is a health economist and econometrician in the Department of Economics at Indiana University–Purdue University Indianapolis. His research focuses on the development

and application of methods for estimating qualitative and limited dependent variable models with endogeneity. Two of his methods have been implemented as Stata commands. He was keynote speaker at the Stata Users Group meeting in Mexico City in November 2014.

Appendix A: Establishment of (3) and (4) in the text

When the second stage is MLE so that $q_2(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{V}_2)$ is a log likelihood, the true value of the parameter vector $\boldsymbol{\omega}_0' = [\boldsymbol{\alpha}_0' \;\; \boldsymbol{\beta}_0']$, the unconditional information matrix equality (see expression 13.27 on page 479 of Wooldridge [2010]), yields

$$\mathbf{E}\left(\boldsymbol{\nabla}_{\omega\omega}\mathbf{q}_2\right) = -\mathbf{E}\left(\boldsymbol{\nabla}_{\omega}\mathbf{q}_2'\boldsymbol{\nabla}_{\omega}\mathbf{q}_2\right)$$

or

$$\begin{bmatrix} \mathbf{E}\left(\boldsymbol{\nabla}_{\alpha\alpha}\mathbf{q}_2\right) & \mathbf{E}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right)' \\ \mathbf{E}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right) & \mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right) \end{bmatrix} = - \begin{bmatrix} \mathbf{E}\left(\boldsymbol{\nabla}_{\alpha}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_2\right) & \mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_2\right)' \\ \mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_2\right) & \mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\beta}\mathbf{q}_2\right) \end{bmatrix}$$

Therefore,

$$\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta}\mathbf{q}_2\right) = -\mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\beta}\mathbf{q}_2\right)$$

and

$$\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\alpha}\mathbf{q}_2\right) = -\mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_2\right)$$

so (3) and (4) in the text hold.

Appendix B: Establishment of (5) in the text

When the second stage is NLS,

$$q_2(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{V}_2) = -\{Y - J(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{Z}_2)\}^2$$

where $J(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{Z}_2)$ denotes the relevant nonlinear regression function, $\mathbf{V}_{2i} = [Y_i \;\; \mathbf{Z}_{2i}]$,

$$J(\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0, \mathbf{Z}_2) = \mathbf{E}\left(Y|\mathbf{Z}_2\right) \tag{34}$$

and $\boldsymbol{\omega}_0' = [\boldsymbol{\alpha}_0' \;\; \boldsymbol{\beta}_0']$ denotes the true value of the parameter vector. Moreover, we can write

$$\begin{aligned} \mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_1\right) &= \mathbf{E}\left(\mathbf{E}\left(\boldsymbol{\nabla}_{\beta}\mathbf{q}_2\left(\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0, \mathbf{V}_2\right)|\mathbf{Z}_2\right)'\boldsymbol{\nabla}_{\alpha}\mathbf{q}_1(\boldsymbol{\delta}_0, \mathbf{V}_1)\right) \\ &= 2\mathbf{E}\left\{\left(\mathbf{E}\left[\{Y - J\left(\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0, \mathbf{Z}_2\right)\}|\mathbf{Z}_2\right]\boldsymbol{\nabla}_{\beta}\mathbf{J}(\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0, \mathbf{Z}_2)\right)'\right. \\ &\quad \left.\boldsymbol{\nabla}_{\alpha}\mathbf{q}_1(\boldsymbol{\alpha}_0, \mathbf{V}_1)\right\} \end{aligned}$$

because $q_1(\boldsymbol{\alpha}_0, \mathbf{V}_1)$ is not a function of Y and $\mathbf{Z}_2$ is a subvector of $\mathbf{V}_1$. But using (34), we have that

$$\mathbf{E}\left[\{Y - J\left(\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0, \mathbf{Z}_2\right)\}|\mathbf{Z}_2\right] = 0$$

Therefore, (5) in the text holds.

When the second stage is MLE, we have that

$$q_2(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{V}_{2i}) = \ln f(Y|\mathbf{Z}_2; \boldsymbol{\alpha}, \boldsymbol{\beta})$$

where $f(Y|\mathbf{Z}_2; \boldsymbol{\alpha}_0, \boldsymbol{\beta}_0)$ denotes the true conditional density of Y given $\mathbf{Z}_2$. Accordingly, we can write

$$\mathbf{E}\left(\boldsymbol{\nabla}_\beta \mathbf{q}_2' \boldsymbol{\nabla}_\alpha \mathbf{q}_1\right) = \mathbf{E}\left[\mathbf{E}\left\{\boldsymbol{\nabla}_\beta \mathbf{f}(Y|\mathbf{Z}_2; \boldsymbol{\alpha}_0, \boldsymbol{\beta}_0)|\mathbf{Z}_2\right\}' \boldsymbol{\nabla}_\alpha \mathbf{q}_1(\boldsymbol{\alpha}_0, \mathbf{V}_1)\right]$$

Now, using (13.20) on page 477 of Wooldridge (2010), we have that

$$\mathbf{E}\left\{\boldsymbol{\nabla}_\beta \mathbf{f}\left(Y|\mathbf{Z}_2; \boldsymbol{\alpha}_0, \boldsymbol{\beta}_0\right)|\mathbf{Z}_2\right\} = \mathbf{0}$$

because it is the score of the second-stage log-likelihood function. Therefore, (5) in the text holds.

Appendix C: Derivation of consistent estimators (19) and (20) in the text

When the second stage is NLS

$$q_2(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{V}_2) = -\{Y - J(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{Z}_2)\}^2 \tag{35}$$

where $J(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{Z}_2)$ denotes the relevant nonlinear regression function and $\mathbf{V}_{2i} = [Y_i \;\; \mathbf{Z}_{2i}]$. From (35), we get

$$\boldsymbol{\nabla}_\beta \mathbf{q}_2 = 2e\boldsymbol{\nabla}_\beta \mathbf{J}$$

so

$$\boldsymbol{\nabla}_{\beta\alpha} \mathbf{q}_2 = 2(\boldsymbol{\nabla}_\alpha e \boldsymbol{\nabla}_\beta \mathbf{J} + e\boldsymbol{\nabla}_{\beta\alpha} \mathbf{J})$$

and

$$\boldsymbol{\nabla}_{\beta\beta} \mathbf{q}_2 = 2(\boldsymbol{\nabla}_\beta e \boldsymbol{\nabla}_\beta \mathbf{J} + e\boldsymbol{\nabla}_{\beta\beta} \mathbf{J})$$

where $e = (Y - J(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{Z}_2))$ and J is shorthand notation for $J(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{Z}_2)$. Now,

$$\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\alpha} \mathbf{q}_2\right) = 2\mathbf{E}\left\{\boldsymbol{\nabla}_\beta \mathbf{J}' \boldsymbol{\nabla}_\alpha e + \mathbf{E}\left(e|\mathbf{Z}_2\right) \boldsymbol{\nabla}_{\beta\alpha} \mathbf{J}\right\}$$

where

$$\boldsymbol{\nabla}_\alpha e = -\boldsymbol{\nabla}_\alpha \mathbf{J}$$

and at the true value of $\boldsymbol{\omega}$ (say, $\boldsymbol{\omega}_0' = [\boldsymbol{\alpha}_0' \;\; \boldsymbol{\beta}_0']$),

$$E\left(e|\mathbf{Z}_2\right) = 0$$

Therefore,

$$\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\alpha} \mathbf{q}_2\right) = -2\mathbf{E}\left(\boldsymbol{\nabla}_\beta \mathbf{J}' \boldsymbol{\nabla}_\alpha \mathbf{J}\right) \tag{36}$$

It can similarly be shown that

$$\mathbf{E}\left(\boldsymbol{\nabla}_{\beta\beta} \mathbf{q}_2\right) = -2\mathbf{E}\left(\boldsymbol{\nabla}_\beta \mathbf{J}' \boldsymbol{\nabla}_\beta \mathbf{J}\right) \tag{37}$$

The consistent estimators in (17) and (18) are the sample analogs to (36) and (37).

Appendix D: The do-file for 2SRI estimation of the birthweight model

```
/***********************************************
** Purpose: Mullahy (1997) birthweight model.
** Estimation of the model using the 2SRI and
** GMM approaches to account for endogeneity
** of smoking.
***********************************************/

/***********************************************
** Read in the data.
***********************************************/
use mullahy-birthweight-data.dta

/***********************************************
** Transform birthweight ounces to pounds.
***********************************************/
generate BIRTHWTLB=BIRTHWT/16

***********************************************
** Compute and display descriptive statistics.
***********************************************/
summarize

/***********************************************
** Simple NLS estimation.
***********************************************/
glm BIRTHWTLB CIGSPREG PARITY WHITE MALE, ///
   family(gaussian) link(log) vce(robust)

/***********************************************
** 2SRI estimation begins here.
***********************************************/
/***********************************************
** First-stage NLS estimation of the auxiliary
** exponential regression (via GLM). The
** exponential regression for the 2SRI first
** stage is   xp = exp(w*a)+ xu.
** Conduct Wald test of joint significance of
** the instruments.
** Save xuhat and the predicted values from the
** regression.
***********************************************/
glm CIGSPREG PARITY WHITE MALE EDFATHER EDMOTHER FAMINCOM CIGTAX88, ///
   family(gaussian) link(log) vce(robust)
test (EDFATHER = 0) (EDMOTHER = 0) (FAMINCOM = 0) (CIGTAX88 = 0)
predict xuhat, response
predict expWalpha, mu

/***********************************************
** Load the coefficient vector and covariance
** matrix from first-stage GLM into Mata
** matrices.
***********************************************/
mata: alpha=st_matrix("e(b)")´
mata: covalpha=st_matrix("e(V)")

/***********************************************
```

```
** Apply GLM for the 2SRI second stage.
************************************************/
glm BIRTHWTLB CIGSPREG PARITY WHITE MALE xuhat, ///
   family(gaussian) link(log) vce(robust)
predict expXbeta, mu

/************************************************
** Load the coefficient vector and covariance
** matrix from second-stage GLM into Mata
** matrices.
************************************************/
mata: beta=st_matrix("e(b)")´
mata: covbeta=st_matrix("e(V)")

/************************************************
** Do GMM estimation.
************************************************/
gmm (BIRTHWTLB/exp(xb:CIGSPREG PARITY WHITE MALE + b0)-1), ///
   instruments(PARITY WHITE MALE EDFATHER EDMOTHER FAMINCOM CIGTAX88)

/************************************************
** Use the Stata "putmata" command to send
** Stata data variables into Mata vectors.
************************************************/
putmata CIGSPREG BIRTHWTLB PARITY WHITE MALE EDFATHER ///
   EDMOTHER FAMINCOM CIGTAX88 xuhat expWalpha expXbeta

/************************************************
** Mata start-up.
************************************************/
mata:

/************************************************
** Make the estimate of betau explicit.
************************************************/
bxu=beta[5]

/************************************************
** Load the W-variables for the RHS of the
** first-stage GLM equation into a Mata matrix.
** -- Don´t include the policy variable or xuhat.
** -- Do include the IVs.
** -- Do include a constant term.
************************************************/
W=PARITY, WHITE, MALE, EDFATHER, EDMOTHER, FAMINCOM, ///
   CIGTAX88, J(rows(PARITY),1,1)

/************************************************
** Load the X-variables for the RHS of the
** second-stage GLM equation into a Mata matrix.
************************************************/
X=CIGSPREG,PARITY, WHITE, MALE, xuhat,J(rows(CIGSPREG),1,1)

/************************************************
** Compute the Xbeta index by multiplying the
** matrix of exogenous variables (X) by the
** coefficient vectors.
************************************************/
Xbeta=X*beta
```

```
/***********************************************
** Compute the asymptotic covariance matrix of
** the 2SRI estimate of beta.
***********************************************/
paJ=-bxu:*expXbeta:*expWalpha:*W
pbJ=expXbeta:*X
Bba=pbJ´*paJ
Bbb=pbJ´*pbJ
d22=invsym(Bbb)*Bba*covalpha*Bba´*invsym(Bbb)+covbeta

/***********************************************
**  2SRI estimate of beta with correct
**  asymptotic t statistic
***********************************************/
/***********************************************
**  2SRI second-stage results for beta
***********************************************/
/***********************************************
**  First, the uncorrected t statistics
***********************************************/
tstatwrong=beta:/sqrt(diagonal(covbeta))

/***********************************************
**  Now, the corrected t statistics
***********************************************/
tstatbeta=beta:/sqrt(diagonal(d22))

/***********************************************
**  Display the results.
***********************************************/
"Second-Stage Estimates, True Asy t-stats and p-values"
pvalues=2:*(1:-normal(abs(tstatbeta)))
header="variable","estimate","t-stat","wrong-t-stat","p-value" \
"","","","",""
varnames="CIGSPREG", "PARITY", "WHITE", "MALE", "xuhat","constant"
results=beta,tstatbeta,tstatwrong,pvalues
resview=strofreal(results)
header \ (varnames´,resview)

/***********************************************
** Close Mata.
***********************************************/
end
```

The Stata Journal (2016)
16, Number 2, pp. 386–402

igmobil: A command for intergenerational mobility analysis in Stata

Marco Savegnago
Bank of Italy
Rome, Italy
marco.savegnago@bancaditalia.it

Abstract. In this article, I describe a new command, `igmobil`, that computes up to 20 intergenerational mobility (IGM) indices for continuous (that is, income or years of education) or discrete (that is, educational or occupational level) variables. I consider three classes of IGM indices: 1) single-stage indices, 2) indices derived from a transition matrix between parents' and children's socioeconomic status, and 3) indices based on inequality measures. Users may add a fourth class to specify any possible IGM index not included in `igmobil`. Standard errors and confidence intervals are calculated using a bootstrap procedure. Users can customize many aspects of the program output, including the type and dimension of the transition matrix, the parameters for some IGM indices (like the ones involving generalized entropy measures and the Atkinson index), and how standard errors and confidence intervals are calculated.

Keywords: st0437, igmobil, intergenerational mobility, IGM, bootstrap

1 Introduction

Intergenerational mobility (IGM) refers to the extent to which the advantages and disadvantages of individuals (according to dimensions such as income, wealth, or education) are transmitted across generations (Black and Devereux 2011). A key aspect is that IGM is a complex concept and may mean different things to different people. Fields (2008) identifies six different concepts of income mobility—namely, time independence, positional movement, share movement, nondirectional income movement, directional income movement, and equalizer of long-term incomes—and each concept can be quantified by a specific set of indices. For example, share movement mobility occurs when an individual's income increases with respect to the population mean.[1] A possible index for this example would be $\mathcal{M}(X_i, Y_i) = N^{-1} \sum_{i=1}^{n} |Y_i/\mu_Y - X_i/\mu_X|$, where Y_i and X_i denote, respectively, a child's and his or her parent's income for family i (see section 2). A natural consequence in this case is that there is no consensus on how IGM should be measured, so many indices are available to an applied researcher.[2] Therefore, because the complexity of IGM cannot be captured by a unique index, Fields and Ok (1999a) suggest using different measures.

1. Fields (2008) notes that a child can experience share movement even if he or she has the same income as his or her parent.
2. See Fields and Ok (1999a), Checchi and Dardanoni (2002), Black and Devereux (2011), and Jäntti and Jenkins (2013).

`igmobil` computes point estimates and inferential measures for 19 IGM indices commonly used in the applied research. These 19 indices are appropriate for measuring income mobility, although only some of them are suitable for analyzing mobility according to other dimensions, such as years of education or occupational level. Users can add a 20th index to this list by specifying their own user-written program, an example of which is provided in section 4.6.

`igmobil` also computes standard errors and confidence intervals using an embedded bootstrap procedure.

2 Intergenerational mobility indices

Let the vector (Y_i, X_i) describe the socioeconomic status (SES) of a child and his or her parent for family i. We are interested in the extent to which the child's SES, Y_i, depends on the parent's SES, X_i.[3] From a practical point of view, the abstract and multifaceted notion of SES must be proxied by an observable variable, which is typically chosen among income, wealth, health, education, occupational prestige, and the like. For simplicity, I will assume that the vectors $(\mathbf{Y}, \mathbf{X})$ contain information on permanent income. In the final paragraph of this section, I will briefly discuss how analysis can be carried out on other dimensions of SES other than income.

An intergenerational mobility index, $\mathcal{M}(Y, X)$, is any function $\mathcal{M} : \mathbb{R}^{2n} \to \mathbb{R}$, which maps the vectors of incomes $(\mathbf{Y}, \mathbf{X})$ into a scalar. Following the distinctions made in Cowell and Schluter (1998) and Checchi and Dardanoni (2002), I divide the IGM indices computed by `igmobil` into the following three broad classes: 1) single-stage indices, which are computed directly on microdata; 2) indices based on a transition matrix, which require discrete or discretized data; and 3) inequality reduction indices, which are based on inequality measures (such as the Gini coefficient) computed on the cross-sectional distribution of income for both generations. Users may add a fourth class to specify any possible IGM index not included in `igmobil`.

Table 1 describes the IGM indices estimated by the program. For each index, I report the bibliographic reference where that index was either first proposed or described in an applied work.

3. Alternatively, one can study the evolution of SES for the same individuals at different points in time. What follows applies to both intergenerational and intragenerational mobility, even if I refer to the former only.

Table 1. Intergenerational mobility indices estimated by `igmobil`

Index	Alias	Formula	Reference
		Single-stage indices	
$\mathcal{M}_1$	Abs. difference	$n^{-1}\sum\lvert X_i - Y_i\rvert$	Fields and Ok (1996)
$\mathcal{M}_2$	Sq. difference	$n^{-1}\sum(X_i - Y_i)^2$	Checchi and Dardanoni (2002)
$\mathcal{M}_3$	Fields and Ok	$n^{-1}\sum\lvert\ln X_i - \ln Y_i\rvert$	Fields and Ok (1999b)
$\mathcal{M}_4$	Share	$n^{-1}\sum(X_i/\mu_X - Y_i/\mu_Y)^2$	Fields (2008)
$\mathcal{M}_5$	Hart	$1 - \text{Corr}(\ln Y_i, \ln X_i)$	Hart (1981)
$\mathcal{M}_6$	Spearman	$1 - \text{Spearman}(\ln Y_i, \ln X_i)$	Black and Devereux (2011)
$\mathcal{M}_7$	Abs. CDF	$n^{-1}\sum\lvert F_Y(Y_i) - F_X(X_i)\rvert$	Checchi and Dardanoni (2002)
$\mathcal{M}_8$	Sq. CDF	$n^{-1}\sum\{F_Y(Y_i) - F_X(X_i)\}^2$	Checchi and Dardanoni (2002)
$\mathcal{M}_9$	1-OLS(levels)	$1 - \text{OLS}(Y_i, X_i)$	Black and Devereux (2011)
$\mathcal{M}_{10}$	1-OLS(logs)	$1 - \text{OLS}(\ln Y_i, \ln X_i)$	Black and Devereux (2011)
		Indices based on $\mathbf{P}_{K\times K}$ *transition matrix*	
$\mathcal{M}_{11}$	Prais (trace)	$(K-1)^{-1}\{K - \text{trace}(\mathbf{P})\}$	Shorrocks (1978b)
$\mathcal{M}_{12}$	Bartholomew	$\{K(K-1)\}^{-1}\sum_i\sum_j p_{ij}\lvert i-j\rvert$	Bartholomew (1973)
$\mathcal{M}_{13}$	Eigenvalue2	$1 - \lvert\text{2nd largest eigenvalue}\rvert$	Sommers and Conlisk (1979)
$\mathcal{M}_{14}$	Determinant	$1 - \lvert\det(\mathbf{P})\rvert$	Shorrocks (1978b)
		Indices based on inequality reduction	
$\mathcal{M}_{15}$	$\mathcal{S}$ or $\mathcal{F}$ – Gini	See (1), (2)	Fields (2010); Shorrocks (1978a)
$\mathcal{M}_{16}$	$\mathcal{S}$ or $\mathcal{F}$ – GE(a_1)	See (1), (2), and *Appendix*	Fields (2010); Shorrocks (1978a)
$\mathcal{M}_{17}$	$\mathcal{S}$ or $\mathcal{F}$ – GE(a_2)	See (1), (2), and *Appendix*	Fields (2010); Shorrocks (1978a)
$\mathcal{M}_{18}$	$\mathcal{S}$ or $\mathcal{F}$ – Atk(ϵ_1)	See (1), (2), and *Appendix*	Fields (2010); Shorrocks (1978a)
$\mathcal{M}_{19}$	$\mathcal{S}$ or $\mathcal{F}$ – Atk(ϵ_2)	See (1), (2), and *Appendix*	Fields (2010); Shorrocks (1978a)
		Index based on user-written program	
$\mathcal{M}_{20}$	user written (ex.)	See sections 2 and 4.6	

`igmobil` estimates the following classes of indices:

Class 1: Single-stage indices. The defining characteristic of indices belonging to this class is that the final IGM index is an aggregation of mobility that occurs between the families in the population (see Checchi and Dardanoni [2002]). Many well-known indices belong to this class, including the Fields and Ok index ($\mathcal{M}_3$; Fields and Ok [1999b]) and the indices based on the Pearson correlation ($\mathcal{M}_5$), the Spearman correlation ($\mathcal{M}_6$), and the intergenerational elasticity ($\mathcal{M}_{10}$, which is one minus the ordinary least-squares coefficient in a regression of child log-income on parent log-income). The other indices in this class are the average absolute difference ($\mathcal{M}_1$), the average squared difference ($\mathcal{M}_2$),

the share index ($\mathcal{M}_4$, introduced in section 1), the average absolute ($\mathcal{M}_7$) and squared ($\mathcal{M}_8$) difference of the individuals' empirical cumulative density functions (CDFs), and the ordinary least-squares coefficient in a regression of child income on parent income ($\mathcal{M}_9$, which uses income levels instead of logs as in the intergenerational elasticity, $\mathcal{M}_{10}$).

Class 2: Indices based on a transition matrix. These indices are functional of the transition matrix $\mathbf{P}_{K\times K}$ between the parent's income level and the child's income level. The generic element p_{jk} represents the probability that the child's income falls in the kth class given that the parent's income falls in the jth class. Income levels can be either absolute (a size transition matrix) or set on the basis of quantiles of the marginal distributions of Y_i and X_i (quantile or fractile transition matrix). Indices derived from a transition matrix combine the elements on the main diagonal (as in the Shorrocks and Prais index, $\mathcal{M}_{11}$; see Shorrocks [1978b]); they consider the average "jump" of income classes (as in the Bartholomew index, $\mathcal{M}_{12}$; see Bartholomew [1973]); and they account for the second-largest eigenvalues ($\mathcal{M}_{13}$; see Sommers and Conlisk [1979]) or the determinant of the matrix itself ($\mathcal{M}_{14}$; see Shorrocks [1978b]). By default, `igmobil` computes 5×5 quantile matrices, although one can specify the desired number of quantiles or can switch to a size transition matrix (in this case, however, data must be discrete).

Class 3: Inequality reduction indices. This class captures the notion of mobility as a long-term income equalizer. The intuition is that, in the case of upward mobility, inequality in the average father–son income (a measure of "long-term" or "dynasty" income) will be smaller than inequality in only the father's income (a measure of "short-term" income). Let $Z = (Y+X)/2$ be the "long-term income" and $\mathcal{I}(\cdot)$ a cross-sectional measure of inequality (such as the Gini index). Then, consider the Shorrocks (1978a) $\mathcal{S}(\mathcal{I}, Y, X)$ and the Fields (2010) $\mathcal{F}(\mathcal{I}, Y, X)$ families as follows:

$$\text{Shorrocks: } \mathcal{S}(\mathcal{I}, Y, X) = 1 - \frac{\mu_Z\ \mathcal{I}(Z)}{\mu_X\ \mathcal{I}(X) + \mu_Y\ \mathcal{I}(Y)} \tag{1}$$

$$\text{Fields: } \mathcal{F}(\mathcal{I}, Y, X) = 1 - \frac{\mathcal{I}\,(Z)}{\mathcal{I}(X)} \tag{2}$$

The families $\mathcal{S}(\cdot)$ and $\mathcal{F}(\cdot)$ are conceptually similar: the only difference is that the Shorrocks family's indices do not distinguish whether the income dynamics are equalizing or disequalizing, while the Fields family's indices do. Moreover, the Fields family is very close to the Chakravarty, Dutta, and Weymark (1985) index, although these indices differ in their normative implications (see Fields [2010] for a detailed discussion on this point).

As cross-sectional inequality measures, I will consider the Gini index; the generalized entropy measure, GE(a); and the Atkinson index, Atk(ϵ). I provide a brief description of

these last two inequality measures in the *Appendix*. Users can specify that the `igmobil` command uses either the $\mathcal{S}(\cdot)$ or the $\mathcal{F}(\cdot)$ family (the default is $\mathcal{F}(\cdot)$) and can specify up to two parameters each for the generalized entropy (by default, $a = 0$ and $a = 1$, for which the generalized entropy becomes the mean log deviation and the Theil index) and for the Atkinson index (by default, $\epsilon = 0.5$ and $\epsilon = 2$).

Class 4: Index from a user-written program. This option is useful if one wants to enrich the above list with another IGM index. For example, consider the following upward-mobility index suggested by Bhattacharya and Mazumder (2011):

$$\text{UP}_{\tau,s} = \Pr(r_{Y_i} - r_{X_i} > \tau \mid r_{X_i} \leq s)$$

Here r_{Y_i} and r_{X_i} are the child's and the parent's relative positions in their marginal income distribution; that is, $r_{Y_i} = F_Y(Y_i)$ and $r_{X_i} = F_X(X_i)$. Thus $\text{UP}_{\tau,s}$ is the probability that the child's rank exceeds the parent's rank by τ given that the parent's rank is below s. I provide code for estimating such an index and for passing its value to the `igmobil` command.

Final remarks and inference. In the given examples, SES of each generation is proxied by income: In this case, all IGM indices computed by `igmobil` make sense, although the specific concept of mobility that one has in mind can lead to a preference for certain indices. If we use other continuous variables (such as years of education or a continuous measure of health status) as SES indicators, then we should probably estimate only some of the single-stage indices [like the indices based on absolute and squared differences ($\mathcal{M}_1$ and $\mathcal{M}_2$), the ones based on empirical CDFs ($\mathcal{M}_7$ and $\mathcal{M}_8$), and the ordinary least-squares coefficient (on levels, as in $\mathcal{M}_9$)]. If we use purely categorical variables (such as occupational status or educational attainment), we should instead estimate only indices based on a transition matrix ($\mathcal{M}_{11}$–$\mathcal{M}_{14}$), and so on. One advantage of `igmobil` is that it can accommodate all of these needs within the same framework.

An important—and probably undervalued—aspect of empirical research on IGM regards statistical inference. `igmobil` computes some complex IGM indices (like those derived from the quantile transition matrix, where we have extra variability resulting from the estimation of quantiles) for which we might expect poor inference if based on asymptotic arguments. For this reason, I implement a bootstrap procedure in the `igmobil` command; options for this procedure can be modified by the user.

3 The igmobil command

3.1 Description

The `igmobil` command provides point estimates, standard errors, and confidence intervals for the three classes of IGM indices discussed in section 2. By default, `igmobil` assumes that data are continuous and computes 10 single-stage indices, 4 indices based on a 5×5 quantile transition matrix, and 5 indices based on inequality measures (in-

cluding the Fields class applied to the Gini index, generalized entropy measures with parameters -1 and 2, and the Atkinson index with parameters 0.5 and 2). Standard errors are computed with 50 bootstrap replications, and the 95% confidence intervals are based on normal approximation.

The user can modify the program output in the following ways:

- One can omit estimation of some (but not all) classes of IGM indices. Omission of classes can be motivated by lack of interest in that particular class or can be done to decrease computing time.
- One can change the dimension of the quantile transition matrix or switch the program to use a size transition matrix. In the latter, data must be discrete; consequently, single-stage and inequality-based indices will not be computed.
- For inequality-based indices, one can specify that the program use either the Fields or the Shorrocks class and modify the parameters of the generalized entropy measures and the Atkinson index (no more than two scalars for each index).
- Through a proper user-written program, one can include an extra IGM index (see the example in section 4.6).
- One can modify the number of bootstrap replications, the type of confidence intervals (the normal approximation, the percentile method, or the bias-corrected method), and the confidence level.

When accessing results, recall that each index has a progressive number from 1 to 20. The list of indices is reported in table 1 and in the help file. Therefore, to display the standard error of the trace index ($\mathcal{M}_{11}$), we type `display _se[i11]`.

3.2 Syntax

The syntax of `igmobil` is as follows:

`igmobil` *varname1* *varname2* [*if*] [*in*] [`, nosingle notrans noinequal`
`userwritten(`*userwrittenstr*`) classes(`#`) discrete matrix(`*matname*`)`
`family(`*familystr*`) ge(`#[#]`) atk(`#[#]`) bootstrap(`*bootstrapstr*`)`
`citype(`*citypestr*`) format(`*formatstr*`)`]

varname1 and *varname2* denote, respectively, the most and the least recent observation (that is, the child and the parent in the intergenerational setting, or Y_t and Y_{t-1} in the intragenerational setting).

3.3 Options

Main

`nosingle` specifies to not calculate single-stage indices. `nosingle` is appropriate when *varname1* and *varname2* are discrete variables (such as occupational status or income class). This is the default option when the option `discrete` is specified.

`notrans` specifies to not calculate transition-matrix indices. `notrans` is appropriate when the variables are continuous and there is no interest in calculating the transition matrices. `notrans` must not be used with the option `classes()` or `discrete`.

`noinequal` specifies to not calculate indices based on inequality measures. `noinequal` must not be used with the option `family()`, `ge()`, or `atk()`.

`userwritten(`*userwrittenstr*`)` specifies that the output include any IGM index defined in *userwrittenstr*. The program must be r-class and return the IGM index in a scalar named `r(UW)`. An example is provided in section 4.6.

Indices based on a transition matrix

`classes(#)` specifies the size of the quantile transition matrix on which transition-matrix indices are to be calculated. The default is `classes(5)`. Quantiles are computed using the `xtile` command (see [D] **pctile**). `classes(#)` can be used when *varname1* and *varname2* are continuous and the user wants to specify a quantile transition matrix with a size different from 5. `classes(#)` must not be used when variables are discrete or when the option `discrete` or `notrans` is used.

`discrete` specifies that *varname1* and *varname2* are discrete (or already discretized) variables (such as types of jobs, levels of education, or income categories). When `discrete` is used, single-stage and inequality-based indices will not be computed because we are dealing with discrete random variables. `discrete` must not be used with `classes()`.

`matrix(`*matname*`)` saves the resulting transition matrix in *matname*. If the option `notrans` is used, `matrix()` is ignored.

Indices based on inequality measures

`family(`*familystr*`)` specifies what indices will be used to compare inequality measures across generations. *familystr* can be `fields` or `shorrocks` (see Fields [2010] and Shorrocks [1978a]). The default is `family(fields)`.

`ge(#[ #])` specifies the values of the generalized entropy measure parameter. The maximum two values of `ge()` can be listed; if only one value is chosen, it is repeated twice. The default is `ge(0 1)`.

`atk(#[ #])` specifies the values of the Atkinson index parameter. The maximum two values of `atk()` can be listed; if only one value is chosen, it is repeated twice. The default is `atk(0.5 2)`.

Inference and reporting

`bootstrap(`*bootstrapstr*`)` allows the user to customize almost every aspect of the bootstrap procedure. *bootstrapstr* can be any valid option of the `bootstrap` command (see [R] **bootstrap**), including `reps()`, `strata()`, `size()`, `saving()`, `level()`, or `seed()`. The options `notable`, `nolegend`, and `nowarn` are already "built-in". The computation of the bootstrapped standard error can be avoided using the option `bootstrap(off)`.

`citype(`*citypestr*`)` specifies how confidence intervals are to be computed and displayed. *citypestr* can be `normal`, `percentile`, or `bc`, which stand, respectively, for normal approximation, percentile method, and bias-corrected confidence intervals.

`format(`*formatstr*`)` displays results accordingly; see [D] **format**.

3.4 Stored results

`igmobil` stores results in `e()`. The results stored are the same as in any `bootstrap` command.

4 Examples

4.1 Preliminary: Artificial dataset

Let's generate Y_i and X_i from a bivariate lognormal distribution with parameters $\mu_Y = \mu_X = 0$, $\sigma_Y^2 = \sigma_X^2 = 0.25$, and $\rho = 0.5$. Then, let's generate Y_i^{disc} and X_i^{disc} to simulate the case where we have discrete random variables.

```
. clear
. matrix C = (.25, .5*.25 \ .5*.25, .25)
. set seed 12345
. drawnorm u0 u1, n(2000) cov(C)                        /* normal r.v. */
(obs 2,000)
. generate son = exp(u1)                                /* lognormal r.v.*/
. generate dad = exp(u0)
. generate son_disc = irecode(u1, -1, -0.5, 0, 0.5, 1)  /*discrete r.v.*/
. generate dad_disc = irecode(u0, -1, -0.5, 0, 0.5, 1)
. drop u*
```

4.2 Basic use of igmobil

In its simplest form, `igmobil` requires two inputs: the *child* variable and the *parent* variable, both expressed in levels (that is, no logs).

```
. igmobil son dad
(running igmobil_1 on estimation sample)
Bootstrap replications (50)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
Bootstrap results                               Number of obs     =      2,000
                                                Replications      =         50

Child generation:      son = Y                  Type of variables: continuous
Parent generation:     dad = X
```

Type of indices	IGM estimate	Bootstrap Std. Err.	[95% Conf. Interv.] normal approx.	
Single-stage Indices				
(1) 1/N * sum |X - Y|	0.446	0.010	0.427	0.465
(2) 1/N * sum (X - Y)^2	0.385	0.022	0.341	0.428
(3) 1/N * sum |ln X - ln Y|	0.395	0.007	0.382	0.408
(4) 1/N * sum |X/mu(X) - Y/mu(Y)|	0.389	0.006	0.377	0.402
(5) 1 - Pearson coef. (on logs)	0.487	0.015	0.457	0.517
(6) 1 - Spearman coef. (on logs)	0.511	0.016	0.479	0.543
(7) 1/N * sum |CDF X - CDF Y|	0.229	0.004	0.221	0.237
(8) 1/N * sum (CDF X - CDF Y)^2	0.085	0.003	0.080	0.090
(9) 1 - OLS(Y,X)	0.540	0.026	0.489	0.591
(10) 1 - OLS(ln Y,ln X)	0.499	0.018	0.464	0.534
Transition matrix Indices (based on 5 quantiles)				
(11) Shorrock/Prais	0.849	0.013	0.823	0.875
(12) Bartholomew	0.269	0.005	0.259	0.279
(13) 1-Second largest eigenvalue	0.522	0.017	0.489	0.556
(14) Determinant index	1.000	0.000	1.000	1.000
Inequality related Indices				
(15) Fields - Gini	0.134	0.010	0.115	0.152
(16) Fields - GE(0)	0.254	0.016	0.223	0.286
(17) Fields - GE(1)	0.255	0.018	0.219	0.291
(18) Fields - Atkinson(.5)	0.249	0.017	0.216	0.281
(19) Fields - Atkinson(2)	0.229	0.014	0.202	0.257

The output indicates that the Bartholomew index ($\mathcal{M}_{12}$, the average "jump" of income classes) from our sample is 0.269 with a bootstrapped standard error of 0.005 and a symmetric 95% confidence interval of [0.259; 0.279] obtained with the normal approximation.

In the next example, we want our transition matrix to be based on 10 quantiles. Also, we omit the computation of single-stage and inequality-based indices.

```
. igmobil son dad, nosingle noinequal classes(10)
(running igmobil_1 on estimation sample)
Bootstrap replications (50)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
Bootstrap results                               Number of obs     =      2,000
                                                Replications      =         50

Child generation:      son = Y                  Type of variables: continuous
Parent generation:     dad = X
```

Type of indices	IGM estimate	Bootstrap Std. Err.	[95% Conf. Interv.] normal approx.	
Transition matrix Indices (based on 10 quantiles)				
(11) Shorrock/Prais	0.919	0.010	0.900	0.939
(12) Bartholomew	0.251	0.005	0.242	0.260
(13) 1-Second largest eigenvalue	0.508	0.016	0.476	0.540
(14) Determinant index	1.000	0.000	1.000	1.000

We see that our $\mathcal{M}_{11}$–$\mathcal{M}_{14}$ indices change when we double the size of the transition matrix (for example, $\mathcal{M}_{11}$ increases, but $\mathcal{M}_{12}$ decreases). This should warn us against comparing transition-matrix indices derived from transition matrices that have different numbers of classes.

Now assume that the data are already in discrete form, such as income classes or educational achievement. Here we need to use the option `discrete`; otherwise, `igmobil` will assume continuous variables. Note that the option `discrete` automatically sets `nosingle` and `noinequal`, because single-stage and inequality indices cannot be computed from discrete variables.

```
. igmobil son_disc dad_disc, discrete
(running igmobil_1 on estimation sample)
Bootstrap replications (50)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
Bootstrap results                               Number of obs     =      2,000
                                                Replications      =         50

Child generation:      son_disc = Y             Type of variables: discrete
Parent generation:     dad_disc = X
```

Type of indices	IGM estimate	Bootstrap Std. Err.	[95% Conf. Interv.] normal approx.	
Transition matrix Indices (original categories of X,Y)				
(11) Shorrock/Prais	0.834	0.016	0.801	0.866
(12) Bartholomew	0.195	0.006	0.182	0.207
(13) 1-Second largest eigenvalue	0.547	0.020	0.507	0.586
(14) Determinant index	1.000	0.000	1.000	1.000

4.3 igmobil and inequality-based indices

`igmobil` allows users to customize both the IGM indices family (from the default Fields to Shorrocks) and the parameters of the generalized entropy and the Atkinson indices (a maximum of two parameters each; if only one parameter is specified, it is repeated twice). These options can be combined with the others previously given.

