



^b
**UNIVERSITÄT
BERN**

Faculty of Business, Economics and
Social Sciences

Department of Social Sciences

University of Bern Social Sciences Working Paper No. 14

Creating LaTeX documents from within Stata using texdoc

Ben Jann

This paper is forthcoming in the Stata Journal.

Current version: March 14, 2016
First version: September 2, 2015

<http://ideas.repec.org/p/bss/wpaper/14.html>
<http://econpapers.repec.org/paper/bsswpaper/14.htm>

Creating LaTeX documents from within Stata using texdoc

Ben Jann
Institute of Sociology, University of Bern
ben.jann@soz.unibe.ch
March 14, 2016

Abstract. This paper discusses 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. The command can be used, for example, to assemble automatic reports, write a Stata Journal article, prepare slides for classes, or put together solutions for homework assignments.

Keywords: Stata, `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. Such inclusion of 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 their usage I put together the `texdoc` utility. `texdoc` is based on `sjlatex`, but it automizes most of the relevant tasks. Specifically, `texdoc` allows maintaining 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 (on weaving

1. The `texdoc` command was first released in the SSC Archive in 2009. This article describes a heavily revised and expanded version of the command. The most important addition, in my opinion, is the `nodot` option that allows working on the text without having to rerun the Stata commands. Due to this option and a number of 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 (e.g., output only or commands only), much effort has been put 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).

for Stata see, e.g., Rising 2008). Moreover, the use of `texdoc` is not limited to including facsimiles of Stata output in a LaTeX document. It may be of general use 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 `texdoc` and the required `sjlatex` package on your system, type

```
. ssc install texdoc
. 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 command `texdoc do`. The 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 ]
```

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

`init(docname)` to initialize the LaTeX document by calling the `texdoc init` command. 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 target file, possibly including a path.

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

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

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

`savecmd(newfile [, replace append])` to save a copy of the do-file from which all `/*tex tex*/` blocks (see below) have been removed. Suboption `replace` allows overwriting

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 only be applied to the nested do-file. This is also true for applications of **texdoc init** or **texdoc close** within the nested do-file: After terminating 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 L^AT_EX document in the nested do-file, **texdoc** closes the new L^AT_EX 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 L^AT_EX document will be reopened after termination). An exception is if you change the L^AT_EX document in the nested do-file and specify the **noclose** option. In this case, **texdoc** will continue writing to the new L^AT_EX document.

2.2 Initializing the LaTeX document

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

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

where *docname* is the name of the L^AT_EX target file, possibly including a path. Alternatively, use **texdoc init** without *docname* to change existing settings after the L^AT_EX document has been initialized by **texdoc do** or **texdoc init**. *init_options* are:

replace to allow overwriting an existing L^AT_EX file.

append to append results to an existing L^AT_EX file.

[no]logdir[(*path*)] to specify 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 L^AT_EX document, using the name of the L^AT_EX document as a prefix for the names of the log files; also see the **prefix()** option below. Option **logdir** without argument causes the log files to be stored in a subdirectory with the name of the L^AT_EX document. Option **logdir**(*path*) causes the log files to be stored in subdirectory *path*, where *path* is a relative path starting from the folder of the L^AT_EX document.

[no]prefix[(*prefix*)] to specify a prefix for the automatic names that will be used for the Stata output log files. The names are constructed as “*prefix*#”, where # is a counter (1, 2, 3, etc.). Option **noprefix** omits the prefix; option **prefix** without argument causes “*basename.*” to be used as prefix, where *basename* is the name of the L^AT_EX document (without path); option **prefix**(*prefix*) causes *prefix* to be used as 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)]` to specify 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 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 `stpath(path)` to add a custom path. The default is `nostpath`. Specifying `stpath()` might be necessary if the LaTeX document is itself an input to a master LaTeX file somewhere else in the file system.

`grdir(path)` to specify 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)` to specify 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` to set the default behavior of `texdoc stlog`. See section 2.4 for details.

