UNIVERSITÄT
BERN

Faculty of Business, Economics and Social Sciences

Department of Social Sciences

An earlier version of this paper has been published as:

Jann, Ben (2016). Creating LaTeX documents from within Stata using texdoc. The Stata Journal 16(2): 245-263.

See: http://www.stata-journal.com/article.html?article=pr0062

University of Bern Social Sciences Working Paper No. 14

Creating LaTeX documents from within Stata using texdoc

Ben Jann

Current version: November 27, 2016
First version: September 2, 2015

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

University of Bern
Department of Social Sciences
Fabrikstrasse 8
CH-3012 Bern

Tel. +41 (0)31 631 48 11
Fax +41 (0)31 631 48 17
info@sowi.unibe.ch
www.sowi.unibe.ch

Creating LaTeX documents from within Stata using texdoc

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

November 27, 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 or a Stata Press book, prepare slides for classes, or put together solutions for homework assignments.

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

Contents

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` 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 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. Further examples can also be found at `http://repec.sowi.unibe.ch/stata/texdoc`.

2 Installation

To install the `texdoc` package on your system, type:

```
. ssc install texdoc
```

To be able to compile a LaTeX document containing Stata output you need to copy the Stata LaTeX files to your system and include `\usepackage{stata}` in the preamble of your LaTeX document. To obtain the Stata LaTeX files, first type

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

[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 `nodo` 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).

to install the `sjlatex` package. After that, use command `sjlatex install` to download the Stata LaTeX files; see `help 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).

3 The texdoc command

3.1 Processing a do-file by texdoc do

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* [*arguments*] [`,` *options*]

where *filename* is the name of the do-file to be processed (as usual, include the file name in double quotes if it contains spaces) and *arguments* are optional arguments passed through to the do-file (as local macros `1`, `2`, `3`, and so on; see [R] **do**). *options* are as follows.

[`no`]`init`[`(`*docname*`)`] specifies whether and how to initialize the LaTeX document. If the processed do-file contains a command to initialize the LaTeX document (i.e. if the do-file contains `texdoc init` *docname*; see section 3.2) or if the LaTeX document is already open (e.g. in a nested application of `texdoc do`), the default for `texdoc do` is not to initialize the LaTeX document. Otherwise, `texdoc do` will automatically initialize the LaTeX document in the folder of the do-file using *basename*`.tex` as name for the LaTeX document, where *basename* is the name of the do-file without suffix. Use the `init` option to override these defaults: `noinit` will deactivate automatic initialization; `init` will enforce automatic initialization; `init(`*docname*`)` will enforce initialization using *docname* as name for the LaTeX document (*docname* may include an absolute or relative path; the base folder is the current working directory or the folder of the do-file, depending on whether option `cd` is specified; default suffix "`.tex`" will be added to *docname* if no suffix is specified).

init_options are options to be passed through to `texdoc init`. See section 3.2 for details on available options.

`nostop` allows continuing execution even if an error occurs. Use the `nostop` option if you want to make sure that `texdoc do` runs the do-file all the way to the end even if some of the commands return error. Usage of this option is not recommended. Use the `nostop` option with `texdoc stlog using` if you want to log output from a command that returns error (see section 3.4).

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

`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 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).

3.2 Initializing the LaTeX document

Within a do-file, use `texdoc init` to initialize the LaTeX document (if the do-file does not contain a `texdoc init` command to initialize the LaTeX document, `texdoc do` will automatically call `texdoc init`; see the the `init()` option in section 3.1 above). The syntax of `texdoc init` is

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

where *docname* is the name of the LaTeX target file, possibly including a path. You may also apply `texdoc init` without *docname* in later parts of the do-file to change settings. *init_options* are as follows.

`replace` allows overwriting an existing LaTeX file.

`append` appends results to an existing LaTeX file.

`force` causes `texdoc init` to initialize the LaTeX document even though `texdoc do` is not running. By default `texdoc init` has no effect if typed in Stata's command window or if included in a do-file that is not processed by `texdoc do`. Specify `force` to enforce initialization in these cases. The LaTeX document will remain active until you type `texdoc close`. Note that `texdoc` has only limited functionality if `texdoc do` is not running (for example, `/*tex` `tex*/` blocks and `//` `texdoc exit` will be ignored and some of the options of `texdoc stlog` will not work). Specifying `force` is not recommended.