In the following example, we estimate both 4×4 transition-matrix and inequality-based indices. For the last class, we specify that the Shorrocks family should be used.

```
. igmobil son dad, nosingle classes(4) family(shorrocks)
(running igmobil_1 on estimation sample)
Bootstrap replications (50)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
Bootstrap results                               Number of obs     =      2,000
                                                Replications      =         50

Child generation:      son = Y                  Type of variables: continuous
Parent generation:     dad = X
```

Type of indices	IGM estimate	Bootstrap Std. Err.	[95% Conf. Interv.] normal approx.	
Transition matrix Indices (based on 4 quantiles)				
(11) Shorrock/Prais	0.807	0.016	0.776	0.837
(12) Bartholomew	0.282	0.007	0.269	0.295
(13) 1-Second largest eigenvalue	0.547	0.018	0.512	0.582
(14) Determinant index	0.999	0.001	0.997	1.001
Inequality related Indices				
(15) Shorrocks - Gini	0.140	0.015	0.110	0.170
(16) Shorrocks - GE(0)	0.265	0.024	0.217	0.313
(17) Shorrocks - GE(1)	0.266	0.029	0.209	0.323
(18) Shorrocks - Atkinson(.5)	0.259	0.026	0.209	0.310
(19) Shorrocks - Atkinson(2)	0.239	0.021	0.198	0.280

In the next example, we modify the parameter of the generalized entropy (2, for which the generalized entropy becomes the half-squared coefficient of variation) and of the Atkinson index (2 and 5).

```
. igmobil son dad, nosingle notrans family(fields) ge(2) atk(2 5)
(running igmobil_1 on estimation sample)
Bootstrap replications (50)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
Bootstrap results                                 Number of obs     =      2,000
                                                  Replications      =         50

Child generation:      son = Y                    Type of variables: continuous
Parent generation:     dad = X
```

Type of indices	IGM estimate	Bootstrap Std. Err.	[95% Conf. Interv.] normal approx.	
Inequality related Indices				
(15) Fields - Gini	0.134	0.010	0.114	0.154
(16) Fields - GE(2)	0.278	0.023	0.233	0.324
(17) Fields - GE(2)	0.278	0.023	0.233	0.324
(18) Fields - Atkinson(2)	0.229	0.016	0.198	0.261
(19) Fields - Atkinson(5)	0.194	0.022	0.150	0.237

4.4 Standard errors and confidence intervals

One important feature of `igmobil` is the embedded bootstrap procedure, which allows users to make inference on the estimated indices without the need of extra programming. By default, the number of replications is set to 50 to reduce computational burden. In the next example, we modify the relevant `bootstrap()` option by increasing the number of replications and saving each bootstrap result for later use. We also fix the `seed` number so that results can be replicated, and we omit replication dots to preserve space.

```
. igmobil son dad, nosingle noinequal
> bootstrap(reps(200) seed(12345) saving(myfile, replace) nodots)
Bootstrap results                                 Number of obs     =      2,000
                                                  Replications      =        200

Child generation:      son = Y                    Type of variables: continuous
Parent generation:     dad = X
```

Type of indices	IGM estimate	Bootstrap Std. Err.	[95% Conf. Interv.] normal approx.	
Transition matrix Indices (based on 5 quantiles)				
(11) Shorrock/Prais	0.849	0.015	0.820	0.878
(12) Bartholomew	0.269	0.006	0.257	0.281
(13) 1-Second largest eigenvalue	0.522	0.019	0.485	0.560
(14) Determinant index	1.000	0.000	1.000	1.000

Comparing the output with the corresponding part of the table shown in section 4.2, we see that the standard errors and the (normal approximated) confidence intervals hardly changed when we went from 50 to 200 bootstrap replications. However, this might be due to the specific data-generating process used for this example.

In this example, we specify that the program displays the 99% percentile method confidence interval. Note that the confidence level is set as a suboption of the `bootstrap()` option, while the type of confidence interval is chosen with the option `citype()`.

```
. igmobil son dad, nosingle noinequal
> boot(reps(200) seed(12345) saving(myfile, replace) level(99) nodots)
> citype(percentile)
Bootstrap results                                 Number of obs     =      2,000
                                                  Replications      =        200

Child generation:      son = Y                    Type of variables: continuous
Parent generation:     dad = X
```

Type of indices	IGM estimate	Bootstrap Std. Err.	[99% Conf. Interv.] percentile	method
Transition matrix Indices (based on 5 quantiles)				
(11) Shorrock/Prais	0.849	0.015	0.817	0.903
(12) Bartholomew	0.269	0.006	0.257	0.288
(13) 1-Second largest eigenvalue	0.522	0.019	0.482	0.568
(14) Determinant index	1.000	0.000	1.000	1.000

In this case, we can no longer expect the confidence interval to be symmetric with respect to the estimated coefficient. Here the estimated Bartholomew index (0.269) is slightly closer to the left bound of the interval (0.257) than to the right one (0.288).

4.5 Reporting and postestimation

Because `igmobil` is an estimation command, we can easily recall parameter estimates and standard errors by typing `_b[i#]` and `_se[i#]`, where # is the IGM progressive number. We first recall the last estimate of the Eigenvalue2 index ($\mathcal{M}_{13}$), and then we test the hypothesis that this index equals 0.5. We find that the Wald test would not reject the null hypothesis at the 5% level.

```
. display _b[i13]
.52210281
. test i13 = 0.5
 ( 1)  i13 = .5
           chi2(  1) =    1.33
         Prob > chi2 =    0.2485
```

Another advantage of an estimation command is that it can be used with the commands `estimates store`, `estimates restore`, `estimates table`, etc., to manipulate estimation results. Assume that a quarter of the data belongs to country A and the remaining to country B and that we want to estimate the $\mathcal{M}_{11}$, $\mathcal{M}_{12}$, and $\mathcal{M}_{13}$ separately for those countries. We can then apply the following commands:

```
. generate country = cond(_n<= 500, "A", "B")
. quietly igmobil son dad if country == "A",  nosingle noinequal
    Transition matrix Indices (based on 5 quantiles)
. estimate store igm_A
```

```
. quietly igmobil son dad if country == "B",  nosingle noinequal
    Transition matrix Indices (based on 5 quantiles)
. estimate store igm_B
. estimates table igm_A igm_B, stats(N) b(%9.4f) se(%9.4f) keep(i11-i13)
```

Variable	igm_A	igm_B
i11	0.8700	0.8475
	0.0286	0.0153
i12	0.2680	0.2713
	0.0111	0.0070
i13	0.5106	0.5299
	0.0499	0.0235
N	500	1500

legend: b/se

4.6 Advanced use: Adding a new IGM index

Assume that we want to add the Bhattacharya and Mazumder (2011) upward-mobility (UP) index described in section 2 to our standard transition-matrix indices. In particular, we want to estimate the probability that a child's percentile exceeds the one of his or her parent by 10, given that the parent belonged to the lowest quartile. In other words, we seek the sample counterpart of $\text{UP}_{\tau,s} = \text{UP}_{0.10,0.25} = \Pr(r_{Y_i} - r_{X_i} > 0.10 |\, r_{X_i} \leq 0.25)$.

To do this, we write an r-class program that returns the desired IGM estimate in a macro named `UW` (which stands for user written) and save it as a do-file.

```
* upward-mobility index
capture program drop myindex
        program myindex, rclass
        syntax varlist(min=2 max=2 numeric) [if] [in] [, tau(real 0) s(real 0.25)]

        marksample touse

        tempvar y x ry rx diff
        tempname num den

        tokenize `varlist'
        quietly {
                generate `y'  = `1' if `touse'
                generate `x'  = `2' if `touse'

                cumul `y', gen(`ry')
                cumul `x', gen(`rx')

                count if (`ry'-`rx') > `tau' & `rx' <= `s' & `touse'
                scalar `num' = r(N)

                count if `rx' <= `s' & `touse'
                scalar `den' = r(N)
```

```
                return scalar UW = `num'/`den'
        }
end
```

Finally, we incorporate the `myindex` command into `igmobil`:

```
. quietly do myindex.do
. igmobil son dad, nosingle noinequal
> userwritten(myindex son dad, tau(0.1) s(0.25)) classes(4)
(running igmobil_1 on estimation sample)
Bootstrap replications (50)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
Bootstrap results                                 Number of obs     =      2,000
                                                  Replications      =         50
Child generation:      son = Y                    Type of variables: continuous
Parent generation:     dad = X
```

Type of indices	IGM estimate	Bootstrap Std. Err.	[95% Conf. Interv.] normal approx.	
Transition matrix Indices (based on 4 quantiles)				
(11) Shorrock/Prais	0.807	0.015	0.778	0.835
(12) Bartholomew	0.282	0.006	0.269	0.294
(13) 1-Second largest eigenvalue	0.547	0.019	0.509	0.584
(14) Determinant index	0.999	0.001	0.996	1.001
(20) User written program	0.550	0.015	0.520	0.580

Among the families whose parents belonged to 25% of the poorest, 55.0% of the children outperformed the rank of their parents by more than 0.1 (or 10%).

5 Acknowledgments

I thank the editor and an anonymous referee for helpful comments; any remaining errors are my responsibility. Any views expressed in this article are those of the author and do not necessarily reflect those of the Bank of Italy.

6 References

Bartholomew, D. J. 1973. *Stochastic Models for Social Processes*. New York: Wiley.

Bhattacharya, D., and B. Mazumder. 2011. A nonparametric analysis of black–white differences in intergenerational income mobility in the United States. *Quantitative Economics* 2: 335–379.

Black, S. E., and P. J. Devereux. 2011. Recent developments in intergenerational mobility. In *Handbook of Labor Economics*, ed. O. Ashenfelter and D. Card, vol. 4B, 1487–1541. San Diego: Elsevier.

Chakravarty, S. R., B. Dutta, and J. A. Weymark. 1985. Ethical indices of income mobility. *Social Choice and Welfare* 2: 1–21.

Checchi, D., and V. Dardanoni. 2002. Mobility comparisons: Does using different measures matter? Departmental Working Papers 2002-15, Department of Economics, Management, and Quantitative Methods at Università degli Studi di Milano. http://ideas.repec.org/p/mil/wpdepa/2002-15.html.

Cowell, F. A. 2000. Measurement of inequality. In *Handbook of Income Distribution, vol. 1*, ed. A. B. Atkinson and F. Bourguignon, chap. 2. The Netherlands: Elsevier.

Cowell, F. A., and C. Schluter. 1998. Income mobility: A robust approach. Discussion paper DARP/37, Suntory and Toyota International Centres for Economics and Related Disciplines, London School of Economics and Political Science. http://sticerd.lse.ac.uk/dps/darp/darp37.pdf.

Fields, G. S. 2008. Income mobility. In *The New Palgrave Dictionary of Economics*, ed. S. N. Durlauf and L. E. Blume, 2nd ed. London: Palgrave Macmillan. http://www.dictionaryofeconomics.com/article?id=pde2008_I000271.

———. 2010. Does income mobility equalize longer-term incomes? New measures of an old concept. *Journal of Economic Inequality* 8: 409–427.

Fields, G. S., and E. A. Ok. 1996. The meaning and measurement of income mobility. *Journal of Economic Theory* 71: 349–377.

———. 1999a. The measurement of income mobility: An introduction to the literature. In *Handbook of Income Inequality Measurement, vol. 71*, ed. J. Silber, 557–598. The Netherlands: Springer.

———. 1999b. Measuring movement of incomes. *Economica* 66: 455–471.

Hart, P. E. 1981. The Statics and Dynamics of Income Distributions: A Survey. In *The Statics and Dynamics of Income*, ed. N. A. Klevmarken and J. A. Lybeck. Clevedon, UK: Tieto.

Jäntti, M., and S. P. Jenkins. 2013. Income Mobility. SOEPpapers No. 607, German Institute for Economic Research. http://www.diw.de/documents/pubikationen/73/diw_01.c.432440.de/diw_sp0607.pdf.

Jenkins, S. P. 2006. Estimation and interpretation of measures of inequality, poverty, and social welfare using Stata. 2006 North American Stata Users Group meeting proceedings. http://ideas.repec.org/p/boc/asug06/16.html.

Shorrocks, A. 1978a. Income inequality and income mobility. *Journal of Economic Theory* 19: 376–393.

Shorrocks, A. F. 1978b. The measurement of mobility. *Econometrica* 1013–1024.

Sommers, P. M., and J. Conlisk. 1979. Eigenvalue immobility measures for Markov chains. *Journal of Mathematical Sociology* 6: 253–276.

Appendix: Generalized entropy and Atkinson indices

In this appendix, I provide some background on the generalized entropy measure and the Atkinson index, relying mostly on Jenkins (2006) and Cowell (2000). The generalized entropy family takes the following form:

$$\text{GE}(a) = \frac{1}{Na(a-1)} \sum_{i=1}^{N} \left\{ \left(\frac{x_i}{\mu_x} \right)^a - 1 \right\} \qquad a \neq [0,1]$$

$$\text{GE}(a) = \frac{1}{N} \sum_{i=1}^{N} \left\{ \frac{x_i}{\mu_x} \log \left(\frac{x_i}{\mu_x} \right) \right\} \qquad a = 1$$

$$\text{GE}(a) = \frac{1}{N} \sum_{i=1}^{N} \log \left(\frac{\mu_x}{x_i} \right) \qquad a = 0$$

The parameter a captures the sensitivity of the $\text{GE}(a)$ family to a particular part of the distribution: a large positive a increases sensitivity to changes in the upper tail, and a negative a increases sensitivity to changes in the lower tail. For specific values of a, the $\text{GE}(a)$ assumes the following known forms: $\text{GE}(0)$ is the mean log deviation, $\text{GE}(1)$ is the Theil index, and $\text{GE}(2)$ is the half-squared coefficient of variation. The default option will have $a_1 = 0$ and $a_2 = 1$.

The Atkinson index is a welfare-based measure of inequality that assumes an explicit formulation of the social welfare function (that is, the way individual utilities are aggregated) and an explicit level of income inequality aversion. At the core of the Atkinson index is the equally distributed equivalent income, $y_e \leq \mu_y$, which is the income value that—if equally distributed—would equal the same level of social welfare as the actual income distribution. The larger the difference between μ_Y and y_e, the higher the cost of inequality. Finally,

$$\text{Atk}(\epsilon) = 1 - \left\{ \frac{1}{N} \sum_{i=1}^{N} \left(\frac{x_i}{\mu_x} \right)^{1-\epsilon} \right\}^{\frac{1}{1-\epsilon}} \qquad \epsilon \geq 0, \epsilon \neq 1$$

$$\text{Atk}(\epsilon) = 1 - \exp \left\{ \frac{1}{N} \sum_{i=1}^{N} \left(\frac{x_i}{\mu_x} \right) \right\} \qquad \epsilon = 1$$

where the parameter ϵ captures the inequality aversion in a society. A larger ϵ means that the society is more inequality averse. The default option will have $\epsilon_1 = 0.5$ and $\epsilon_2 = 2$.

About the author

Marco Savegnago is an economist at the Bank of Italy.

The Stata Journal (2016)
16, Number 2, pp. 403–415

Fixed effects in unconditional quantile regression

Nicolai T. Borgen
Department of Sociology and Human Geography
University of Oslo
Oslo, Norway
n.t.borgen@sosgeo.uio.no

Abstract. Unconditional quantile regression has quickly become popular after being introduced by Firpo, Fortin, and Lemieux (2009, *Econometrica* 77: 953–973) and is easily implemented using the user-written command `rifreg` by the same authors. However, including high-dimensional fixed effects in `rifreg` is quite burdensome and sometimes even impossible. In this article, I show that when the number of fixed effects is large, the computational speed is massively increased by using `xtreg` rather than `regress` to fit the unconditional quantile regression models. I also introduce the `xtrifreg` command, which should be considered a supplement to `rifreg`. The `xtrifreg` command has many of the same features as `rifreg` but can be used to include a large number of fixed effects, to estimate cluster–robust standard errors, and to estimate cluster–bootstrapped standard errors.

Keywords: st0438, xtrifreg, unconditional quantile regression, fixed effects

1 Introduction

In the last few years, researchers have discussed the usefulness of the conditional quantile regression (CQR) method in social science research. In CQR, the quantiles are defined conditional on the control variables. Thus including control variables not only adjusts for selection bias, but also redefines the quantiles. This redefinition of the quantiles is sometimes advantageous, for instance, when investigating student growth in test scores (Castellano and Ho 2013).

However, in most cases, researchers are not interested in the effects of conditional quantiles. Rather, they want to investigate the effects of a treatment variable on unconditional quantiles (Porter 2015). Thus Firpo, Fortin, and Lemieux (2009) developed the unconditional quantile regression (UQR) model. The advantage of the UQR model is that the quantiles are defined preregression; therefore, the model is not influenced by any right-hand-side variables (Killewald and Bearak 2014). In UQR, one can, for instance, include fixed effects to adjust for selection bias without redefining the quantiles.

However, implementing UQR in Stata with high-dimensional fixed effects (that is, a large number of groups) is either computationally slow or quite burdensome, especially if the researcher wants bootstrapped standard errors. In two recent commentary articles in the *American Sociological Review*, Killewald and Bearak (2014) and Budig and Hodges

(2014) discuss, among other things, how to include a large number of fixed effects in UQR. Killewald and Bearak (2014) use a computationally undemanding approach of demeaning their variables and including the demeaned variables in an ordinary least-squares (OLS) model. This approach introduces some complications regarding, for example, estimation of standard errors (Budig and Hodges 2014; Allison 2009, 18). Even introductory texts to quantile regression do not always acknowledge that when using UQR, the conventional `bootstrap` command will produce incorrect standard errors (for an example, see Porter [2015, 377]).

Budig and Hodges (2014) suggest using the more computationally demanding strategy of including dummy variables, in their case, $N - 1$ person dummy variables. The advantage of this least-squares dummy variables (LSDV) approach is that it is less prone to coding errors. But with high-dimensional fixed effects, the LSDV estimator is very slow (Allison 2009). With large datasets, such as administrative data, the number of fixed effects may also exceed the allowed number of right-hand-side variables (10,998 in Stata/MP and Stata/SE, 798 in Stata/IC); for an example, see Borgen (2015). Demeaning is often preferable to the LSDV estimator because of the incidental-parameter problem (Cameron and Trivedi 2009, 259).[1]

In this article, I demonstrate how to include high-dimensional fixed effects in Stata without any tradeoff between computational speed and ease of implementation. When you have a large number of fixed effects, I suggest a two-step approach based on the intuition in the seminal paper by Firpo, Fortin, and Lemieux (2009). The first step is to obtain the recentered influence function (RIF), which is convenient with the user-written `rifreg` command (Firpo, Fortin, and Lemieux 2009). The second step is to use the `xtreg` command—rather than the `regress` command used in, for instance, `rifreg`—with this RIF as the outcome variable.

The two-step approach I describe may have a few potential pitfalls. I discuss these pitfalls and explain how researchers can avoid them by using the `xtrifreg` command, which I also introduce in this article. Because the `xtrifreg` command is basically a wrapper around `rifreg` and `xtreg`, it makes it more comfortable, reliable, and streamlined to include high-dimensional fixed effects in UQR. This command should be considered a supplement to the `rifreg` command. `xtrifreg` is particularly handy for including bootstrapped standard errors. Unlike the `rifreg` command, it also reports cluster–robust standard errors and cluster–bootstrapped standard errors. In fixed-effects models, using cluster–robust standard errors and bootstrapped standard errors is often advisable (Cameron and Miller 2015).

There are two main advantages to using `xtrifreg` rather than `rifreg`: 1) `xtrifreg` can be used even if the number of fixed effects exceeds 10,997[2], and 2) it is a lot faster than `regress` when the number of fixed effects is large. I will demonstrate

1. Unlike the demeaning approach (`xtreg`), the LSDV approach (`regress` or `areg`) assumes that the number of fixed effects does not grow with sample size. The estimated variance–covariance matrix, therefore, differs in the LSDV approach compared with the demeaning approach when `vce(cluster` *clustvar*`)` is specified.
2. Assuming at least one other independent variable, the maximum number of fixed effects in Stata/MP and Stata/SE is $10998 - 1$.

that computational speed is massively increased by using `xtrifreg`. With the current "data revolution" in social sciences, with more big data—such as administrative data (Einav and Levin 2013)—and multiple high-dimensional fixed effects (McCaffrey et al. 2012; Guimarães and Portugal 2010), computational speed is becoming increasingly important.

The remainder of the article is arranged as follows. In section 2, I describe UQR. In section 3, I describe the two-step approach to including fixed effects in UQR. In section 4, I discuss the standard errors in this two-step approach. In section 5, I present the `xtrifreg` command. Finally, in section 6, I conclude.

2 Unconditional quantile regression basics

Researchers can estimate UQR by simply replacing the outcome variable in OLS with the RIF. Firpo, Fortin, and Lemieux (2009) provide a technical introduction, while Porter (2015) and Killewald and Bearak (2014) provide more easily accessible introductions.[3]

RIF is defined as

$$\text{RIF}(Y; q_\tau, F_Y) = q_\tau + \frac{\tau - \mathbb{1}\{Y \leq q_\tau\}}{f_Y(q_\tau)} \tag{1}$$

where q_τ is the value of the outcome variable, Y, at the quantile τ. F_Y is the cumulative distribution function of Y, and $f_Y(q_\tau)$ is the density of Y at q_τ. The indicator function, $\mathbb{1}\{Y \leq q_\tau\}$, identifies whether the value of the outcome variable, Y, for the individual is below q_τ.

Consider the 75th quantile ($\tau = 0.75$). To identify the RIF for this quantile, one needs to 1) estimate the value of the outcome variable, Y, at that quantile, $q_{0.75}$; 2) estimate the density $f_Y(q_{0.75})$ at $q_{0.75}$ using, for instance, kernel methods; and 3) generate a dummy variable, $\mathbb{1}\{Y \leq q_{0.75}\}$, which indicates whether the value of the outcome variable is at or below the value of Y at the 75th quantile, $q_{0.75}$. The resulting RIF is a dummy variable, holding the values $q_{0.75} + \{0.75/f_Y(q_{0.75})\}$ for those above the 75th quantile and the values $q_{0.75} - \{0.25/f_Y(q_{0.75})\}$ for those at or below the 75th quantile. This RIF could serve as the outcome variable in an OLS model (linear probability model), a so-called RIF-OLS (Firpo, Fortin, and Lemieux 2009).[4]

Equation (1) provides two main insights (Porter 2015). First, the transformed outcome variable (the RIF) is defined preregression. Thus, unlike CQR, including any control variables does not change the definition of the quantile. Second, the transformed outcome variable (the RIF) depends heavily on the estimated density, $f_Y(q_\tau)$. It is thus wise to check the sensitivity of the results by using different kernels and bandwidths.

3. For readers interested in CQR, see Koenker (2005) and Hao and Naiman (2007)
4. Firpo, Fortin, and Lemieux (2009) also describe two other ways to estimate UQR: RIF-logit and RIF-NP. The differences between the three estimation methods are minor in their application.

3 Including fixed effects in UQR

Fitting UQR models in Stata is made easy by the user-written command `rifreg` (Firpo, Fortin, and Lemieux 2009).[5] This command 1) computes the RIF and 2) includes this RIF as an outcome variable in `regress` along with any right-hand-side variables. In most applications, using `rifreg` is a good choice.[6]

To include fixed effects in `rifreg`, one could simply add the fixed effects as a set of dummy variables. However, in some cases, the number of fixed effects may exceed the number of allowed right-hand-side variables. In Borgen (2015), for instance, the number of sibling fixed effects is 42,860, which makes it impossible to use `rifreg`. More generally, when the number of fixed effects is large, the dummy-variable approach to fixed effects gets burdensome (Allison 2009). Luckily, we can speed up the process by substituting the `regress` command in step 2 with `xtreg`.[7]

I will use a subsample of the National Longitudinal Survey (`nlswork.dta`) to exemplify how to obtain UQR point estimates and standard errors using `xtreg` (see Borgen [2015] for an empirical application). `nlswork.dta` contains information on 4,711 young working women aged 14–26 years in 1968, followed over the years 1968–1988. The total number of observations is 28,453.

To load this dataset, type

```
. use http://www.stata-press.com/data/r14/nlswork
(National Longitudinal Survey.  Young Women 14-26 years of age in 1968)
```

Using `nlswork.dta`, I will estimate the effect of union membership (1 if union member) on log wages at the 50th quantile, with fixed effects on individuals (`idcode`). The model is

$$Y_{it} = \beta_0 + \beta_1 \text{union}_{it} + \alpha_i + \varepsilon_{it}$$

where i indexes individuals and t indexes time, β_0 is the constant term, β_1 is the effect of union membership, α_i are the individual fixed effects, and ε_{it} is the error term. I use this model specification to illustrate how to include fixed effects in UQR, but it is not a correct specification of the causal effect of union membership.

3.1 Obtaining the RIF

In Stata, generating the transformed outcome variable (the RIF) is easy. By using the `retain(`*string*`)` option of `rifreg`, the transformed outcome variable is stored to the dataset.

5. To install this program, go to http://faculty.arts.ubc.ca/nfortin/datahead.html.
6. Another useful command is the `ivqte` command by Frölich and Melly (2010), which can implement CQR, the instrumental-variable conditional quantile regression estimator, UQR, and the instrumental-variable unconditional quantile regression estimator.
7. Using `xtreg` in the second step is similar to the demeaning strategy of Killewald and Bearak (2014) but is easier to implement (Allison 2009).

```
. rifreg ln_wage, quantile(50) retain(q50)
  (output omitted)
```

The `rifreg` command uses the `pctile` command to identify the value of the outcome variable at the 50th quantile ($q_{.50}$) and the `kdensity` command to estimate the density of the outcome variable at the quantile [$f_Y(q_{.50})$]. `rifreg` then includes these values in (1). We could also do this ourselves, without the `rifreg` command, as follows:[8]

```
pctile quantiles=ln_wage, n(100)
kdensity ln_wage, at(quantiles) kernel(gaussian) bwidth(0.0) ///
     generate(quantile density) nodraw
generate indicator=ln_wage<quantile[50]
generate q50=quantile[50]+((.50-indicator)/density[50])
```

When one obtains the transformed outcome variable in a separate first step, there are two potential caveats (neither of which pose a problem when using the `xtrifreg` command, which I will introduce in section 5). First, one should make sure that the sample used to obtain the RIF variable is identical to the sample that will be used in the regression analyses. One could do this by using, for instance, the `marksample` and `markout` commands. If the sample differs, the coefficients and inference will be incorrect.

```
. marksample touse
. markout `touse´ ln_wage union idcode
. rifreg ln_wage if `touse´, quantile(50) retain(q50)
  (output omitted)
```

Second, the `rifreg` command gives all observations that are excluded from the analysis (`e(sample)==0`) the value 0 on the retained variable. In `nlswork.dta`, approximately a third of the observations have missing on the union variable and get the value 0 on the outcome variable.

```
. tabulate q50
```

q50	Freq.	Percent	Cum.
0	9,296	32.58	32.58
1.155391	9,617	33.70	66.28
2.305247	9,621	33.72	100.00
Total	28,534	100.00	

It may be wise to replace the 0 values on the retained variable with missing.

```
. replace q50=. if e(sample)!=1
(9,296 real changes made, 9,296 to missing)
```

We are then left with a sample of 19,238 observations on 4,150 women.

8. Following the `rifreg` command, the indicator function is here defined as 1 if log wages is below the 50th quantile. Note that this is slightly different from how the RIF is defined in (1).

3.2 Regression model

Next, we can include the transformed outcome variable in any linear regression model. For example, we can use `xtreg`, which is considerably more efficient than `regress` if the regression model includes high-dimensional fixed effects. To estimate UQR with fixed effects using `xtreg`, simply type

```
xtreg q50 union, i(idcode) fe
```

The coefficient of union is equivalent to the much more time-demanding approach of including $N-1$ person-dummy variables in `rifreg`.

```
xi: rifreg ln_wage union i.idcode, quantile(50)
```

3.3 Weights

One can also include weights in this two-step approach. However, the weights must be included in both `rifreg` (or `pctile` and `kdensity`) and `xtreg`.

4 Standard errors

The two-step approach outlined in section 3 and the `rifreg` command do not automatically produce identical standard errors. With `rifreg`, we have the option of conventional standard errors (the default in `regress` and `xtreg`),

```
xi: rifreg ln_wage union i.idcode, quantile(50) norobust
```

robust standard errors (the Huber/White/sandwich estimator, which is the default in `rifreg`),

```
xi: rifreg ln_wage union i.idcode, quantile(50)
```

and bootstrapped standard errors.

```
xi: rifreg ln_wage union i.idcode, quantile(50) bootstrap
```

4.1 Conventional standard errors

The conventional standard errors assume that the error term is independent and identically distributed. To replicate the conventional standard errors in `rifreg` using the `xtreg` command, we type the following:

```
xtreg q50 union, fe i(idcode)
```

4.2 Robust standard errors

Reporting the default standard errors in `rifreg` relaxes the assumption that the error term is identically distributed (but not the independence assumption) by using the Huber/White/sandwich estimator. Replicating these standard errors using `xtreg` is complicated, because `xtreg` with the `robust` option reports standard errors that not only relax the assumption that the error term is identically distributed but also relax the independence assumption. With the `vce(robust)` option, `xtreg` requires only that the observations are independent across the panel variable, otherwise known as cluster–robust standard errors in Stata.[9]

However, that `xtreg` reports cluster–robust standard errors is certainly not a drawback. Using cluster–robust standard errors in fixed-effects models is often a better choice than using robust standard errors (Cameron and Miller 2015).

```
xtreg q50 union, fe i(idcode) robust
```

4.3 Bootstrapped standard errors

Most studies using quantile regression report bootstrapped standard errors, which is also what Firpo, Fortin, and Lemieux (2009) report in their main examples. Unfortunately, bootstrapping the standard errors in UQR is slightly more complex than bootstrapping the standard errors in the OLS analysis, because the distribution of Y changes in each bootstrapped sample. Thus the value of the outcome variable, Y, at the 50th quantile, $q_{.50}$, and the estimated density at the 50th quantile, $f_Y(q_{.50})$, differ in each bootstrapped sample [see equation (1)].

Thus the transformed outcome variable must be recalculated in each of the bootstrapped samples. However, the `bootstrap` prefix command and the `vce(bootstrap)` option in `xtreg` bootstrap only the coefficients (Cameron and Trivedi 2009) and leave the transformed outcome variable unchanged.[10] Not all researchers acknowledge this limitation of the `bootstrap` command and the `vce(bootstrap)` option when fitting UQR models. In a highly accessible and useful introductory text to quantile regression, which also shows how to implement CQR and UQR in Stata, Porter (2015, 377) unfortunately makes exactly this error when bootstrapping the UQR standard errors.

The following occur when using the `bootstrap` option of `rifreg`:

1. A dataset of size `_N` is sampled with replacement.
2. In this dataset, the transformed outcome variable is generated.

9. The user-written command `xtivreg2` provides robust standard errors in fixed-effects models (Schaffer 2005).
10. With CQR, however, using the `bootstrap` prefix command yields identical standard errors as the `bsqreg` command, because the quantiles are defined (conditional on the covariates) in the regression model, and not before the regression model, as with UQR.

3. Using this transformed outcome variable, one runs a linear regression model and stores the value of the coefficient.

4. The three first steps are then repeated, with the default being 50 replications.

5. The standard deviation of the coefficients from the bootstrapped samples is the bootstrapped standard errors.

To get bootstrapped standard errors in UQR, we need to calculate the transformed outcome variable and run the regression model in each of the bootstrapped samples. We can do this, for instance, by using the following code:

```
. marksample touse
. markout `touse´ ln_wage union idcode
. forvalues i=1/50 {
  2.         preserve
  3.                 bsample if `touse´
  4.                 qui rifreg ln_wage if `touse´, quantile(50) retain(q50b)
  5.                 qui xtreg q50b union if `touse´, i(idcode) fe
  6.                 matrix b=nullmat(b)\e(b)
  7.         restore
  8.         }
. mata VCEmata=st_matrix("b")
. mata st_matrix("vce", variance(VCEmata))
. matrix colnames vce=union _cons
. matrix rownames vce=union _cons
. matrix list vce
symmetric vce[2,2]
            union       _cons
union   .00021213
_cons  -.00004102   .00002818
```

The square root of the diagonal elements in this variance–covariance matrix is the bootstrapped standard errors. With the `bsample` command, clustered observations could be accounted for by using the `cluster(`*varlist*`)` option.

When there are many independent variables, this bootstrap approach is tedious. I suggest using the `xtrifreg` command in these cases.

5 xtrifreg

5.1 Description

`xtrifreg` builds on the user-written `rifreg` command and can be used to fit UQR models with fixed effects. More specifically, `xtrifreg` uses `pctile` to identify the value of the outcome variable, Y, at the quantile τ (q_τ), uses `kdensity` to estimate the density of Y at q_τ [$f_Y(q_\tau)$], and then includes these values in (1). `xtreg` is subsequently used to fit the regression model with the RIF as the outcome variable. When bootstrapping the standard errors, `xtrifreg` uses the `bsample` command.

5.2 Syntax

`xtrifreg` *depvar indepvars* [*if*] [*in*] [*weight*], `fe i(`*varname*`)` [`quantile(`#`)` `kernop(`*string*`)` `width(`#`)` `norobust` `bootstrap` `clusterbootstrap` `reps(`#`)`]

`aweight`s, `fweight`s, and `iweight`s are allowed; see [U] **11.1.6 weight**.

By using this `xtrifreg` command, the two-step process is automated (as it is in the original `rifreg` command). Thus the outcome variable should be the original outcome variable (not the transformed outcome variable).

5.3 Options

`fe` specifies that a fixed-effects estimator (that is, `xtreg`) should be used. `fe` is required.

`i(`*varname*`)` specifies the fixed-effects variable. Only one fixed-effects variable can be included in `i(`*varname*`)`. `i()` is required.

`quantile(`#`)` specifies the quantile. The 75th quantile, for instance, can be written as either `quantile(.75)` or `quantile(75)`. The default is `quantile(50)`.

`kernop(`*string*`)` specifies the kernel function, where *string* is `gaussian`, `epanechnikov`, `epan2`, `biweight`, `cosine`, `parzen`, `rectangle`, or `triangle`. The default is `kernop(gaussian)`.

`width(`#`)` specifies the halfwidth of the kernel. The default is `width(0.0)`, which calculates the "optimal value" (see the help file for `rifreg`).

`norobust` specifies to include conventional standard errors. The default is to include cluster–robust standard errors.

`bootstrap` specifies to include bootstrapped standard errors.

`clusterbootstrap` specifies to include cluster–bootstrapped standard errors, with clustering on the fixed-effects variable specified in `i(`*varname*`)`.

`reps(`#`)` specifies the number of bootstrap replications. The default is `reps(50)`.

5.4 Examples

To illustrate the use of `xtrifreg`, I will estimate the effect of union membership on wages at the 50th quantile with fixed effects on individuals. I will show how to include conventional standard errors, cluster–robust standard errors, bootstrapped standard errors, and cluster–bootstrapped standard errors.

For comparison, I begin by fitting the model using the `rifreg` command with conventional standard errors, robust standard errors, and bootstrapped standard errors (ignore the `timer` command for now).

```
. xi: rifreg ln_wage union i.idcode, quantile(50) norobust
  (output omitted)
. estimates store norobust
. xi: rifreg ln_wage union i.idcode, quantile(50)
  (output omitted)
. estimates store robust
. timer on 1
. set seed 339487731
. xi: rifreg ln_wage union i.idcode, quantile(50) bootstrap reps(50)
  (output omitted)
. estimates store bootstrap
. timer off 1
. esttab norobust robust bootstrap, keep(union) se mtitle
```

	(1) norobust	(2) robust	(3) bootstrap
union	0.126***	0.126***	0.126***
	(0.0104)	(0.0121)	(0.0145)
N	19238	19238	19238

Standard errors in parentheses
* p<0.05, ** p<0.01, *** p<0.001

The syntax for `xtrifreg` is similar to that of `rifreg`, but instead of including $N-1$ person-dummy variables, we include the `fe` and `i(idcode)` options. The following are examples using `xtrifreg`:

```
. xtrifreg ln_wage union, quantile(50) norobust fe i(idcode)
  (output omitted)
. estimates store norobust
. xtrifreg ln_wage union, quantile(50) fe i(idcode)
  (output omitted)
. estimates store clusterrobust
. timer on 2
. set seed 339487731
. xtrifreg ln_wage union, quantile(50) bootstrap reps(50) fe i(idcode)
  (output omitted)
. estimates store bootstrap
. timer off 2
. set seed 339487731
. xtrifreg ln_wage union, quantile(50) clusterbootstrap reps(50) fe i(idcode)
  (output omitted)
. estimates store clusterbootstrap
```

```
. esttab norobust clusterrobust bootstrap clusterbootstrap, keep(union) se mtitle
```

	(1) norobust	(2) clusterrob~t	(3) bootstrap	(4) clusterboo~p
union	0.126***	0.126***	0.126***	0.126***
	(0.0104)	(0.0158)	(0.0145)	(0.0139)
N	19238	19238	19238	19238

Standard errors in parentheses
* p<0.05, ** p<0.01, *** p<0.001

This demonstrates that `rifreg` and `xtrifreg` produce identical point estimates, as well as identical conventional and bootstrapped standard errors. The standard errors in column 2 in the two tables above are not identical, because `xtrifreg` reports cluster–robust standard errors while `rifreg` reports robust standard errors. Also, unlike with `rifreg`, we can include cluster–bootstrapped standard errors using `xtrifreg`. In this particular example, they are somewhat smaller than the cluster–robust and bootstrapped standard errors.

The main advantages of `xtrifreg` are that it can be used when the number of fixed effects exceeds the number of allowed right-hand-side variables and that it is much faster than `regress` when the number of fixed effects is large. In the above code, I used the `timer` command to investigate the number of seconds used to fit the models with bootstrapped standard errors using `rifreg` (timer 1) and `xtrifreg` (timer 2). The `rifreg` command needed more than 8 hours to fit the model. The `xtrifreg` command fit the same model in only about 30 seconds.

```
. timer list
   1:  29784.74 /        1 =   29784.7380
   2:     28.59 /        1 =      28.5950
```

Thus computational speed is massively increased by using `xtrifreg` rather than `rifreg`. Note that the number of bootstrapped replications was only 50, which is far less than the recommended lower limit of at least 200 (Cameron and Trivedi 2009, 433). With 200 bootstrap replications, we should expect the `rifreg` command to take almost one and a half days, and that is for only the effects on one quantile. To estimate the effect throughout the wage distribution, say, at deciles, we would need two weeks if using `rifreg`. However, with `xtrifreg`, we need fewer than 20 minutes.

```
. timer on 3
. forvalues i=10(10)90 {
  2.         set seed 339487731
  3.         qui xtrifreg ln_wage union, q(`i´) fe i(idcode) bootstrap reps(200)
  4.                 estimates store decile`i´
  5. }
. timer off 3
. timer list 3
   3:   1005.48 /        1 =    1005.4820
```

The difference in computational speed between `rifreg` and `xtrifreg` depends on many factors, including the Stata version being used, the computer specifications, the sample size, and the number of fixed effects. In this article, I used an analysis sample of 19,238 observations and 4,150 fixed effects. This is indeed a large dataset, but with the "data revolution" (Einav and Levin 2013) and, particularly, the call for expanding access to administrative data (Card et al. 2010), researchers will most likely routinely deal with sample sizes substantially larger than this.

6 Conclusion

In this article, I presented an easy-to-use two-step approach to include high-dimensional fixed effects in UQR. I suggested 1) using either the user-written `rifreg` command or the official `pctile` and `kdensity` commands to obtain the RIF and 2) using this RIF variable as an outcome variable in `xtreg`. I also introduced a new command, `xtrifreg`, that automates the process. `xtrifreg` is especially convenient when bootstrapping the standard errors.

The two-step approach I presented in this article, which is implemented by `xtrifreg`, is useful when you have a large number of fixed effects or when the number of fixed effects exceeds the number of allowed right-hand-side variables in Stata. Using a sample of 19,238 observations and 4,150 fixed effects, I demonstrated that the computational speed is massively increased by using `xtrifreg` rather than `rifreg`.

7 Acknowledgments

I thank the editor, an anonymous reviewer, and Solveig T. Borgen for useful comments and suggestions. This article is part of the project "Long-term effects of school-wide interventions and school environment using longitudinal register data", funded by The Research Council of Norway (grant #238050).

8 References

Allison, P. D. 2009. *Fixed Effects Regression Models*. Thousand Oaks, CA: Sage.

Borgen, N. T. 2015. Changes in the economic returns to attending prestigious institutions in Norway. *European Societies* 17: 219–241.

Budig, M. J., and M. J. Hodges. 2014. Statistical models and empirical evidence for differences in the motherhood penalty across the earnings distribution. *American Sociological Review* 79: 358–364.

Cameron, A. C., and D. L. Miller. 2015. A practitioner's guide to cluster–robust inference. *Journal of Human Resources* 50: 317–372.

Cameron, A. C., and P. K. Trivedi. 2009. *Microeconometrics Using Stata*. College Station, TX: Stata Press.

