

^b
**UNIVERSITÄT
BERN**

Faculty of Business, Economics and
Social Sciences

Department of Social Sciences

University of Bern Social Sciences Working Paper No. 22

Creating HTML or Markdown documents from within Stata using webdoc

Ben Jann

This paper is forthcoming in the Stata Journal.

Current version: January 17, 2017

First version: July 27, 2016

<http://ideas.repec.org/p/bss/wpaper/22.html>

<http://econpapers.repec.org/paper/bsswpaper/22.htm>

Creating HTML or Markdown documents from within Stata using webdoc

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

January 17, 2017

Abstract

This paper discusses the use of `webdoc` for creating HTML or Markdown documents from within Stata. The `webdoc` command provides a way to embed HTML or Markdown 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 create a webpage documenting your data analysis, including all Stata output and graphs. More generally, the command can be used to create and maintain a website that contains results computed by Stata.

Keywords: Stata, `webdoc`, HTML, Markdown, weaving, Stata output, Stata log, reproducible research

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 3 |
| 2 | The webdoc command | 4 |
| 2.1 | Processing a do-file by webdoc do | 4 |
| 2.2 | Initializing the output document | 5 |
| 2.3 | Including HTML or Markdown code | 8 |
| 2.4 | Adding a table of contents | 9 |
| 2.5 | Including Stata output | 10 |
| 2.6 | Including graphs | 14 |
| 2.7 | Changing the HTML settings for Stata output and graphs | 16 |
| 2.8 | Closing the output document and exiting the do-file | 17 |
| 2.9 | Stripping webdoc commands from a do-file | 18 |
| 2.10 | Stored results | 18 |
| 3 | Examples | 19 |
| 3.1 | Basic usage | 19 |
| 3.2 | Using Markdown | 21 |
| 3.3 | Changing the look of the HTML file | 21 |
| 3.4 | Contents of output sections | 24 |
| 3.5 | Generating do-files from output sections | 27 |
| 3.6 | The nodo option | 28 |
| 3.7 | Graphs | 29 |
| 3.8 | Tables | 31 |
| 3.9 | Table of contents | 31 |
| 3.10 | Dynamic text | 33 |
| 4 | Limitations | 35 |

1 Introduction

`webdoc` is a command to process a do-file that contains Stata commands as well as pieces of HTML or Markdown code. A primary use of `webdoc` is to produce a HTML document that displays literal Stata output as it is shown in Stata's Results window. However, `webdoc` can be seen as a general tool for generating HTML documents that combine text sections and results from statistical analysis.

Several other user commands are available to support the production of HTML documents in Stata. For example, you can

- translate Stata output or SMCL files to HTML format using commands such as `log2html` by Baum et al. (2001) or `hlp2html` by Jeanty (2010),
- create HTML documents from within Stata with tools such as the `ht` package by Quintó et al. (2012) or `htmlutil` by Newson (2015),
- export tables or matrixes into a HTML file with commands such as `listtex` by Newson (2001), `matprint` by Bruun (2016b), `tabout` by Watson (2004), or `esttab` by Jann (2007),
- and weave Stata commands and pieces of HTML or Markdown code in a single do-file using tools such as `log2markup` by Bruun (2016a), `weaver` and `markdoc` by Haghish (2014b,a), or `weave` by Germán Rodríguez (see <http://data.princeton.edu/wws509/stata/weave>).

The `webdoc` command covers much of the functionality of these packages. Like `log2html` it transforms Stata output to HTML format (relying, in part, on the undocumented `loghtml` command); like the `ht` package or the `htmlutil` command it allows working on a HTML file from within Stata; like `log2markup` or `markdoc` it allows integrating HTML or Markdown code in a do-file. Furthermore, although `webdoc` does not provide specific tools for producing tables, results from special-purpose programs such as `listtex` can easily be integrated. A major difference to other weaving programs such as `log2markup` or `markdoc` is that `webdoc` pre-processes the do-file. This provides some advantages such as being able to update the HTML output document without having to rerun all Stata commands. It also means, however, that `webdoc` cannot be used interactively.