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 target file 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 at least one 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 at least one 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 target file. Furthermore, to copy the contents of an external file to the LaTeX document, type

```
texdoc append filename
```

where *filename* is the name (and path) of the file to be added. The contents of *filename* will be copied into the L^AT_EX 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 L^AT_EX document containing Stata output. The `stata` L^AT_EX package providing the `stlog` environment is required to display the output (that is, `\usepackage{stata}` should be included in the preamble of the L^AT_EX document). The syntax to include a Stata output log is

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

commands ...

```
texdoc stlog close
```

where `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 generated (see the `prefix()` option in section 2.2 for details). Alternatively, you may type

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

where *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` L^AT_EX package), and

```
texdoc stlog cnp
```

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

`[no]do` to decide whether or not to run the Stata commands. The default is `do`, that is, to 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 re-run the Stata commands. Be aware that the automatic names of Stata output sections change if the order of Stata output sections changes. That is, `nodo` should only be used as long as the order did not change or if a fixed name was assigned to the Stata output section.

- [no] `log` to decide whether the Stata output is to be logged and included in the LaTeX document. The default is `log`, that is, to 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 as 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` to decide whether or not to strip command lines from the Stata output. The default is `nocmdstrip`, that is, to 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` to decide whether or not to strip line break comments from command lines in the Stata output. The default is `no lbstrip`, that is, not to 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` to decide whether or not to remove indentation of commands (that is, whether or not to remove white space on the left of commands) before running the commands and creating the log. This is only relevant in non-interactive 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` to decide whether the Stata output includes commands and their output or only a copy of the commands without output. The default is `nocmdlog`, that is, to 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` to decide whether the command log will be processed by `sjlog` or not. This is only relevant if `cmdlog` has been specified. The default is `noverbatim`, that is, to process the command log by `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 command `\verbatiminput{}`, which requires `\usepackage{verbatim}` in the preamble of the LaTeX document.
- [no] `hardcode` to decide whether the Stata output is physically copied into the LaTeX document or not. The default is `nohardcode`, that is, to 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` to decide whether the external log file will be kept or not. This is only relevant if `hardcode` has been specified. The default is `keep`, that is, to keep the log file so

that `nodo` can be applied later on. Type `nokeep` if you want to erase the external log file.

`[no]_custom` to decide whether or not to use custom code to include the log file in the L^AT_EX document. The default is `nocustom`, that is, to use standard code to include the log. Specify `custom` if you want to skip the standard code and take care of 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 L^AT_EX 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 L^AT_EX document. The syntax of `texdoc graph` is

```
texdoc graph [ , graph_options ]
```

where *graph_options* are:

`as(fileformats)` to set the output format(s). The default is `as(pdf)`. See [G] **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)` to specify the name of the graph window to be exported. The default is to export the topmost graph.

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

`optargs(args)` to pass optional arguments through to the L^AT_EX graph command (as in `\includegraphics[args]{filename}` or `\epsfig{file=filename,args}`).

`[no]_suffix` to specify 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 option `suffix` is specified with multiple output formats, the suffix is determined by the first output format.

`[no]_epsfig` to decide whether or not to use `\epsfig` instead of `\includegraphics` to include the graph in the L^AT_EX document. The default is `noepsfig`, that is, to use `\includegraphics`. Option `epsfig` implies `as(eps)` (unless specified otherwise).

`[no]_center` to decide whether or not to center the graph horizontally in the L^AT_EX document. The default is `center`.

`[no]figure[args]` to decide whether or not 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)` to provide a caption for the figure. `caption()` implies `figure` (unless `nofigure` is specified).

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

`cabove` or `cbelow` to decide whether the caption is printed above or below the figure. Only one of `cabove` and `cbelow` is allowed. `cbelow` is the default.