Card, D., R. Chetty, M. S. Feldstein, and E. Saez. 2010. Expanding access to administrative data for research in the United States. American Economic Association, Ten Years and Beyond: Economists Answer NSF's Call for Long-Term Research Agendas.

Castellano, K. E., and A. D. Ho. 2013. Contrasting OLS and quantile regression approaches to student "growth" percentiles. *Journal of Educational and Behavioral Statistics* 38: 190–215.

Einav, L., and J. D. Levin. 2013. The data revolution and economic analysis. NBER Working Paper No. 19035, The National Bureau of Economic Research. http://www.nber.org/papers/w19035.

Firpo, S., N. M. Fortin, and T. Lemieux. 2009. Unconditional quantile regressions. *Econometrica* 77: 953–973.

Frölich, M., and B. Melly. 2010. Estimation of quantile treatment effects with Stata. *Stata Journal* 10: 423–457.

Guimarães, P., and P. Portugal. 2010. A simple feasible procedure to fit models with high-dimensional fixed effects. *Stata Journal* 10: 628–649.

Hao, L., and D. Q. Naiman. 2007. *Quantile Regression*. Thousand Oaks, CA: Sage.

Killewald, A., and J. Bearak. 2014. Is the motherhood penalty larger for low-wage women? A comment on quantile regression. *American Sociological Review* 79: 350–357.

Koenker, R. 2005. *Quantile Regression*. New York: Cambridge University Press.

McCaffrey, D. F., J. R. Lockwood, K. Mihaly, and T. R. Sass. 2012. A review of Stata commands for fixed-effects estimation in normal linear models. *Stata Journal* 12: 406–432.

Porter, S. R. 2015. Quantile regression: Analyzing changes in distributions instead of means. In *Higher Education, Vol. 30: Handbook of Theory and Research*, ed. M. B. Paulsen, 335–381. Cham, Switzerland: Springer.

Schaffer, M. E. 2005. xtivreg2: Stata module to perform extended IV/2SLS, GMM and AC/HAC, LIML, and k-class regression for panel-data models. Statistical Software Components S456501, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s456501.html.

About the author

Nicolai T. Borgen is a postdoctoral researcher in sociology at the University of Oslo.

The Stata Journal (2016)
16, Number 2, pp. 416–423

Calculate travel time and distance with OpenStreetMap data using the Open Source Routing Machine (OSRM)

Stephan Huber
University of Regensburg
Regensburg, Germany
stephan.huber@wiwi.uni-regensburg.de

Christoph Rust
University of Regensburg
Regensburg, Germany
christoph.rust@stud.uni-regensburg.de

Abstract. In this article, we introduce the `osrmtime` command, which calculates the distance and travel time between two points using latitude and longitude information. The command uses the Open Source Routing Machine (OSRM) and OpenStreetMap to find the optimal route by car, by bicycle, or on foot. The procedure is specially built for large georeferenced datasets. Because it is fast, the command uses the full computational capacity of a PC, allows the user to make unlimited requests, and is independent of the Internet and commercial online providers. Hence, there is no risk of the command becoming obsolete. Moreover, the results can be replicated at any time.

Keywords: dm0088, osrmtime, osrmprepare, mqtime, traveltime3, OSRM, OpenStreetMap, Google Maps, MapQuest, geospatial analysis, ArcGIS, travel time, travel distance, public road network

1 Introduction

The increased availability of large georeferenced datasets for scientific purposes calls for an efficient method to calculate the distances between subjects and the time it takes to travel from A to B. In this article, we introduce the `osrmtime` command, which uses geographic data on latitudes and longitudes to determine the travel time and the distance between two points. In contrast to existing commands like `globaldist`, `vincenty`, `geodist`, or `sphdist`, which compute geodetic distances, `osrmtime` calculates the travel time and distance to find the optimal route over public roads by car, by bicycle, or on foot. This platform-independent method (Windows, Mac, Linux) is innovative because it allows the user to calculate an unlimited number of requests, and it works offline, which ensures that the results can be replicated. Moreover, `osrmtime` works efficiently. It can calculate thousands of requests within seconds,[1] because it is multiprocessor capable and uses the Open Source Routing Machine (OSRM).[2] The OSRM

1. In an example, we calculated the distance and the travel time between 826,256 pairwise combinations of German hospitals. The calculation took about 49 minutes, which is about 280 requests per second on a system with 16 GB RAM and an Intel i7-2600 3.40 GHz CPU.
2. For more information, see Luxen and Vetter (2011) and http://project-osrm.org/.

is a high-performance open-source C++ routing engine that indicates the shortest routes on public roads and runs with open-source maps from OpenStreetMap.[3]

The program's independence from the Internet and commercial providers has some advantages. First, georeferenced data often contain sensitive data, and their rules of use often forbid using an Internet connection because of either legally binding constraints or a nondisclosure agreement.

Second, and probably most important, an offline procedure that uses only open-source software ensures that the results can be replicated at any time and carries no risk of the command becoming obsolete—as was the case with `traveltime` (Ozimek and Miles 2011), `traveltime3`,[4] and `mqtime` (Voorheis 2015). These earlier programs calculated travel time and distance using the application programming interface (API)[5] from commercial providers via the Internet. Third-party providers, however, can change their APIs or their terms of use; thus user-written commands can become obsolete. The `traveltime` command, for example, was created to use the Google Maps API v.2. Unfortunately, this API is now obsolete; therefore, so is `traveltime`. Although Stefan Bernhard adjusted `traveltime` to work with the up-to-date Google Maps API v.3, his program `traveltime3` is no longer available because Google changed its restrictions on using the Distance Matrix API.[6] The most recent approach by Voorheis (2015) suffers a combination of both problems. His command, `mqtime`, was created to use the API of the commercial provider MapQuest to calculate travel time and distance for an unlimited number of requests by using OpenStreetMap. Unfortunately, MapQuest restructured its API licensing, dramatically cutting the number of requests that `mqtime` can process. Hence, `mqtime` no longer works, and John Voorheis has ceased to maintain the command.

Third, unlike approaches that use online-mapping services, our approach is not based on real-time data. Although a real-time calculation is sometimes desired, researchers often want to know the travel time and distance at a certain point in time. Furthermore, they frequently want results that can be replicated at any time. Neither is really possible when using real-time data from online services, because the results are a function of time-specific circumstances. For example, if you use georeferenced data from 2013, you will probably not want to calculate the travel time and the distance on a Monday morning in late 2015 during rush hour. In turn, it would probably be misleading to use the resulting travel-time data to explain economic behavior in 2013.

`osrmtime` implements two tools: the OSRM and OpenStreetMap. Both are provided by the open-source community, which offers some advantages but also a few disadvantages. One advantage is that both tools can be downloaded, used, spread, and adjusted without restrictions, which gives the user full control over the software. One disadvantage is that the maps provided by OpenStreetMap are not validated by a general authority like the maps of a commercial provider but are recorded and maintained

3. For more information, see http://www.openstreetmap.org.
4. The user-written code by Stefan Bernhard is no longer available. For further information, please email stefanbernhard88@gmail.com.
5. An API provides source code-based facilities to develop applications for a system in a given programming language.
6. See https://developers.google.com/maps/documentation/distancematrix/.

by users in a decentralized fashion. However, this does not necessarily devalue OpenStreetMap, because the quality of both ways of recording and updating geographical data is subject to criticism. Commercial providers record and maintain geographical information more intensively for regions that are most profitable in sales, whereas the quality of geographical information from open sources is a function of the effort of users in a given region. Therefore, regions with a lively community probably have better maps than regions with only a few active users. Overall, OpenStreetMap is used heavily in scientific research, as Arsanjani et al. (2015) show in their overview.

In the following section, we describe how to install the OSRM with all its dependencies. In section 3, we explain the `osrmtime` command. In section 4, we illustrate its use. In section 5, we conclude by comparing it with ArcGIS.

2 Prerequisites

`osrmtime` calculates the travel time and distance from a point of origin to a point of destination using the high-performance routing open-source software, OSRM. `osrmtime` automatically starts the OSRM from the hard disk and performs the calculation using an extract from OpenStreetMap, which needs to be saved on the hard disk. To use `osrmtime`, your system must support a 64-bit architecture (for example, Windows 7 or later). Some files from the Microsoft Visual C++ Redistributable package must also be installed. In the next section, we describe this installation procedure.

2.1 Install the files

`osrmtime` uses the OSRM and some files from the Microsoft Visual C++ Redistributable package. Both must be installed on your system to run `osrmtime`. The installation can be done manually or automatically.

Automatic

```
. net install osrmtime, from("http://www.uni-regensburg.de/
> wirtschaftswissenschaften/vwl-moeller/medien/osrmtime")
. net get osrmtime, from("http://www.uni-regensburg.de/
> wirtschaftswissenschaften/vwl-moeller/medien/osrmtime")
. shell osrminstall.cmd
```

Manual

1. Copy the ado-files `osrmtime.ado`, `osrmprepare.ado`, and `osrminterface.ado` into your `PERSONAL` ado-folder.

2. Install the recent Microsoft Visual C++ Redistributable package for Visual Studio 2015.[7]

7. See https://www.microsoft.com/en-us/download/details.aspx?id=48145.

3. Install the OSRM by downloading[8] and unpacking the OSRM executables to a folder of your choice in which Stata has write access, for example, `C:/osrm/`.

Note that implementing the OSRM is different on Linux and Mac OS X systems. For instructions on how to build the OSRM on these systems, see https://github.com/Project-OSRM/osrm-backend/wiki/Building%20OSRM.

2.2 Prepare maps with osrmprepare

To use `osrmtime`, you must download at least one map covering the region of interest and prepare it for routing. This is necessary for several reasons. Most importantly, raw OpenStreetMap data also include information that are not relevant for routing, such as public toilets or memorials. The preparation ensures that only relevant information is extracted and that this information can be used efficiently by the OSRM. We offer the `osrmprepare` command to execute all necessary steps automatically. The execution speed for `osrmprepare` depends on the size of your map and the capacity of your system.[9] Note that you have to prepare your map only once. The prepared map can be used as often as you like. To update your map, however, you have to download a more recent map and prepare it again.

The following steps explain how to proceed:

1. Download an OpenStreetMap data file in the `osm.pbf` format to a folder of your choice, for example, `C:/mymaps/mymap.osm.pbf`.[10]

2. Prepare a map for routing. To make this step easier for the user, we wrote the `osrmprepare` command. Install the command and use it as explained below.

Syntax of osrmprepare

`osrmprepare, mapfile(`*pbf_path*`)` [`osrmdir(`*path*`) diskspace(#` `MB)` `profile(`*speed_profile*`)`]

Options of osrmprepare

`mapfile(`*pbf_path*`)` specifies the location of the downloaded map file from OpenStreetMap in *`*.osm.pbf` format, for example, `mapfile("C:/mymap/examplemap.osm.pbf")`. `mapfile()` is required.

8. See http://www.uni-regensburg.de/wirtschaftswissenschaften/vwl-moeller/medien/osrmtime/osrm.zip.
9. For instance, it takes about 27 minutes to extract a map for Germany (about 2.6 GB) on a system with 16 GB RAM with an Intel i7-2600 3.40 GHz CPU.
10. Maps can be downloaded, for example, from http://download.geofabrik.de.

`osrmdir(`*path*`)` specifies the path in which the OSRM executables are saved. The default is `osrmdir("C:/osrm/")` for Windows and `osrmdir("/usr/local/osrm/")` for Linux.

`diskspace(# MB)` specifies the allocation of disk space for preparation. The default is `diskspace(5000 MB)`. If your system cannot allocate 5,000 MB, you must adjust this number here; otherwise, the command will not work.

`profile(`*speed_profile*`)` specifies to prepare a map that contains the routes for traveling by car, by bicycle, or on foot. *speed_profile* can be `car`, `bicycle`, or `foot`.

3 The osrmtime command

3.1 Syntax

`osrmtime` *latitude1 longitude1 latitude2 longitude2* [`, mapfile(`*osrm_path*`)`
`osrmdir(`*path*`) nocleanup threads(#) servers(#) ports(`*numlist*`)`]

latitude1, *longitude1*, *latitude2*, and *longitude2* are numeric variables, denoted in decimal degrees.[11] They contain the starting point (*latitude1 longitude1*) and the destination (*latitude2 longitude2*) in a system of coordinates.

3.2 Options

`mapfile(`*osrm_path*`)` specifies the location of the *`*.osrm`* file format map, for example, `mapfile("C:/mymap/examplemap.osrm")`. This file can be extracted by using the `osrmprepare` command as explained above.

`osrmdir(`*path*`)` specifies the path in which the OSRM binary (see step 1 of preparation) is saved. The default is `osrmdir("C:/osrm/")` for Windows and `osrmdir("/usr/local/osrm/")` for Linux.

`nocleanup` indicates to keep temporary files that are generated during the process and prevents the OSRM from being shut down. This can speed up the calculation if `osrmtime` is used consecutively with the same map, because `osrmtime` does not need to shut down and start the OSRM over and over again.

Advanced users with large datasets can optimize the parallel computing to speed up calculation on their system by using the following options: `threads(#)` specifies the number of parallel Stata threads per running OSRM instance, the default value being `4`; `servers(#)` starts several instances of the OSRM—at least if your system permits, the default being `1`; `ports(`*numlist*`)` resolves problems with used TCP ports by manually specifying the port to use, the default being `5000`.

11. We use the standard coordinate system in its latest revision, World Geodetic System (WGS 84). It also works as the reference coordinate system of the Global Positioning System (GPS).

3.3 Description

`osrmtime` provides an interface to the free high-performance OSRM. This enables the calculation of travel time and distance from a point of origin to a point of destination in Stata. Provided that the OSRM is already installed on your system and you already have prepared your map of interest, `osrmtime` automatically starts the OSRM and performs the calculation. `osrmtime` already implements parallel computation, so the time for calculating shortest distances can be reduced significantly depending on your system.

`osrmtime` generates the following five variables:

- `distance`: the distance of the shortest route in meters
- `duration`: the travel time of the shortest route in seconds
- `jumpdist1`: the (spheric) distance between the specified input location (origin) and a matched location to the road network in meters
- `jumpdist2`: the (spheric) distance between the specified input location (destination) and a matched location to the road network in meters
- `return_code`: 0 ⇒ everything is fine; 1 ⇒ no route was found by the OSRM with points specified; 2 ⇒ the OSRM did not respond; and 3 ⇒ something else went wrong.

Note that large values for `jumpdist1` or `jumpdist2` can be a signal that the map is incomplete, meaning that an existing street is not listed in the map. Hence, we recommend to check the length of both jump distances, especially because the jump distance is not considered in the travel-time calculation, which means that large jump distances can yield an underestimated travel time. One way to solve this problem, for example, is to assign a certain number of seconds per meter that it takes to travel the jump distances and add this time to the travel time.

Advanced users can manipulate the routing using the OSRM in various ways. It is possible, for instance, to exclude certain kinds of roads or to adjust the speed profile (for example, change the maximum speed allowed on highways). Moreover, the map file from OpenStreetMap itself can be manipulated.

4 Example

The following results exemplify how `osrmtime` and `osrmprepare` can be used. In the example, we calculate the travel time and distance from Alexanderplatz in Berlin to 3,374 restaurants also located in Berlin.

```
. *download the map of Berlin
. capture mkdir mymaps
. copy "http://download.geofabrik.de/europe/germany/berlin-latest.osm.pbf"
> "mymaps/berlin.osm.pbf", replace
(note: file mymaps/berlin.osm.pbf not found)
```

```
. *prepare the map (this takes some time ~2 minutes, depending on your system):
. osrmprepare, mapfile("mymaps/berlin.osm.pbf") osrmdir("C:\osrm\") profile(car)
. *open coordinates of restaurants in Berlin
. discard
. import delimited "http://www.uni-regensburg.de/wirtschaftswissenschaften/
> vwl-moeller/medien/osrmtime/restaurants_berlin.csv", delimiter(";") clear
(4 vars, 3,374 obs)
. *add destination Alexanderplatz
. generate lat_alex = 52.5219184
. generate lon_alex = 13.4132147
. list in 1/3
```

	lon	lat	osm_id	name	lat_alex	lon_alex
1.	13.32283	52.50691	26735749	Aida	52.52192	13.41321
2.	13.31732	52.50624	26735760	La Forneria	52.52192	13.41321
3.	13.32078	52.50734	26735763	Sakana	52.52192	13.41321

```
. * calculate travel time and distances:
. osrmtime lat lon lat_alex lon_alex, mapfile("mymaps/berlin.osrm")
> osrmdir("C:\osrm\")
------------------------------------------------------------------------------
Traveltime and Distance with OSRM
Check for running OSRM: not running!
Starting OSRM           now running!
Writing do-files:       done!
Partitioning datasets:  done!
Calculating:
0%---10%---20%---30%--40%---50%---60%--70%---80%---90%--100%
finished calculation!
------------------------------------------------------------------------------
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
lon	3,374	13.37962	.0867841	13.09485	13.75691
lat	3,374	52.50509	.0399598	52.35291	52.66253
osm_id	3,374	1.54e+09	1.08e+09	2.67e+07	3.80e+09
name	0				
lat_alex	3,374	52.52192	0	52.52192	52.52192
lon_alex	3,374	13.41321	0	13.41321	13.41321
distance	3,374	7797.189	5203.881	287	31190
duration	3,374	618.3402	380.2277	28	2469
jumpdist1	3,374	17.75756	16.54522	0	292
jumpdist2	3,374	103	0	103	103
return_code	3,374	0	0	0	0

```
. list name distance duration jumpdist1 jumpdist2 in 1/3
```

	name	distance	duration	jumpdi~1	jumpdi~2
1.	Aida	7360	612	18	103
2.	La Forneria	7710	634	8	103
3.	Sakana	7416	618	11	103

5 Conclusion

In this article, we introduced a fast procedure to calculate travel time and distance using public roads by car, by bicycle, and on foot. This kind of geographic information is fundamental to regional sciences and can be applied to empirical research in various subjects, including economics, sociology, and epidemiology. `osrmtime` has advantages over other offline routing software. The high-end mapping software ArcGIS, for example, also allows the user to calculate the travel time and distance but has some drawbacks compared with `osrmtime`. First, the Network Analyst Extension required is costly. Second, the routing algorithm works less efficiently than the OSRM. Third, ArcGIS does not have a tool that easily allows the user to calculate hundreds of requests. Thus the processing of many requests requires experience with Python. In a previous project, we succeeded in calculating thousands of routing requests using ArcGIS on a cluster of eight PCs. However, when calculating the same requests with one PC and `osrmtime`, we find that ArcGIS is outperformed by a factor of at least 100.

6 References

Arsanjani, J. J., A. Zipf, P. Mooney, and M. Helbich, eds. 2015. *OpenStreetMap in GIScience: Experiences, Research, and Applications*. Cham, Switzerland: Springer.

Luxen, D., and C. Vetter. 2011. Real-time routing with OpenStreetMap data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 513–516. New York: Association for Computing Machinery.

Ozimek, A., and D. Miles. 2011. Stata utilities for geocoding and generating travel time and travel distance information. *Stata Journal* 11: 106–119.

Voorheis, J. 2015. mqtime: A Stata tool for calculating travel time and distance using MapQuest web services. *Stata Journal* 15: 845–853.

About the authors

Stephan Huber and Christoph Rust are research assistants of Joachim Möller at the University of Regensburg. Huber is also a doctoral candidate at the University of Trier. His thesis is about disaggregated international bilateral trade flows and the impact of FDI and international trade on economic development.

The Stata Journal (2016)
16, Number 2, pp. 424–442

Panel time series: Review of the methodological evolution

Tamara Burdisso
Central Bank of Argentina and Universidad de Buenos Aires
Buenos Aires, Argentina
tburdisso@bcra.gob.ar

Máximo Sangiácomo
Central Bank of Argentina and Universidad Nacional de La Plata
La Plata, Argentina
maximo.sangiacomo@bcra.gob.ar

Abstract. In this article, we discuss the econometric treatment of macropanels, also known as panel time series. This new approach rejects the assumption of slope homogeneity and handles nonstationarity. It also recognizes that cross-section dependence (that is, some correlation structure in the error term between units due to unobservable common factors) squanders efficiency gains by operating with a panel. This approach uses a new set of estimators known in the literature as the common correlated effect, which essentially consists of increasing the model to be fit by adding the averages of the individuals in each time t, of both the dependent variable and the specific regressors of each individual. We present two commands developed for the evaluation and treatment of cross-section dependence.

Keywords: st0439, xtcsi, xtcips, panel time series, time series, cross-section dependence

1 Introduction

Panel-data models became very popular in empirical econometrics in the late 20th century and the beginning of the 21st century, mainly because they can capture the heterogeneity of the agents' socioeconomic performance against both cross-section and time-series models.

Panel data[1] are used to describe various econometric situations. Basically, panel data consist of a sample of units[2] over time, and the data provide multiple observations for each unit using periodic surveys on families or companies. Panel-data models use either micropanels or macropanels. A micropanel consists of a large number of N units—hundreds or thousands—over a short period of time, from $T = 2$ observations per unit to a maximum of $T = 10/20$. In contrast, macropanels generally involve an N number of countries from a few nations (such as the G7 members to all countries of

1. Panel data may also be referred to as longitudinal data. The term used may vary depending on the discipline analyzing the data.
2. By units, we mean workers, families, companies, industries, regions, countries, etc.

the Penn World Table or the World Development Indicators), and the data are usually given quarterly or yearly, with ranges from 20 to 60 years (Arellano 2003; Hsiao 2014).

Micropanels and macropanels require different econometric treatment (Baltagi 2013). For instance, in micropanels, the asymptotic analysis must be performed for large N and fixed T; in macropanels, the asymptotic analysis is performed allowing both N and T to tend to infinity (Phillips and Moon 1999). Likewise, a large T in a macropanel must deal with nonstationarity issues inherent in the time-series analysis.

The first theoretical developments involving panel data were applied to the treatment of micropanels. The panel-data literature in the second half of the 1980s and in most of the 1990s focused on the structure of micropanels (a large N and a small T). The fixed-effect estimator, the Anderson–Hsiao estimator, the Arellano and Bond estimator, or the system generalized method of moments estimators were conceived to address the design of the micropanel (see Arellano [2003]; Hsiao [2014]; Baltagi [2013]). However, in the late 1990s, the first articles were published to warn that the selection of the estimator crucially depends on the design of the panel (that is, on the relative size of N and T) (Pesaran and Smith 1995; Im, Pesaran, and Shin [IPS] 2003).

In this article, we focus on the econometric treatment of macropanels, known in the literature as "panel time series". We introduce the main attributes of the panel time-series literature, and we present two new commands for evaluating and treating of cross-section dependence (CSD).

2 Panel time series

Concepts such as the purchasing power parity, the savings-to-investment ratio, or the problem of convergence in the theory of growth, among others, have benefited from using panels formed by countries with large T. The fact that T may tend to infinity contributed to the dramatic increase of the literature on panel data. The earliest literature rejected the assumption of homogeneity of slopes, as assumed in standard pooled estimators (fixed effects, difference, or system generalized method of moments), and proposed heterogeneous slopes (that is, a regression per unit[3]) (for example, see Pesaran and Smith [1995]; Pesaran, Shin, and Smith [1999]; and IPS [2003]). This literature is based on a T large enough to estimate each regression separately (that is, a regression per country).

Other literature focused on the time-series methods applied to a panel, dealing with nonstationarity, spurious regressions, and cointegration relationships. This work discussed how including the cross-section dimension in the time dimension offers important advantages when evaluating nonstationarity and cointegration. Confidence in the econometrics of nonstationarity panels lies in combining the best of both worlds: the treatment of nonstationarity according to time-series models and, simultaneously, the possibility to increase the data and the power of the tests based on the cross-section dimension. Particularly, adding the cross-section dimension under specific assumptions

3. See the user-written command `xtmg` by Eberhardt (2012).

may be interpreted as different samples of the same population distribution. By combining the time dimension with the cross-section dimension, one increases the power of the statistical tests, and the estimators may converge in distribution to normal random variables (Baltagi and Kao 2000).

As with the empirical analysis of time-series models, unit-root tests are now a frequent practice in panel models. In the late 1990s, the first panel unit-root tests were developed.[4] Theoretically, these tests make different assumptions about the rates at which the number of units, N, and the numbers of time periods, T, tend to infinity or about whether N or T is fixed. The way in which N and T tend to infinity is critical when determining the asymptotic properties of the estimators and deciding which test is the most appropriate (Phillips and Moon 1999, 2000; LL 1992).[5] The IPS (2003) test is one of the most widely used because it is less restrictive than that of Levin, Lin, and Chu (2002). According to IPS (2003), a sample of N units (countries) over T periods is considered, and the stochastic process y_{it} is generated by the following first-order autoregressive process with initial y_{it} values:

$$y_{it} = (1 - \phi_i)\mu_i + \phi_i y_{i,t-1} + \epsilon_{it} \quad i = 1, \ldots, N \quad t = 1, \ldots, T \tag{1}$$

As in the Dickey–Fuller (DF) test, the interest lies in testing the unit-root null hypothesis of $\phi_i = 1$ for every i (unit). The previous equation may be rewritten as

$$\Delta y_{it} = \alpha_i + \beta_i y_{i,t-1} + \epsilon_{it}$$

where

$$\alpha_i = (1 - \phi_i)\mu_i$$
$$\beta_i = -(1 - \phi_i)$$

and

$$\Delta y_{it} = y_{it} - y_{i,t-1}$$

Then, the unit-root test is

$H_0 : \beta_i = 0$ for all i, versus the alternatives
$H_1 : \beta_i < 0, \quad i = 1, \ldots, N_1, \quad \beta_i = 0, \quad i = N_1 + 1, N_1 + 2, \ldots, N \quad 0 < N_1 \leq N$

The alternative hypothesis allows β_i to differ among units, unlike the homogeneous alternative of the test of Levin, Lin, and Chu (2002), which states $\beta_i = \beta < 0$ for every i. Likewise, the fraction of units following a stationary process is different from zero; that is, $\lim_{N\to\infty}(N_1/N) = \delta$, $0 < \delta \leq 1$. This condition is required for the consistency of the IPS test. The statistic proposed by IPS, $\bar{t}_{\text{IPS}}$, is defined as the average of the unit t statistics of the DF (augmented) regression.

4. The first test was used by Levin and Lin (LL, 1992) and was followed by the IPS (2003), Maddala and Wu (1999), Choi (2001), and Hadri (2000) tests.
5. For a detailed analysis of the asymptotic properties of different panel unit-root tests, see [XT] **xtunitroot**.

Maddala and Wu (1999) agree on the advantage of the heterogeneous alternative of IPS, but they highlight that averaging DF statistics is not the most efficient way to use information. Following Fisher (1932), they propose a statistical test that is an average of the logarithms of the p-values associated with statistic t in each unit. According to his simulations, in several situations, the Maddala and Wu test performs better (both in terms of size and power) than the IPS test, which, in turn, is more powerful than the LL test (Smith and Fuertes 2010). On the other hand, Breitung (2000) uses Monte Carlo simulations to study the power of the LL and IPS tests and finds a dramatic loss of power in both tests when deterministic terms are included.

Breitung and Pesaran (2008) discuss the evolution of panel unit-root tests. They highlight that one of the primary objectives when using these panel tests is to improve the poor performance of time-series unit-root tests. For instance, the augmented DF test generally does not reject the null hypothesis that the real exchange rate is nonstationary. However, the panel unit-root tests applied to an ensemble of industrialized countries generally reject the hypothesis of one unit root—that is, the real exchange rate shows a stationary behavior, empirically supporting the purchasing power parity (Coakley and Fuertes 1997).

Although panel unit-root tests were conceived to correct the lack of power in time-series tests, they also caused several inconveniences. One key assumption of panel unit-root tests is the independence of units, which is an essential condition for the average statistic of unit DFs, $\bar{t}_{\text{IPS}}$, to converge to the normal distribution.[6] Furthermore, if the unit-root null hypothesis is rejected, then interpreting this result becomes more difficult because the best conclusion we may draw is that a fraction of units is stationary. Nothing can be stated about how many units there are or which units are stationary.

One reason why empirical research is concerned about the presence of unit roots in time-series models is to avoid the problem of spurious correlation. As is well known, cointegration is required among variables $I(1)$ for the regression not to be spurious and for the estimator of interest to be consistent. This means that if the variables are cointegrated, then they share a common stochastic trend that is canceled in their linear combination. Pesaran and Smith (1995) indicate that spurious regression does not arise in a cross-section regression when the time dimension collapses, even when the time series of each unit has a unit root.[7] As a result of this observation, the problem of spurious correlation was largely mitigated by averaging the units. Phillips and Moon (1999, 2000), Pedroni (1996, 1997b,a), and Kao and Chiang (2000) show that the mean group (MG) estimator that they propose is more efficient than the estimator given by a cross-section regression.

Panel-cointegration models are used to study long-term economic relationships, which are typical in macroeconomic and financial data analysis. These long-term relationships are frequently predicted by economic theory. Consequently, empirical re-

6. The assumption of independence of the units is critical to meet the requirements of the Linderberg–Levy central limit theorem in the elaboration of the unit-root statistic and of the estimators and tests that are an average of individual relationships (Baltagi and Kao 2000).
7. See the user-written command `xtpmg` by Blackburne and Frank (2007).

search is interested in the estimation of regression coefficients to then evaluate if the theoretical restrictions are satisfied. Kao and Chen (1995) show that the ordinary least-squares (OLS) estimator in the cointegrated panel is asymptotically normal but biased. Chen, McCoskey, and Kao (1999) find that the OLS estimator corrected for bias does not improve the results relative to the general OLS estimator. The authors suggest using the fully modified OLS (FMOLS) estimator or the dynamic OLS estimator. Phillips and Moon (1999) and Pedroni (1996) propose the fully modified (FM) estimator as a generalization of the Phillips and Hansen (1990) estimator. Kao and Chiang (2000) study the limit distribution in a cointegration regression according to the FM estimator and show that it is asymptotically normal.

Likewise, Pedroni (1996) and Phillips and Moon (1999) also obtain similar results for the limit distribution of the FM estimator. Phillips and Moon (1999) analyze the different types of relationships present in nonstationary panels.[8] The authors require that $N/T \to 0$; consequently, the results are valid for panels with moderate N and large T (that is, macropanels) but not for panels with moderate T and large N (typical micropanels). Within possible estimators, the MG estimator proposed by Pesaran and Smith (1995), also called the average long-run estimator by Phillips and Moon (1999, 2000), estimates the time-series model $y_{it} = \eta_i + \lambda_i x_{it} + u_{it}$ for each country and then obtains the $\widehat{\lambda}$, as $\widehat{\lambda} = \sum_i \widehat{\lambda}_i / N$. Likewise, $E(\widehat{\lambda}_i) = \lambda$ represents the average behavior of the countries. The MG estimator is consistent even when the $\widehat{\lambda}_i$ are not.

As it occurs with unit-root tests mentioned above, the key to obtaining the consistent estimators is the independence of cross-section units so as to add information when averaging the estimated parameters that result from the unit time-series analysis, thus mitigating the virtual spurious correlation. Phillips and Moon (1999, 2000) propose a variation for the MG estimator, which they call the FMOLS. Apart from its capacity to consider heterogeneity among the panel units, the FMOLS estimator can control for the bias induced by the potential endogeneity of regressors and the serial correlation and heteroskedasticity of residuals (Pedroni 2000, 2001, 2007).[9]

One interesting result of nonstationary panels is that several statistical tests and estimators converge to the normal distribution. This applies to the above-mentioned distribution of the IPS statistic and to the FM and dynamic OLS estimators (Kao and Chiang 2000). This asymptotic convergence contrasts markedly with the unit-root test behavior and the spurious correlation problems of time-series models.

8. The authors allow the series under analysis to cointegrate or not, and they introduce a framework for the sequential and joint study of the asymptotic theory in nonstationary panels. The panel model considers the following four cases: 1) spurious regression in panel data, where there is no cointegration among series; 2) heterogeneous cointegration in panel data, where each unit has its own cointegration relationship; 3) homogeneous cointegration in panel data; and 4) near-homogeneous cointegration in panel data.
9. While the MG estimator uses parametric short-term dynamics, the FM estimator is based on nonparametric methods to eliminate the effects of dynamics and any type of endogeneity of the residuals on long-term coefficients (Smith and Fuertes 2010). The estimator is highly consistent under cointegration and robust when there are omitted variables that are not part of the cointegration relationship (Pedroni 2007).

The econometric theory developed so far for panel unit-root tests and the asymptotic convergence to normal distribution of the proposed estimators that entail heterogeneous slopes was based on the independence of the units (countries) of the panel, a situation rarely seen in the empirical study of macropanels. The lack of independence among units is known in the literature as CSD, and its presence is natural in the study of these types of data, such as the global economic and financial cycle through the globalization of economic activity, the common trade areas, technological progress, and the spillover effects. Disregarding the CSD—that is, some correlation structure in the error term between units due to unobservable common factors—squanders the efficiency gains of operating with a panel and leads to inconsistent estimators of the parameters, thus invalidating the theoretical inference in panel-data models (Kapetanios, Pesaran, and Yamagata 2011; Banerjee and Carrion-i-Silvestre 2011). Note that unobservable common factors are nothing but variables that have been omitted in the specification of the model to be fit.

Empirical researchers first incorporated time dummies to deal with the lack of independence among units and to remove unobservable common factors. Nevertheless, this solution assumed slopes to be homogeneous; that is, $\lambda_i = \lambda$. Another proposal consisted in deducting the corresponding mean from each variable (that is, $\widetilde{y}_{it} = y_{it} - \overline{y}_t$, where $\overline{y}_t = \sum_{i=1}^{N} y_{it}/N$) and similarly from regressors, x_{it}. This process is known as de-meaning. But once again, one can prove that this potential solution for estimating unobservable factors requires an assumption of homogeneity in the impact of unobservable factors on units.

To model the CSD, one estimates the unobservable common factors by using the principal components techniques (Coakley, Fuertes, and Smith 2002; Bai 2004; Bai and Ng 2004). Pesaran (2006) refutes the proposal of principal components for estimating the CSD made by Coakley, Fuertes, and Smith (2002) and shows that a linear combination of unobservable common factors can be proxied using the averages on the units of the model regressors and of the dependent variable. This led to a new set of estimators known in the literature as the common correlated effect (CCE).[10] CCE essentially consists of increasing the model to be fit by including the average of the units in each t of time, both of the dependent variable and of the specific regressors of each unit.

One can illustrate different intensities in the types of CSD manifestations such as neighborhood effects, network effects, the influence of a dominant unit, or simply unobservable common factors.[11] According to Pesaran (2006), the econometric model can be represented as

$$\begin{aligned} y_{it} &= \eta_i' z_t + \lambda_i' x_{it} + e_{it} \quad i = 1, \ldots, N \quad t = 1, \ldots, T \\ e_{it} &= \gamma_i' f_t + \epsilon_{it} \end{aligned} \tag{2}$$

10. See the user-written command `xtcce` for CCE estimation for static and dynamic panels with cross-sectional dependence.

11. There are two types of CSDs: weak and strong. The weak CSD implies that dependencies are of a local nature and decline with N. This may be true of spatial correlations, where each unit is correlated with only its neighbors, while the strong CSD implies that the dependence affects all the units.

where y_{it} is the observation of the ith unit at moment **t**; z_t is a $k_z \times 1$ vector of the variables that do not differ over units (that is, the y intercept, trend, or seasonal dummies); x_{it} is a $k_x \times 1$ vector of observable regressors specific to each unit at moment t; f_t is a $r \times 1$ vector of unobservable factors that may affect each unit differently and may be correlated with x_{it}; and ϵ_{it} is the unobservable disturbance with $E(\epsilon_{it}) = 0$, $E(\epsilon_{it}^2) = \sigma_i^2$, which is independently distributed through i and t. The covariance between errors, e_{it}, is determined by the loading factor, γ_i. If f_t is correlated with x_{it}—as is generally the case in many empirical applications, such as global cycles— then disregarding the CSD by omitting factor f_t results in biased and inconsistent λ_i estimators.

Pesaran (2006) proposed to treat the unobservable factors as the nuisance parameters that we want to control for to get a better estimate of λ_i. The estimator proposed, the CCE, seeks to enrich the model to be fit by including cross-section averages in each t of time to control for the unobservable factors. This involves both independent and dependent variables, as follows:

$$y_{it} = \eta_i' z_t + \lambda_i' x_{it} + \delta_{oi} \overline{y}_t + \delta_i' \overline{x}_t + u_{it}$$

To understand the motivation of this procedure, let's assume one single factor and make an average of (2) on the units:

$$\overline{y}_t = \overline{\eta}' z_t + \overline{\lambda}' \overline{x}_t + \overline{\gamma} f_t + \overline{\epsilon}_t + \frac{1}{N} \sum (\lambda_i - \overline{\lambda})' x_{it}$$

$$f_t = \overline{\gamma}^{-1} \left[\overline{y}_t - \left\{ \overline{\eta}' z_t + \overline{\lambda}' \overline{x}_t + \overline{\epsilon}_t + \frac{1}{N} \sum \left(\lambda_i - \overline{\lambda} \right)' x_{it} \right\} \right]$$

Then, $\overline{y}_t$ and $\overline{x}_t$ operate as proxy of the unobservable factor. Note that the covariance between $\overline{y}_t$ and ϵ_{it} tends to zero with N; consequently, for a large N, there are no endogeneity problems. This formulation presupposes the existence of heterogeneous coefficients, but there are also homogeneous versions of them (see Eberhardt and Teal [2011]). This idea of incorporating the averages to the regressions per country is also used by Pesaran (2007) to immunize the IPS unit-root test against the presence of unobservable factors. These unit-root tests that control for the CSD are known as second-generation tests.

In addition to the heterogeneity on the observable regressors, Pesaran (2006) permits i) that unobservable common effects may affect units differently; ii) that errors per unit may show a serial correlation and heteroskedasticity; and iii) that it is not necessary for individual-specific regressors to be identical or to be distributed independently through the individuals, a relevant feature in country-panel analysis. However, Pesaran (2006) presupposes that both individual-specific regressors and common unobservable factors are stationary and exogenous. Kapetanios, Pesaran, and Yamagata (2011) extend this analysis and include $I(1)$ processes of the individual-specific regressors and of the unobservable effects. The extension is far from trivial and resorts to very different intermediate results to obtain the asymptotic distribution of the estimators when data are $I(1)$ versus when data are $I(0)$. But, surprisingly, Monte Carlo simulations suggest that the CCE method proposed by Pesaran (2006) to address CSD is robust for many

data-generating processes. This result is quite dissimilar in terms of the substantial differences inherent in time-series models for the distribution of $I(1)$ processes versus $I(0)$ processes.

Although this second generation of unit-root tests considered the lack of independence of the units when admitting the presence of unobservable common factors, it led to new challenges when interpreting both the unit-root test and the cointegration test (Breitung and Pesaran 2008). These unobservable common factors may exhibit a stationary behavior (for example, global economic cycles) or a nonstationary behavior (for example, global technological progress). If the unobservable factor exhibits a $I(1)$ behavior (that is, it reveals a unit root), then we should consider the possibility that this factor may cointegrate inside each unit and also between units. This is why the interpretation of the second-generation unit-root tests differs from the standard interpretation of a unit-root test.

Let's go back to (1), but we will now consider the unobservable factor:

$$y_{it} = (1 - \phi_i)\mu_i + \phi_i y_{i,t-1} + \gamma_i f_t + \epsilon_{it} \quad i = 1, \ldots, N \quad t = 1, \ldots, T$$

Rejecting the null hypothesis $\phi_i = 1$ in favor of the alternative $\phi_i < 1$ might be for very different reasons. It might be due to i) both y_{it} and f_t being stationary processes or to ii) y_{it} and f_t being $I(1)$ and cointegrated. And this is independent from the method used to account for the CSD.

Let's return to the solution offered by Pesaran (2006), the CCE, to control for the presence of the CSD, which was extended to the unit-root tests by Pesaran (2007) and Pesaran, Smith, and Yamagata (2013). The relevant equation to evaluate the presence of a unit root is

$$\Delta y_{it} = \alpha_i + \beta_i y_{i,t-1} + \delta_{0i}\overline{\Delta y_t} + \delta_{1i}\overline{y}_{t-1} + \epsilon_{it}, \quad i = 1, \ldots, N \quad t = 1, \ldots, T \qquad (3)$$

that is, the conventional equation augmented by the averages of the units of both the regressor, y_{it}, and the dependent variable Δy_{it}. The hypothesis would consist in evaluating $\beta_i = 0$ using a panel test. Like the IPS (2003), the proposal of Pesaran (2007) consists in averaging the t_i statistics corresponding to β_i of (3). The new statistic, called the cross-sectional Im, Pesaran, and Shin (CIPS) by Pesaran, has a nonstandard distribution, even with a large N. This is different from the result obtained by IPS (2003), in which, under the assumption of the independence of units, the IPS statistic is distributed according to a normal distribution for a large N.

Note that (3) might be considered a correction-toward-equilibrium model, where $\overline{y}_t$ and y_{it} might be $I(1)$ despite $\beta_i < 0$, simply because they are cointegrated. The latter discourages the use of a panel unit-root test because interpretation can become more difficult. Under both H_0 and the alternative, we would face joint hypotheses. Under H_0, a simultaneous evaluation reveals that all units are $I(1)$ and that they do not cointegrate, while the alternative reveals that $\beta_i < 0$, with the possibility that $y_{it} \sim I(1)$ may cointegrate with the unobservable factor.

To summarize, this new approach of time-series panel econometrics combines the two discussed lines of work proposed by the end of the 1990s. The questioning of parameter homogeneity of a macropanel model may come from the impact of both the observable factors (the regressors) and unobservable factors (the factor loadings). Ignoring the potential heterogeneity of both observable regressors and unobservable factors may have more serious implications if the observable variables or the unobservable factors are nonstationary. The following example illustrates this: an equation in levels estimated through a standard pooled estimator imposes common parameters for all countries and, at the same time, leads to nonstationary errors if the true parameters of the model are heterogeneous and the variables are nonstationary (Eberhardt and Teal 2011). Specifically, disregarding the heterogeneity of the model's observable regressor's parameters may disrupt the cointegration relationship between the regressors and the dependent variable and may produce spurious results (Smith and Fuertes 2010). Similarly, an equation in levels estimated through a standard pooled estimator and augmented with $T-1$ dummy variables imposes a common evolution of unobservable factors to all countries, which creates stationary errors if the true unobservable factors exhibit a nonstationary behavior.

3 Evaluation of CSD: xtcsi

Although the CSD is a fact rather than an exception in macropanels, there are several tests for its evaluation. The Lagrange multiplier (LM) test of Breusch and Pagan (1980) could be one such test. It consists in the average of the squared pairwise correlation coefficients of the residuals and was designed using apparently nonrelated equations, or seemingly unrelated regression (Zellner 1962), with a fixed N and $T \to \infty$ (that is, a small N relative to T). Pesaran (2004) shows that the LM test exhibits serious size distortions when N is large relative to T, a situation easily observed in many empirical applications. To overcome the LM test bias, Pesaran (2004) proposes another test that he calls CD, which consists in averaging the pairwise correlations of the residuals. Under the null hypothesis, for a sufficiently large T, the CD statistic converges in distribution to $N(0,1)$, when $N \to \infty$. However, as Pesaran (2004) notes, the CD test may be inconsistent in several relevant alternatives.

Based on Breusch–Pagan's LM test, Pesaran, Ullah, and Yamagata (2008) propose a new LM test that corrects the bias of the previous one in panels with strictly exogenous regressors and normal errors. Monte Carlo simulations analyze the power and size of the three available statistics. The authors conclude that the bias-adjusted LM test successfully controls for the size of the test and keeps a reasonable power.

Following Pesaran, Ullah, and Yamagata (2008), we have designed the `xtcsi` command, which computes the above mentioned three statistics as follows:

Consider the following panel-data model

$$y_{it} = \lambda_i' x_{it} + u_{it}, \text{ for } i = 1, 2, \ldots, N; \quad t = 1, 2, \ldots, T \tag{4}$$

where the $X_i = (x_{i1}, \ldots, x_{iT})'$ matrix of regressors may contain the unit vector for the constant in the first column and a trend in the second column.[12] For each i, $u_{it} \sim \text{IIDN}(0, \sigma_{ui}^2)$. For every t, however, they might be cross-section correlated.

Breusch and Pagan (1980) propose the following LM statistic for testing the null of zero cross-equation error correlations:

$$\text{LM} = T \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \widehat{\rho}_{ij}^2$$

Here $\widehat{\rho}_{ij}$ is the sample estimate of the pairwise correlation of the residuals arising from the Monte Carlo optimization estimate of the regression for each unit of the panel.

Under the null hypothesis,

$$H_0 : \text{Cov}\,(u_{it}, u_{jt}) = 0 \quad \text{for all } t \text{ and } i \neq j \tag{5}$$

The LM statistic is distributed asymptotically as a chi-squared with $N(N-1)/2$ degrees of freedom. However, the LM test may exhibit substantial distortions of size for large N and small T, a frequent situation in empirical applications.

Ullah (2004) offers unified techniques to obtain the exact and approximate moments of the econometric estimators and statistical tests. Pesaran, Ullah, and Yamagata (2008) use this approach to correct the bias in small samples of the LM statistic.

Assume the following:

Assumption 1: For each i, the disturbances, u_{it}, are serially independent with the mean 0 and the variance $0 < \sigma_i^2 < \infty$.

Assumption 2: Under the null hypothesis defined by $H_0 : u_{it} = \sigma_i \epsilon_{it}$, $\epsilon_{it} \sim \text{IIDN}(0, 1)$ for all i and t.

Assumption 3: The regressors, x_{it}, are strictly exogenous such that $E(u_{it}/X_i) = 0$ for all i and t, where $X_i = (x_{i1}, \ldots, x_{iT})'$ and $X_i' X_i$ is a positive defined matrix.

The authors introduce the following idempotent matrix of rank $T - k$:

$$M_i = I_T - H_i; \quad H_i = X_i (X_i' X_i)^{-1} X_i'$$

Considering (4) and under assumptions 1 to 3, the exact mean and variance of $(T-k)\widehat{\rho}_{ij}^2$ are, respectively, given by

$$\mu_{Tij} = E\left\{(T-k)\widehat{\rho}_{ij}^2\right\} = \frac{1}{T-k} Tr\left\{E(M_i M_j)\right\} \tag{6}$$

12. We include the constant and the trend in the X_i matrix [unlike that illustrated in (2), where both appeared explicitly in the Z_t matrix] to simplify the matrix calculation used to obtain the LM_{adj} statistic.

and

$$v_{Tij}^2 \text{Var}\left\{(T-k)\widehat{\rho}_{ij}^2\right\} = \left[Tr\left\{E(M_i M_J)\right\}\right]^2 a_{1T} + 2Tr\left\{E\lfloor (M_i M_j)^2 \rfloor\right\} a_{2T} \tag{7}$$

where

$$a_{1T} = a_{2T} - \frac{1}{(T-k)^2}, \quad a_{2T} = 3\left\{\frac{(T-k-8)(T-k+2)+24}{(T-k+2)(T-k-2)(T-k-4)}\right\}^2$$

Using (6) and (7), we define the bias-adjusted LM statistical test as

$$\text{LM}_{\text{adj}} = \sqrt{\frac{2}{N(N-1)}} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{(T-k)\widehat{\rho}_{ij}^2 - \mu_{Tij}}{v_{Tij}} \tag{8}$$

Under assumptions 1 to 3 and assuming that H_0 is defined by (5), $T \to \infty$ and then $N \to \infty$, we have

$$\text{LM}_{\text{adj}} \to_d N(0,1)$$

To deal with the large N bias of the LM test, Pesaran (2004) suggests using the CD statistic defined by

$$\text{CD} = \sqrt{\frac{2T}{N(N-1)}} \left(\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \widehat{\rho}_{ij} \right)$$

and shows that under H_0 and for sufficiently large T, CD $\to_d N(0,1)$ as $N \to \infty$.

The test proposed by Pesaran, Ullah, and Yamagata (2008) is reported for balanced panels. According to Yamagata, the bias-adjusted test to the unbalanced panels case can be extended as follows: suppose we are considering two units (unit i and unit j), and suppose that y_{it} and x_{it} are observed between $[\text{date}_1, \text{date}_2]$, that y_{jt} and x_{jt} are observed between $[\text{date}_1', \text{date}_2']$, and that both have some overlapping period $[\text{date}_1'', \text{date}_2'']$, where $\text{date}_1'' = \max(\text{date}_1, \text{date}_1')$ and $\text{date}_2'' = \min(\text{date}_2, \text{date}_2')$. Then, the $\widehat{\rho}_{ij}$ will be based on the residual regressions of y_{it} on x_{it} and y_{jt} on x_{jt} between the overlapping sample. Thus mean and variance adjustments as well as "T" in (8) should be adjusted appropriately (that is, these can be different for each combination of i and j).

3.1 Syntax

`xtcsi` *depvar indepvars* [*if*] [*in*] [`, trend`]

3.2 Description

`xtcsi` implements several error cross-section independence tests in (balanced) heterogeneous panels. These tests include the LM test by Breusch and Pagan (1980); the bias-adjusted LM test by Pesaran, Ullah, and Yamagata (2008); and the CD test by Pesaran (2004).

3.3 Option

`trend` specifies a linear trend to be included in each individual regression model.

3.4 Stored results

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

Scalars

`r(N_g)`	number of units of the panel
`r(lm)`	Breusch and Pagan (1980) LM test statistic
`r(p_lm)`	p-value of chi-squared with $N(N-1)/2$ degrees of freedom
`r(lm_adj)`	Pesaran, Ullah, and Yamagata (2008) bias-adjusted LM test statistic
`r(p_lm_adj)`	two-sided p-value of normal $(0, 1)$
`r(lm_cd)`	Pesaran (2004) CD test statistic
`r(p_lm_cd)`	two-sided p-value of normal $(0, 1)$

3.5 Empirical example: xtcsi

Here we illustrate `xtcsi`. To evaluate the null hypothesis of no correlation among units, we take data from Katz (2014) on the determinants of foreign direct investment flows in eight Latin American countries during 1981–2012. Following his article, the dependent variable is the foreign direct investment as a percentage of gross domestic product (GDP) (`lfdi_gdp_1`), while the regressors are the change in the GDP per capita (`dlgdp_pc`), the change in the consumer price index (`infla`), the real exchange rate (`ltcr`), the terms of trade (`ltot`), and the rule of law (`rule_of_law`) and openness (`laper_gdp`). All variables are in logs but rule of law.