[`no`]`logall` specifies whether to include the output of all Stata commands in the LaTeX document. The default is `nologall`, that is, to include only the output selected by `texdoc stlog` (see section 3.4). Specify `logall` if you want to log all output. When `logall` is specified, `texdoc do` will insert appropriate `texdoc stlog` and `texdoc stlog close` commands automatically at each `/*tex` `tex*/` or `/***` `***/` block or `texdoc` command (but not at `texdoc stlog oom` and `texdoc stlog cnp`). Empty lines (or lines that only contain white space) at the beginning and end of each command section will be skipped.

stlog_ options are options to set the default behavior of `texdoc stlog`. See section 3.4 for details.

`gropts(`*graph_ options*`)` specifies default options to be passed through to `texdoc graph`. See section 3.5 for details. Updating `gropts()` in repeated calls to `texdoc init` will replace the option as a whole.

[`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. Option `logdir` without argument causes the log files to be stored in a subdirectory with the name of the LaTeX 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 LaTeX document.

`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.

[`no`]`prefix`[`(`*prefix*`)`] specifies a prefix for the automatic names that will be used for the Stata output log files and graphs. 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 LaTeX 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*`_`" (note that reinitializing `logdir` may reset the prefix). The prefix will be ignored if a custom *name* is provided when calling `texdoc stlog` (see section 3.4). 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 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.

3.3 Including LaTeX code

After initializing the LaTeX document, use

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

to include a section of LaTeX code. *text* can contain any text, including multiple lines and paragraphs. 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 `***/`). The provided text will be passed through to the LaTeX target file as is, that is, without expanding Stata macros (although see section 4.10). However, you can use command `texdoc substitute` to define a set of substitutions that will be applied to the text. The syntax of `texdoc substitute` is:

`texdoc substitute` [*from to* [*from to* ...]] [`, add`]

The substitutions defined by `texdoc substitute` will be applied to all subsequent `/*tex tex*/` or `/*** ***/` blocks until a new set of substitutions is defined or until the substitutions are turned off by calling `texdoc substitute` without arguments. To extend an existing set of substitution definitions, specify `texdoc substitute` with the `add` option.

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. `texdoc write` adds a new-line character at the end of the line. If you want to omit the new-line character, you can type `texdoc _write`. Furthermore, to copy the contents of an external file to the LaTeX document, use

`texdoc append` *filename* [`, substitute(`*from to* [*from to* ...]`)`]

where *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. If `substitute()` is specified, all occurrences of *from* will be replaced by *to*. Include *from* and *to* in double quotes if they contain spaces. For example, to replace "`@title`" by "`My Title`" and "`@author`" by "`My Name`", you could type `substitute(@title "My Title" @author "My Name")`.

3.4 Including Stata output

The `texdoc stlog` command creates a section in the LaTeX document containing Stata output. The `stata` LaTeX 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`

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 3.2 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. Furthermore, to include just the output of a single command (without input), you can type

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

(note that `texdoc stlog close` is not needed after the using-form or the colon-form of `texdoc stlog`). *stlog_options* are as follows.

`linesize(`#`)` sets the line width (number of characters) to be used in the output log. # must be an integer between between 40 and 255. The default is to use the current `set linesize` setting; see [R] **log**.

[`no`]`do` specifies whether 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. An exception is if `nodo` is used together with the `cmdlog` option (see below). In this case the log file will always be recreated (as running the commands is not necessary to recreate the log file).

[`no`]`log` specifies 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]cmdlog specifies whether to print a plain copy of the commands instead of using a Stata output log. The default is nocmdlog, that is, to include a Stata output log. If you type cmdlog then only a copy of the commands without output will be included (note that the commands will still be executed; add the nodo option if you want to skip running the commands). cmdlog is similar to nooutput. A difference is that nooutput prints ". " at the beginning of each command whereas cmdlog displays a plain copy of the commands. Furthermore, cmdlog can be combined with nodo to include a copy of the commands without executing the commands. cmdlog is not allowed with the colon-form of texdoc stlog.

[no]verbatim specifies whether the command log will be processed by sjlog. This is only relevant if cmdlog has been specified. The default is noverbatim, that is, to processes 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]output specifies whether to suppress command output in the log. The default is output, that is, to display the output. If nooutput is specified, set output inform is applied before running the commands and, after closing the log, set output proc is applied to turn output back on (see [P] **quietly**). nooutput has no effect if cmdlog is specified. Furthermore, nooutput has no effect if specified with the using-form or the colon-form of texdoc stlog.

[no]matastrip specifies whether to strip Mata opening and ending commands from the Stata output. The default is nomatastrip, that is, to retain the Mata opening and ending commands. If you type matastrip, the "mata" or "mata:" command invoking Mata and the subsequent "end" command exiting Mata will be removed from the log. matastrip only has an effect if the Mata opening command is the first command in the output section.