`[no]custom` to specify 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 take care of including 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]
```

where *filename* is the name of the do-file to be stripped and *newname* is the name of the file to be written to. Option `replace` allows replacing an existing file; option `append` 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` returns the following `s()` macros:

<code>s(docname)</code>	name of L ^A T _E X document (including absolute path)	<code>s(basename)</code>	base name of L ^A T _E X document (excluding path)
<code>s(path)</code>	(absolute) path of L ^A T _E X document	<code>s(logdir)</code>	subdirectory used for Stata log files
<code>s(prefix)</code>	prefix for automatic Stata log names	<code>s(stpath)</code>	include-path to be used for Stata log in L ^A T _E X document
<code>s(grdir)</code>	subdirectory used for graphs (if unequal <code>s(logdir)</code>)	<code>s(gropts)</code>	default graph export options
<code>s(nodo)</code>	nodo or empty	<code>s(nolog)</code>	nolog or empty
<code>s(cmdstrip)</code>	cmdstrip or empty	<code>s(lbstrip)</code>	lbstrip or empty
<code>s(noltrim)</code>	ltrim or empty	<code>s(cmdlog)</code>	cmdlog or empty
<code>s(verbatim)</code>	verbatim or empty	<code>s(hardcode)</code>	hardcode or empty
<code>s(keep)</code>	keep or empty	<code>s(custom)</code>	custom or empty

`texdoc stlog close` and `texdoc stlog using` return the following `s()` macros:

<code>s(name)</code>	name of the Stata output log, including <code>logdir()</code> path	<code>s(name0)</code>	<code>s(name)</code> without <code>logdir()</code> path
<code>s(filename)</code>	name of log file on disk (including path and suffix)	<code>s(filename0)</code>	<code>s(filename)</code> without suffix
<code>s(texname)</code>	name of log file with include-path for use in L ^A T _E X document	<code>s(texname0)</code>	<code>s(texname)</code> without suffix
<code>s(indent)</code>	size of indentation	<code>s(nodo)</code>	nodo or empty
<code>s(nolog)</code>	nolog or empty	<code>s(cmdstrip)</code>	cmdstrip or empty
<code>s(lbstrip)</code>	lbstrip or empty	<code>s(noltrim)</code>	ltrim or empty
<code>s(cmdlog)</code>	cmdlog or empty	<code>s(verbatim)</code>	verbatim or empty
<code>s(hardcode)</code>	hardcode or empty	<code>s(keep)</code>	keep or empty
<code>s(custom)</code>	custom or empty		

3 Examples

3.1 Basic usage

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

```

— 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 milage and weight.

tex*/

texdoc stlog
regress price mpg weight
texdoc stlog close

/*tex

\end{document}
tex*/
— end of file —

```

To process the file, type

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

This will create 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:

```

— 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 milage and weight.

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

\end{document}
— end of file —

```

You can then use your favorite L^AT_EX compiler to generate the final document, which will look about as displayed in figure 1.

3.2 Varieties of log files

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

```

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

```

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 milage and weight.

```
. regress price mpg weight
```

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

the result in the \LaTeX document will be:

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

To only include 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:

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

Conversely, if you only want 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:

```
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 about as follows 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. If, as above, 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. The option allows you to recompile your document without re-running the Stata commands. `texdoc` keeps the log files from previous runs so that re-running 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 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.

Be aware 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 it is always a good idea to assign a fixed name to the output section. For example,

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

would assign 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 L^AT_EX document and only display 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 scatter plot) 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}
```

Option `caption()` has been added to provide a title for the figure, option `label()` 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 L^AT_EX 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 2007), you could proceed as follows:

```
texdoc stlog, nolog
```


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

Standard errors in parentheses
 * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Figure 2: Compiled LaTeX table

```

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 ///
    nontitles booktabs align(D{.}{.}{-1}) ///
    title(Some regression table\label{table1})
}
texdoc write \input{`s(texname0)'.tex}

```

This would include a table such as the one shown in figure 2 in your document. 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 or not).

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 editors can also be set up in a way such 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`). It may also be helpful 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 re-running the Stata commands.

Furthermore, for larger projects it is usually 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 (such as, for example, inline comments and line breaks in commands), there are some limitations and technical issues that should be kept in mind when working with `texdoc`:

- `texdoc` tries to create missing subdirectories using Mata's `mkdir()` function; see [M-5] `chdir()`. Usually, this only works 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 [P] `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 (non-comment) word on the line. For example, do *not* type

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

or similar.

- `texdoc` only provides 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 help [P] `macro`) will only be available until the next `/*tex tex*/` block (or the next `texdoc init` command, or the next `texdoc stlog close` command if option `cmdlog` has been specified, as 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`). A consequence, `/*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`, `cmdlog`, and `ltrim` options of `texdoc stlog` do not work in interactive mode.
- If you apply `texdoc` commands without having initialized 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 help [R] `log`.

5 References

Jann, B. 2007. Making regression tables simplified. *The Stata Journal* 7(2): 227–244.

Rising, B. 2008. Reproducible Research: Weaving with Stata. Italian Stata Users Group Meeting 2008. Available from 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 papers in *Sociological Methodology*, *Sociological Methods & Research*, the *Stata Journal*, *Public Opinion Quarterly*, and the *American Sociological Review*.