```
. use b_csd.dta
. tsset id year, y
       panel variable:  id (strongly balanced)
        time variable:  year, 1981 to 2012
                delta:  1 year
. xtcsi lfdi_gdp_1 dlgdp_pc infla ltcr ltot rule_of_law_1 laper_gdp, trend
Bias-adjusted LM test of error cross-section independence
H0: Cov(uit,ujt) = 0 for all t and i!=j
```

Test	Statistic	p-value
LM	25.9	0.5786
LM adj*	-2.077	0.0378
LM CD*	.6665	0.5051

```
*two-sided test
```

Test results show that while LM and LM CD cannot reject the null hypothesis of no correlation among the countries for all t, the bias-adjusted LM test by Pesaran, Ullah, and Yamagata (2008) rejects it at a confidence level of 3.8%

4 IPS test in the presence of the CSD: xtcips

Following Pesaran (2007), we used Stata to implement the CIPS test and the CIPS* test (a truncated alternative of the CIPS), which we called `xtcips`. As previously mentioned, the limit distribution of the CIPS statistic is not normal, and the corresponding critical values are tabulated in Pesaran (2007). The command is designed for balanced panels but may be adapted for unbalanced panels, although this would imply making a Monte Carlo simulation to obtain the critical values according to the panel structure. This is precisely what Bebczuk, Burdisso, and Sangiácomo (2012) do.

A simple guideline for obtaining the critical values for an unbalanced panel is to replicate the panel's structure. For example, suppose there are N cross-section units with a varying number of observations, $T_1, T_2 \dots T_N$. We then perform the following steps:

1. Generate a $T_1, T_2 \dots T_N$ nonstationary series, $y_1, y_2 \dots y_n$. For example, $y_{1t} = y_{1t-1} + u_{1t}$ $t = 1, 2, \dots, T_1$, and u_1 is from a standard normal process.
2. Compute CADF$_i$ for each unit.
3. Compute the t-bar (the average of CADF$_i$) according to Pesaran (2007) and keep it.
4. Repeat steps 1–3 10,000 times.
5. Obtain the 1%, 5%, and 10% quantiles for the sample distribution of t-bars.

Nevertheless, as mentioned previously, the command is designed for balanced panels.

4.1 Syntax

`xtcips` *varname* [*if*] [*in*] , `maxlags(#)` `bglags(`*numlist*`)` [`q` `trend` `noc`]

`xtcips` is used with balanced panel data. You must `tsset` your data before using `xtcips` with the panel form of `tsset`; see `help tsset`. *varname* may contain time-series operators; see `help tsvarlist`.

4.2 Description

`xtcips` estimates the unit-root CIPS test in heterogeneous (balanced) panels developed by Pesaran (2007, sec. 4, 275–279).

There are three possible specifications:

Case I: models without an intercept or trend (see the `noc` option)

Case II: models with an individual-specific intercept (default)

Case III: models with an incidental linear trend (see the `trend` option)

The command allows the user to define individual dynamic specifications in each regression using two alternative criteria (see the `maxlags(#)` option):

i) the Wald test of compound linear hypothesis on the model parameters (default)

ii) the Portmanteau test (Q) of white noise (see the `q` option)

`xtcips` reports the p-value of the LM test on the Breusch–Godfrey serial correlation of each individual regression (see the `bglags()` option).

The null hypothesis is (homogeneous nonstationary)

$$H_0\colon\ \beta_i = 0 \text{ for all } i$$

versus the alternatives

$$H_1\colon\ \beta_i < 0, \quad i = 1, \ldots, N_i, \quad \beta_i = 0, \quad i = N_1 + 1,\ N_1 + 2, \ldots, N$$

in the following regression of the cross-section augmented DF (CADF) test:

$$\Delta y_{it} = \alpha_i + \beta_i y_{i,t-1} + \delta_{0i}\Delta\overline{y}_t + \delta_{1i}\overline{y}_{t-1} + \epsilon_{it}, \quad i = 1, \ldots, N \quad t = 1, \ldots, T$$

4.3 Options

`maxlags(#)` defines the individual dynamic specification and specifies the maximum number of lags to be included in the model to be estimated for each unit. Then, `xtcips` determines the number of lags to be included in each individual regression with an iterative process of 0 to `maxlags()` based on the level of significance of the test established to select the dynamics. This may be done by selecting the highest significant lag, either i) by rejecting H_0 (at 5% or lower) in the Wald test[13] or ii) by not rejecting H_0 (at 95% or higher) in the Portmanteau test (Q) of white noise or `maxlags()`, whichever occurs first. `maxlags()` is required and must be a positive integer.

`bglags(`*numlist*`)` establishes the serial correlation order to be tested in the LM test by Breusch–Godfrey in each individual regression. If only one value is provided (a positive integer), then that order is used for all units. If a list of numbers is provided, its length must match the number of units in the panel. `bglags()` is required.

13. The model is estimated with the number of lags specified according to `maxlags(L)`, and the steps for performing the Wald test are as follows. Test the null hypothesis of the test $H_0: \delta_{0i}^1 = \delta_{0i}^2 = \delta_{0i}^3 = \cdots = \delta_{0i}^L = \delta_{1i}^1 = \delta_{1i}^2 = \delta_{1i}^3 = \cdots = \delta_{1i}^L = 0$. If H_0 is not rejected, then the specification suggested would be the standard DF without augmenting. If H_0 is rejected, then $H_0: \delta_{0i}^2 = \delta_{0i}^3 = \cdots = \delta_{0i}^L = \delta_{1i}^2 = \delta_{1i}^3 = \cdots = \delta_{1i}^L = 0$ is tested. If H_0 is not rejected, the specification suggested would be the DF augmented by one lag. If H_0 is rejected, the same procedure applies until the maximum lag for the specified `maxlags(L)` is determined.

`q` establishes the Portmanteau (Q) test of white noise as the criterion to determine the dynamic specification.

`trend` includes a time trend in the estimated equation (case III).

`noc` eliminates the constant term (case I).

4.4 Stored results

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

Scalars
- `r(cips)` CIPS statistic

Matrices
- `r(cv)` critical values of average of individual cross-sectionally augmented DF distribution
- `r(W)` individual regression diagnostics

4.5 Empirical example: xtcips

In this section, we illustrate `xtcips`. With the same dataset used in section 3.5, we evaluate the presence of a unit root for the log of foreign direct investment as %GDP (`lfdi_gdp_1`) when we consider the CSD.

```
. use b_csd.dta, clear
. tsset id year, y
       panel variable:  id (strongly balanced)
        time variable:  year, 1981 to 2012
                delta:  1 year
. xtcips lfdi_gdp_1, maxl(5) bglag(1) trend
Pesaran Panel Unit Root Test with cross-sectional and first difference mean
> included for lfdi_gdp_1
Deterministics chosen: constant & trend
Dynamics: lags criterion decision General to Particular based on F joint test
H0 (homogeneous non-stationary): bi = 0 for all i
CIPS =    -3.920        N,T = (8,32)
```

	10%	5%	1%
Critical values at	-2.71	-2.86	-3.15

As can be seen, the statistic value is -3.92, which is below the critical value at the 1% significance level. Thus this second-generation test rejects the null hypothesis of a unit-root process for the foreign direct investment as a percentage of GDP.

5 Acknowledgment

We thank Ricardo Bebczuk for his inspiring ideas and motivating support. We also thank an anonymous referee for his helpful suggestions on a previous version of this article.

6 References

Arellano, M. 2003. *Panel Data Econometrics*. Oxford: Oxford University Press.

Bai, J. 2004. Estimating cross-section common stochastic trends in nonstationary panel data. *Journal of Econometrics* 122: 137–183.

Bai, J., and S. Ng. 2004. A PANIC attack on unit roots and cointegration. *Econometrica* 72: 1127–1177.

Baltagi, B. H. 2013. *Econometric Analysis of Panel Data*. 5th ed. New York: Wiley.

Baltagi, B. H., and C. Kao. 2000. Nonstationary panels, cointegration in panels and dynamic panels: A survey. In *Advances in Econometrics: Vol. 15—Nonstationary Panels, Panel Cointegration, and Dynamic Panels*, ed. B. H. Baltagi, 7–52. New York: Elsevier.

Banerjee, A., and J. L. Carrion-i-Silvestre. 2011. Testing for panel cointegration using common correlated effects estimators. Discussion Paper No. 11-16, Deparment of Economics, University of Birmingham. ftp://ftp.bham.ac.uk/pub/RePEc/pdf/11-16.pdf.

Bebczuk, R., T. Burdisso, and M. Sangiácomo. 2012. Credit vs. payment services: Financial development and economic activity revisited. Working Paper 56, BCRA. http://www.bcra.gov.ar/pdfs/investigaciones/WP_56_2012i.pdf.

Blackburne, E. F., III, and M. W. Frank. 2007. Estimation of nonstationary heterogeneous panels. *Stata Journal* 7: 197–208.

Breitung, J. 2000. The local power of some unit root tests for panel data. In *Advances in Econometrics: Vol. 15—Nonstationary Panels, Panel Cointegration, and Dynamic Panels*, ed. B. H. Baltagi, 161–178. New York: Elsevier.

Breitung, J., and M. H. Pesaran. 2008. Unit roots and cointegration in panels. In *The Econometrics of Panel Data: Fundamentals and Recent Developments in Theory and Practice*, ed. L. Mátyás and P. Sevestre, 3rd ed., 279–322. Verlag: Springer.

Breusch, T. S., and A. R. Pagan. 1980. The Lagrange multiplier test and its applications to model specification in econometrics. *Review of Economic Studies* 47: 239–253.

Chen, B., S. K. McCoskey, and C. Kao. 1999. Estimation and inference of a cointegrated regression in panel data: A Monte Carlo study. *American Journal of Mathematical and Management Sciences* 19: 75–114.

Choi, I. 2001. Unit root tests for panel data. *Journal of International Money and Finance* 20: 249–272.

Coakley, J., and A. M. Fuertes. 1997. New panel unit root tests of PPP. *Economics Letters* 57: 17–22.

Coakley, J., A.-M. Fuertes, and R. Smith. 2002. A principal components approach to cross-section dependence in panels. https://www.diw.de/sixcms/detail.php/39312.

Eberhardt, M. 2012. Estimating panel time-series models with heterogeneous slopes. *Stata Journal* 12: 61–71.

Eberhardt, M., and F. Teal. 2011. Econometrics for grumblers: A new look at the literature on cross-country growth empirics. *Journal of Economic Surveys* 25: 109–155.

Fisher, R. A. 1932. *Statistical Methods for Research Workers*. 4th ed. Edinburgh: Oliver & Boyd.

Hadri, K. 2000. Testing for stationarity in heterogeneous panel data. *Econometrics Journal* 3: 148–161.

Hsiao, C. 2014. *Analysis of Panel Data*. 3rd ed. Cambridge: Cambridge University Press.

Im, K. S., M. H. Pesaran, and Y. Shin. 2003. Testing for unit roots in heterogeneous panels. *Journal of Econometrics* 115: 53–74.

Kao, C., and B. Chen. 1995. On the estimation and inference for cointegration in panel data when the cross-section and time series dimensions are comparable. Manuscript, Center for Policy Research, Syracuse University.

Kao, C., and M.-H. Chiang. 2000. On the estimation and inference of a cointegrated regression in panel data. In *Advances in Econometrics: Vol. 15—Nonstationary Panels, Panel Cointegration, and Dynamic Panels*, ed. B. H. Baltagi, 179–222. New York: Elsevier.

Kapetanios, G., M. H. Pesaran, and T. Yamagata. 2011. Panels with non-stationary multifactor error structures. *Journal of Econometrics* 160: 326–348.

Katz, S. 2014. Análisis de los determinantes de los flujos de inversión extranjera directa (IED) en América Latina. Mimeo: Banco Interamericano de Desarrollo.

Levin, A., and C. F. Lin. 1992. Unit root test in panel data: Asymptotic and finite sample properties. Unpublished manuscript, University of California, San Diego.

Levin, A. T., C.-F. Lin, and C.-S. J. Chu. 2002. Unit root tests in panel data: Asymptotic and finite-sample properties. *Journal of Econometrics* 108: 1–24.

Maddala, G. S., and S. Wu. 1999. A comparative study of unit root tests with panel data and a new simple test. *Oxford Bulletin of Economics and Statistics* 61: 631–652.

Pedroni, P. 1996. Fully modified OLS for heterogeneous cointegrated panels and the case of purchasing power parity. Working Paper No. 96-020, Department of Economics, Indiana University.

———. 1997a. Cross sectional dependence in cointegration tests of purchasing power parity in panels. Working Paper, Department of Economics, Indiana University.

———. 1997b. Panel cointegration: Asymptotic and finite sample properties of pooled time series tests with an application to the PPP hypothesis. Working Paper No. 2004-15, Department of Economics, Williams College.

———. 2000. Fully modified OLS for heterogeneous cointegrated panels. In *Advances in Econometrics: Vol. 15—Nonstationary Panels, Panel Cointegration, and Dynamic Panels*, ed. B. H. Baltagi, 93–130. New York: Elsevier.

———. 2001. Purchasing power parity tests in cointegrated panels. *Review of Economics and Statistics* 83: 727–731.

———. 2007. Social capital, barriers to production and capital shares: Implications for the importance of parameter heterogeneity from a nonstationary panel approach. *Journal of Applied Econometrics* 22: 429–451.

Pesaran, M. H. 2004. General diagnostic tests for cross section dependence in panels. Cambridge Working Papers in Economics No. 0435, University of Cambridge, Faculty of Economics.

———. 2006. Estimation and inference in large heterogeneous panels with a multifactor error structure. *Econometrica* 74: 967–1012.

———. 2007. A simple panel unit root test in the presence of cross-section dependence. *Journal of Applied Econometrics* 22: 265–312.

Pesaran, M. H., Y. Shin, and R. P. Smith. 1999. Pooled mean group estimation of dynamic heterogeneous panels. *Journal of the American Statistical Association* 94: 621–634.

Pesaran, M. H., L. V. Smith, and T. Yamagata. 2013. Panel unit root tests in the presence of a multifactor error structure. *Journal of Econometrics* 175: 94–115.

Pesaran, M. H., and R. P. Smith. 1995. Estimating long-run relationships from dynamic heterogeneous panels. *Journal of Econometrics* 68: 79–113.

Pesaran, M. H., A. Ullah, and T. Yamagata. 2008. A bias-adjusted LM test of error cross-section independence. *Econometrics Journal* 11: 105–127.

Phillips, P. C. B., and B. E. Hansen. 1990. Statistical inference in instrumental variables regression with I(1) processes. *Review of Economics Studies* 57: 99–125.

Phillips, P. C. B., and H. R. Moon. 1999. Linear regression limit theory for nonstationary panel data. *Econometrica* 67: 1057–1111.

———. 2000. Nonstationary panel data analysis: An overview of some recent developments. *Econometric Reviews* 19: 263–286.

Smith, R. P., and A. M. Fuertes. 2010. Panel time series. Mimeo: Centre for Microdata Methods and Practice.

Ullah, A. 2004. *Finite Sample Econometrics*. Oxford: Oxford University Press.

Zellner, A. 1962. An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias. *Journal of the American Statistical Association* 57: 348–368.

About the authors

Tamara Burdisso holds a position as Senior Analyst at the Economic Research Department of Banco Central de la República Argentina and is associate professor of Statistics at the Universidad de Buenos Aires. She specializes in macroeconometrics.

Máximo Sangiácomo holds a position as Chief of Economic Research at Central Bank of Argentina and is teacher assistant of Political Economics II at Universidad Nacional de La Plata. He specializes in topics related to the structure, functioning and regulation of financial markets.

The Stata Journal (2016)
16, Number 2, pp. 443–463

Reference-based sensitivity analysis via multiple imputation for longitudinal trials with protocol deviation

Suzie Cro
MRC Clinical Trials Unit at UCL
London School of Hygiene and Tropical Medicine
London, UK
s.cro@ucl.ac.uk

Tim P. Morris
MRC Clinical Trials Unit at UCL
London School of Hygiene and Tropical Medicine
London, UK
tim.morris@ucl.ac.uk

Michael G. Kenward
London School of Hygiene and Tropical Medicine
London, UK
mike.kenward@lshtm.ac.uk

James R. Carpenter
MRC Clinical Trials Unit at UCL
London School of Hygiene and Tropical Medicine
London, UK
james.carpenter@lshtm.ac.uk

Abstract. Randomized controlled trials provide essential evidence for the evaluation of new and existing medical treatments. Unfortunately, the statistical analysis is often complicated by the occurrence of protocol deviations, which mean we cannot always measure the intended outcomes for individuals who deviate, resulting in a missing-data problem. In such settings, however one approaches the analysis, an untestable assumption about the distribution of the unobserved data must be made. To understand how far the results depend on these assumptions, the primary analysis should be supplemented by a range of sensitivity analyses, which explore how the conclusions vary over a range of different credible assumptions for the missing data. In this article, we describe a new command, `mimix`, that can be used to perform reference-based sensitivity analyses for randomized controlled trials with longitudinal quantitative outcome data, using the approach proposed by Carpenter, Roger, and Kenward (2013, *Journal of Biopharmaceutical Statistics* 23: 1352–1371). Under this approach, we make qualitative assumptions about how individuals' missing outcomes relate to those observed in relevant groups in the trial, based on plausible clinical scenarios. Statistical analysis then proceeds using the method of multiple imputation.

Keywords: st0440, mimix, clinical trial, protocol deviation, missing data, multiple imputation, sensitivity analysis

1 Introduction

Randomized controlled trials that collect longitudinal response data are widely used in medical research because they provide essential evidence for the evaluation of new and existing treatments. Unfortunately, protocol deviations—such as treatment withdrawal, unblinding, or loss to follow-up—are unavoidable during the full course of a trial. Consequently, we often cannot measure what we intended for deviating individuals. Planned outcomes may be unobtainable because of the type of deviation. In addition, depending on the nature of the analysis, even values that were recorded postdeviation may be best regarded as missing. The result is a missing-data problem, complicating the analysis.

Complexity arises because—as in any analysis with missing data—we are forced to make an assumption about the distribution of the unobserved data that crucially cannot be verified from the observed data. Therefore, to understand how far the results depend on these assumptions, the primary analysis should be supplemented by a range of sensitivity analyses, which explore how the conclusions vary over a range of different credible assumptions for the missing data (White et al. 2011).

The importance of sensitivity analysis in this context is highlighted in recent regulatory guidelines from the European Medicines Agency (Committee for Medicinal Products for Human Use 2010) and the U.S. National Research Council (2010), which recommends that "examining sensitivity to the assumptions about the missing data mechanism should be a mandatory component of reporting." Ideally, inferences will be stable across sensitivity analyses, indicating that the impact of the missing data does not seriously affect the interpretation of results. However, it is even more important to report the results of sensitivity analyses when they are contradictory.

When framing a sensitivity analysis, we need to consider carefully both the quantity we wish to estimate and the population for which we wish to estimate it. Following the National Research Council (2010) report, the term estimand is used to describe both the target of inference and the population in which this is estimated. Thus, with missing observations, we need to specify the statistical distribution of individuals' postdeviation responses. This is often done by specifying one or more parameters that relate individuals' predeviation and postdeviation data (for example, see Carpenter and Kenward [2013, chap. 10]). However, with reference-based sensitivity analysis, such statements are made by reference to other groups of individuals in the trial (typically to individuals in different treatment arms), obviating the need for explicit parameter specification (Carpenter, Roger, and Kenward 2013).

Before describing this approach further, we follow Carpenter, Roger, and Kenward (2013) and distinguish between two main classes of estimands. The first considers the estimated treatment effect when we assume that postdeviation individuals continue to follow the trial "rules"—that is, abide by the protocol. This is referred to as a de jure

estimand. The second explores robustness of inferences to various assumptions about what might have happened—in other words, various de facto scenarios.

Under both de jure and de facto assumptions, we specify the joint distribution of each individual's predeviation and postdeviation data by reference to relevant groups of individuals in the study. We can then calculate the distribution of each individual's postdeviation data given his or her predeviation data and use this to multiply impute m completed datasets, fitting our substantive scientific model to each in turn and combining the results for inference using Rubin's rules (Rubin 1987; Carpenter and Kenward 2013).

A natural assumption for the de jure estimand is that, in each treatment arm, conditional distributions of later follow-up data given earlier follow-up data are the same, whether or not an individual deviates. This corresponds to Rubin's (1976) missing at random (MAR) assumption that, conditional on observed variables, missing data are equal in distribution to observed data. Under MAR, it is assumed that postdeviation individuals continued to abide by the protocol. Hence, we refer to it as "randomized-arm MAR" below. Under this assumption, the resulting estimates and inferences may also be obtained by fitting a saturated repeated-measures model with separate covariance matrices for each treatment arm (Carpenter and Kenward 2007, chap. 3).

For de facto estimands, we may wish to explore a range of assumptions, described in more detail below. For example, we may assume that postdeviation individuals behave as if they were on a reference (or control) treatment or that the responses stabilize postdeviation. In this context, a distinct advantage of multiple imputation (MI) is that it provides a convenient pathway for sensitivity analysis, because the imputation model need not be formally consistent with the analysis model. Thus, Carpenter, Roger, and Kenward (2013) extend the usual MAR-based MI approach and build on the ideas of Little and Yau (1996) to define a collection of MI methods for the inference under a range of contextually relevant de facto assumptions.

The approach falls into the pattern-mixture modeling framework (Little 1993, 1994), where different distributions are specified for fully and partially observed cases such that the overall outcome distribution is a mixture of the two. Each de facto assumption typically corresponds to a different missing not at random data mechanism (Rubin 1976), where conditional distributions of later follow-up data given earlier follow-up data differ between individuals who do and do not deviate. In this setting, some thought has to be given to the appropriate variance of the MI estimator. We have argued elsewhere (Carpenter et al. 2014) that Rubin's (1987) rules give an appropriate estimate of the variance, inflating the variance that would have been seen—had postdeviation data followed the assumption and been observed—to allow for the information lost because of the missing data.

The purpose of this article is to describe a new command, `mimix`, that can be used to implement the reference-based sensitivity analyses described by Carpenter, Roger, and Kenward (2013) for quantitative longitudinal data. This command can therefore be used to perform sensitivity analyses for longitudinal continuous clinical trials under a range of qualitative assumptions about the postdeviation behavior.

In the next section, we give more details about the methodology of Carpenter, Roger, and Kenward (2013) and present their generic algorithm for a continuous outcome. In section 3, we outline the syntax of the `mimix` command. In section 4, we demonstrate the `mimix` command by using data from a randomized double-blind controlled trial of budesonide delivered by Turbuhaler for the treatment of adult patients with chronic asthma. We discuss and conclude in section 5.

2 Methodology

In this section, we present the methodology of Carpenter, Roger, and Kenward (2013) underlying the `mimix` command.

Consider a randomized clinical trial with continuous longitudinal follow-up and two treatment arms, active and reference. Let $i = 1, \ldots, n$ index individuals, and let T_i denote the randomized treatment arm. Let $j = 0, \ldots, J$ index the J scheduled observation times, with $j = 0$ denoting the baseline; then, the outcome for each individual i at time j we denote by Y_{ij}. We assume that all individuals are observed at baseline, and following protocol deviation, data are missing. For simplicity, we also assume that there are no interim missing values, that is, no individuals with missing data at some point in the follow-up that are later observed. Define D_i as the last observation time prior to deviation for each individual; D_i therefore can take values $0, \ldots, J$. The column vector $\mathbf{Y}_{Oi} = (Y_{i0}, \ldots, Y_{iD_i})^T$ denotes an individual's observed outcomes up to D_i, and if $D_i < J$, then the column vector $\mathbf{Y}_{Mi} = (Y_{i(D_i+1)}, \ldots, Y_{iJ})^T$ denotes the missing outcomes at times $D_i + 1, \ldots, J$.

For imputation, for each deviating individual where $D_i < J$, we require the distribution of missing outcomes given their observed outcomes, treatment arm, and deviation time, denoted as

$$(\mathbf{Y}_{Mi} | \mathbf{Y}_{Oi}, D_i, T_i, \eta) \tag{1}$$

where η are the parameters of this distribution whose values we must first estimate before we can impute missing data from (1). Under MAR, (1) does not depend on D_i and is simply $(\mathbf{Y}_{Mi} | \mathbf{Y}_{Oi}, T_i, \eta)$. However, where missing data are missing not at random, this distribution will depend on D_i, and we define a form for (1) that reflects a specific assumption. Given this, MI is used for inference (Rubin 1987; Schafer 1997). That is, we create m complete datasets by drawing from the appropriate Bayesian posterior distribution of $(\eta | \mathbf{Y}_O)$, and we then draw the missing data from (1) by using the current draw of η.

To obtain η, we must choose a model for the observed data. With quantitative longitudinal response data measured at scheduled times, we assume the data can be modeled using the multivariate normal (MVN) distribution. In particular, we assume an unstructured MVN model, with a separate mean for each timepoint in each arm and a separate unstructured covariance matrix in each arm, to allow for the correlation between repeated measures.

The generic algorithm of Carpenter, Roger, and Kenward (2013) that is implemented by the `mimix` command can be summarized as follows:

1. Separately for each treatment arm, take all the observed data, assume MAR, and fit an MVN distribution with an unstructured mean (that is, a separate mean for each of the baseline and the postrandomization observation times) and a variance–covariance matrix using a Bayesian approach with an improper prior for the mean and an uninformative Jeffreys prior for the covariance matrix.

2. Draw a mean vector and covariance matrix from the posterior distribution for each treatment arm. Specifically, we use the Markov chain Monte Carlo (MCMC) method to draw from the appropriate Bayesian posterior, with a sufficient burn-in, and we update the chain sufficiently in between to ensure that subsequent draws are independent. The sampler is initiated using the expectation maximization (EM) algorithm. Refer to Carpenter and Kenward (2013) and Gilks, Richardson, and Spiegelhalter (1996) for a more in-depth discussion of MCMC methods and their applications to missing data, and refer to Schafer (1997) for a description of the applicable EM algorithm.

3. Use the draws in step 2 to form the joint distribution for each deviating individual's observed and missing outcome data as required. This can be done under a range of assumptions to explore the robustness of inference about treatment effects. The five options available in the software are described in detail in section 2.1.

4. Construct the conditional distribution of missing (postdeviation) given observed outcome data (1) for each individual who deviated, using the individual's joint distribution formed in step 3. Sample the missing postdeviation data from this conditional distribution to create a completed dataset.

5. Repeat steps 2–4 m times, resulting in m imputed datasets.

We use this algorithm to generate m imputed datasets. The substantive model of interest is then fit to each imputed dataset in turn, and the results are summarized for inference using Rubin's rules. For example, the substantive analysis model is often an analysis of covariance in which the final outcome is regressed on a randomized group and adjusted for baseline. For a single scalar parameter of interest, θ, estimates $\widehat{\theta}_m$ are obtained with standard error $\widehat{\sigma}_m$. Results across imputations can then be combined using Rubin's (1987) rules to estimate the overall treatment effect and its associated standard error under the given assumption.

Because Rubin's rules condition on the number of imputations, estimates, confidence intervals, and inferences will be sensible with two or more imputations. However, with a small number of imputations, results will be imprecise (Rubin 1987). As discussed by Carpenter and Kenward (2007), 5–10 imputations is sufficient to get a reasonably accurate answer for most applications. For more critical inferences, at least 100 imputations are recommended (Carpenter and Kenward 2013).

2.1 Constructing the joint distributions

The proposed framework revolves around the construction of appropriate joint distributions for the observed and unobserved data for deviating individuals. These joint distributions imply conditional distributions for the missing data given the observed data, which are required for imputation (1). The five options described below were proposed by Carpenter, Roger, and Kenward (2013), and they are available in the software.

Randomized-arm MAR. The joint distribution of an individual's observed and missing outcome data is MVN with a mean and covariance matrix from the individual's randomized treatment group. This option is natural for a de jure estimand.

Jump to reference (J2R). The joint distribution of an individual's observed and missing outcome data is MVN with a mean vector from the individual's randomized group up to his or her last observation time before deviating. Postdeviation, the individual's mean response profile follows that observed for a reference (typically the control) group. The covariance matrix matches that from the randomized arm for the predeviation measurements and the reference arm for the conditional components for the postdeviation given the predeviation measurements. For individuals in the reference group with missing data, this means that the joint distribution of those individuals' observed and missing outcome data is formed as MVN with a mean and covariance matrix from the individual's randomized treatment for predeviation and postdeviation measurements (as under randomized-arm MAR). This option is appropriate when the postdeviation individuals ceased their randomized treatment and started treatment similar to that available in one of the other trial arms (the reference).

Last mean carried forward. The joint distribution of an individual's observed and missing outcome data is MVN with a mean vector from the individual's randomized group up to his or her last observed time before deviating. Postdeviation, the individual's means are set equal to the value of the marginal mean for his or her randomized treatment group at the last predeviation measurement. The covariance matrix remains that from the individual's randomized treatment group. This is an appropriate option when the effect of treatment is maintained, on average, postdeviation.

Copy increments in reference (CIR). The joint distribution of an individual's observed and missing outcome data is MVN with a mean vector from the individual's

randomized group up to his or her last observation time before deviating. Postdeviation, the individual's mean increments follow those from a reference (typically the control) group. The covariance matrix is the same as for J2R. For individuals in the reference group with missing data, this means that the joint distribution of those individuals' observed and missing outcome data is formed as under randomized-arm MAR. This is an appropriate assumption when we wish to assume that, postdeviation, the disease resumes the course observed in the reference arm.

Copy reference (CR). The joint distribution of an individual's observed and missing outcome data is MVN with a mean and covariance matrix from a reference (typically the control) group, regardless of deviation time. For individuals in the reference group with missing data, this means that the joint distribution of those individuals' observed and missing outcome data is formed as under randomized-arm MAR. This is a natural option for individuals who in fact followed a different (reference) treatment from their randomized allocation.

For the J2R, CIR, and CR options, we need to specify a reference group (typically the control arm). In many settings, it is then appropriate to impute missing data for individuals in the reference group under randomized-arm MAR, and this is the default in the software.

Full technical details on the construction of the appropriate covariance structure can be found in Carpenter, Roger, and Kenward (2013) and Carpenter and Kenward (2013). There is great flexibility for contextually appropriate sensitivity analysis because different assumptions about the missing data can be made for different groups or specific individuals.

We have not yet discussed interim missing data, which is when individuals have missing data at some point in the follow-up but data are observed later. Interim missing values can also be imputed under any of the assumptions outlined above following the generic algorithm of Carpenter, Roger, and Kenward (2013). In some circumstances, the assumption made for interim missing values may be different from that specified for postdeviation data, and `mimix` allows for this. Interim missing observations may often be reasonably imputed under randomized-arm MAR.

3 The mimix command

3.1 Syntax

The `mimix` command conducts MI under the distinct treatment arm–based assumptions for missing data outlined in section 2.1. Optionally, two substantive models can also be fit to each imputed dataset and the results summarized using Rubin's (1987) rules. The two substantive model options in `mimix` are a) a linear regression of the final timepoint on treatment and baseline or b) a saturated repeated-measures model (that is, including treatment crossed with visit and baseline crossed with visit) with separate covariance matrices for each treatment arm. Other substantive models can be fit to the imputed data in the usual way by using `mi estimate`.

The syntax of the `mimix` command is the following:

`mimix` *depvar treatvar*`, id(`*varname*`) time(`*varname*`) [clear`
`saving(`*filename*`[ , replace ]) covariates(`*varlist*`) interim(`*string*`)`
`iref(`*string*`) {method(`*string*`)|methodvar(`*varname*`)} mixed`
`{refgroup(`*string*`)|refgroupvar(`*varname*`)} regress burnbetween(`#`)`
`burnin(`#`) m(`#`) seed(`#`) ]`

Data are required in long format with one record per individual per timepoint, where *depvar* is the numeric outcome variable with missing data in the existing dataset and *treatvar* identifies the treatment group variable in the existing dataset and may be either a numeric or string variable.

`id(`*varname*`)` specifies the variable identifying individuals in the existing dataset. `id()` is required and may be either a numeric or a string variable.

`time(`*varname*`)` specifies the variable identifying units of time in the original dataset. `time()` is required and must be a numeric variable.

`clear` specifies that the original data in memory be cleared and replaced by the imputed dataset. The imputed dataset must be saved manually if required. One of `clear` or `saving()` is required.

`saving(`*filename*`[ , replace ])` saves the imputed datasets. A new filename is required unless `replace` is also specified. `replace` allows the *filename* to be overwritten with new data. One of `clear` or `saving()` is required.

`covariates(`*varlist*`)` specifies any additional baseline covariates to be included in the MI model and analysis if either the `regress` or the `mixed` option is specified. Any specified covariates must be fully observed numerical variables. Dummy variables must be generated for any factor covariates.

`interim(`*string*`)` specifies an alternative imputation method for all interim missing values (where the individual has data observed later). *string* may be `mar`, `j2r`, `lmcf`, `cir`, or `cr` (not case sensitive). See section 3.3 for further details on specifying the imputation method.

`iref(`*string*`)` specifies the level of *treatvar* chosen for the reference for all interim missing values (where the individual has data observed later). `iref()` is required when using the `j2r`, `cir`, or `cr` imputation method. See section 3.3 for further details on specifying the imputation method.

`method(`*string*`)` defines the imputation method for all individuals. *string* may be `mar`, `j2r`, `lmcf`, `cir`, or `cr` (not case sensitive). `method()` and `methodvar()` are mutually exclusive; specifying both will return an error message. See section 3.3 for further details on specifying the imputation method.

`methodvar(`*varname*`)` specifies the variable in the original dataset that contains the individual-specific imputation method(s). This option should be used if different imputation methods are required for different individuals. `methodvar()` must be a string variable containing one of `mar`, `j2r`, `lmcf`, `cir`, or `cr` (not case sensitive) for each individual. `methodvar()` and `method()` are mutually exclusive; specifying both will return an error message. See section 3.3 for further details on specifying the imputation method.

`mixed` uses `mi estimate` with Stata's default options to fit a saturated repeated-measures model using restricted maximum likelihood—with a separate mean for each treatment and time, full covariate–time interactions for any included `covariates()`, and a separate unstructured covariance matrix for each arm—to each of the imputed datasets. `mixed` combines results using Rubin's (1987) rules for inference. This option may add substantially to the postimputation computation time if a large number of imputations have been specified.

`refgroup(`*string*`)` specifies the level of *treatvar* chosen for the reference for all individuals. This option is required when using the `j2r`, `cir`, or `cr` imputation method. `refgroup()` and `refgroupvar()` are mutually exclusive; specifying both will return an error message. See section 3.3 for further details on specifying the imputation method.

`refgroupvar(`*varname*`)` specifies the variable in the original dataset that identifies the level of *treatvar* chosen for the reference for each individual. This option is required when using the `j2r`, `cir`, or `cr` imputation method. `refgroupvar()` and `refgroup()` are mutually exclusive; specifying both will return an error message. See section 3.3 for further details on specifying the imputation method.

`regress` uses `mi estimate` with Stata's default options to fit a linear regression of *depvar* at the final timepoint on *treatvar*, and any included `covariates()`, to each of the imputed datasets. It combines results using Rubin's (1987) rules for inference.

`burnbetween(`#`)` specifies the number of iterations between pulls for the posterior in the MCMC. The default is `burnbetween(100)`.

`burnin(`#`)` specifies the number of iterations in the MCMC burn-in. The default is `burnin(100)`.

`m(`#`)` specifies the number of imputations required. The default is `m(5)`.

`seed(`#`)` specifies the seed for the random-number generator. The default is `seed(0)`, meaning that no seed is specified by the user and so the current value of Stata's random-number seed will be used; this will result in different sets of imputations for multiple program runs. To reproduce a set of imputations, the same random-number seed should be used with the original data sorted in exactly the same order.

3.2 Implementation details

Required data format

Data are required in long format with one record per individual per timepoint. If data are in wide format, consult [D] **reshape** to convert data into long format.

Baseline covariates

Any additional included baseline covariates are required to be complete. Individuals with missing covariate information will be highlighted for the user by `mimix` and will be discarded in the imputation process and any requested analysis.

Potential error with sparse data

Stata's `mi impute mvn` command (which uses the MCMC method initialized by the EM algorithm to impute missing values) is used to complete steps 1 and 2 of the general procedure, as detailed in section 2. If the response variable of interest is measured at an occasion with only a few complete cases, `mi impute mvn` may terminate with an error message if there is not enough information in the observed data to reliably estimate aspects of the covariance structure in the required MVN model. If this is the case, we advise the user to explore an alternative viable MVN model for the data by using the `mi impute mvn` command. The response at the occasion with few observed outcomes may need to be excluded from the analysis and `mimix` rerun.

Data output

The imputed datasets are produced in long format, with one record per individual per timepoint per imputation, and are `mi set` in `flong` style, ready to analyze using `mi estimate`. The imputed datasets are output in memory if `clear` is specified, and they are saved in *filename*`.dta` if `saving()` is specified.

Analysis options

If the `regress` or `mixed` analysis option is specified, Stata's `mi estimate` command is used with default options to fit the specified analysis model to each imputed dataset and to combine results using Rubin's (1987) rules (see [MI] **mi estimate**). The usual output will be displayed in the Results window. If alternative `mi estimate` options or other substantive models are required following the completion of `mimix`, then `mi estimate` can be used in the usual way for further analysis.

Use of data preserve

Because of extensive manipulation of the data, `mimix` uses the `preserve` and `restore` commands. While `mimix` can be successfully run on data that are already preserved, we recommend that users cancel any previous data preserve by using `restore, not` to ensure the `clear` and `saving()` options of `mimix` work as intended.

3.3 Specifying the imputation method

The `mimix` command must contain either the `method()` option or the `methodvar()` option. `method()` indicates which imputation method should be employed for all individuals, while `methodvar()` indicates which imputation method should be employed for each individual. `method()` and `methodvar()` are mutually exclusive options; specifying both will return an error message.

If the `method()` option is used to request the same imputation method for all individuals, then values specified in `method()` must be one of those presented in table 1 (not case sensitive). If the `methodvar()` option is used to request different imputation methods for different individuals, then a new variable that contains individual-specific imputation methods must be generated and specified in `methodvar()`. The variable that holds the individual imputation methods must only contain values presented in table 1 (not case sensitive), and the method specification cannot vary within an individual over time.

If the `j2r`, `cir`, or `cr` imputation method is used, then either the `refgroup()` option must also be used to specify the reference level of the *treatvar* for all individuals or the `refgroupvar()` option must also be used to indicate the reference level of the *treatvar* for each individual. Together, these variables allow for the required assumptions outlined in section 2.1. If one of the imputation methods that includes a reference group is specified for all individuals (or for specific individuals via `methodvar()`), then missing data for individuals in that reference group (with the reference-imputation specification) are imputed under randomized-arm MAR.

The `interim()` option specifies the imputation method for all interim missing values. If this option is not used, any interim missing values will be imputed following the method specified by the `methodvar()` or `method()` option, in the same way as missing postdeviation data.

Table 1. Specifying the imputation method

Method name	Name to specify in `method()` or `methodvar()`
Randomized-arm MAR	`mar`
Jump to reference	`j2r`
Last mean carried forward	`lmcf`
Copy increments in reference	`cir` or `ciir`
Copy reference	`cr`

3.4 Stored results

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

Scalars

`r(N)`	total sample size
`r(Nmiss)`	total number of individuals with incomplete data
`r(Ncomp)`	total number of individuals with complete data
`r(M)`	number of imputations
`r(burnin)`	number of MCMC burn-in iterations
`r(bbetween)`	number of MCMC burn-between iterations

Macros

`r(depvar)`	name of dependent variable
`r(treatvar)`	name of treatment group variable
`r(covariates)`	names of covariates
`r(method)`	imputation method (with `method()` only)
`r(methodvar)`	imputation method variable (with `methodvar()` only)
`r(rgroup)`	name of reference group (with `refgroup()` only)
`r(rgroupvar)`	name of reference group variable (with `refgroupvar()` only)
`r(rseed)`	random-number seed

Matrices

`r(Ntreat)`	sample size in each treatment group
`r(Ntreat_mis)`	number of individuals with incomplete data in each treatment group
`r(Ntreat_comp)`	number of individuals with complete data in each treatment group
`r(Ntreat_pat)`	number of unique missing-value patterns in each treatment group
`r(niter_em)`	number of iterations EM takes to converge in each treatment group
`r(lpobs_em)`	observed log posterior in EM in each treatment group
`r(conv_em)`	convergence flag for EM in each treatment group

If the `regress` or `mixed` analysis option is used, then `mi estimate` is called within the program run and the associated `mi estimate` results will also be stored in `e()` (see [MI] **mi estimate**). If both `regress` and `mixed` are specified, then only the `mi estimate` results of `mixed` will be stored in `e()`.

4 Example

Here we demonstrate the `mimix` command with data from a randomized double-blind clinical trial of budesonide delivered by Turbuhaler for the treatment of adult patients with chronic asthma (Busse et al. 1998). A total of 473 individuals were randomized to a daily dose of either 200, 400, 800, or 1,600μg of budesonide or a placebo. The primary outcome—measured at weeks 0 (baseline), 2, 4, 8, and 12—was forced expiratory volume in one second (FEV_1), recorded in liters (L); however, several individuals deviated and did not complete the full 12-week follow-up.

In this article, we focus our attention on only the placebo and the lowest dose active arm (200μg budesonide) for sensitivity analysis. The observed mean profiles by treatment arm and the various missing-data patterns are shown in figure 1. Only 38 of the 92 individuals in the placebo arm (41%) and 72 of the 91 individuals in the active arm (79%) remained in the trial at 12 weeks; 3 individuals (2 placebo and 1 active) had interim missing data.

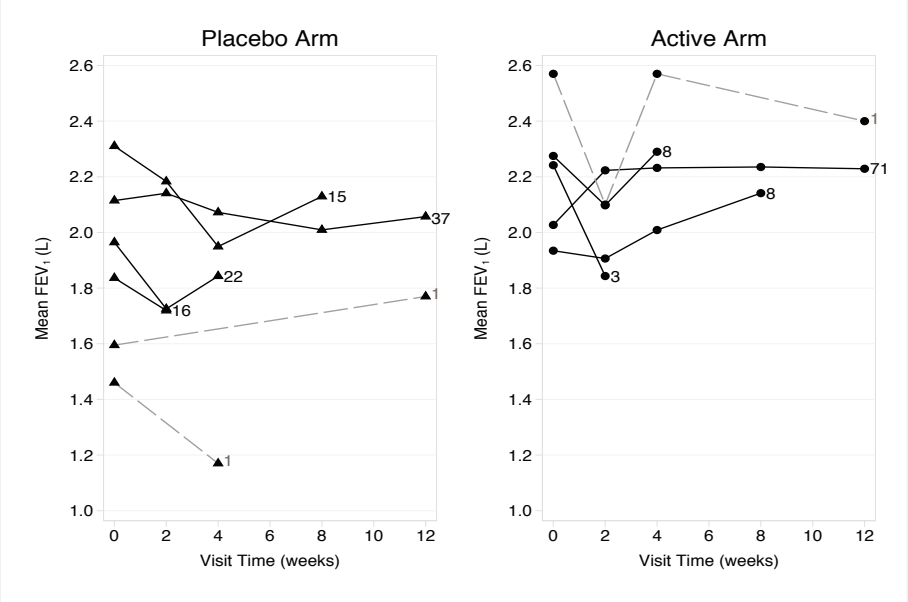


Figure 1. Observed mean FEV_1 by treatment arm and deviation profile against time. Solid lines join observed means at each timepoint for the various deviation (withdrawal) patterns; dashed lines join observed means of the three individuals with interim missing data. Numbers indicate the counts of individuals with the associated profile.

The primary analysis of the original trial consisted of a linear regression of the 12-week FEV_1 outcome on the treatment group, adjusted for baseline FEV_1, using data from the 110 individuals measured at week 12. This gives a treatment effect of 0.239 L, $p = 0.017$. We will use `mimix` to assess the robustness of the results to various postdeviation

assumptions outlined in section 2.1. The interim missing outcomes will be imputed under MAR.

In the following output, we describe the variables in the asthma trial dataset and list their contents for one arbitrarily selected deviating individual.