[no]cmdstrip specifies whether to strip command lines (input) 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 ". " (or ": " in Mata) 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, not to strip the line break comments. Specify lbstrip to delete the line break comments. Specifically, " ///..." at the end of lines staring with ". " or of subsequent lines starting with "> " will be removed.

[no]gtstrip specifies whether to strip continuation symbols from command lines in the Stata output. The default is nogtstrip, that is, not to strip the continuation symbols. Specify gtstrip to delete the continuation symbols. Specifically, "> " at the beginning

of command lines that were broken by a line break comment will be replaced by white space.

[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. 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. `ltrim` has no effect on commands called from an external do-file by `texdoc stlog using`.

`alert(`*strlist*`)` adds the `\alert{}` command to all occurrences of the specified strings, where *strlist* is

string [*string* ...]

Enclose *string* in double quotes if it contains blanks; use compound double quotes if it contains double quotes.

`tag(`*matchlist*`)` applies custom tags to all occurrences of the specified strings, where *matchlist* is

strlist = *begin end* [*strlist* = *begin end* ...]

and *strlist* is

string [*string* ...]

strlist specifies the strings to be tagged, *begin* specifies the start tag, *end* specifies the end tag. Enclose an element in double quotes if it contains blanks; use compound double quotes if the element contains double quotes.

[no]`hardcode` specifies whether the Stata output is physically copied into the LaTeX document. 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.

[no]`keep` specifies whether the external log file will be kept. 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` specifies whether to use custom code to include the log file in the LaTeX 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`.

[no]`certify` specifies whether to compare the current results to the previous version of the log file (if a previous version exists). The default is `nocertify`. Specify `certify` if you want to confirm that the output did not change. In case of a difference, `texdoc` will stop execution and display an error message. `certify` has no effect if `nolog` or `cmdlog` is specified.

`nostop` allows continuing execution even if an error occurs. Use the `nostop` option if you want to log output from a command that returns error. The `nostop` option is only allowed with `texdoc stlog using`.

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` LaTeX package),

`texdoc stlog quietly` *command*

to suppress the output of a command without inserting an "output omitted" message, and

`texdoc stlog cnp`

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

Within or after a Stata output section, you can use the `texdoc local` command to define local macros that will be backed up on disk. This is useful if you want include specific results in your text and want to ensure that the results will be available in later runs when suppressing the Stata commands using the `nodo` option. The syntax of `texdoc local` is

`texdoc local` *name definition*

where possible definitions are as for the Stata's regular `local` command; see [P] **macro**. The locals will be backed up in a library that has the same name as the Stata output section (using file suffix "`.stloc`"). Each output section has its own library, so that the names of the locals can be reused between sections.

The defined locals will be expanded in subsequent `/*tex tex*/` or `/*** ***/` blocks up until the next `texdoc stlog` command. Alternatively, you can write the locals to your document using `webdoc write`. See the example in section 4.10 below.

3.5 Including graphs

`texdoc graph` can be used to export the current graph and include appropriate code in the LaTeX document to display the graph. `texdoc graph` can be specified within a `texdoc stlog` section or directly after `texdoc stlog close`. If `texdoc graph` is specified within a `texdoc stlog` section, the graph is included in the LaTeX document before the Stata output; if `texdoc graph` is specified after `texdoc stlog close`, the graph is included after the Stata output. Furthermore, if `texdoc graph` is used outside a `texdoc stlog` section while `logall` is on, the graph will be placed at the position in the output where the `texdoc graph` command occurs. In general, if `nodo` is on, 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` [*name*] [, *graph_ options*]

where *name* specifies the name to be used for the graph. If *name* is omitted, the name of the `texdoc stlog` section is used to name the graph (possibly suffixed by a counter if the `texdoc stlog` section contains more than one `texdoc graph` command). *graph_ options* are as follows.

`as(`*fileformats*`)` sets 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*`)` specifies the name of the graph window to be exported. The default is to export the topmost graph.

override_ options modifies how the graph is converted. See [G] **graph export** for details.

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

`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 option `suffix` 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`. Option `epsfig` implies `as(eps)` (unless specified otherwise).

[`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 take care of including the graph yourself.

3.6 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.

To cause `texdoc do` exit a do-file, type

```
// texdoc exit
```

(without anything else on the same line). `texdoc do` will only read the do-file up to this line.

3.7 Stripping texdoc commands from a do-file

To clear a 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*/` or `/*** ***/` blocks and all `texdoc` commands from the do-file.