```
. use asthma
. describe
Contains data from asthma.dta
  obs:           732
 vars:             5                          12 Feb 2015 10:18
 size:        11,712
```

variable name	storage type	display format	value label	variable label
id	int	%8.0g		Patient ID
time	byte	%9.0g		Measurement time (weeks)
treat	byte	%8.0g	treat1	Randomised treatment assignment
base	double	%12.0g		Baseline FEV1 (L)
fev	float	%9.0g		FEV1 (L)

```
Sorted by: id
. list in 37/40, noobs sepby(id)
```

id	time	treat	base	fev
5030	2	Placebo	1.14	.85
5030	4	Placebo	1.14	1.51
5030	8	Placebo	1.14	.
5030	12	Placebo	1.14	.

`id` is the unique individual identifier, and `treat` is the randomized treatment assignment to placebo (`treat` = 2) or active (`treat` = 3). `fev` is the postbaseline FEV_1 measurement (L), and `time` is the time of the FEV_1 measurement in weeks. `base` is the baseline FEV_1 measurement. The dataset is already in long format with one observation per individual per timepoint, as required for `mimix`. We can see that the selected individual deviated sometime between week 4 and week 8; consequently, the individual has missing outcomes for weeks 8 and 12.

4.1 Sensitivity analysis using the mimix command

In this section, we perform a sensitivity analysis using each of the five options listed in section 2.1 for constructing joint distributions. Results of these analyses are summarized in table 2.

We first analyze the data under the randomized-arm MAR assumption for all individuals, in other words, the de jure assumption that—postdeviation—individuals continued on their randomized treatment as specified in the protocol. We create 50 imputations and take the default MCMC burn-in of 100 iterations and burn-between of 100 iterations. We include the baseline FEV_1 measure in the imputation model as a covariate, but if

this fully observed variable were used as an outcome, the results would be stochastically identical. We use the `regress` option to specify that the substantive analysis is a linear regression of 12-week FEV_1 on randomized treatment and baseline FEV_1. Imputation with the (randomized-arm) `mar` option automatically means the interim missing values will be imputed under MAR in each treatment group.

```
. mimix fev treat, id(id) time(time) method(mar) covariates(base) regress m(50)
> clear seed(101)
Performing imputation procedure for group 1 of 2...
Performing imputation procedure for group 2 of 2...

Performing regress procedure ...

i.treat           _Itreat_2-3         (naturally coded; _Itreat_2 omitted)

Multiple-imputation estimates                     Imputations       =         50
Linear regression                                 Number of obs     =        183
                                                  Average RVI       =     0.4106
                                                  Largest FMI       =     0.3495
                                                  Complete DF       =        180
DF adjustment:   Small sample                     DF:     min       =      91.39
                                                          avg       =      99.15
                                                          max       =     105.79
Model F test:       Equal FMI                     F(   2,  149.8)   =      40.69
Within VCE type:          OLS                     Prob > F          =     0.0000

------------------------------------------------------------------------------
         fev |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
   _Itreat_3 |   .3230728   .1042794     3.10   0.002     .1163241    .5298215
        base |   .7240691   .0861441     8.41   0.000     .5531672    .8949709
       _cons |   .3959986   .1971734     2.01   0.048     .0043602     .787637
------------------------------------------------------------------------------

Imputed dataset now loaded in memory
Imputed data created in variable fev using mar
```

The output displays the results from the requested analysis, along with a description of the variable that now contains imputed data. Under randomized-arm MAR, the treatment estimate is increased from the complete records regression reported above, to 0.323 L with a p-value of 0.002. The results of this analysis are shown in the top panel of figure 2.

Because we used the `clear` option, the imputed dataset is stored in memory. The imputed data are output using `mi set flong`. Note that the imputed dataset has not yet been saved. If the `saving()` option is specified, then the imputed data will be saved when the command is executed.

We now reimpute the asthma trial under the J2R assumption for all individuals, with the placebo arm (`treat` = 2) first set as the reference. The `interim()` option is included to impute the interim missing values under randomized-arm MAR. Including the `interim()` option here does not actually affect the results because our substantive model of interest considers the treatment effect at the final timepoint. Imputation of interim values under MAR will have an impact when the `mixed` option is specified to fit a saturated repeated-measures model, using all follow-up outcomes, to estimate a separate baseline-adjusted treatment effect at each follow-up time.

```
. mimix fev treat, id(id) time(time) method(j2r) refgroup(2) covariates(base)
> interim(mar) regress m(50) clear seed(101)
Performing imputation procedure for group 1 of 2...
Performing imputation procedure for group 2 of 2...
Performing regress procedure ...
i.treat           _Itreat_2-3         (naturally coded; _Itreat_2 omitted)
Multiple-imputation estimates                     Imputations       =         50
Linear regression                                 Number of obs     =        183
                                                  Average RVI       =     0.4483
                                                  Largest FMI       =     0.3510
                                                  Complete DF       =        180
DF adjustment:   Small sample                     DF:     min       =      91.07
                                                          avg       =     109.09
                                                          max       =     140.18
Model F test:       Equal FMI                     F(   2,  156.9)   =      32.45
Within VCE type:          OLS                     Prob > F          =     0.0000
------------------------------------------------------------------------------
         fev |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
   _Itreat_3 |   .2261827   .1028346     2.20   0.029     .0228754      .42949
        base |   .6894261   .0933944     7.38   0.000     .5040403    .8748119
       _cons |   .4669997   .2112431     2.21   0.030     .0473954    .8866041
------------------------------------------------------------------------------
Imputed dataset now loaded in memory
Imputed data created in variable fev using j2r
Interim missing data imputed using mar
```

The results of the J2R analysis with placebo as the reference are summarized in table 2 along with the results of a J2R analysis with active as the reference. These address the de facto assumption, when postdeviation individuals not randomized to the reference treatment change to the reference treatment. Both of these analyses result in a reduced treatment estimate relative to the de jure randomized-arm MAR assumption. However, while J2R with placebo as the reference still gives a treatment effect that is statistically significant at the 5% level, J2R with active as the reference does not. This is because more individuals deviate in the placebo arm than in the active arm (figure 1), and they tend to be individuals whose lung function is lower. The effect of this versus analysis under randomized-arm MAR is shown in figure 2. The change in placebo individuals under J2R-active reduces the treatment estimate by the greatest amount.

Our next analysis is last mean carried forward. Figure 1 shows that the arm-specific means begin to stabilize quite early in the follow-up. It is therefore to be expected that last mean carried forward gives a slightly reduced estimate relative to randomized-arm MAR, with a slightly higher p-value (see table 2). If we wish to assume that individuals' lung function at deviation is broadly maintained postdeviation, then last mean carried forward would be appropriate.

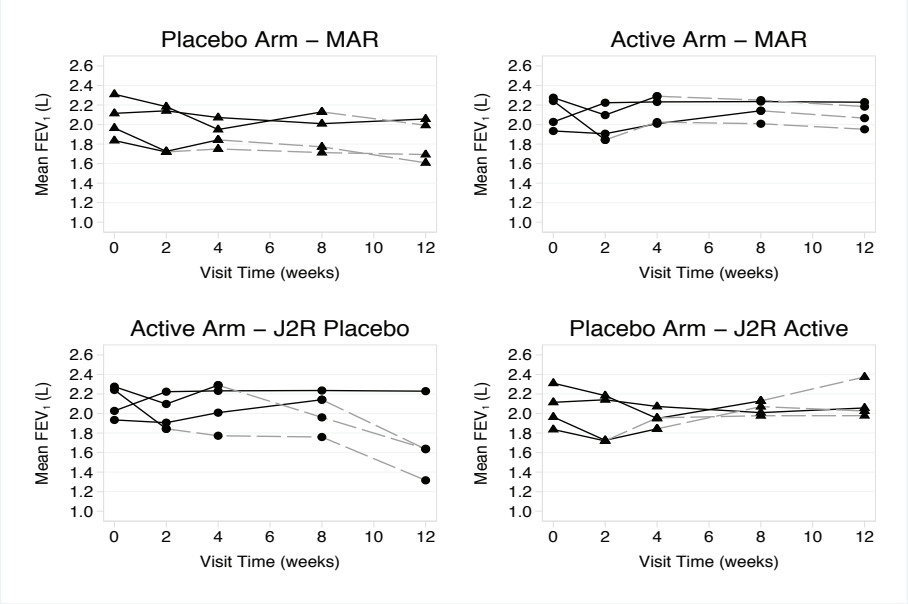


Figure 2. Mean FEV_1 against time, by treatment arm, for the four different deviation (withdrawal) patterns under randomized-arm MAR (top panel) and J2R (bottom panel). Solid lines join observed means before deviation, and dashed lines join the means of the imputed data for that pattern.

The next two analyses are both CIR. In the first, the reference is the placebo, and in the second, the reference is the active arm. Because more individuals deviate in the placebo arm than in the active arm and because the placebo arm profiles tend to decrease while those of the active arm increase, we again see a slightly larger treatment estimate when the reference arm is placebo (see table 2). CIR with placebo reference is appropriate if, postdeviation, we wish to assume that active individuals' lung function starts to decline from its current value at the same rate as seen in the placebo arm. CIR with active reference is appropriate if, postdeviation, we wish to assume that postdeviation placebo individuals access an active treatment and their lung function increases from its current value at the rate seen in the active arm.

Finally, we consider CR with placebo reference and with active reference. Under this assumption, an individual's postdeviation data are imputed as if they had always belonged to the reference arm. CR with placebo reference may be an appropriate de facto assumption for individuals who could not tolerate the active treatment. Under CR, predeviation individual-specific residuals about the mean are typically greater than under J2R. This means that postdeviation profiles typically change less abruptly than with J2R, which is what we observe here (see table 2). For CR with active reference, the treatment estimate is greater than both treatment estimates under J2R but less than the treatment estimates for all the other de facto assumptions.

Table 2. Sensitivity analysis results

Analysis	Treatment estimate (L)	Standard error	p-value
De jure			
Primary analysis (analysis of covariance)	0.239	0.099	0.017
Randomized-arm MAR	0.323	0.104	0.002
De facto			
Jump to placebo	0.226	0.103	0.029
Jump to active	0.128	0.095	0.181
Last mean carried forward	0.296	0.096	0.003
Copy increments in placebo	0.281	0.103	0.007
Copy increments in active	0.277	0.082	0.001
Copy placebo	0.289	0.101	0.005
Copy active	0.251	0.082	0.003

We therefore conclude that if, postdeviation, medication has a comparable effect with the lowest active dose, then individuals will have comparable lung function at the end of the study. Otherwise, the sensitivity analysis is consistent with the primary analysis of the trial in identifying a significant beneficial effect of treatment relative to placebo.

5 Discussion

In this article, we introduced the `mimix` command to implement the reference-based sensitivity analysis approach described by Carpenter, Roger, and Kenward (2013). This approach sets out to provide contextually relevant sensitivity analysis of a longitudinal clinical trial with continuous outcome data subject to individual deviation. As we described, the approach constructs each individual's joint predeviation and postdeviation data distribution by reference to treatment groups and then imputes the individual's missing postdeviation data accordingly. The `mimix` program automates the steps of constructing the required joint distributions and the corresponding imputation distributions under a range of assumptions. Further, if desired, the program will automatically fit one of two substantive models to the resulting imputed data and combine the results by using Rubin's (1987) rules. The available substantive models are either a linear regression of the final timepoint on baseline or treatment or a saturated repeated-measures model, as detailed above.

This method is appealing for sensitivity analysis because it does not require the formal specification of any sensitivity parameters, which is notoriously difficult (White et al. 2007). Rather than requiring quantitative assumptions, it asks for qualitative

assumptions in respect to certain study arms. The associated quantitative assumptions are then estimated from the data and used to produce imputations. Different qualitative assumptions can be made for different individuals (or similar groups of individuals), and the `mimix` command allows this flexibility through the `methodvar()` option, providing contextually plausible sensitivity analyses.

Interim missing data, which in practice are likely to be inevitable to some extent, are also accommodated by `mimix`. These may be imputed under randomized-arm MAR (often the most appropriate assumption) or one of the alternative reference-based assumptions.

The approach can be used for individuals who deviate immediately, that is, for those who have no outcome data, as long as these individuals are included in the original dataset. The relevant postdeviation distribution is constructed as outlined in section 2 for such individuals, and all outcome data are imputed from it.

Recall that we are modeling data from a clinical trial where patient outcome data are collected according to a prespecified common schedule. The imputation model is MVN, with a separate unstructured covariance matrix for each trial arm and a separate mean for each timepoint. This is the most general, and by far the most appropriate, model for such data (Molenberghs and Kenward 2007, chap. 5.6). If individual patients' data are collected irregularly, the unstructured covariance matrix is no longer as natural of an option, and other options may be considered. While it is possible that this may encounter convergence difficulties with a very large number of timepoints and limited number of patients, in our experience this is not common. In such situations, one may need to consider alternative, more structured, forms of covariance matrix, but this is beyond our current scope. If the data are skewed, one can consider transformation to approximate normality, and then impute and transform back. Schafer (1997, chap. 6.4), however, reports simulations showing that imputation drawn under the MVN model are robust to moderate skewness.

If we have several baseline covariates on which we wish to condition the imputations, then a current restriction is that these must be fully observed. Moreover, at the imputation step they are formally treated as continuous in the MVN imputation. Fully binary variables can simply be included as they are (however they are coded). However, fully observed c-level categorical variables must be included as $(c-1)$ dummy indicator variables.

Following the general algorithm of Carpenter, Roger, and Kenward (2013), separate models for the predeviation data are required in each treatment arm. Any covariates potentially including the baseline response are consequently fit separately in each arm prior to the construction stage, where the treatment arm parameters may be mixed for subsequent imputation. If the covariates are markedly imbalanced across treatment arms, this may result in inappropriate data distributions. However, in the randomized controlled trial setting, the expected distribution of the covariates in the two arms will be the same. Randomization should therefore ensure any covariates are well matched and clinically similar in the two arms.

Throughout this article, we focused on the two-arm randomized clinical trial setting; however, this is not a constraint. `mimix` can be used to conduct reference arm–based imputation for trials with more than two arms.

To summarize, the `mimix` command provides a computationally accessible tool for reference-based sensitivity analysis. The assumptions available in the program correspond to both de jure and de facto estimands, allowing sensitivity analysis that explores the effect of contrasting assumptions concerning the individual's postdeviation outcomes. We hope that this implementation will remove a barrier to trialists performing sensitivity analysis in practice.

6 Acknowledgments

The `mimix` command is a development of an SAS macro written by James Roger, who we would like to thank. This work was funded by the MRC London Hub for Trials Methodology Research (grant MC_EX_G0800814).

7 References

Busse, W. W., P. Chervinsky, J. Condemi, W. R. Lumry, T. L. Petty, S. Rennard, and R. G. Townley. 1998. Budesonide delivered by Turbuhaler is effective in a dose-dependent fashion when used in the treatment of adult patients with chronic asthma. *Journal of Allergy and Clinical Immunology* 101: 457–463.

Carpenter, J. R., and M. G. Kenward. 2007. *Missing Data in Randomised Controlled Trials—A Practical Guide*. Birmingham: National Health Service Co-ordinating Centre for Research Methodology.

———. 2013. *Multiple Imputation and Its Application*. Chichester, UK: Wiley.

Carpenter, J. R., J. H. Roger, S. Cro, and M. G. Kenward. 2014. Response to comments by Seaman et al. on "Analysis of Longitudinal Trials with Protocol Deviation: A Framework for Relevant, Accessible Assumptions, and Inference via Multiple Imputation", Journal of Biopharmaceutical Statistics 23: 1352–1371. *Journal of Biopharmaceutical Statistics* 24: 1363–1369.

Carpenter, J. R., J. H. Roger, and M. G. Kenward. 2013. Analysis of longitudinal trials with protocol deviation: A framework for relevant, accessible assumptions, and inference via multiple imputation. *Journal of Biopharmaceutical Statistics* 23: 1352–1371.

Committee for Medicinal Products for Human Use. 2010. *Guideline on Missing Data in Confirmatory Clinical Trials*. London, UK: European Medicines Agency.

Gilks, W. R., S. Richardson, and D. J. Spiegelhalter, eds. 1996. *Markov Chain Monte Carlo in Practice*. London: Chapman & Hall/CRC.

Little, R. J. A. 1993. Pattern-mixture models for multivariate incomplete data. *Journal of the American Statistical Association* 88: 125–134.

———. 1994. A class of pattern-mixture models for normal incomplete data. *Biometrika* 81: 471–483.

Little, R. J. A., and L. Yau. 1996. Intent-to-treat analysis for longitudinal studies with drop-outs. *Biometrics* 52: 1324–1333.

Molenberghs, G., and M. G. Kenward. 2007. *Missing Data in Clinical Studies*. Chichester, UK: Wiley.

National Research Council. 2010. *The Prevention and Treatment of Missing Data in Clinical Trials*. Panel on Handling Missing Data in Clinical Trials. Committee on National Statistics, Division of Behavioural and Social Sciences Education, Washington, DC: National Academies Press.

Rubin, D. B. 1976. Inference and missing data. *Biometrika* 63: 581–592.

———. 1987. *Multiple Imputation for Nonresponse in Surveys*. New York: Wiley.

Schafer, J. L. 1997. *Analysis of Incomplete Multivariate Data*. Boca Raton, FL: Chapman & Hall/CRC.

White, I. R., J. Carpenter, S. Evans, and S. Schroter. 2007. Eliciting and using expert opinions about dropout bias in randomized controlled trials. *Clinical Trials* 4: 125–139.

White, I. R., N. J. Horton, J. Carpenter, and S. J. Pocock. 2011. Strategy for intention to treat analysis in randomised trials with missing outcome data. *British Medical Journal* 342: d40.

About the authors

Suzie Cro is a PhD student at the London School of Hygiene and Tropical Medicine. Her thesis topic is relevant accessible sensitivity analysis for clinical trials with missing data. She is funded by the MRC Clinical Trials Unit at University College London, where she also works part-time as a medical statistician.

Tim P. Morris is a medical statistician with eight years of experience. He is interested in statistical methods for improving the design and analysis of randomized trials and meta-analysis. His PhD was on practical methods for MI. He is a Stata enthusiast.

Michael G. Kenward has been GlaxoSmithKline professor of biostatistics at the London School of Hygiene and Tropical Medicine since 1999 and has been a consultant in biostatistics for over 25 years. His research interests include longitudinal data analysis and the problem of missing data.

James R. Carpenter holds a joint appointment at the London School of Hygiene and Tropical Medicine and the MRC Clinical Trials Unit at University College London. He has a longstanding interest in longitudinal data, MI, and trials, and he is comaintainer of http://www.missingdata.org.uk.

The Stata Journal (2016)
16, Number 2, pp. 464–481

Partial credit model: Estimations and tests of fit with pcmodel

Jean-François Hamel
University Hospital of Angers
Methodology and Biostatistics Unit
Angers, France
jeanfrancois.hamel@chu-angers.fr

Véronique Sébille
University Hospital of Nantes
Biometric Platform
Nantes, France

Gaëlle Challet-Bouju
University Hospital of Nantes
Addictology Department
Nantes, France

Jean-Benoit Hardouin
University Hospital of Nantes
Biometric Platform
Nantes, France

Abstract. The partial credit and rating scale models are classical models from item response theory; they belong to the generalized linear latent and mixed model family and allow one to analyze questionnaires such as patient-reported outcomes. Few goodness-of-fit testing procedures have been proposed for such models, and few computer programs implement such tests. Here we describe two tests: the R1m test (which tests the overall adequacy of the model to the data) and the Si test (which evaluates the contribution of each item to a possible lack of fit). We also propose two commands: `pcmodel`, which implements partial credit or rating scale models, and `pcmtest`, which tests the adequacy of such models to the data.

Keywords: st0441, pcmodel, pcmtest, partial credit model, rating scale model, item response theory, fit tests

1 Introduction

Several scientific studies investigate phenomena such as intelligence, anxiety, quality of life, or welfare. Such phenomena, not directly observed or measured, are called latent variables. Usually, latent variables are indirectly investigated using questionnaires including different items called patient-reported outcomes.

Item response theory (IRT) provides a conceptual framework suitable for modeling such data (Lord and Novick 1968). With IRT, the observed item responses are modeled based on the unobservable respondent characteristics (that is, the latent variables of interest) and the item characteristics.

One of the most famous IRT models is the Rasch model (Rasch 1960). With this model, which is suitable for only dichotomous items, items are characterized only by their difficulty, defined as the latent trait of an individual having exactly the same probability of responding to each of the two proposed answers to the item. Extensions have been proposed for dealing with polytomous items with ordered response categories (0, 1, ..., m_j for each item j), such as the rating scale model (RSM: Andrich [1978]) and

the partial credit model (PCM: Masters [1982]). A PCM can be used even if the number of response categories differs depending on the items. In contrast, an RSM can be used only if the number of response categories are equal for all the items and if one may assume that the differences in the step difficulties for the different response categories are the same for all the items. With these two Rasch-family models, respondents are characterized by their latent trait (that is, the individual value of the latent variable of interest); the items are characterized by the difficulties associated with each of their response categories.

Such IRT models are particularly suitable for performing population-based measurements of latent traits and for studying the effect of associated covariates on these latent variables. These models consist in mixed models: the latent trait is considered a random variable, and the item difficulties (and the potentially included covariates) are considered fixed effects. Parameters are then classically estimated using marginal maximum likelihood (MML) (Thissen 1982; Hamel et al. 2012).

There are several limitations of such models. The magnitude order of latent variables (or covariates associated with such latent variables) remains unknown. Thus interpreting numerical estimations for such variables remains a challenge. Moreover (and as is the case for any statistical model), using such models requires the ability to test their fit to the analyzed data. Few goodness-of-fit testing procedures have been proposed in the literature for fitting the PCM or RSM with the MML procedure. Glas and Verhelst (1995) proposed three tests, all asymptotically distributed as chi-squares: the R1m test (which tests the assumption of monotone increasing and parallel item response functions); the R2 test (which tests the unidimensionality of the latent trait); and the Si test (which evaluates the contribution of each item to a possible lack of fit in case of poor model fit).

Few computer programs allow for both estimating the parameters of IRT models using MML and performing tests of fit. The SAS macroprogram `%AnaQol` (Hardouin and Mesbah 2007) and the R library `ltm` (Rizopoulos 2006) estimate the parameters of a PCM using MML but do not test the fit. Hardouin (2007) proposed the `raschtest` command for estimating parameters of a Rasch model and for testing the fit to the observed data. However, this command is suitable for only dichotomous items and does not estimate the parameters of a PCM or an RSM. With the `gllamm` command (Rabe-Hesketh, Pickles, and Taylor 2000; Zheng and Rabe-Hesketh 2007), one can estimate the parameters of a PCM or an RSM using MML, but it is not adapted for testing the fit.

In this article, we present a command, `pcmodel`, for modeling a latent process using PCM or RSM and estimating its parameters using MML. This command allows for introducing covariates possibly affecting the individual's latent trait (for example, group covariates) and assists in the interpretation of their effect (by estimating pseudo-type-III sum of squares for these covariates and the proportion of the latent-trait variance they can explain). An additional command, `pcmtest`, tests the fit of a PCM or an RSM to observed data using both R1 and Si tests. `pcmtest` can be used after the `pcmodel` command but also after the `irt pcm` and `irt rsm` commands.

2 PCM

Let's consider a questionnaire consisting of k polytomous items. Each item j ($j = 1, \ldots, k$) comprises m_j ordered response categories. The PCM defines the probability of observing the lth ($l = 0, 1, \ldots, m_j$) response category to the jth item as a function of the latent trait (considered as a random-effects covariate assumed to follow a normal distribution with mean μ—usually constrained to 0—and variance σ^2) and of the difficulties associated with each of the item response categories δ_{jl} ($l > 0$) (considered as fixed-effects covariates). δ_{jl} can be interpreted as the value of the latent trait of an individual with equal probability of choosing the $(l-1)$th or the lth response for the jth item.

$$P\left(X_{ij} = l|\theta, \boldsymbol{\delta}_j\right) = \frac{\exp\left(l\theta - \sum_{a=1}^{l} \delta_{ja}\right)}{\sum_{b=0}^{m_j} \exp\left(b\theta - \sum_{a=1}^{b} \delta_{ja}\right)} \qquad \theta \sim \mathcal{N}(\mu, \sigma^2)$$

Group covariates can be included in such a model for explaining variations of the latent trait. When one introduces covariates, the latent trait is split into two parts, the first one corresponding to the component explained by the observed covariate values and the second one to the residual component explained by individual variation (Christensen 2007).

Consider a set of n covariates possibly associated with the latent trait. $\mathbf{c}_i$ is the indicator vector of dimension n specifying the observed values of the k covariates for the ith individual, and $\boldsymbol{\beta}$ is the vector of dimension n of the regression parameters. The latent trait is equal to $\theta = \boldsymbol{\beta}'\mathbf{c}_i + \theta_{\text{Res}}$ with $\theta_{\text{Res}} \sim \mathcal{N}(0, \sigma^2_{\text{Res}})$. A PCM including group covariates can then be written as follows:

$$P\left(X_{ij} = l|\boldsymbol{\beta}, \mathbf{c}_i, \theta_{\text{Res}}, \boldsymbol{\delta}_j\right) = \frac{\exp\left\{l(\boldsymbol{\beta}'\mathbf{c}_i + \theta_{\text{Res}}) - \sum_{a=1}^{l} \delta_{ja}\right\}}{\sum_{b=0}^{m_j} \exp\left\{b(\boldsymbol{\beta}'\mathbf{c}_i + \theta_{\text{Res}}) - \sum_{a=1}^{b} \delta_{ja}\right\}}$$
$$\theta_{\text{Res}} \sim \mathcal{N}(0, \sigma^2_{\text{Res}})$$

The effects of group covariates—considered as fixed effects—can then be classically tested using Wald tests.

3 RSM

The RSM is a special case of the PCM. With this model, for example suitable for items with the same response categories, the item difficulties are split into two parts: one based on the item, δ_j, and the other based on the response category, τ_l. An identifiability constraint for τ_l can be $\tau_1 = 0$. Then, δ_j represents the first-step difficulty for item j,

and τ_l $(l = 1, 2, \ldots, m)$ represents the extrastep difficulty of subsequent steps compared with the first step. An RSM including group covariates can then be written as follows:

$$P\left(X_{ij} = l|\boldsymbol{\beta}, \mathbf{c}_i, \theta_{\text{Res}}, \delta_j, \boldsymbol{\tau}\right) = \frac{\exp\left\{l(\boldsymbol{\beta}'\mathbf{c}_i + \theta_{\text{Res}}) - \sum_{a=1}^{l}(\delta_j + \tau_a)\right\}}{\sum_{b=0}^{m_j}\exp\left\{b(\boldsymbol{\beta}'\mathbf{c}_i + \theta_{\text{Res}}) - \sum_{a=1}^{b}(\delta_j + \tau_a)\right\}}$$
$$\theta_{\text{Res}} \sim \mathcal{N}(0, \sigma^2_{\text{Res}})$$

4 Interpreting the effect of covariates included in a PCM or an RSM

Usually, interpreting the effect of a covariate is performed using a two-steps procedure. First, the statistical significance is checked through the p-value. Then, if significant, the covariate's order of magnitude is evaluated for determining whether this effect has practical implications in real life.

Checking the statistical significance of a covariate introduced in a Rasch-family model can be easily performed using Wald tests. However, interpreting its magnitude is quite a challenge because that refers to unobservable latent-variable magnitude.

Nevertheless, solutions can be proposed for assisting in such an interpretation. For example, the estimated latent-trait variance can be partitioned into different components: parts explained by covariates and a residual part. Thus the proportion of latent-trait variance explained by introducing a covariate in the PCM can be fit through the estimate of the pseudo-type-III sum of squares associated with this covariate.

The pseudo-type-III sum of squares associated with a given covariate can be fit using nested models: one containing all the covariates to be introduced (full model) and another containing all of these covariates except the studied one (reduced model). The pseudo-type-III sum of squares is then computed as the difference between the residual sum of squares of the reduced model and the residual sum of squares of the full model. The proportion of latent-trait variance explained by introducing a covariate can finally be estimated as the ratio between the pseudo-type-III sum of squares and the residual sum of squares of the reduced model.

5 Tests of fit

5.1 Tests of fit computation

Testing the fit of a PCM or an RSM is one of the big issues when estimating parameters using MML procedures. Few tests of fit have been proposed, except the R1m test (testing the assumption of monotone increasing and parallel item response functions), the R2m test (testing the latent-trait unidimensionality), and the Si test (identifying the items contributing to a possible lack of fit) (Glas and Verhelst 1995).

We propose two tests in `pcmtest`: the R1m test and the Si test. These tests are based on grouping the individuals into G mutually exclusive subgroups by partitioning the latent traits in continuous and disjoint regions. Because the score is a sufficient statistic of the latent trait with the Rasch-family models, these subgroups are created by partitioning the observed scores of the studied questionnaire (Glas 1988). Then, the R1m and Si tests can be computed based on the differences observed in each region g ($g \in \{1, \ldots, G\}$) between the observed and expected number (based on a PCM or an RSM) of individuals responding l ($l \in \{0, \ldots, m_j\}$) to the item j.

These two tests are based on the linear function

$$\mathbf{d} = N^{1/2}\mathbf{U}' \left\{ \mathbf{p} - \pi\left(\widehat{\boldsymbol{\phi}}\right) \right\}$$

where N is the total number of individuals, $\widehat{\boldsymbol{\phi}}$ is a vector of MML estimates of the model parameters, $\pi(\widehat{\boldsymbol{\phi}})$ is the vector of the response pattern probabilities evaluated at $\widehat{\boldsymbol{\phi}}$, $\mathbf{p}$ is the associated vector of observed proportions, and $\mathbf{U}$ is the contrast matrix based on whether the performed test is the R1m test or the Si test.

The generalized Pearson statistic can then be written as

$$Q = \mathbf{d}'\mathbf{W}^{-}\mathbf{d}$$

where $\mathbf{W} = \mathbf{U}'\widehat{\mathbf{D}}_{\pi}\mathbf{U}$ and $\widehat{\mathbf{D}}_{\pi}$ is the diagonal matrix of the $\pi(\widehat{\boldsymbol{\phi}})$ elements. Q then has an asymptotic chi-squared distribution of rank($\mathbf{U}'\widehat{\mathbf{D}}_{\pi}\mathbf{U}$)-order($\boldsymbol{\phi}$)-1 degrees of freedom.

The differences between the R1m and the Si tests are therefore based on the constitution of the contrast matrix $\mathbf{U}$. For the R1m test, this contrast matrix is defined as a block diagonal matrix with $\mathbf{U} = (\mathbf{U}^1 \oplus \ldots \oplus \mathbf{U}^G)$. Each of the $\mathbf{U}^g$ is constructed so that $\mathbf{U}^g = [\mathbf{T}_1^g | \mathbf{T}_2^g]$. $\mathbf{T}_1^g$ is the complete disjunctive table of the item responses for each pattern response observable in the g subgroup. $\mathbf{T}_2^g$ is the complete disjunctive table of the possible scores for each pattern response observable in the g subgroup.

For example, consider a questionnaire composed of 3 items. Item 1 and item 2 have 2 response categories (0 and 1), whereas item 3 has 3 response categories (0, 1, and 2). The possible scores range from 0 to 4. We consider 2 subgroups, the first corresponding to individuals with scores ranging from 0 to 2 and the second to individuals with scores ranging from 3 to 4. Eight response patterns may lead to a score compatible with the first subgroup. Four response patterns may lead to a score compatible with the second subgroup.

$\mathbf{T}_1^1$ can be defined as follows ($r = 0$ means response to the item equal to 0):

item 1		item 2		item 3		
$r = 0$	$r = 1$	$r = 0$	$r = 1$	$r = 0$	$r = 1$	$r = 2$
1	0	1	0	1	0	0
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	0	1	0
1	0	0	1	0	1	0
1	0	1	0	0	0	1

$\mathbf{T}_2^1$ can be defined as follows ($s = 0$ means score equal to 0):

$s = 0$	$s = 1$	$s = 2$
1	0	0
0	1	0
0	1	0
0	1	0
0	0	1
0	0	1
0	0	1
0	0	1

Thus $\mathbf{U}^1 = [\mathbf{T}_1^1|\mathbf{T}_2^1]$ is shown below:

1	0	1	0	1	0	0	1	0	0
0	1	1	0	1	0	0	0	1	0
1	0	0	1	1	0	0	0	1	0
1	0	1	0	0	1	0	0	1	0
0	1	0	1	1	0	0	0	0	1
0	1	1	0	0	1	0	0	0	1
1	0	0	1	0	1	0	0	0	1
1	0	1	0	0	0	1	0	0	1

$\mathbf{U}^2$ is constructed in the same way:

0	1	0	1	0	1	0	1	0
0	1	1	0	0	0	1	1	0
1	0	0	1	0	0	1	1	0
0	1	0	1	0	0	1	0	1

Finally, the block diagonal matrix $\mathbf{U} = (\mathbf{U}^1 \oplus \ldots \oplus \mathbf{U}^G)$ is shown below:

$$\left(\begin{array}{ccccccc|ccc||cccccccc|cc}
1&0&1&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&1&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&1&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&1&1&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
1&0&0&1&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0\\
\hline
0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&1&0&1&0\\
\end{array}\right)$$

For the Si test, the contrast matrix $\mathbf{U}$ is constructed so that $\mathbf{U} = [\mathbf{T}_1|\mathbf{T}_2|\mathbf{Y}]$. $\mathbf{T}_1$ is the complete disjunctive table of the item responses for each pattern response. $\mathbf{T}_2$ is the complete disjunctive table of the possible scores for each pattern response. Thus, for a given survey and regardless of the tested item, $\mathbf{T}_1$ and $\mathbf{T}_2$ will always be the same. $\mathbf{Y}$ contains the relevant contrasts based on the G predefined subgroups. $\mathbf{Y}$ is therefore constructed as a block diagonal matrix with $\mathbf{Y} = (\mathbf{Y}^1 \oplus \ldots \oplus \mathbf{Y}^G)$. $\mathbf{Y}^g$ is the complete disjunctive table of the possible responses of the tested item for each pattern response observable in the g subgroup.

Let's continue the previous example to test the contribution of the first item to a possible lack of model fit. The $\mathbf{U}$ matrix for such a test is shown below:

$$\left(\begin{array}{ccccccc||ccccc||cc|cc}
1&0&1&0&1&0&0&1&0&0&0&0&1&0&0&0\\
0&1&1&0&1&0&0&0&1&0&0&0&0&1&0&0\\
1&0&0&1&1&0&0&0&1&0&0&0&1&0&0&0\\
1&0&1&0&0&1&0&0&1&0&0&0&1&0&0&0\\
0&1&0&1&1&0&0&0&0&1&0&0&0&1&0&0\\
0&1&1&0&0&1&0&0&0&1&0&0&0&1&0&0\\
1&0&0&1&0&1&0&0&0&1&0&0&1&0&0&0\\
1&0&1&0&0&0&1&0&0&1&0&0&1&0&0&0\\
\hline
0&1&0&1&0&1&0&0&0&0&1&0&0&0&0&1\\
0&1&1&0&0&0&1&0&0&0&1&0&0&0&0&1\\
1&0&0&1&0&0&1&0&0&0&1&0&0&0&1&0\\
0&1&0&1&0&0&1&0&0&0&0&1&0&0&0&1\\
\end{array}\right)$$

5.2 Power issues for tests of fit

Interpreting these tests of fit can be facilitated by estimating their a posteriori power. For example, if such a test is performed on a very large sample size, the rejection of the null hypothesis can be due either to the existence of a truly important inadequacy of the model to the data or to an inadequacy of minimal importance that has become

statistically significant given the excessive sample size (this situation can be related to a problem of overpower).

Such an a posteriori power is estimated using the noncentral chi-squared distribution (Patnaik 1949; Satorra and Saris 1985). Under the null hypothesis, the R1m and Si statistics follow an asymptotic chi-squared distribution. Under the alternative hypothesis, these test statistics follow an asymptotic noncentral chi-squared distribution with noncentrality parameter equal to the observed likelihood-ratio chi-squared statistic.

Let λ be the noncentrality parameter, η the number of degrees of freedom, and χ^2_t the threshold for rejecting the null hypothesis at a significance level equal to α. Let $F\chi^2_{\lambda,\eta}$ be the cumulative distribution function of a noncentral chi-squared distribution with noncentrality parameter equal to λ and η degrees of freedom. The a posteriori power of a test of fit with a type I error equal to α is $1 - F\chi^2_{\lambda,\eta}(\chi^2_t)$

Assuming an invariant distribution of the observed patterns of item responses, we can estimate the sample size for obtaining a given power and estimating a chi-squared statistic corresponding to a different sample size or to a different a posteriori power. Estimating a chi-squared statistic for a different sample size under the assumption of an identical pattern of item response distribution simply involves weighting the observed chi-squared statistic by the ratio between the desired sample size and the observed sample size.

6 The pcmodel command

The `pcmodel` command estimates the parameters of a PCM or an RSM using MML and includes covariates that can explain individuals' latent-trait differences. This command can run under Stata 11 and later.

6.1 Syntax

The syntax of the `pcmodel` command is detailed below:

```
pcmodel varlist [if] [in] [, categorical(varlist) continuous(varlist)
    difficulties(matrix_list) iterate(#) adapt robust from(matrix) rsm
    estimateonly nip(#) trace level(#)]
```

The `gllamm` (Rabe-Hesketh, Skrondal, and Pickles 2004, 2005) and `gausshermite` (Hardouin 2007) commands must be installed for `pcmodel` to work; type `ssc install gllamm` and `ssc install gausshermite`.

6.2 Options

`categorical(`*varlist*`)` specifies the categorical covariates included in the PCM or the RSM (that is, the potential categorical covariates that might explain latent-trait differences between individuals).

`continuous(`*varlist*`)` lists the continuous covariates included in the PCM or the RSM.

`difficulties(`*matrix_list*`)` specifies a list of row vectors containing the known values of each item difficulty (if they are known). If the `difficulties()` option is specified, there must be a vector for each item, with the same name as the corresponding items. If the `difficulties()` option is not filled, the item difficulties are considered unknown and are estimated during the analysis. This option cannot be used with the `rsm` option.

`iterate(#)` specifies the (maximum) number of iterations. With the `adapt` option, the `iterate(#)` option will cause `pcmodel` to skip the Newton–Raphson iterations usually performed at the end without updating the quadrature locations.

`adapt` causes adaptive quadrature to be used instead of ordinary quadrature.

`robust` specifies that the Huber/White/sandwich estimator of the covariance matrix of the parameter estimates be used.

`from(`*matrix*`)` specifies a row vector to be used for the initial values of the estimation iterative process. This vector must have exactly the number of parameters to be estimated, starting with the difficulties parameters, followed by the parameters associated with the covariates, and ending with the estimated standard deviation of the latent trait.

`rsm` performs an RSM instead of a PCM.

`estimateonly` specifies that the marginal McFadden's pseudo-R^2 and the type-III sum of squares computations not be performed.

`nip(#)` specifies the number of integration points to be used for each integral or summation. Only the following degrees are available: 5, 7, 9, 11, and 15.

`trace` causes more output to be displayed. Before estimation begins, details of the specified model are displayed. In addition, a detailed iteration log is shown including parameter estimates and log-likelihood values for each iteration.

`level(#)` sets confidence level. The default is `level(95)`.

6.3 Displayed outputs

`pcmodel` displays a first table corresponding to the estimation of the latent-trait parameters and a second table corresponding to the estimations of the items response category difficulties. If covariates are included in the model, their effects are displayed in the first table, together with the type-III sum of squares associated with them and the percentage of latent-trait variance they explain.

6.4 Stored results

`pcmodel` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(ll)`	marginal log-likelihood
`e(cn)`	condition number
`e(Nit)`	number of items
`e(Ncat)`	number of categorical covariates
`e(Ncont)`	number of continuous covariates
`e(sigma)`	estimated standard deviation of the latent trait
`e(Varsigma)`	variance of the estimated standard deviation of the latent trait

Matrices

`e(theta)`	coefficient vector of the parameters associated with the latent-trait covariates (if no covariate is included in the model, value of the average latent trait)
`e(Vartheta)`	covariance matrix for the latent-trait covariates.
`e(delta)`	estimated difficulty parameters
`e(Vardelta)`	covariance matrix for the estimated difficulty parameters
`e(b)`	overall estimated parameters of the PCM (or RSM)
`e(V)`	covariance matrix for the overall estimated parameters

7 The pcmtest command

The `pcmtest` command tests the fit of a PCM or an RSM to the observed data. The PCM or RSM should have been fit with one of the following commands before using the `pcmtest` command: `pcmodel`, `irt pcm`, or `irt rsm`.

7.1 Syntax

The syntax of the `pcmtest` command is

`pcmtest` [, `group(`*numlist*`)` `nfit(#)` `power(#)` `alpha(#)` `approximation` `new sitest` `graphics` `filegraph(`*filegraph*[, `replace`]`)`]

7.2 Options

`group(`*numlist*`)` defines the groups of individuals for performing fit tests by specifying the upper score limit of each group. If `group()` is not specified, groups are formed based on the score quartiles.

`nfit(#)` defines the sample size for which the test power must be calculated (`nfit()` deals with overpower problems when fit tests are performed on large samples). If `nfit()` is not filled, tests are performed on only the observed sample without adjusting sample size.

`power(#)` estimates the sample size required for performing the R1m test at a given power. If `power()` is not filled, tests are performed on only the observed sample without adjusting power.

`alpha(#)` specifies the type I error used to perform the tests of fit. The default is `alpha(0.05)`.

`approximation` specifies that the pattern response probabilities computation must be performed using simulations instead of Gauss–Hermite quadratures. This option shortens the computation time when the number of item or response categories is high.

`new` changes the computation methodology of the pattern response probabilities between several tests of fit rather than using the pattern response probabilities stored in Stata memory.

`sitest` performs item-specific test of fit (Si tests).

`graphics` displays several graphs: the distribution of the latent trait depending on the individual scores, the graph of `MAP`, the graph of the group contributions to the R1m statistic, and the graph of the observed and expected score distribution.

`filegraph(`*filegraph*`[ , replace ])` indicates the path and filename for saving the graphs (four graphs are stored: *filegraph*`_LT_Sc`, *filegraph*`_MAP`, *filegraph*`_Contrib`, and *filegraph*`_Score_Distrib`).

7.3 Displayed outputs

`pcmtest` displays a first table corresponding to the R1m test and a second table corresponding to the Si tests for each of the items. Tests are performed on the observed sample and, depending on the chosen options, possibly on virtual samples with sample size set to `nfit()` or sample size set so that the R1m test power is equal to `power()`.

7.4 Stored results

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

Matrices

`r(globalFitTot)`	R1m test results performed on the observed sample
`r(itemFitTot)`	Si test results performed on the observed sample
`r(globalFitPo)`	R1m test results performed on a virtual sample with sample size set so that the R1m test power is equal to `power()`
`r(itemFitPo)`	Si test results performed on a virtual sample with sample size set so that the R1m test power is equal to `power()`
`r(globalFitNu)`	R1m test results performed on a virtual sample with sample size set to `nfit()`
`r(itemFitNu)`	Si test results performed on a virtual sample with sample size set to `nfit()`

8 Example

The Gambling Attitudes and Beliefs Scale (Breen and Zuckerman 1999; Bouju et al. 2014) is used to illustrate both `pcmodel` and `pcmtest`. The scale is a 35-item question-

naire measuring a wide range of cognitive biases, irrational beliefs, subjective excitement, and positive attitudes experienced through gambling.

In this example, we analyze only the emotion subscale, measuring the subjective excitement experienced during gambling. This subscale consists of five items recorded under the names `gabs1`, `gabs18`, `gabs26`, `gabs27`, and `gabs35`. These items are recorded on a four-point scale ranging from "strongly agree" to "strongly disagree".

Players are characterized using three variables: gender (`Gender` = 1 for male and `Gender` = 2 for female); favorite type of game (`FavourGame` = 1 for pure chance games, `FavourGame` = 2 for chance games with quasiskill, and `FavourGame` = 3 for chance games with elements of skill); and the characteristics of their game practice (`PracticeChar` = 1 for nonpathological gamblers, `PracticeChar` = 2 for untreated pathological gamblers, and `PracticeChar` = 3 for pathological gamblers treated for their gambling practice).

We first perform a PCM including the three considered covariates:

```
. use data
. pcmodel gabs1 gabs18 gabs26 gabs27 gabs35,
> categorical(Gender FavourGame PracticeChar)
Iteration 0:   log likelihood = -4124.2257  (not concave)
Iteration 1:   log likelihood = -3656.9816  (not concave)
Iteration 2:   log likelihood = -3514.1578
Iteration 3:   log likelihood = -3465.3162
Iteration 4:   log likelihood = -3463.7758
Iteration 5:   log likelihood = -3463.7726
Iteration 6:   log likelihood = -3463.7726

McFadden´s pseudo R square and type III Sums of squares computation
                for Gender covariate
                for FavourGame covariate
                for PracticeChar covariate


Model : Partial Credit Model
                log likelihood: -3463.773
                Marginal McFadden´s pseudo R2: 8.6 %
                Number of individuals: 628
                Number of items: 5
                Number of covariates: 3

Parameters of the Latent trait distribution:
                Identifiability constraint: latent trait for Gender = 1,
> FavourGame = 1, PracticeChar = 1: set to 0
                Variance of the Latent trait: Sigma=0.579    (SE:0.070   )
```

Latent trait group effect:

	Coef.	S.E.	z	P>|z|	[95% C.I.]	
Gender:						
Gender: 1	0	.	.	.	.	.
Gender: 2	0.169	0.094	1.80	0.072	0.077	0.261
FavourGame:						
FavourGame: 1	0	.	.	.	.	.
FavourGame: 2	-0.008	0.097	-0.08	0.934	-0.103	0.087
FavourGame: 3	0.286	0.126	2.27	0.023	0.163	0.409
PracticeChar:						
Practice~r: 1	0	.	.	.	.	.
Practice~r: 2	1.040	0.100	10.38	0.000	0.942	1.138
Practice~r: 3	1.260	0.101	12.45	0.000	1.161	1.359

Proportion of latent trait variance explained by covariates

	SS.III	df	V.exp.	R2.exp.
Gender:	25.563	1	1.5%	0.5%
FavourGame:	5.225	2	0.3%	12.1%
PracticeChar:	956.797	2	35.7%	28.8%

	SS.res	df
Model without cov.	2693.981	2983
Full model	1724.453	2978

```
Items difficulty parameters:
```

Item	Coef.	S.E.	[95% C.I.]	
gabs1:				
response: 2	1.077	0.142	0.939	1.216
response: 3	0.834	0.157	0.681	0.987
response: 4	2.080	0.179	1.905	2.255
gabs18:				
response: 2	1.535	0.136	1.402	1.668
response: 3	2.006	0.182	1.828	2.183
response: 4	2.153	0.231	1.928	2.378
gabs26:				
response: 2	0.531	0.155	0.380	0.682
response: 3	0.306	0.154	0.155	0.456
response: 4	1.309	0.153	1.160	1.458
gabs27:				
response: 2	1.061	0.138	0.927	1.196
response: 3	1.176	0.158	1.022	1.330
response: 4	2.224	0.192	2.037	2.411
gabs35:				
response: 2	0.612	0.148	0.468	0.756
response: 3	0.599	0.153	0.449	0.748
response: 4	1.483	0.160	1.327	1.639

Then, we proceed to the test of fit. To obtain accurate statistics of the test, we form groups of at least 60 individuals based on the scores. The ranges of the scores in each of these groups is 0, 1–3, 4–5, 6–7, 8–9, and 10–15.

```
. pcmtest, power(0.95) group(0 3 5 7 9 15) sitest
Performing R1m test
1024 response pattern probabilities to compute
Percentage of completion
----|---10%---|---20%---|---30%---|---40%---|---50%
..................................................    50
..................................................   100
U matrix computation
W matrix computation
Performing Si test for the 1th item
----|---25%---|---50%---|---75%---|---100%
.....................................
Performing Si test for the 2th item
----|---25%---|---50%---|---75%---|---100%
.....................................
Performing Si test for the 3th item
----|---25%---|---50%---|---75%---|---100%
.....................................
Performing Si test for the 4th item
----|---25%---|---50%---|---75%---|---100%
.....................................
```

```
Performing Si test for the 5th item
----|---25%---|---50%---|---75%---|---100%
.....................................

Global tests of the fit : test R1m
                        groups : 0 3 5 7 9 15
                        Number of individuals with missing data : 28 (4.46%)
```

		N = 600			N = 113		
	df	R1m	p-val	Power	R1m	p-val	Power
R1m	70	262.7	0.0000	1.0000	49.2	0.9723	0.9500

```
Items specific tests of the fit : tests Si
```

		N = 600			N = 113		
Item	df	Si	p-val	Power	Si	p-val	Power
gabs1 :	15	66.2	0.0000	1.0000	12.4	0.6498	0.5694
gabs18 :	15	128.7	0.0000	1.0000	24.1	0.0637	0.9075
gabs26 :	15	81.3	0.0000	1.0000	15.2	0.4358	0.6849
gabs27 :	15	120.9	0.0000	1.0000	22.6	0.0925	0.8841
gabs35 :	15	80.9	0.0000	1.0000	15.1	0.4408	0.6823

The R1m global test of fit and all the Si item-oriented tests performed on the observed sample—that is, on the 600 individuals who completed all the items—show a bad fit. The estimated power of these tests is almost 100%. The large size of the studied sample is probably responsible for overpowering. Such an overpowering issue could be solved by performing tests on a smaller sample with the same distribution of patterns of item responses. The required sample size to perform the R1m test with a power equal to 95% is 113 subjects. With such a sample size, the tests of fit are no longer significant, which allows us to use and interpret the previous results from the `pcmodel` command.

`pcmodel` displays two tables of results. The first one corresponds to the estimates of the latent-trait distribution parameters and the association between the gambling subjective excitement and the considered covariates. The second table corresponds to the estimates of the difficulty parameters associated with each of the responses categories of the considered items.

In the first table, no association between gender and gambling subjective excitement is highlighted because the p-value associated with the "gender" covariate is greater than 5%. A statistical association is found between the gamblers' favorite types of games and their subjective excitement experienced during gambling. Those who play to chance games with elements of skill present a significantly higher excitement than pure chance gamers (p-value = 0.02). Finally, the gamblers' game-practice characteristics are statistically associated with their subjective excitement experienced: pathological gamblers (treated or untreated) present a significantly higher excitement than nonpathological

gamblers (p-value $< 10^{-3}$). From a statistical point of view, adjusting the gambling subjective excitement level on both the favorite type of game and the game-practice characteristics seems relevant. This can be confirmed by performing a likelihood-ratio test using `lrtest`.

Yet these statistical associations do not appear to explain equivalent parts of the gambling subjective excitement: the introduction in the model of the `FavourGame` variable explains only 0.3% of the overall experienced excitement variance, whereas the gamblers' game-practice characteristics explains about 36% of the overall experienced excitement variance. With these results, some may wonder whether it is necessary to adjust the gambling subjective excitement on the gamblers' favorite type of game. However, we should recall that the estimate of latent-trait variance proportion explained by introducing a covariate is only a tool for assisting in interpreting the effect of a covariate included in the model—not necessarily a rule for constructing statistical models.

Finally, we can explore whether the untreated pathological gamblers present an equivalent excitement as the pathological gamblers treated for their gambling practice. We can classically resolve such an issue using linear combinations of the covariate category estimators just after the `pcmodel` command with the Stata `lincom` command:

```
. lincom PracticeChar_3-PracticeChar_2
 ( 1)  - [estimates]PracticeChar_2 + [estimates]PracticeChar_3 = 0
```

theta	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
(1)	.2201164	.1007149	2.19	0.029	.0227187	.417514

Such linear combinations highlight the fact that treated and untreated pathological gamblers present a significantly different level of excitement (p-value = 2.9%).

9 Discussion

The `pcmodel` and `pcmtest` commands provide features not previously available in a statistical software, such as the inclusion of covariates in a PCM or an RSM, the assistance for interpreting the parameters associated with these covariates (by estimating the percentage of latent-trait variability explained by these covariates), and the implementation of tests of fit adapted for a PCM or an RSM estimated using MML.

Note that when data are missing, all the information of the data is used for estimating the parameters of the PCM or RSM: all the observed item responses are used even if questionnaires are incompletely filled out. However, the tests of fit can be performed on only the complete cases. Individuals who have not responded to all the items are necessarily excluded when the R1m and Si tests of fit are performed.

One limitation of the proposed commands is that only the first-order tests of fit (that is, the R1m and the Si tests) are proposed. Other important tests are not yet available with `pcmtest`, such as the second-order R2m test, which tests the unidimensionality principle underlying both a PCM and an RSM. Such a test should be developed and included in `pcmtest` in the future. Furthermore, other tests could also be implemented, such as the Martin Lof test for testing the unidimensionality assumption, the Anferson test for testing the specific objectivity, and other fit measurements such as the infit and outfit indexes.

10 References

Andrich, D. 1978. A rating formulation for ordered response categories. *Psychometrika* 43: 561–573.

Bouju, G., J.-B. Hardouin, C. Boutin, P. Gorwood, J.-D. Le Bourvellec, F. Feuillet, J.-L. Venisse, and M. Grall-Bronnec. 2014. A shorter and multidimensional version of the Gambling Attitudes and Beliefs Survey (GABS-23). *Journal of Gambling Studies* 30: 349–367.

Breen, R. B., and M. Zuckerman. 1999. 'Chasing' in gambling behavior: Personality and cognitive determinants. *Personality and Individual Differences* 27: 1097–1111.

Christensen, K. B. 2007. Latent covariates in generalized linear models: A Rasch model approach. In *Advances in Statistical Methods for the Health Sciences*, ed. J.-L. Auget, N. Balakrishnan, M. Mesbah, and G. Molenberghs, 95–108. Basel, Switzerland: Birkhäuser.

Glas, C. A. W. 1988. The derivation of some tests for the Rasch model from the multinomial distribution. *Psychometrika* 53: 525–546.

Glas, C. A. W., and N. D. Verhelst. 1995. Tests of fit for polytomous Rasch models. In *Rasch Models: Foundations, Recent Developments, and Applications*, ed. G. H. Fischer and I. W. Molenaar, 325–352. New York: Springer.

Hamel, J.-F., J.-B. Hardouin, T. Le Neel, G. Kubis, Y. Roquelaure, and V. Sébille. 2012. Biases and Power for Groups Comparison on Subjective Health Measurements. *PLOS ONE* 7: e44695.

Hardouin, J.-B. 2007. Rasch analysis: Estimation and tests with raschtest. *Stata Journal* 7: 22–44.

Hardouin, J.-B., and M. Mesbah. 2007. The SAS macro-program %AnaQol to estimate the parameters of item responses theory models. *Communications in Statistics—Simulation and Computation* 36: 437–453.

Lord, F. M., and M. R. Novick, eds. 1968. *Statistical Theories of Mental Test Scores*. Reading, MA: Addison–Wesley.

Masters, G. N. 1982. A Rasch model for partial credit scoring. *Psychometrika* 47: 149–174.

Patnaik, P. B. 1949. The non-central χ^2- and F-distributions and their applications. *Biometrika* 36: 202–232.

Rabe-Hesketh, S., A. Pickles, and C. Taylor. 2000. sg129: Generalized linear latent and mixed models. *Stata Technical Bulletin* 53: 47–57. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 293–307. College Station, TX: Stata Press.

Rabe-Hesketh, S., A. Skrondal, and A. Pickles. 2004. Generalized multilevel structural equation modeling. *Psychometrika* 69: 167–190.

———. 2005. Maximum likelihood estimation of limited and discrete dependent variable models with nested random effects. *Journal of Econometrics* 128: 301–323.

Rasch, G. 1960. *Probabilistic Models for Some Intelligence and Attainment Tests.* Copenhagen: Danmarks Paedagogiske Institute.

Rizopoulos, D. 2006. ltm: An R package for latent variable modeling and item response theory analyses. *Journal of Statistical Software* 17: 1–25.

Satorra, A., and W. E. Saris. 1985. Power of the likelihood ratio test in covariance structure analysis. *Psychometrika* 51: 83–90.

Thissen, D. 1982. Marginal maximum likelihood estimation for the one-parameter logistic model. *Psychometrika* 47: 175–186.

Zheng, X., and S. Rabe-Hesketh. 2007. Estimating parameters of dichotomous and ordinal item response models with gllamm. *Stata Journal* 7: 313–333.

About the authors

Jean-François Hamel is an assistant professor in biostatistics at the University Hospital of Angers. Jean-Benoit Hardouin and Véronique Sébille are, respectively, attached professor and full professor in biostatistics at the Faculty of Pharmaceutical Sciences of the University of Nantes. Their research applies item response theory in clinical research, especially differential item functioning and response shift. Gaëlle Challet-Bouju is coordinator of clinical studies for the Federal Institute of Behavioral Addictions (IFAC) at Nantes.

The Stata Journal (2016)
16, Number 2, pp. 482–490

Abadie's semiparametric difference-in-differences estimator

Kenneth Houngbedji
Paris School of Economics
Paris, France
kenneth.houngbedji@psemail.eu

Abstract. The difference-in-differences estimator measures the effect of a treatment or policy intervention by comparing change over time of the outcome variable across treatment groups. To interpret the estimate as a causal effect, this strategy requires that, in the absence of the treatment, the outcome variable followed the same trend in treated and untreated groups. This assumption may be implausible if selection for treatment is correlated with characteristics that affect the dynamic of the outcome variable. In this article, I describe the command `asdid`, which implements the semiparametric difference-in-differences (SDID) estimator of Abadie (2005, *Review of Economic Studies* 72: 1–19). The SDID is a reweighing technique that addresses the imbalance of characteristics between treated and untreated groups. Hence, it makes the parallel trend assumption more credible. In addition, the SDID estimator allows the use of covariates to describe how the average effect of the treatment varies for different groups of the treated population.

Keywords: st0442, absdid, semiparametric estimations, difference-in-differences, propensity score

1 The semiparametric difference-in-differences estimator

Let's consider the general setting of studies of causal effects used by Rosenbaum and Rubin (1983). We want to estimate the causal effect of a treatment on a variable of interest $\mathbf{y}$ at some time t. Each participant has two potential outcomes: $\mathbf{y}_{1t}$ and $\mathbf{y}_{0t}$. $\mathbf{y}_{1t}$ is the value of $\mathbf{y}$ if the participant received the treatment by time t. $\mathbf{y}_{0t}$ is the value of $\mathbf{y}$ if the participant had not received the treatment by time t. $\mathbf{d}$ is an indicator of whether or not a participant was treated by time t. At time $t = 0$, which is the baseline b, no one is treated. At time $t \neq 0$, $\mathbf{d}$ is equal to 1 for a treated participant and is equal to 0 otherwise. We want to estimate the average treatment effect on the treated (ATT):

$$\text{ATT} \equiv \mathbb{E}\left(\mathbf{y}_{1t} - \mathbf{y}_{0t} \,\middle|\, \mathbf{d} = 1\right)$$

Because $\mathbf{y}_{0t}$ is never observed for a treated participant, the ATT cannot be directly estimated. Assume $\mathbf{y}_{0b}$ is the value of $\mathbf{y}$ at time $t = 0$—that is, the baseline. $\mathbf{x}_b$ is a set of pretreatment characteristics, $\Delta\mathbf{y}_t \equiv \mathbf{y}_t - \mathbf{y}_b$ is the change of $\mathbf{y}$ between time t and the baseline b, and $\pi(\mathbf{x}_b) \equiv \mathbb{P}\left(\mathbf{d} = 1 \,\middle|\, \mathbf{x}_b\right)$ is the conditional probability to be in the treatment group (also called the propensity score). Abadie (2005) shows that the sample analog of

$$\mathbb{E}\left\{\frac{\Delta \mathbf{y}_{t}}{\mathbb{P}(\mathbf{d}=1)} \times \frac{\mathbf{d}-\pi(\mathbf{x}_{b})}{1-\pi(\mathbf{x}_{b})}\right\} \tag{1}$$

gives an unbiased estimate of the ATT if (2) and (3) hold.

$$\mathbb{E}\left(\mathbf{y}_{0t}-\mathbf{y}_{0b}\,\middle|\,\mathbf{d}=1\,,\,\mathbf{x}_{b}\right)=\mathbb{E}\left(\mathbf{y}_{0t}-\mathbf{y}_{0b}\,\middle|\,\mathbf{d}=0\,,\,\mathbf{x}_{b}\right) \tag{2}$$

$$\mathbb{P}(\mathbf{d}=1)>0 \quad \text{and} \quad \pi(\mathbf{x}_{b})<1 \tag{3}$$

The estimator is a weighted average of the difference of trend—$\Delta \mathbf{y}_{t}$—across treatment groups. It proceeds by reweighing the trend for the untreated participants based on their propensity score $\pi(\mathbf{x}_{b})$. Because $\{\pi(\mathbf{x}_{b})\}/\{1-\pi(\mathbf{x}_{b})\}$ is an increasing function of $\pi(\mathbf{x}_{b})$, untreated participants with a higher propensity score are given a higher weight.

Abadie (2005) suggests to approximate the propensity score $\pi(\mathbf{x}_{b})$ semiparametrically using a polynomial series of the predictors. Thereafter, the values predicted are plugged into the sample analogue of (1). Even though the approximation improves for higher polynomial order, the estimation becomes less precise. It is also possible to estimate $\pi(\mathbf{x}_{b})$ with the series logit estimator (SLE) (see Hirano, Imbens, and Ridder [2003]). This method uses a logit specification to constrain the estimated propensity score to vary between 0 and 1.

Consider, for instance, that $\widehat{\pi}(\mathbf{x}_{b})$ is the approximated propensity score and k is the order the polynomial function used to approximate $\pi(\mathbf{x}_{b})$. The approximation of $\pi(\mathbf{x}_{b})$ produced by the linear probability model (LPM) can be written as

$$\widehat{\pi}(\mathbf{x}_{b})=\widehat{\gamma}_{0}+\widehat{\gamma}_{1}\times\mathbf{x}_{1}+\sum_{i=1}^{\mathrm{k}}\widehat{\gamma}_{2i}\times\mathbf{x}_{2}^{i}$$

where $\mathbf{x}_{1}$ is a binary variable, $\mathbf{x}_{2}^{i}=\prod_{j=1}^{i}\mathbf{x}_{2}$, and $\mathbf{x}_{2}$ is a continuous variable. The coefficients $\widehat{\gamma}_{0},\widehat{\gamma}_{1},\widehat{\gamma}_{21},\ldots,\widehat{\gamma}_{2i},\ldots,\widehat{\gamma}_{2k}$ are estimated using an ordinary least-squares estimator.

With an SLE estimator approach, the propensity score $\pi(\mathbf{x}_{b})$ is estimated as follows:

$$\widehat{\pi}(\mathbf{x}_{b})=\Lambda\left(\widehat{\gamma}_{0}+\widehat{\gamma}_{1}\times\mathbf{x}_{1}+\sum_{k=1}^{\mathrm{K}}\widehat{\gamma}_{2k}\times\mathbf{x}_{2}^{k}\right)$$

where $\Lambda(x)=\exp(x)/\{1+\exp(x)\}$ is the logistic function. Higher order binary variables—like $\mathbf{x}_{1}$—are not considered because $\mathbf{x}_{1}^{k}=\mathbf{x}_{1}$ for any value $k>1$.

Independently of the approximation method used, the errors related to the estimation of the propensity scores are considered when estimating the standard error of the ATT as described in Abadie (2005). Other estimators use the propensity score to estimate the ATT. The kernel matching and nearest neighbor matching estimators are among the most widely used estimators for quasi experimental identification. However, both estimators assume that the propensity score is given, not estimated, and produce on average estimates with smaller standard errors than the estimator of Abadie.

2 The absdid command

The `absdid` command is the Stata equivalent of a MATLAB code written by Abadie (2005) in an empirical application of the semiparametric difference-in-differences (SDID) estimator.[1] `absdid` estimates the ATT by comparing change over time of the outcome of interest across treatment groups while adjusting for differences between treatment groups on the observable characteristics at baseline that are correlated to the propensity score.

2.1 Syntax

`absdid` *depvar* [*if*] [*in*], `tvar(`*varname*`)` `xvar(`*varlist*`)` [`yxvar(`*varlist*`)` `order(`#`)` `sle` `csinf(`#`)` `csup(`#`)`]

depvar is a variable that represents the change of the outcome of interest between baseline and post treatment for each observation.

2.2 Options

`tvar(`*varname*`)` is the binary treatment variable. It takes the value `1` when the observation is treated and takes the value `0` otherwise. `tvar()` is required.

`xvar(`*varlist*`)` are the control variables. They can be either continuous or binary and are used to estimate the propensity score. `xvar()` is required.

`yxvar(`*varlist*`)` is a list of variables that can modify the treatment effect. By default, the treatment effect is assumed to be constant.

`order(`#`)` represents the order of the polynomial function used to estimate the propensity score. It takes integer values and the default is `order(1)`.

`sle` forces the use of a logistic specification to estimate the propensity score (see Hirano, Imbens, and Ridder [2003]). This ensures, for instance, that the estimated propensity score is always greater than 0 and less than 1. By default, the propensity score is estimated with a linear regression.

`csinf(`#`)` drops the observations of which the propensity score is less than #. The default is `csinf(0)`.

`csup(`#`)` drops the observations of which the propensity score is greater than #. The default is `csup(1)`.

1. The original code is tailored to measure the effect of union membership on wages for workers. It is available at http://www.hks.harvard.edu/fs/aabadie/cdid_union.m.

3 Example

To illustrate how `absdid` works, let's reproduce the application exercise available on Abadie's website. We estimate the effect of participation in a worker union on wages of unionized female workers. The data used are an excerpt of the current population survey—a U.S. government monthly survey of unemployment and labor force participation. The data consist of female workers observed in 1996 and resurveyed in 1997 (see table 1). The workers were not unionized in 1996, so we can identify the union–wage effect on the workers who joined a worker union between 1996 and 1997.

Let $w_{1,97}$ be the wage of a worker in 1997 if she joined a worker union, and let $w_{0,97}$ be the wage if she had not joined a union. Because wage variations are traditionally modeled through a lognormal distribution, the parameter of interest is as follows:

$$\text{ATT}\left\{\log(w)\right\} \equiv \mathbb{E}\left\{\log(w_{1,97}) - \log(w_{0,97}) \,\middle|\, \mathbf{union}_{97} = 1\right\}$$

For simplicity, we report estimates of $\text{ATT}\{\log(w)\}$ and interpret the results as the percentage effect of worker union on wage.[2]

If female workers were randomly selected to join a union in 1997, one could estimate $\text{ATT}\{\log(w)\}$ by comparing the log of wages of unionized and nonunionized workers in 1997. To account for the female workers who joined a union in 1997 differing from those who remained nonunionized with respect to age, education level, and race—see table 1—we use an SDID approach.

Assume that, in the absence of worker unions, the wage dynamics of unionized workers would have been similar to that of nonunionized workers with the same age, education level, race, state of residence, and sector of activity. If that assumption holds, we can use the `absdid` command to compute the SDID estimator of the union–wage effect for female workers.

2. Actually, a more accurate estimate of the percentage effect of worker union on wage can be obtained using the transformation suggested by Kennedy (1981).

Table 1. Characteristics of female workers across treatment groups

Variables	Entire sample	Unionized	Non-unionized	Diff.
Union coverage in 1997	0.05			
	[0.22]			
Wage variables:				
Log wage in 1997	2.36	2.43	2.36	0.07 ***
	[0.52]	[0.49]	[0.53]	(0.02)
Log wage in 1996	2.30	2.34	2.30	0.04 **
	[0.54]	[0.52]	[0.54]	(0.02)
Covariates in 1996:				
Age (years)	39.33	40.37	39.27	1.09 ***
	[11.01]	[10.55]	[11.03]	(0.37)
High school	0.93	0.92	0.93	−0.01
	[0.26]	[0.27]	[0.26]	(0.01)
College	0.25	0.35	0.24	0.10 ***
	[0.43]	[0.48]	[0.43]	(0.01)
African American	0.10	0.19	0.09	0.10 ***
	[0.29]	[0.39]	[0.29]	(0.01)
Hispanic	0.06	0.07	0.06	0.01
	[0.24]	[0.26]	[0.24]	(0.01)
Married	0.63	0.63	0.63	−0.00
	[0.48]	[0.48]	[0.48]	(0.02)
Number of workers	18,470	958	17,512	18,470

Notes: Standard deviations are in brackets. Standard errors are in parentheses. Significance levels are denoted as follows: * p<0.10, ** p<0.05, and *** p<0.01.

First, we need a variable (`dlwage`) that, as suggested in (1), measures the change of log wage between baseline and follow-up. Second, we need a binary variable (`union97`) that indicates treated and untreated observations. Third, we need a list of control variables among which unionized and nonunionized workers differ from one another; let's consider the variables `age`, `black`, `hispanic`, and `grade`, which report the age, ethnic background, and education level of the workers in 1996. With these inputs, we can estimate the SDID estimator of the union–wage effect for female workers:

```
. absdid dlwage, tvar(union97) xvar(age black hispanic married grade)
Abadie´s semi-parametric diff-in-diff           Number of obs     =      18469
```

dlwage	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATT						
_cons	.0361469	.0163367	2.21	0.027	.0041276	.0681663

`Number of obs` shows the number of observations used for the estimation that satisfy (3), that is, the observations for which the estimated propensity score is bigger than 0 and smaller than 1. Though the sample has 18,470 observations, only 18,469 are used to estimate the ATT. This suggests that 1 observation has an estimated propensity score that either is smaller than or equal to 0 or is bigger than or equal to 1. This is not surprising because, by default, `absdid` uses a linear regression to estimate the propensity score; hence, the predicted values often can be either negative or bigger than 1. To avoid any loss of information, we can add the `sle` option.[3]

```
. absdid dlwage, tvar(union97) xvar(age black hispanic married grade) sle
Abadie´s semi-parametric diff-in-diff           Number of obs     =      18470
```

dlwage	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATT						
_cons	.0364533	.0163435	2.23	0.026	.0044207	.0684859

To discard observations with very small or very large propensity scores, we can use the `csinf` and `csup` options to indicate the lowest and highest acceptable values of the propensity score. In the example below, we restrict the estimation of the ATT to female workers whose propensity score is between 0.01 and 0.99.

```
. absdid dlwage, tvar(union97) xvar(age black hispanic married grade)
> csinf(0.01) csup(0.99)
Abadie´s semi-parametric diff-in-diff           Number of obs     =      18447
```

dlwage	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATT						
_cons	.0362135	.0163374	2.22	0.027	.0041928	.0682343

Independently of the method used to estimate the propensity score, the outputs of `absdid` show a point estimate of the ATT when the union–wage premium is constant and does not vary with worker characteristics. Overall, the results suggest that joining a worker union increased the wage of female workers by 3.6% in 1997. The effect does not vary with the option `sle`.

3. When `sle` is used, some observations can still be left out of the propensity score estimation when there is perfect prediction. This is the case, for instance, when all the workers in a given industry are either unionized or nonunionized. In those cases, the ATT is estimated only for the observations for which the treatment status is not perfectly predicted by observed characteristics.

Similarly, we can also consider that the effect of being in a union on wage varies with worker characteristics. For instance, the union–wage premium may vary with the age of the worker. Experienced workers—based on age—are often scarce in the economy. As such, they have more bargaining power and may not need to join a worker union to negotiate their wage. Hence, we may expect the union–wage premium to decrease with the age of the worker. Likewise, the union–wage premium may also vary with education level. Workers who have not completed high school should expect a higher premium compared with similar workers who have completed either high school or college. We see below the command for estimating how the union premium for female workers varies with age and education level.

```
. absdid dlwage, tvar(union97) xvar(age black hispanic married grade)
> yxvar(age hschool college) sle
Abadie´s semi-parametric diff-in-diff           Number of obs     =      18470
```

dlwage	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATT						
age	-.0036144	.0016146	-2.24	0.025	-.0067789	-.0004499
hschool	-.3099432	.1043214	-2.97	0.003	-.5144095	-.105477
college	.0562573	.0374896	1.50	0.133	-.0172211	.1297356
_cons	.4582553	.1356808	3.38	0.001	.1923259	.7241847

As expected, the results indicate that union premium decreases with age and education level. Considering that the average female worker of the sample was 39 years old in 1996, joining a worker union should increase the wage of the average female worker by 31.8% (that is, $0.458 - 39 \times 0.0036 = 0.3176$). In contrast, the premium is estimated at 16.1% for a worker who was 50 years old in 1996. Likewise, compared with workers who have no diploma in 1996, the union premium decreases by 31% for workers whose highest diploma is high school. Surprisingly, there is no statistically significant difference between the union premium of workers with a college diploma and those with no diploma. This is likely because of the small sample size: only 7.3% of female workers with a college diploma joined a union between 1996 and 1997.

To reproduce the results from table II of the empirical illustration available from Abadie's website, we need to consider other control variables that may affect the propensity score. We also need to increase the order of the polynomial function used to estimate the propensity score.

First, Abadie considers a larger list of control variables, including age, ethnic group, and fixed effects for education level, state of residence, sector of activity, and date of interview. Let's call this list `cvars` and save it in a macro:

```
. local cvars age black hispanic married i.grade i.state i.dind i.month
```

Second, Abadie uses a polynomial function of order 4 to estimate the propensity score. Using the control variables listed above and using 4 as the order of the polynomial function, we reproduce the results shown on Abadie's website for female workers:

```
. absdid dlwage, tvar(union97) xvar(`cvars´) order(4)
Abadie´s semi-parametric diff-in-diff           Number of obs     =      16374
```

dlwage	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATT						
_cons	.0327631	.0159989	2.05	0.041	.0014058	.0641203

```
. absdid dlwage, tvar(union97) xvar(`cvars´) yxvar(age hschool college) order(4)
Abadie´s semi-parametric diff-in-diff           Number of obs     =      16374
```

dlwage	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATT						
age	-.0031764	.001577	-2.01	0.044	-.0062673	-.0000856
hschool	-.1505565	.0648411	-2.32	0.020	-.2776427	-.0234703
college	.0388147	.0349236	1.11	0.266	-.0296343	.1072637
_cons	.2865646	.0955502	3.00	0.003	.0992897	.4738394

Those results are presented in columns (1) and (2) of table 2. They are similar to the union–wage premium for female workers found by Abadie in his empirical exercise.

Table 2. Effects of worker union on log of wage of female workers

Union premium (ATT)	LPM		SLE	
	(1)	(2)	(3)	(4)
Constant	0.0328 **	0.2866 ***	0.0399 **	0.3426 ***
	(0.0160)	(0.0956)	(0.0168)	(0.1082)
Age (years)		−0.0032 **		−0.0036 **
		(0.0016)		(0.0017)
High school		−0.1506 **		−0.1869 ***
		(0.0648)		(0.0724)
College		0.0388		0.0422
		(0.0349)		(0.0361)
Number of workers	16,374	16,374	18,273	18,273

Notes: Models (1) and (3) report estimates of the average union premium for unionized workers. Models (2) and (4) show how the union premium varies with worker age and education level. The average union premiums reported in (1) and (2) are estimated using a linear polynomial function of degree 4 to approximate the propensity score. The premiums reported in (3) and (4) are estimated using a logit specification of degree 4 to estimate the propensity score. Standard errors are in parentheses. Significance levels are denoted as follows: * p<0.10, ** p<0.05, and *** p<0.01.

4 Discussion

For a given set of control variables and predictors, the SDID estimates vary with the type of approximation used—`sle` or simple LPM (the default)—and the order of the polynomial approximation used—`order(#)`. To reduce the margin for arbitrage, one could use a cross validation technique to decide the combination of methods that best suits the semiparametric approximation of the propensity score. It can also help to consider that the LPM is likely to produce estimates of the propensity score that are either negative or greater than 1. When the SLE approximation is used, the observations for which the treatment status is perfectly predicted by a control variable are discarded from the estimation. In most cases, however, the sample size used to estimate the ATT is larger when the propensity score is approximated with the `sle` option.

Using our latest example as benchmark, table 2 shows how our estimates of the union premium for unionized workers vary depending on the type of approximation used.

To conclude, the SDID approach is mostly suited for longitudinal surveys with a baseline and follow-up rounds. To use `absdid`, the user needs to have a measure of the change of the main outcome variable over time for each observation along with treatment status and baseline characteristics.

5 References

Abadie, A. 2005. Semiparametric difference-in-differences estimators. *Review of Economic Studies* 72: 1–19.

Hirano, K., G. W. Imbens, and G. Ridder. 2003. Efficient estimation of average treatment effects using the estimated propensity score. *Econometrica* 71: 1161–1189.

Kennedy, P. 1981. Estimation with correctly interpreted dummy variables in semilogarithmic equations. *American Economic Review* 71: 801.

Rosenbaum, P. R., and D. B. Rubin. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika* 70: 41–55.

About the author

Kenneth Houngbedji is a researcher at the Paris School of Economics. His main research interests are studies of economic behavior and decision-making processes of households in developing countries to help design better public policies.

The Stata Journal (2016)
16, Number 2, pp. 491–510

Speaking Stata: Multiple bar charts in table form

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

Abstract. Tables that are one way, two way, or three way in structure may often be helpfully represented as multiple bar charts. The one, two, or three variables that define the structure of the table thus determine rows, columns, and panels in which bars are arranged. The merits of this design include easy focus on individual values or groups of values; leaving space for numeric information to be shown as in a table; and convenient axis or panel labeling through text rather than through a key or legend. A Stata command for these purposes, `tabplot`, is discussed systematically.

Keywords: gr0066, tabplot, bar charts, histograms, mosaic plots, spineplots, graphics, tables, categorical data

1 Introduction

Articles in this journal often introduce new Stata commands. This column introduces an *old* Stata command, which goes back to 1999. `tabplot` for plotting multiple bar charts has been discussed briefly in previous columns (Cox 2004, 2008b, 2012). Here I discuss the main ideas more systematically. Without contradiction or paradox, one may say that while the leading idea behind `tabplot` has been well known in many sciences for decades, it is nevertheless neglected in several others, and possibly even unknown in some quarters.

"Multiple bar charts" is one simple umbrella term that covers what `tabplot` can do. However, that does not mean stacked or divided bars, and it does not mean bars side by side on the same axis. A lengthier but less ambiguous explanation might be one-, two- or three-way bar charts in table form.

Why do we need another bar chart command in Stata? Bar charts are basic and may seem well supported in Stata. Only a little acquaintance with the manual documentation reveals four main official commands, `graph bar`, `graph hbar`, `twoway bar`, and `twoway rbar`. That might seem already three more than one might need. Indeed, `tabplot` is just a wrapper for `twoway rbar`. However, it can produce various plots more easily and more quickly than you could do yourself, unless you were willing to do a little programming and a lot of fiddling around.

In a nutshell, the main conceit of `tabplot` is tablelike plots. The name is intended to evoke commands like `tabulate` with their structured output of tables in rows and columns.

Section 2 dives straight in with two substantial examples of `tabplot`. Section 3 discusses other features, and section 4 links to previous literature. Section 5 gives a different kind of example. Section 6 is a more formal statement of `tabplot` syntax.

2 Illustrations

2.1 Two-way table example

Greenacre (2007, 42) published the dataset we will use for the first example. The data come from the 1997 Encuesta Nacional de la Salud (Spanish National Health Survey). They are interesting in themselves, but for present purposes, they are useful as an example simple enough not to require specialist knowledge but large enough to be a little challenging. As with many tables, the main handle for understanding is to look at the probability distribution of the response, which is here an ordered (ordinal, graded) variable, `health`, given a predictor, here another ordered variable, `agegroup`. As often, a cross-sectional dataset is partly of interest for what it might convey indirectly about longitudinal patterns.

`tabplot` offers options to calculate percent or proportional (fractional) breakdowns on the fly. Aesthetic preferences or social conventions often encourage presentation in terms of percents rather than proportions. ("Percentage" seems to me too long a word, whatever dictionaries may say.)

```
clear
input byte(agegroup health) long freq
1 1 243
1 2 789
1 3 167
1 4 18
1 5 6
2 1 220
2 2 809
2 3 164
2 4 35
2 5 6
3 1 147
3 2 658
3 3 181
3 4 41
3 5 8
4 1 90
4 2 469
4 3 236
4 4 50
4 5 16
5 1 53
5 2 414
5 3 306
```

```
5 4 106
5 5 30
6 1 44
6 2 267
6 3 284
6 4 98
6 5 20
7 1 20
7 2 136
7 3 157
7 4 66
7 5 17
end
label values agegroup agegroup
label def agegroup 1 "16-24", modify
label def agegroup 2 "25-34", modify
label def agegroup 3 "35-44", modify
label def agegroup 4 "45-54", modify
label def agegroup 5 "55-64", modify
label def agegroup 6 "65-74", modify
label def agegroup 7 "75+", modify
label values health health
label def health 1 "very good", modify
label def health 2 "good", modify
label def health 3 "regular", modify
label def health 4 "bad", modify
label def health 5 "very bad", modify
tabplot health agegroup [w=freq], percent(agegroup) showval   ///
     subtitle(% of age group) xtitle("") bfcolor(none)
```

Figure 1 shows the result.

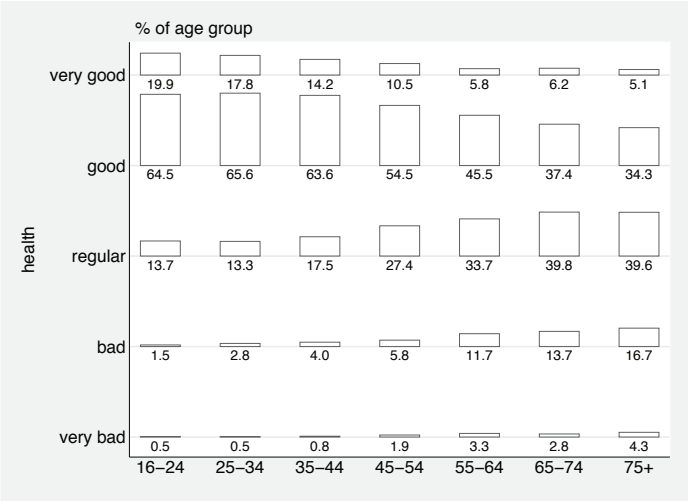


Figure 1. Two-way bar chart showing perception of health, with percent breakdown by age group according to the 1997 Encuesta Nacional de la Salud (Spanish National Health Survey). Source of data: Greenacre (2007, 42).

`tabplot` with two variables matches a standard `twoway` convention in that the first-named variable is plotted on the y axis. Usually, but not invariably, this variable is the response or outcome variable of interest. The first-named variable thus defines rows of the two-way bar chart, and the second-named variable, plotted on the x axis, defines columns of the chart.

In a simple application like this, `tabplot` counts frequencies of cross-combinations before producing a graph. As seen, it accepts frequency weights for the common case in which data are aggregated by cross-combinations of categories and frequencies are held in another variable. Although plots of raw frequencies are often of interest, it is common to show proportional or percent breakdowns, usually conditional on one or more predictor variables.

Figure 1 exemplifies a common feature with categorical data. Some categories may be relatively rare. There are some very small percents in the plot. A merit of the multiple-bar-charts design is that small values are discernible as such.

The `showval` option (think "show values") is a crucial detail of `tabplot`. Although not a default, it is likely to be used in most applications of the command. If we do insist on showing values too, then we are deliberately making use of table form as well as graph form. If percents are shown, one decimal place is the default (12.3, say). If proportions are shown, three decimal places are the default (0.123, say).

The facility to show individual values is most valuable when there is a desire to "look up" those values. This can come either before or after the graph is drawn. What is the value for such-and-such compared with others? Look, that value is a worrying 4.2%! But that 66.6% is good news!

Some researchers argue graphs are graphs and tables are tables, and ne'er the twain shall meet. An intriguing suggestion, which I have borrowed elsewhere (Cox 2003), is that the conventional distinction between graphs and tables was a side effect of the development of printing. Before printing, there were manuscripts—those scripted manually or written by hand—to which writers could freely add instructive and entertaining illustrations, whether factual, fabulous, or fantastic. Printed documents encouraged, or even enforced, a division of labor between typesetters and those who prepared illustrations. But now that division is largely obsolete. Writers and publishers have more flexibility in mixing text and illustrations of whatever kind, including not only figures and tables but also hybrid graphs and tables (see also Cox [2008a]).

Given this dataset, how else would you represent the patterns graphically? Setting aside any temptation to draw multiple pie charts, you might opt for the alternative of a stacked bar chart. Figure 2 shows one result.

```
graph bar (count) [fw=freq], over(health, descending) over(agegroup)     ///
     percent subtitle(% of age group) stack asyvars bar(5, bcolor(gs1))  ///
     bar(4, bcolor(gs4)) bar(3, bcolor(gs7)) bar(2, bcolor(gs10))        ///
     bar(1, bcolor(gs14)) legend(pos(3) column(1)) yscale(r(-5 100))     ///
     ylabel(, angle(h))
```

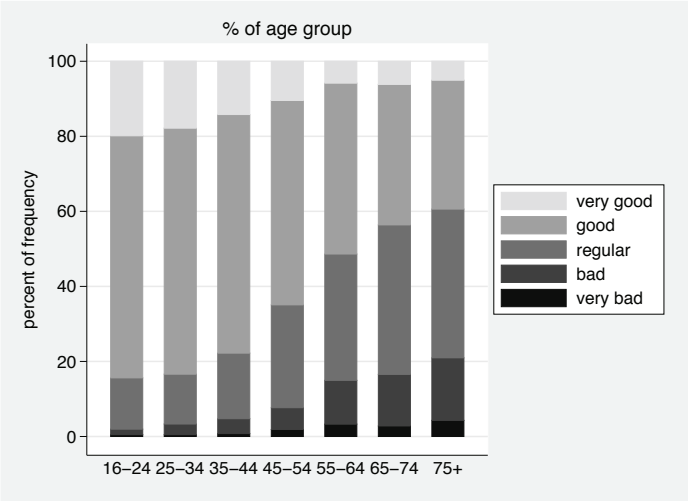


Figure 2. Stacked bar chart showing perception of health, with percent breakdown by age group according to the 1997 Encuesta Nacional de la Salud (Spanish National Health Survey). Source of data: Greenacre (2007, 42).

This example shows the constraints of printing in black and white. You might well prefer to experiment without that constraint. Here is one command with less inhibited use of color, which you can try for yourself if interested:

```
graph bar (count) [fw=freq], over(health, descending) over(agegroup)        ///
     percent subtitle(% of age group) stack asyvars                         ///
     bar(5, bfcolor(red*0.8)) bar(4, bfcolor(red*0.3) blcolor(red*0.8))     ///
     bar(3, bfcolor(blue*0.2) blcolor(blue*1.2))                            ///
     bar(2, bfcolor(blue*0.7) blcolor(blue*1.2)) bar(1, bcolor(blue*1.2))   ///
     legend(position(3) column(1)) yscale(range(-5 100)) ylabel(, angle(h))
```

Another graph that is popular in some fields is a mosaic plot, or more specifically a spineplot, here made using the `spineplot` command (Cox 2008b, 2016). Figure 3 is one such plot.

```
spineplot health age [w=freq], xlabel(, labsize(*0.8) axis(2)) percent      ///
     xlabel(0(20)100, tposition(outside)) ylabel(0(20)100, axis(2))         ///
     bar1(bcolor(gs1)) bar2(bcolor(gs4)) bar3(bcolor(gs7))                  ///
     bar4(bcolor(gs10)) bar5(bcolor(gs14)) barall(blwidth(none))
```

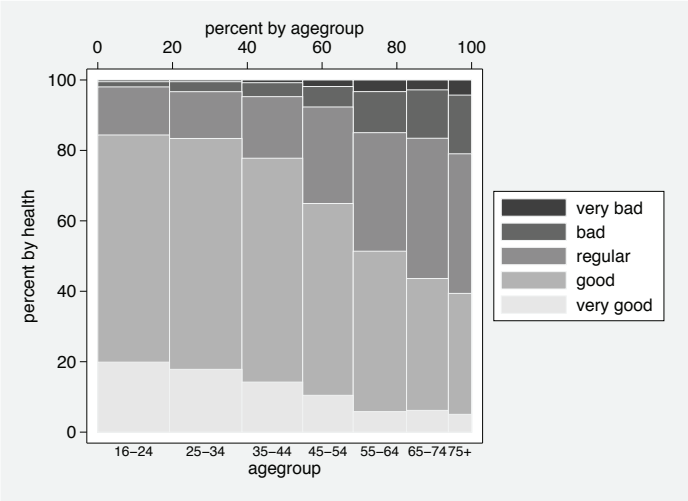


Figure 3. Spineplot (mosaic plot) showing perception of health, with percent breakdown by age group according to the 1997 Encuesta Nacional de la Salud (Spanish National Health Survey). Source of data: Greenacre (2007, 42).

As often emphasized, spineplots have various distinctive advantages. They faithfully show raw frequencies as well as row and column proportions. The area of each tile is proportional to absolute frequency, here the number of people in each cross-combination. The vertical and horizontal sides of each tile represent the proportions defined by the row and column variables. Collectively, the pattern of tiles shows the relationship between the variables. Independence of variables would imply a simple pattern in which row tiles are aligned, just as column tiles are. Departures from independence show departures from that benchmark.

Conversely, the spineplot design necessarily implies that tiles may be very small (or even not exist). In figure 3, three small tiles are occluded completely. However, that could be seen as a fault of the implementation of `spineplot`: in some other software, it is common to insert small spaces between tiles for readability.

In general, however, spineplots are based on the principle that tile areas represent absolute or relative frequencies. `tabplot`, like most bar chart programs, shows bars equal in width or height in one dimension and so is based on the simpler principle that those frequencies are encoded by bar heights or lengths in the other dimension.

So, which graphs are better (say, clearer, more effective, more attractive)? People can easily disagree here, depending partly on personal experience and taste and partly on ideas about what the readers of any graph will appreciate. There is often resistance to adopting unfamiliar designs. Sometimes that disinclination seems to imply that you must never show a graph that readers may not have seen before, which raises the question of how anyone ever learned about a graph for the first time.

Stacking is a well-understood design, but very small amounts are hard to spot on a stacked graph. Similarly, any legend obliges mental "back and forth" from readers (or else they give up on looking at the detail). Both `graph bar` and `spineplot` let you add numeric values on top of the bars or tiles. However, for this example, and for many others with some small amounts to show, that would be at least a little messy. The tradeoff is difficult to get right: different colors or shadings are essential to distinguish the different bars in stacked graphs, but that undermines showing numeric values too.

2.2 Three-way table example

Aitkin et al. (1989, 242) reported data from a survey of student opinion on the Vietnam War taken at the University of North Carolina at Chapel Hill in May 1967. Students were classified by sex, year of study, and the policy they supported, given the following choices:

A. The United States should defeat the power of North Vietnam by widespread bombing of its industries, ports, and harbors and by land invasion.

B. The United States should follow the present policy in Vietnam.

C. The United States should de-escalate its military activity, stop bombing North Vietnam, and intensify its efforts to begin negotiation.

D. The United States should withdraw its military forces from Vietnam immediately.

The labels A through D are charmless, but even at this distance, suggesting better ones might be thought contentious politically, so I will let them stand. What is important is that the sequence is ordered, from "hawk" to "dove" in contemporary terms.

```
clear
input str6 sex str8 year str1 policy int freq
"male" "1" "A" 175
"male" "1" "B" 116
"male" "1" "C" 131
"male" "1" "D" 17
"male" "2" "A" 160
"male" "2" "B" 126
"male" "2" "C" 135
"male" "2" "D" 21
"male" "3" "A" 132
"male" "3" "B" 120
"male" "3" "C" 154
"male" "3" "D" 29
"male" "4" "A" 145
"male" "4" "B" 95
"male" "4" "C" 185
"male" "4" "D" 44
"male" "Graduate" "A" 118
"male" "Graduate" "B" 176
"male" "Graduate" "C" 345
```

```
"male" "Graduate" "D" 141
"female" "1" "A" 13
"female" "1" "B" 19
"female" "1" "C" 40
"female" "1" "D" 5
"female" "2" "A" 5
"female" "2" "B" 9
"female" "2" "C" 33
"female" "2" "D" 3
"female" "3" "A" 22
"female" "3" "B" 29
"female" "3" "C" 110
"female" "3" "D" 6
"female" "4" "A" 12
"female" "4" "B" 21
"female" "4" "C" 58
"female" "4" "D" 10
"female" "Graduate" "A" 19
"female" "Graduate" "B" 27
"female" "Graduate" "C" 128
"female" "Graduate" "D" 13
end
tabplot policy year [w=freq], by(sex, subtitle(% by sex and year, place(w)) ///
     note("")) percent(sex year) showval
```

Figure 4 shows a three-way bar chart.

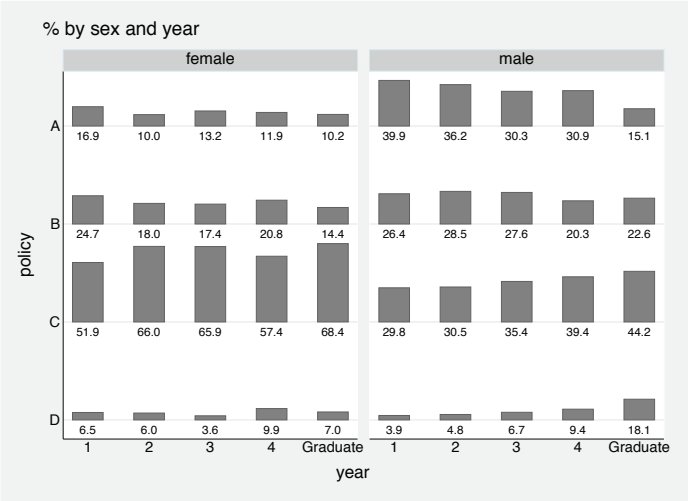


Figure 4. Three-way bar chart showing student preferences for policy options in the Vietnam War, University of North Carolina at Chapel Hill in May 1967, with percent breakdown by sex and year of study. Source of data: Aitkin et al. (1989, 242).

As with the previous example, the data are cross-sectional, not longitudinal, however tempting an interpretation may be in terms of students changing their opinions over time as they get older.

The way to plot three-way tables is by using a `by()` option to repeat two-way tables. As with two-way tables, it is usually best to specify the response or outcome variable first, as defining rows of the plot and as to be shown on the y axis. The order of the other two variables is at choice. There can be tradeoffs or compromises, because no layout is best for all purposes. Big differences can safely be put at a distance (so males and females here differ markedly in their mix of views), while finer comparisons are easier to make if bars are close. On top of all that, any ordinal scales should naturally be respected as such. A wider issue, not addressed directly here, is the scope for correspondence analysis or some other multivariate method to choose a good ordering for other sets of categories, an approach known in archeology and other sciences as seriation (Hahsler, Hornik, and Buchta 2008). This was precisely the machinery behind table 1 in Roberts et al. (2013).

As with the previous example, are there graph forms that make the data easier to think about?

3 More features

One-way bar charts are also possible with `tabplot`. The main reason for using them is if `graph bar`, `graph hbar`, and `histogram` do not satisfy, particularly if you prefer the display provided by the `showval()` option.

We have shown the default of vertical bars. A simple alternative is to use the `horizontal` option to get horizontal bars. In that case, it is usually easier to compare down columns. Some experimentation with both forms will often be helpful.

`tabplot` *varname*`, by()` is another way to plot two-way tables.

Four-way or higher charts would often not be readable or interpretable, but there are three evident ways to attempt them. First, try to `reshape` or otherwise restructure the data concerned to fewer variables. Second, combine variables, usually predictor variables, into a composite variable to be shown on one axis. See Cox (2007) for discussion of how to do that. Third, use `tabplot` repeatedly and then use `graph combine`.

`tabplot` with the `xasis` option may be useful for stacking histograms vertically. Less commonly, with the `yasis` and `horizontal` options, it may be useful for stacking them horizontally. The help file gives examples. In general, specify a variable containing equally spaced midpoints, and assign to it an appropriate variable label. `tabplot` will do the rest. Omit the `percent()` option for display of frequencies.

A method for subverting `tabplot` to plot any variable that takes on a single value for each cross-combination of categories is illustrated in the examples in the help file. The key is to select precisely one observation for each cross-combination and to specify that variable as (most generally) an `iweight`.

Furthermore, using an `iweight` is the only possible method whenever a variable has at least some negative values. In that case, you might do the following:

1. Consider changing the maximum height with `height()` to avoid overlap of bars variously representing positive and negative values. By default, `tabplot` chooses the scale to accommodate the longest bar to be shown, but it contains no special intelligence otherwise to avoid overlap of bars in the same column or row.
2. If also using `showval` or `showval()`, consider changing `offset()` and using a transparent `bfcolor()`.

Yet further variants can be specified through other options.

No example here shows a large table, partly because you need a big monitor or a bigger paper size than we have in the *Stata Journal* to do one justice. I have had reasonable results with `tabplot` and tables on the order of a thousand cells. The emphasis is then typically on overall patterns, although striking details can sometimes be seen. `showval()` is usually out of the question. For larger tables, other tricks can be helpful, such as suppressing small amounts or truncating large amounts using the `minimum()` or `maximum()` option; coloring different kinds of bars differently; and showing bar counts or amounts on some transformed scale, typically a square-root scale. The last is achieved by transforming before the program is called and passing the transformed amounts as importance weights.

4 Remarks on the literature

Bar charts presented as one row or one column of bars go back at least as far as Playfair (1786). See, for example, Playfair (2005, 25) or Wainer (2005, 45; 2009, 174).

Bar charts presented in table form with two or more rows and two or more columns are less common. They have been used in one form of pollen diagram. Sears (1933, 1935) gave some early examples. See also Emeny (1934) for a well-illustrated monograph on raw materials. The help file for `tabplot` gives many other references. Among more recent work, the examples and exhortations of Bertin (1981, 1983) are especially notable. See also Morrison (1985), de Falguerolles, Friedrich, and Sawitzki (1997), and Chauchat and Risson (1998) for articles influenced by Bertin.

As the example of pollen diagrams shows, the same form of graph can be used for showing on any one axis either the categories of what is regarded as one variable or two or more variables considered similar or comparable. Such bar charts, or similar displays, are also known as

- aligned bar charts and multipane bar charts (see Mackinlay [1986] and McDaniel and McDaniel [2012a; 2012b]);
- survey plots (see Lohninger [1994; 1996], Hoffman and Grinstein [2002], Grinstein et al. [2002], and Ward, Grinstein, and Keim [2010]);

- table lens (Rao and Card [1994], Pirolli and Rao [1996], Spence [2007], Ward, Grinstein, and Keim [2010], and Few [2012]); and

- multiple bar charts (as here) and fluctuation diagrams (see Becker, Chambers, and Wilks [1988], Unwin, Theus, and Hofmann [2006], Hofmann [2008], Theus and Urbanek [2009], and Unwin [2015]).

Such bar charts may require no more than `reshape`. Shortly, I will give an example with archaeological data.

Displays such as bar and pie charts with added numeric labels have been called "grables" (Hink, Wogalter, and Eustace 1996; Hink, Eustace, and Wogalter 1998; Bradstreet 2012). Somehow, this term has not proved widely attractive.

Note also what are often called Hinton diagrams or Hinton plots in machine learning. Rumelhart, Hinton, and Williams (1986) is a token reference. Examples occur in mainstream machine-learning texts such as MacKay (2003), Bishop (2006), Barber (2012), and Murphy (2012).

Brinton (1939, 142, 505) uses the term "two-way bar chart" for back-to-back or bilateral bar charts, a use different from that here.

5 A different kind of example

Doran and Hodson (1975, 259) gave these archaeological data from the rock shelter of Ksar Akil, near Beirut, which relate to the Upper Paleolithic. The data are counts of three kinds of artifacts. Although the data arrive as three count variables together with level, an indicator of depth (highest-numbered level is uppermost), it is easy to reshape them to a two-way table.