3.8 Stored results

`texdoc init` clears `s()` and `texdoc close` returns the following `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 log in LaTeX document
`s(grdir)`	subdirectory used for graphs (if unequal `s(logdir)`)	`s(gropts)`	default graph export options
`s(logall)`	`logall` or empty	`s(nodo)`	`nodo` or empty
`s(nolog)`	`nolog` or empty	`s(matastrip)`	`matastrip` or empty
`s(cmdstrip)`	`cmdstrip` or empty	`s(lbstrip)`	`lbstrip` or empty
`s(gtstrip)`	`gtstrip` or empty	`s(noltrim)`	`ltrim` or empty
`s(nooutput)`	`nooutput` or empty	`s(cmdlog)`	`cmdlog` or empty
`s(verbatim)`	`verbatim` or empty	`s(hardcode)`	`hardcode` or empty
`s(nokeep)`	`nokeep` or empty	`s(custom)`	`custom` or empty
`s(certify)`	`certify` or empty	`s(linesize)`	specified line width or empty
`s(alert)`	contents of `alert()` option	`s(tag)`	contents of `tag()` option

`texdoc stlog close` returns the following `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(matastrip)`	`matastrip` or empty
`s(cmdstrip)`	`cmdstrip` or empty	`s(lbstrip)`	`lbstrip` or empty
`s(gtstrip)`	`gtstrip` or empty	`s(noltrim)`	`ltrim` or empty
`s(nooutput)`	`nooutput` or empty	`s(cmdlog)`	`cmdlog` or empty
`s(verbatim)`	`verbatim` or empty	`s(hardcode)`	`hardcode` or empty
`s(nokeep)`	`nokeep` or empty	`s(custom)`	`custom` or empty
`s(certify)`	`certify` or empty	`s(linesize)`	line width used for the output log
`s(alert)`	contents of `alert()` option	`s(tag)`	contents of `tag()` option

4 Examples

4.1 Basic usage

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

— *example.texdoc* —

```
texdoc init example.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 example.texdoc
```

This will create file "example.tex" and two log files, "example_1.log.tex" and "example_2.log.tex", in the same directory. The contents of "example.tex" will be:

— example.tex —
```
\documentclass{article}
\usepackage{stata}
\begin{document}

\section*{Exercise 1}
Open the 1978 Automobile Data and summarize the variables.
\begin{stlog}\input{example_1.log.tex}\end{stlog}

\section*{Exercise 2}
Run a regression of price on milage and weight.
\begin{stlog}\input{example_2.log.tex}\end{stlog}

\end{document}
```
— end of file —

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

4.2 The logall option

In the example above, the `texdoc stlog` command was used to select the Stata output to be included in the LaTeX document. If you want to include the output of all Stata commands in the do-file by default, you can specify the `logall` option with `texdoc init` (or with `texdoc do`). Furthermore, note that you can use `/*** ***/` to mark text sections, which is easier to type than `/*tex tex*/`. That is, applying `texdoc do` to the following do-file will produce the exact same result as above:

— example.texdoc —
```
texdoc init example.tex, replace logall

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

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

sysuse auto
summarize
```

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		
Model	186321280	2	93160639.9	Number of obs	= 74
Residual	448744116	71	6320339.67	F(2, 71)	= 14.74
				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 LaTeX document

```
/***

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

regress price mpg weight

/***

\end{document}
***/
```
— end of file —

If the `logall` option has been specified, you can still use the `texdoc stlog` command (for example, with option `nolog` to suppress parts of the output), but it is no longer needed.

4.3 Automatic initialization

If a do-file does not contain a `texdoc init` command to initialize the LaTeX document, `texdoc do` will automatically initialize the document using the name of the do-file. Alternatively, you can use the `init()` option with `texdoc do` to provide a name for the LaTeX document. For example, if your do-file looks as follows

— example.texdoc —
```
/***
\documentclass{article}
\usepackage{stata}
\begin{document}

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

sysuse auto
summarize

/***

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

regress price mpg weight

/***

\end{document}
***/
```
— end of file —

you could type

```
. texdoc do example.texdoc, logall replace
```

to generate `example.tex` (which will, again, contain the same result as above). Alternatively, typing

```
. texdoc do example.texdoc, init(myexample) logall replace
```

will write the results to file `myexample.tex`.

4.4 Changing settings on the fly

You can use the `texdoc init` command to change settings on the fly. For example, you could start off processing a do-file with `logall` on, but then at a later point in the do-file turn `logall` off:

```
texdoc init example.tex, replace logall
commands ...
texdoc init, nologall
commands ...
```

Use `texdoc init` in the same way to change the default behavior of `texdoc stlog` and `texdoc graph` between different sections of the do-file.