```
clear
input levels freqcores freqblanks freqtools
    25 21 32 70
    24 36 52 115
    23 126 650 549
    22 159 2342 1633
    21 75 487 511
    20 176 1090 912
    19 132 713 578
    18 46 374 266
    17 550 6182 1541
    16 76 846 349
    15 17 182 51
    14 4 51 14
    13 29 228 130
    12 135 2227 729
end
reshape long freq, i(levels) j(type) string
tabplot levels type [w=freq], bfcolor(none) horizontal barwidth(1)      ///
     percent(levels) subtitle(% at each level) showval(offset(0.45))    ///
     xscale(r(0.8 .)) yasis
```

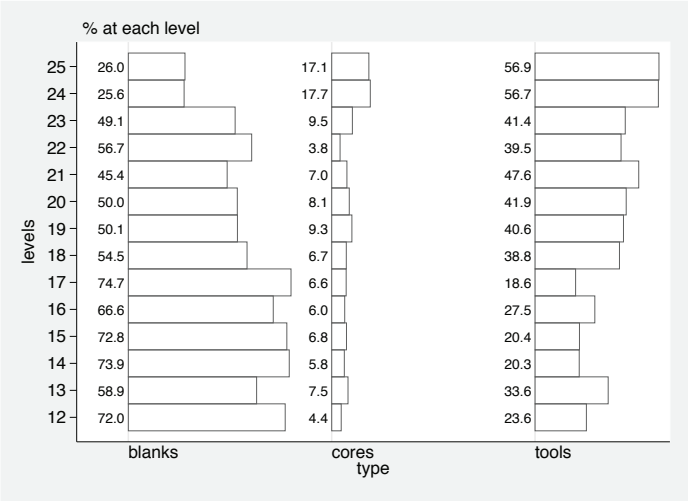


Figure 5. Two-way bar chart showing variations in relative abundance of cores, blanks, and tools from various levels at Ksar Akil. Source of data: Doran and Hodson (1975, 259).

Figure 5 shows how other choices can seem natural, or at least convenient. In archaeology, and also commonly in the earth and environmental sciences, variations with depth below (or height above) the surface are commonly plotted with depth or height on the y axis. That is how data were collected, and it is how researchers often think about their problems (see Cox and Barlow [2008] for another example).

Given a choice of depth for the y axis, and the `yasis` option to enforce a literal reading of levels, horizontal bars are called for. The percents go to the left of each bar, and for variety, as well as logic, the bars are made to touch using the `barwidth()` option. A subtle default is that axis ticks are explicit with the `yasis` or `xasis` option, the inference being that the scale is measured or counted, so that ticks are then natural. That is just the default, and it can be overruled.

6 The tabplot command

6.1 Syntax

`tabplot` *varname* [*if*] [*in*] [*weight*] [`,` *options*]

`tabplot` *rowvar colvar* [*if*] [*in*] [*weight*] [`,` *options*]

`fweight`s, `aweight`s, and `iweight`s may be specified.

See section 6.3 for the available *options*.

6.2 Description

`tabplot` plots a table of numerical values (for example, frequencies, fractions, or percents) in graphical form as a bar chart. It is mainly intended for representing contingency tables for one, two, or three categorical variables. It also has uses for producing multiple histograms and graphs for general one-, two-, or three-way tables.

`tabplot` *varname* creates a bar chart that by default displays one set of vertical bars; with the `horizontal` option, it displays one set of horizontal bars. The categories of *varname* thus define either columns from left (low values) to right (high values) or rows from top (low values) to bottom (high values). The value (for example, frequency, fraction, or percent) for each column or row is shown as a bar.

`tabplot` *rowvar colvar* follows standard tabular alignment: the categories of *rowvar* define rows from top (low values) to bottom (high values), and the categories of *colvar* define columns from left (low values) to right (high values). The value (for example, frequency, fraction, or percent) for each combination of row and column is shown as a bar with default alignment vertical.

The default bar width is 0.5. Use the `barwidth()` option to vary width, but note that all bars will have the same width.

By default, both variables are mapped on the fly in sort order to successive integers from 1 and up, but original values or value labels are used as value labels: this may be varied by use of the `yasis` or `xasis` option. The maximum bar height is by default 0.8. Use the `height()` option to vary this.

6.3 Options

`fraction` indicates that all frequencies should be shown as fractions (with sum 1) of the total frequency of all values being represented in the graph.

`fraction(`*varlist*`)` indicates that all frequencies should be shown as fractions (with sum 1) of the total frequency for each distinct category defined by the combinations of *varlist*. Usually, *varlist* will be one or more of the variables specified.

`percent` indicates that all frequencies should be shown as percents (with sum 100) of the total frequency of all values being represented in the graph.

`percent(`*varlist*`)` indicates that all frequencies should be shown as percents (with sum 100) of the total frequency for each distinct category defined by the combinations of *varlist*. Usually, *varlist* will be one or more of the variables specified.

Only one of the `fraction[()]` and `percent[()]` options may be specified.

`missing` specifies that any missing values of any of the variables specified should also be included within their own categories.

`yasis` and `xasis` specify, respectively, that the y (row) variable and the x (column) variable are to be treated literally (that is, numerically). Most commonly, each option will be specified if the variable in question is a measured scale or a graded variable with gaps. If values 1 to 5 are labeled A to E, but no value of 4 (D) is present in the data, `yasis` or `xasis` prevents a mapping to 1 (A) ... 4 (E).

`horizontal` specifies horizontal bars. The default is vertical bars.

`height(#)` controls the amount of graph space taken up by bars. The default is `height(0.8)`. Note that the height may need to be much smaller or much larger with `yasis` or `xasis`, given that the latter take values literally.

`showval` specifies that numeric values be shown beneath (or if `horizontal` is specified, to the left of) bars.

`showval` may also be specified with a variable name and options. If options alone are specified, no comma is necessary. In particular,

`showval(`*varname*`)` specifies that the values to be shown are those of *varname*. For example, the values of some kind of residuals might be shown alongside frequency bars.

`showval(offset(#))` specifies an offset between the base (or left-hand edge) of the bar and the position of the numeric value. The default is 0.1 with two variables or 0.02 with one variable. Tweak this if the spacing is too large or too small.

`showval(format(`*format*`))` specifies a format with which to show values. Specifying a format will often be advisable with nonintegers; for example, using `showval(format(%2.1f))` specifies rounding to 1 decimal place. Note that with a specified variable, the format defaults to the format of that variable; with percent options, the format defaults to %2.1f (1 decimal place); with fraction options, the format defaults to %4.3f (3 decimal places).

`showval(`*varname*`, format(%2.1f))` is an example of *varname* specified with options. As usual, a comma is needed in such cases.

Otherwise, the options of `showval()` can be options of `scatter` (see [G-2] **graph twoway scatter**), most usually marker label options.

`minimum(#)` suppresses plotting of bars with values less than the minimum specified, in effect setting them to zero.

`maximum(#)` truncates bars with values more than the maximum specified to show that maximum.

`separate(`*sepspec*`)` specifies that bars associated with different *sepspec* will be shown differently, most obviously using different colors. *sepspec* is passed as an argument to the `by()` option of `separate`, except that references to `@` are first translated to be references to the quantity being plotted.

A call to `separate()` may be supplemented with calls to options `bar1()` ... `bar20()` or to `barall()`. The arguments should be options of `twoway rbar`.

`bar1(`*rbar_options*`)` ... `bar20(`*rbar_options*`)` are provided to allow overriding the defaults on up to 20 categories, the first, second, etc., shown. The limit of 20 is arbitrary and more than any user should really want.

`barall(`*rbar_options*`)` overrides the defaults for all bars. Any `bar#()` option always overrides `barall()`. Thus if you wanted thicker `blwidth()` on all bars, you could specify `barall(blwidth(thick))`. If you wanted to highlight the first category only, you could specify `bar1(blwidth(thick))`.

graph_options refers to options of `twoway rbar`; see [G-2] **graph twoway rbar**. Among others:

`barwidth(#)` specifies the widths of the bars. The default is `barwidth(0.5)`. This may need changing, especially with the `xasis` or `yasis` option or if you wish bars to touch, exactly or nearly.

`bfcolor(`*colorstyle*`)` adjusts the bar fill color. In particular, Stata's defaults often imply that bars are filled with strong colors, but unfilled bars created by using `bfcolor(none)` may be more subtle and just as clear.

`by(`*varlist*`)` specifies another variable used to subdivide the display into panels.

`recast(`*newplottype*`)` recasts the graph as another twoway plottype. In practice, `recast(rspike)` is the main alternative.

`subtitle(`*tinfo*`)`, shown by default outside the graph and at top left, specifies what kind of quantity is being shown: `"frequency"`, `"percent"`, and so forth. The examples in the help file show how the subtitle is changed, which may mean being blanked out.

`plot(`*plots*`)` provides a way to add other plots to the generated graph. Allowed in Stata 8.

`addplot(`*plots*`)` provides a way to add other plots to the generated graph. Allowed in Stata 9 and later.

With large datasets especially, one should ensure that the plot or the extra plots do not contain information repeated for every observation within each combination of *rowvar* and *colvar*. The examples in the help file show one technique for avoiding this.

7 Conclusions

`tabplot` implements a simple twist on bar charts that can impart helpful spin. By separating bars, yet preserving table layout, we allow focus on individual values as well as overview of bars as a collective pattern. This can be especially useful for two- or three-dimensional tables. There is, at least if tables are not too large, space for numeric information to be shown, just as in a table. A further small virtue is convenient axis or panel labeling through text rather than through a key or legend.

8 Acknowledgments

This column is a personal salute to two very different people. Jacques Bertin (1918–2010) was a leader in cartography and visualization, a maverick (and even mischievous) maestro of the graphic. Bar charts in table form were among his recommended designs. Sir David MacKay (1967–2016) was enormously original and influential in a range from machine learning, Bayesian style, to sustainable energy policy. The illustrations in his books (MacKay 2003, 2009) are miniatures of economy and elegance.

Bob Fitzgerald, Friedrich Huebler, and Martyn Sherriff found typos in earlier versions of the `tabplot` help file. Friedrich also pointed to various efficiency issues. Marcello Pagano provided encouragement and found a bug. Vince Wiggins suggested how best to align x-axis labels when bars are horizontal. Jay Goodliffe suggested flagging use of the `subtitle()` option in the help.

9 References

Aitkin, M., D. Anderson, B. Francis, and J. Hinde. 1989. *Statistical Modelling in GLIM*. Oxford: Oxford University Press.

Barber, D. 2012. *Bayesian Reasoning and Machine Learning*. Cambridge: Cambridge University Press.

Becker, R. A., J. M. Chambers, and A. R. Wilks. 1988. *The New S Language: A Programming Environment for Data Analysis and Graphics*. Pacific Grove, CA: Wadsworth and Brooks/Cole.

Bertin, J. 1981. *Graphics and Graphic Information Processing*. Berin: De Gruyter.

———. 1983. *Semiology of Graphics: Diagrams, Networks, Maps*. Madison: University of Wisconsin Press.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*. New York: Springer.

Bradstreet, T. E. 2012. Grables: Visual displays that combine the best attributes of graphs and tables. In *A Picture is Worth a Thousand Tables: Graphics in Life Sciences*, ed. A. Krause and M. O'Connell, 41–69. New York: Springer.

Brinton, W. C. 1939. *Graphic Presentation*. New York: Brinton Associates.

Chauchat, J.-H., and A. Risson. 1998. Bertin's graphics and multidimensional data analysis. In *Visualization of Categorical Data*, ed. J. Blasius and M. Greenacre, 37–45. San Diego, CA: Academic Press.

Cox, N. J. 2003. Speaking Stata: Problems with tables, Part I. *Stata Journal* 3: 309–324.

———. 2004. Speaking Stata: Graphing categorical and compositional data. *Stata Journal* 4: 190–215.

———. 2007. Stata tip 52: Generating composite categorical variables. *Stata Journal* 7: 582–583.

———. 2008a. Speaking Stata: Between tables and graphs. *Stata Journal* 8: 269–289.

———. 2008b. Speaking Stata: Spineplots and their kin. *Stata Journal* 8: 105–121.

———. 2012. Speaking Stata: Axis practice, or what goes where on a graph. *Stata Journal* 12: 549–561.

———. 2016. Software Updates: Spineplots and their kin. *Stata Journal* 16: 521–522.

Cox, N. J., and N. L. M. Barlow. 2008. Stata tip 62: Plotting on reversed scales. *Stata Journal* 8: 295–298.

de Falguerolles, A., F. Friedrich, and G. Sawitzki. 1997. A tribute to J. Bertin's graphical data analysis. In *Softstat '97: Advances in Statistical Software 6: The 9th Conference on the Scientific Use of Statistical Software, March 3–6, 1997*, ed. W. Bandilla and F. Faulbaum, 11–20. Stuttgart: Lucius & Lucius.

Doran, J. E., and F. R. Hodson. 1975. *Mathematics and Computers in Archaeology*. Edinburgh: Edinburgh University Press.

Emeny, B. 1934. *The Strategy of Raw Materials: A Study of America in Peace and War*. New York: Macmillan.

Few, S. 2012. *Show Me the Numbers: Designing Tables and Graphs to Enlighten*. 2nd ed. Burlingame, CA: Analytics Press.

Greenacre, M. 2007. *Correspondence Analysis in Practice*. 2nd ed. Boca Raton, FL: Chapman & Hall/CRC.

Grinstein, G. G., P. E. Hoffman, R. M. Pickett, and S. J. Laskowski. 2002. Benchmark development for the evaluation of visualization for data mining. In *Information Visualization in Data Mining and Knowledge Discovery*, ed. U. Fayyad, G. G. Grinstein, and A. Wierse, 129–176. San Diego, CA: Academic Press.

Hahsler, M., K. Hornik, and C. Buchta. 2008. Getting things in order: An introduction to the R package seriation. *Journal of Statistical Software* 25(3): 1–34. https://www.jstatsoft.org/article/view/v025i03.

Hink, J. K., J. K. Eustace, and M. S. Wogalter. 1998. Do grables enable the extraction of quantitative information better than pure graphs or tables? *International Journal of Industrial Ergonomics* 22: 439–447.

Hink, J. K., M. S. Wogalter, and J. K. Eustace. 1996. Display of quantitative information: Are grables better than plain graphs or tables? *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 40: 1155–1159.

Hoffman, P. E., and G. G. Grinstein. 2002. A survey of visualizations for high-dimensional data mining. In *Information Visualization in Data Mining and Knowledge Discovery*, ed. U. Fayyad, G. G. Grinstein, and A. Wierse, 47–82. San Diego, CA: Academic Press.

Hofmann, H. 2008. Mosaic plots and their variants. In *Handbook of Data Visualization*, ed. C. Chen, W. Härdle, and A. Unwin, 617–642. Berlin: Springer.

Lohninger, H. 1994. INSPECT: A program system to visualize and interpret chemical data. *Chemometrics and Intelligent Laboratory Systems* 22: 147–153.

———. 1996. INSPECT*: A Program System for Scientific and Engineering Data*. Berlin: Springer.

MacKay, D. J. C. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge: Cambridge University Press.

———. 2009. *Sustainable Energy—Without the Hot Air*. Cambridge: UIT Cambridge.

Mackinlay, J. D. 1986. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics* 5: 110–141.

McDaniel, E., and S. McDaniel. 2012a. *The Accidental Analyst: Show Your Data Who's Boss*. Seattle, WA: Freakalytics.

McDaniel, S., and E. McDaniel. 2012b. *Rapid Graphs with Tableau Software 7: Create Intuitive, Actionable Insights in Just 15 Days*. Seattle, WA: Freakalytics.

Morrison, P. S. 1985. Symbolic representation of tabular data. *New Zealand Journal of Geography* 79: 11–18.

Murphy, K. P. 2012. *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: MIT Press.

Pirolli, P., and R. Rao. 1996. Table lens as a tool for making sense of data. In *AVI '96: Proceedings of the Workshop on Advanced Visual Interfaces*, ed. T. Catarci, M. F. Costabile, S. Levialdi, and G. Santucci, 67–80. New York: Association for Computing Machinery.

Playfair, W. H. 1786. *The Commercial and Political Atlas*. London: Robinson, Sewell, and Debrett.

———. 2005. *The Commercial and Political Atlas and Statistical Breviary*. Cambridge University Press: Cambridge. Edited by H. Wainer and I. Spence.

Rao, R., and S. K. Card. 1994. The table lens: Merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *CHI '94: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ed. B. Adelson, S. Dumais, and J. S. Olson, 318–322. New York: Association for Computing Machinery.

Roberts, D. H., D. J. A. Evans, J. Lodwick, and N. J. Cox. 2013. The subglacial and ice-marginal signature of the North Sea Lobe of the British–Irish ice sheet during the last glacial maximum at Upgang, North Yorkshire, UK. *Proceedings of the Geologists' Association* 124: 503–519.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323: 533–536.

Sears, P. B. 1933. Climatic change as a factor in forest succession. *Journal of Forestry* 31: 934–942.

———. 1935. Types of North American pollen profiles. *Ecology* 16: 488–499.

Spence, R. 2007. *Information Visualization: Design for Interaction*. 2nd ed. Harlow, UK: Pearson.

Theus, M., and S. Urbanek. 2009. *Interactive Graphics for Data Analysis: Principles and Examples*. Boca Raton, FL: Chapman & Hall/CRC.

Unwin, A. 2015. *Graphical Data Analysis with R*. Boca Raton, FL: Taylor & Francis.

Unwin, A., M. Theus, and H. Hofmann. 2006. *Graphics of Large Datasets: Visualizing a Million*. New York: Springer.

Wainer, H. 2005. *Graphic Discovery: A Trout in the Milk and Other Visual Adventures*. Princeton, NJ: Princeton University Press.

———. 2009. *Picturing the Uncertain World: How to Understand, Communicate, and Control Uncertainty through Graphical Display*. Princeton, NJ: Princeton University Press.

Ward, M., G. Grinstein, and D. Keim. 2010. *Interactive Data Visualization: Foundations, Techniques, and Applications*. Natick, MA: A K Peters.

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 "Speaking Stata" articles on graphics from 2004 to 2013 have been collected as *Speaking Stata Graphics* (College Station, TX: Stata Press, 2014).

The Stata Journal (2016)
16, Number 2, pp. 511–516

Review of Christopher F. Baum's An Introduction to Stata Programming, Second Edition

Clyde Schechter
Albert Einstein College of Medicine
Bronx, NY

Abstract. In this article, I review *An Introduction to Stata Programming, Second Edition*, by Christopher F. Baum (2016 [Stata Press]).

Keywords: gn0070, book review, introduction, Stata, programming, isp2

1 Introduction

StataCorp has made initial adoption of its software easy through the graphical user interface (GUI), but that will only take one so far. The GUI provides limited capacity for automating analyses: there are occasional commands that the GUI cannot access. It also provides limited ability for users to customize their output. Anyone who uses Stata often will eventually encounter such limitations. After using the GUI, a Stata user may begin using the command interface; from there, it is a short step to saving and executing analyses in do-files, at which point the user is programming. For many Stata users, Stata is their first programming language. Even for those who might be familiar with other statistical packages, or who have some experience with a general-purpose computer language such as C, Python, or Java, Stata can seem difficult. At first glance, an expression like `‘"‘:word ‘=‘‘j’’+1’’ of ‘vlist’"’` can be intimidating. Stata's comprehensive manuals are intended as references and, except for the brief [GS] section, are not designed to teach Stata programming. But there are resources for learning Stata programming, in the form of both courses and books. In this article, I review a recent update of one such self-study book.

In the preface to *An Introduction to Stata Programming, Second Edition*, Baum proposes an ambitious agenda. Starting almost from square one, his book leads the reader through the basics of Stata, do-file programming, ado-file programming, and Mata programming in 395 pages. At first, I was skeptical of the possibility. However, after working with the book for several months, I have become a believer.

The book's structure is innovative. Although the first two chapters are a didactic presentation of Stata fundamentals, thereafter the chapters alternate between a didactic presentation of a topic and a "cookbook" chapter that presents a realistic (and occasionally real) problem and develops its solution. The solutions often begin with a simplified, basic approach, followed by enhancements that improve the code or add capabilities. As such, the flow of the "cookbook" chapters genuinely reflects the process

that an experienced Stata programmer would follow when solving real problems. While it is quite typical of programming textbooks to mix didactics with worked problems, examples that are this realistic and extensive, built up incrementally, are exceptional. The "cookbook" metaphor, though, understates the value of the presentation. None of my kitchen's tomes explain the rationale behind the recipes, nor do they review a recipe afterward and suggest ways to improve it!

The book's order of presentation, in addition to being logical, enables a reader to proceed through each chapter topic while starting and stopping at sections where his or her own goals are addressed. For example, while all regular Stata users must be able to write do-files, the ability to write ado-files is needed only by some. The first 10 chapters suffice for the former, and those readers can stop there. Similarly, an experienced do-file writer interested in developing new estimators for Stata could begin with chapter 11 and continue through the end of the book without missing anything.

2 Content

An Introduction to Stata Programming, Second Edition, is not a comprehensive reference for Stata or Mata commands. The book thoroughly covers the basic commands that regular Stata programmers need to use repeatedly. It discusses graphics but only lightly—there are other lengthy books devoted to Stata's pluripotential `graph` command. The book does not discuss Bayesian analysis and item response theory commands, introduced in Stata 14, nor does it discuss survival analysis or multilevel modeling. These omissions are noted not as criticisms but to emphasize that this book focuses on teaching Stata programming and is not a comprehensive resource for Stata users.

What, then, is in the book? Chapter 1 begins by explaining the difference between using Stata casually and programming in Stata. This chapter explains the reasons why a user might want to become a programmer. Chapter 2 covers the fundamentals: creating and organizing do-files, program files, and datasets and importing data from other formats. One of the most important parts of this book is the brief but content-rich section on programming style near the end of chapter 2. In teaching this, the book refrains from adopting specific variants of coding style that programmers often argue about. However, it still emphasizes that every programmer needs to adopt a set of rules and conventions to follow consistently.

Chapter 3 discusses the creation and transformation of variables, and it introduces by-groups and Stata's macros (which are string constants, not to be confused with parameterized blocks of code sometimes referred to as macros in other software). The centrality of these concepts in Stata programming cannot be overstated. The presentation in this chapter is complete but terse. The topics discussed truly come to life in chapter 4, where we see them used to solve real data management problems.

Chapters 5 and 6 provide the remaining essentials of data management and results presentation. When I teach data analysis, my first law is "Never trust anybody else's data," and my second is "Never trust your own data." These chapters introduce and develop the all-important `assert` command and its application to real-life data cleaning. They also include an exposition of the `reshape`, `append`, and `merge` commands and details on accessing the stored results of Stata's commands.

If statistical programming required a license, the core of the examination would be based on the first six chapters of this book. Anyone who masters these six chapters is well positioned to effectively and efficiently create properly organized and cleaned analytic datasets and develop coherent, tailored displays of analytic results.

Chapters 7 and 8 discuss the automation of repetitive calculations. The versatile `foreach` and `forvalues` commands receive an extensive treatment in these chapters—along with `by:`, these commands are the fundamental constructs for repetitive calculations in Stata. Their importance is reflected in the generous amount of space allocated to them. These chapters also cover specialized repetition commands, such as the `statsby:` and `rolling:` prefixes, as well as the still more specialized commands `permute`, `bootstrap`, `jackknife`, and `simulate`.

Chapters 9 and 10 introduce some less-used techniques, including the `postfile` suite, the `file` suite, `sersets`, and data characteristics. While these all belong in the toolkit of the advanced programmer, they are primarily useful for creating highly customized outputs, and most Stata programmers could function for long intervals without recourse to these.

The reader who learns the content of these first 10 chapters will be prepared to effectively analyze real datasets in Stata and present well-organized, concise output. The book would be well worth the cost and effort even if one stopped here.

Chapters 11 and 12 introduce the fundamentals of ado-file programming, including the `program`, `syntax`, `return`, and `marksample` commands. Ado-files contain Stata programs, which are blocks of code assigned a name by which they can be invoked and which usually accomplish some fairly general-purpose task. The programs written in ado-files, unlike the code written in do-files, are intended to be used with a variety of datasets. Thus, for example, ado-file programs generally do not reference specific variable names. They are written to work with "generic" data. These chapters also teach how to write programs to implement functions for commands such as `egen`, `nl`, and `ml`. The book, starting from chapter 11, becomes more difficult because of the inherently greater complexity of the content. Chapter 11 is also longer than all previous chapters, roughly twice the length of any other. Even moderately experienced do-file programmers should lightly review chapters 1–10 before starting here, because any gaps in earlier knowledge may cause problems. The worked examples in chapter 12 reflect the increased difficulty of the material. Although the first few examples are simple and straightforward, the difficulty soon increases, and, as befits the kinds of things that ordinarily warrant writing an ado-file program, the underlying statistics become more detailed. While there is nothing in the chapter that a well-trained statistician will find

intimidating, users with only an introductory-level statistics course behind them may find some of the content challenging.

The first 12 chapters expose the reader to the most important techniques used in Stata programming. One might consider printing out a "diploma" after mastering these chapters! Of course, to really consolidate this learning, one needs practice and experience.

Chapters 13 and 14 focus on StataCorp's other programming language, Mata. I put off learning Mata for several years, and I can tell you that I have yet to discover anything that can be done in Mata that cannot be somehow accomplished in Stata. But I have also seen on Statalist how Mata can sometimes be used to reduce a long Stata program to just a few lines of code. Mata programs can also be compiled and will execute faster than the corresponding Stata code. For professional statisticians who want to develop and implement their own models and estimators, Mata provides the advanced matrix-calculation and manipulation capabilities needed. Chapter 13 begins, appropriately, by discussing the interface between Stata and Mata and the use of Mata both in programs and interactively from the command line. The chapter then presents basic matrix operations followed by more advanced functions. It then explains how to write a Mata program and how to create libraries of Mata programs.

I found these two final chapters to be the "proof of the pudding". Although I have been writing Stata programs for over two decades and was already familiar with nearly everything in the first 12 chapters, I had never learned Mata. So I took the plunge with this book. I found the book's presentation of Mata to be clear and logical. After completing these final chapters, I felt capable of using Mata with reasonable facility, although I still need more practice before I can consider myself to have truly mastered it. But this book provided me with the necessary groundwork to move forward, and I now understand things that I could never quite grasp from using only the Mata user manual.

3 Strengths and limitations

A difficulty faced by anyone teaching Stata programming is how to address a professionally diverse audience. Baum is an economist, and most of the examples in this book use economic datasets and apply tools and techniques frequently used in economics and finance. While there are some examples based in other fields, they are the exception. Some substantive scientific background is provided with these examples, but readers who are completely unfamiliar with these disciplines may need to turn to other sources (a dictionary, search engine, or online reference materials) to clarify some vocabulary.

Let me emphasize that this book does not purport to teach statistics, and it will not be helpful if that is your need. Furthermore, it is not organized as a reference for experienced Stata programmers. Although the text is well indexed and it is easy to locate information on certain commands, it can be difficult to find details on specific programming techniques without already knowing where to look. Moreover, there are

large swaths of Stata commands never mentioned in the book. In short, this book will not supplant Stata's online help files and user manuals.

The material in chapter 11 is more difficult than that in the earlier chapters. If the reader has mastered all chapters before 11, then he or she should be able to proceed without trouble. But for a reader with only minimal experience writing do-files and wanting only to quickly learn how to write a short special-purpose program, starting with this chapter may prove too difficult.

The layout and size of the book are convenient. The font is easy on the eyes, and the book fits easily into a backpack, briefcase, or medium-sized handbag. Users will find the typographic conventions in the book to be quite familiar from the Stata user manuals.

Baum is also the curator of Boston College's Statistical Software Components Stata code repository. That position enables him to introduce the reader to many user-written tools that make performing important tasks easier and quicker. I learned several new commands this way while reading the book for this review. The book also discusses the pros and cons of using prepackaged solutions that achieve generality by including many user-specified options as well as the pros and cons of writing your own narrowly tailored code from the basics. Any commands used in the book that are not part of official Stata are available from the Statistical Software Components archive. The datasets used in the book and the example do- and ado-files are all available on the Stata Press website.

One aspect of this book that I particularly like is its emphasis on programming style. The bare-minimum requirement of any program is that it must produce correct results. Beyond that, programs often require periodic extension or revision, so it is important that other programmers, and even the original programmer, can easily return to a program and understand how it works. Commenting, code formatting, and following certain conventions greatly enhance the human readability of code and make it easier to maintain. This is a lesson that many programmers (in any language) learn the hard way, often after having a bad experience using a program they wrote several months earlier and having little recall of how it works. The importance of programming style is often overlooked in introductory programming courses, but programming style is not just for advanced programmers. Ideally, good coding style should be practiced starting with the very first line of code that one ever writes. It is gratifying that this point is made early in this book, specifically near the end of chapter 2 for do-files and again in chapter 11 for ado-files, and that the code examples all adhere to the best practices.

I was also pleased by the book's focus on data validation. Data analysis sometimes results in unpleasant surprises, such as unexpected error messages from `xtset` about duplicate observations or regression results that cannot possibly be right and are due to incorrectly entered data (such as data claiming a mother is 15 years younger than her daughter or that an adult weighs 5 kg). Worse still are the results that are not obviously wrong and that you learn about much later when others complain that your results cannot be reproduced. The generous use of `assert`ions in do-files that create analytic datasets, taught in chapter 5, is the best way to prevent these errors.

I would have liked to see more emphasis on program correctness in both the didactics and examples of chapters 11 and 12. Although the book discusses and illustrates validation scripts, little is said about designing programs to catch user errors. A program should not only produce correct answers when given valid inputs but also identify when its inputs are not valid and fail gracefully with an informative error message. While the `syntax` command does a thorough validation of variable lists and option types, it is still the programmer's responsibility to range check the values of variables and arguments and to verify any assumptions needed for the algorithm to work correctly with the data. This type of checking for program correctness deserves more emphasis and illustration.

The book does not include any unworked exercises for the reader to attempt on his or her own. A learner whose work or school environment provides problems to work will not be disadvantaged by this. However, for a learner not in such an environment who wants to expand his or her skills, this will be a limitation. Instructors considering using this book as a text to accompany a course on Stata will need a supplementary workbook with problems.

4 Conclusion

An Introduction to Stata Programming, Second Edition, is a well-written and superbly organized book. It is suitable for a Stata user at any level who wants to learn Stata programming or improve already acquired programming skills. The reader can advance from neophyte to expert in Stata (and Mata) programming in useful but manageable increments by studying this book one or two chapters at a time. The extensively developed examples of realistic programming problems enliven the didactic presentation. The methods taught in this book can be applied to cutting-edge statistical work, but the book's presentation and most of the examples assume readers have only the statistical knowledge commonly possessed by graduate students and early-career data analysts. The book will be especially appealing to learners who have a background in finance or economics.

5 Reference

Baum, C. F. 2016. *An Introduction to Stata Programming*. 2nd ed. College Station, TX: Stata Press.

About the author

Clyde Schechter is a professor of family and social medicine at the Albert Einstein College of Medicine, where he works as an epidemiologist and focuses on clinical and health services research projects involving many specialties and disciplines. He has used Stata since version 4, is an active participant in the Statalist forum, and is an occasional contributor to the *Stata Journal*.

The Stata Journal (2016)
16, Number 2, pp. 517–520

Stata tip 126: Handling irregularly spaced high-frequency transactions data

Christopher F. Baum
Department of Economics
Boston College
Chestnut Hill, MA
baum@bc.edu

Sebastiaan Bibo
University of Amsterdam
Amsterdam, Netherlands

A wealth of high-frequency data is available on the Internet for many financial markets. Many of these datasets are "tick data", or transactions data, for which one observation represents one trade. Trades may occur at irregular intervals in calendar time, depending on market activity. To study these data—in particular, to analyze data from more than one market produced in the same time frame—one must convert irregularly spaced transactions data to time intervals. The observation for one time interval usually contains the number of trades during the interval (which could be zero), the average price of the good traded, and both the high and the low price during the interval. Other measures, such as standard deviations for each interval or total trade volume, may also be recorded.

In this tip, we illustrate how a transactions-based financial dataset of bitcoin trades can be converted to a dataset of standardized time intervals so that these datasets from multiple markets may be juxtaposed to evaluate the presence of arbitrage opportunities, which occur when the same good sells for different prices in different markets. That is only a necessary condition: if arbitrage is to be profitable, transactions costs and bid-ask spreads must be considered.

We begin by downloading the raw data for one market from http://api.bitcoincharts.com/v1/csv/bitstampUSD.csv.gz and by using the `gzip` program[1] to extract the `.csv` formatted data.[2]

```
. !gzip -vd bitstampUSD.csv.gz
bitstampUSD.csv.gz:      83.5% -- replaced with bitstampUSD.csv
```

We can now import the data, using the `asdouble` option to ensure that the integer timestamp is preserved to its full precision. The first column of the input data is the date and time of the transaction, recorded as a Unix timestamp. Unix timestamps count the number of seconds since midnight, 1 January 1970. To create a Stata clocktime variable with format `%tc`, we must scale the Unix timestamp by 1,000 to express it in milliseconds, and we must set the base date for the Stata variable as 1/1/1970. For this market, we have about 8.67 million transactions recorded as of mid-January 2016.

1. `gzip` is built-in on Linux, Unix, and Mac OS X systems.
2. Stata's `unzipfile` command does not deal with `gzip` files. If your system lacks `gzip`, see http://www.gzip.org.


```
. import delimited bitstampUSD.csv, asdouble
(3 vars, 8,671,372 obs)
. generate double tstamp = (v1 * 1000) + mdyhms(1,1,1970,0,0,0)
. format tstamp %tc
. summarize tstamp, f
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
      tstamp |  8,671,372   03aug2014 18:36:19    2.46e+10  13sep2011 13:53:36
> 19jan2016 11:45:20
. rename (v2 v3) (price volume)
. save "bitstampUSD.dta", replace
file bitstampUSD.dta saved
```

We now must specify the range of dates to be studied and the time interval in which transactions are to be collapsed. To illustrate, we choose 1 January 2014–31 December 2015 as the time period, to be expressed in five-minute intervals. We create the clocktime values for the beginning and ending dates as scalars,[3] and we compute the number of new observations needed to provide the regular intervals. Here we have 210,240 five-minute intervals in these two calendar years.

```
. describe
Contains data from bitstampUSD.dta
  obs:     8,671,372
 vars:             4                          6 Mar 2016 16:00
 size:   242,798,416
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
v1              long    %12.0g
price           double  %10.0g
volume          double  %10.0g
tstamp          double  %tc
-------------------------------------------------------------------------------
Sorted by:
. scalar tobs = r(N)
. scalar firstdate = mdyhms(1,1,2014,0,0,0)
. scalar lastdate  = mdyhms(12,31,2015,23,59,59)
. scalar interval  = 5 * 60 * 1000  // ms in 5 minute interval
. scalar addobs = (lastdate - firstdate) / interval
. display _n "interval observations = " addobs
interval observations = 210240
. scalar newobs = int(tobs + addobs)
```

We now expand the dataset by setting the new number of observations, and we fill in the `tstamp` values for the new observations. The transactions outside the range are dropped. We then sort the data by `tstamp` to intersperse the interval observations among the transactions data.

3. If you are unfamiliar with Stata's scalars, see [P] **scalar**.

```
. set obs `=newobs´
number of observations (_N) was 8,671,372, now 8,881,611
. local t1 = tobs + 1
. local t2 = tobs + 2
. replace tstamp = firstdate + interval in `t1´
(1 real change made)
. replace tstamp = tstamp[_n-1] + interval in `t2´/L
(210,238 real changes made)
. drop if tstamp < firstdate
(2,222,393 observations deleted)
. drop if tstamp > lastdate
(123,400 observations deleted)
. sort tstamp
```

Producing the regularly spaced interval data is now straightforward. We create an indicator variable, `intvl`, equal to one in each interval observation. The `sum()` function produces a running sum in `wintvl`, numbering each interval sequentially. This variable is then used to drive the `collapse` command, which produces a new dataset with one observation per interval. Each observation is identified by the starting date and time of the interval. We can also now `tsset` the data, specifying a `delta` of five minutes. If there are concerns over missing data for intervals in which no transactions occurred, [D] **ipolate** or another interpolation method could be used to fill in the price series.

```
. generate wintvl = sum(missing(price)) + 1
. collapse (min) tstamp price (sum) volume (min) minp = price (max) maxp = price
> (sd) sdprice = price (count) ntrans = price, by(wintvl)
. tsset tstamp, delta(5 minutes)
        time variable:  tstamp, 01jan2014 00:00:00 to 31dec2015 23:55:00
                delta:  5 minutes
. list in 1/10, sep(0) noobs
```

wintvl	tstamp	price	volume	minp	maxp
1	01jan2014 00:00:00	729.01	57.979092	729.01	734

sdprice	ntrans
1.3688354	38

wintvl	tstamp	price	volume	minp	maxp
2	01jan2014 00:05:00	730.33	19.98457	730.33	734.34

sdprice	ntrans
1.5530906	23

wintvl	tstamp	price	volume	minp	maxp
3	01jan2014 00:10:00	733.7	20.442605	733.7	734.5

sdprice	ntrans
.35482983	20

wintvl	tstamp	price	volume	minp	maxp
4	01jan2014 00:15:00	732.19	34.47609	732.19	735
sdprice			ntrans		
1.0303945			31		

wintvl	tstamp	price	volume	minp	maxp
5	01jan2014 00:20:00	731.81	127.53413	731.81	738
sdprice			ntrans		
2.1149646			83		

wintvl	tstamp	price	volume	minp	maxp
6	01jan2014 00:25:00	734.56	5.76307	734.56	738.25
sdprice			ntrans		
1.5949494			12		

wintvl	tstamp	price	volume	minp	maxp
7	01jan2014 00:30:00	734.81	6.3947002	734.81	738.23
sdprice			ntrans		
1.0703673			9		

wintvl	tstamp	price	volume	minp	maxp
8	01jan2014 00:35:00	734.81	17.86617	734.81	738.23
sdprice			ntrans		
1.3222282			22		

wintvl	tstamp	price	volume	minp	maxp
9	01jan2014 00:40:00	735.47	51.953554	735.47	738.28
sdprice			ntrans		
1.2391635			23		

wintvl	tstamp	price	volume	minp	maxp
10	01jan2014 00:45:00	734.47	195.32778	734.47	739
sdprice			ntrans		
1.6481093			52		

```
. save bitstampUSD_interval, replace
file bitstampUSD_interval.dta saved
```

To aggregate transactions to intervals of a different length, we need change only the definition of `interval`. Likewise, different ranges of the transactions history may be specified by redefining `firstdate` and `lastdate`.

The Stata Journal (2016)
16, Number 2, pp. 521–522

Software Updates

gr0031_1: Speaking Stata: Spineplots and their kin. N. J. Cox. *Stata Journal* 8: 105–121.

The handling of x-axis labels has been improved. Several old and new references have been added to the help file.

st0133_2: Fitting mixed logit models by using maximum simulated likelihood. A. R. Hole. *Stata Journal* 7: 593; 7: 388–401.

This update fixes a bug that could affect the estimation results in the case of extreme parameter values during the iterations. Very large parameter values could cause Stata to evaluate exp(·) as missing, which was incorrectly treated as a zero value by the log-likelihood evaluator.

st0276_1: A command to calculate age-standardized rates with efficient interval estimation. D. Consonni, E. Coviello, C. Buzzoni, and C. Mensi. *Stata Journal* 12: 688–701.

Ratios of directly standardized rates (standardized rate ratios) are computed when `by(`*varlist*`)` is specified and modified F intervals of standardized rate ratios are estimated as proposed by Tiwari, Clegg, and Zou (2006). Minor bugs have also been fixed.

st0301_3: Fitting the generalized multinomial logit model in Stata. Y. Gu, A. R. Hole, and S. Knox. *Stata Journal* 14: 701; 13: 884; 13: 382–397.

This update fixes a bug that could affect the estimation results in the case of extreme parameter values during the iterations. Very large parameter values could cause Stata to evaluate exp(·) as missing, which was incorrectly treated as a zero value by the log-likelihood evaluator.

st0389_1: Conducting interrupted time-series analysis for single- and multiple-group comparisons. A. Linden. *Stata Journal* 15: 480–500.

The time variable (`_t`) was modified to start at 0 rather than 1 because the model estimate of the constant would represent the baseline level of the time series as described by Simonton (1977) and Linden and Adams (2011). The version of the software and example data were changed to 11.0 from the original 13.0 to make it accessible to users with prior versions of Stata.

st0403_1: Creating summary tables using the sumtable command. L. J. Scott and C. A. Rogers. *Stata Journal* 15: 775–783.

The following updates have been made to the command, with further details in the help file where appropriate:

- Four new data summary types (*vartype_options*) may now be used: binary and categorical variables may be summarized without displaying group totals by using the `vartype(binary2)` and `vartype(categorical2)` options, respectively; continuous variables can be summarized by medians and ranges by using the `vartype(contrange)` option; and count variables can be summarized by the total number of events and the total number of subjects experiencing the event by using the `vartype(events)` option.
- A missing count summary, which can be used to aid in writing missing counts for table footnotes, has been added to the resultant Excel summary file.
- A bug that caused the `vartype(contmed)` option to not work if the user-dataset contained variables `temp1` and `temp2` has been fixed.

st0427_1: conindex: Estimation of concentration indices. O. O'Donnell, S. O'Neill, T. Van Ourti, and B. Walsh. *Stata Journal* 16: 112–138.

The program can now be used on data from complex survey designs by including the `svy` option if `svyset` has been used to identify the survey design characteristics prior to running the program. Note that `svy` is included as an option here rather than including `svy:` as a prefix.

References

Linden, A., and J. L. Adams. 2011. Applying a propensity score-based weighting model to interrupted time series data: Improving causal inference in programme evaluation. *Journal of Evaluation in Clinical Practice* 17: 1231–1238.

Simonton, D. K. 1977. Cross-sectional time-series experiments: Some suggested statistical analyses. *Psychological Bulletin* 84: 489–502.

Tiwari, R. C., L. X. Clegg, and Z. Zou. 2006. Efficient interval estimation for age-adjusted cancer rates. *Statistical Methods in Medical Research* 15: 547–569.