4.5 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:

```
. 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:[2]

```
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
```

Some further options for formatting the log files are `matastrip` (to remove the opening `mata` command and the closing `end` command), `lbstrip` (to remove `///` line break comments), and `gtstrip` (to remove `>` continuation symbols in broken command lines).

4.6 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]Note that the commands will still be executed even though no output is recorded. To suppress execution of the commands, add the `nodo` option (see section 4.7 below).

```
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{14}{16}\selectfont
texdoc write \begin{stlog}[auto]\input{`s(texname)'}\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{14}{16}\selectfont
texdoc write \begin{stlog}[auto]
texdoc append `s(filename)'
texdoc write \end{stlog}
texdoc write }
```

4.7 The nodo option

An indispensable option for lager 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.[3] 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.

4.8 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

[3] An exception are `cmdlog` output sections (see section 4.5 above), as the log files of these sections will always be updated irrespective of whether `nodo` is specified or not.

```
\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 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 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}
```

4.9 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
    sysuse auto
```

Table 1: Some 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

```
    regress price weight
    estimates store m1
    regress price weight mpg
    estimates store m2
    regress price weight mpg foreign
    estimates store m3
    esttab m1 m2 m3 using table1.tex, replace se label ///
        nomtitles booktabs align(D{.}{.}{-1}) ///
        title(Some regression table\label{table1})
texdoc stlog close
texdoc write \input{table1.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.

4.10 Dynamic text

If you want to add results from a Stata output section to the text body, an approach is to store the results as local macros and then insert the contents of these locals at appropriate places in the text body using `texdoc write`. A problem, however, is that these locals will no longer be available in later runs once the `nodo` option is applied. A solution to this problem is provided by the `texdoc local` command, which can be applied within or after a Stata output section. The command can be used just like Stata's regular `local` command, but it maintains a backup of the locals on disk and restores them if needed. Furthermore, the local macros defined by `texdoc local` will be expanded in subsequent `/*tex tex*/` and `/***`

```
. sysuse auto, clear
(1978 Automobile Data)
. regress price weight
      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(1, 72)        =     29.42
       Model |  184233937         1  184233937     Prob > F        =    0.0000
    Residual |  450831459        72  6261548.04    R-squared       =    0.2901
-------------+----------------------------------   Adj R-squared   =    0.2802
       Total |  635065396        73  8699525.97    Root MSE        =    2502.3

------------------------------------------------------------------------------
       price |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |   2.044063   .3768341     5.42   0.000     1.292857    2.795268
       _cons |  -6.707353    1174.43    -0.01   0.995     -2347.89    2334.475
------------------------------------------------------------------------------
```

As can be seen in the output above, the estimate for the effect of weight on price is equal to 2.044 (with a standard error of 0.377).

Figure 3: Example with dynamic text

`***/` blocks (up until the next `texdoc stlog` command, which causes the macro library to be reset). An example is as follows (see figure 3 for the compiled result):

```
texdoc stlog
    sysuse auto, clear
    regress price weight
texdoc stlog close
texdoc local b = strofreal(_b[weight], "%9.3f")
texdoc local se = strofreal(_se[weight], "%9.3f")

/***
As can be seen in the output above, the estimate for the effect of weight on
price is equal to `b' (with a standard error of `se').
***/
```

Alternatively, you may use `texdoc write` to write the locals to the output document. That is, you could also type:

```
texdoc write As can be seen in the output above, the estimate for the
texdoc write effect of weight on price is equal to `b´ (with a standard
texdoc write error of `se´).
```

There is a slight difference between the two approaches: expansion in `/*tex tex*/` and `/*** ***/` blocks is based on the locals as stored on disk; `texdoc write` uses the current values of the locals.

5 Workflow and limitations

Do-files containing `texdoc` commands 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 curser 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. Also note that you can run the do-file without applying `texdoc do`, that is, by submitting the commands to Stata's command window or by applying the regular `do` or `run` command (see [R] **do**). The `texdoc` commands (except `texdoc do`) and `/*tex tex*/` blocks will be ignored in this case. This is useful if you only want to run the Stata commands without updating the LaTeX document.

For larger projects it is usually helpful to break up the project into several 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`.

- `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 between `texdoc` commands. However, do not use semicolons to delimit `texdoc` commands.

- `texdoc stlog` cannot be nested. Furthermore, do not use `texdoc do` or `texdoc init` 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**). As a consequence, `/*tex tex*/` blocks in such a file will be ignored and some `texdoc` options will not work. Use `texdoc do` to include nested do-files.

- 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 \$
  ```

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.