Below I will discuss the features of `webdoc` and provide examples of its usage (for further examples also see <http://repec.sowi.unibe.ch/stata/webdoc/>). `webdoc` has a similar architecture, functionality, and user interface as `texdoc`, a command for producing L^AT_EX documents (Jann, 2016). If you are familiar with `texdoc` you will find `webdoc` easy to use. Of course, however, knowledge of HTML and CSS will be beneficial. A good source for detailed information on HTML and CSS is <http://www.w3schools.com/>. For information on Markdown consult <http://daringfireball.net/projects/markdown/>.

To install `webdoc`, type `ssc install webdoc` in Stata.

2 The webdoc command

2.1 Processing a do-file by webdoc do

The basic procedure is to write a do-file including Stata commands and sections of HTML code and then process the do-file by command **webdoc do**. The command will create the HTML source file, which can then be viewed in a browser. It is also possible to use Markdown code instead of HTML. In this case, the source document has to be processed by a Markdown converter before being viewed in the browser. The syntax of **webdoc do** is

```
webdoc 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 output document. If the processed do-file contains an initialization command (that is, if the do-file contains **webdoc init** *docname*; see section 2.2) or if the output document is already open (i.e. in a nested application of **webdoc do**), the default for **webdoc do** is not to initialize the output document. Otherwise, **webdoc do** will automatically initialize the output document in the folder of the do-file using *basename.html* (or, if option **md** is specified, *basename.md*) as name for the 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 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).

init_options are options to specify defaults to be passed through to **webdoc init**. See section 2.2 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 **webdoc 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 **webdoc stlog using** if you want to log output from a command that returns error (see section 2.5).

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

webdoc do can be nested. That is, **webdoc do** can be applied in a do-file that is processed by **webdoc do**. Options specified with a nested call to **webdoc do** will only be applied to the nested do-file. This is also true for applications of **webdoc init** or **webdoc close** within the nested do-file: After terminating a nested do-file all preexisting **webdoc** settings will be

restored. For example, if you use the `init()` option or `webdoc init` to change the output document in the nested do-file, `webdoc` closes the new output document and switches back to the previous one when exiting the nested do-file (similarly, if you use `webdoc close` in the nested do-file, the document will be reopened after termination).

2.2 Initializing the output document

Within a do-file, use `webdoc init` to initialize the HTML or Markdown output document (alternatively, if the do-file does not contain an initialization command, `webdoc do` will automatically call `webdoc init`; see the `init()` option in section 2.1). The syntax of `webdoc init` is

```
webdoc init [docname] [, init_options]
```

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

`replace` allows overwriting an existing output document.

`append` appends results to an existing output document.

`md` specifies that `.md` instead of `.html` is to be used as default suffix for the output document.

`header[(header_opts)]` causes a HTML header (and a footer) to be added to the output document. *header_opts* are as follows.

`width(width)` sets the maximum width of the HTML page, where *width* is a width specification in CSS units (see http://www.w3schools.com/cssref/css_units.asp), such as 800px or 50em. If you use the `bstheme()` option, an alternative approach is to include the body of your page in a container. For example, type `<div class="container-fluid" style="max-width:800px">` on the first line and `</div>` on the last line.

`nofooter` omits the footer. This is useful if you want to append more material to the same document later on.

`title(str)` provides a title for the meta data of the page. The default is to use the name of the document as title.

`author(str)`, `date(str)`, `description(str)`, and `keywords(str)` provide author information, a date, a description, and a (comma separated) list of keywords to be included in the meta data of the page.

`language(str)` specifies the language of the document, where *str* is a HTML language specification (see <https://www.w3.org/International/articles/language-tags/>). The default is `language(en)`.

`charset(str)` specifies the character encoding of the document, where *str* is a HTML charset specification (see http://www.w3schools.com/html/html_charset.asp). The default depends on the Stata version. If you use Stata 13 or older, the default is `charset(iso-8859-1)` (Windows, Unix) or `charset(mac)` (MacOSX). If you use Stata 14 or newer, the default is `charset(utf-8)`.

`bstheme[(spec)]` includes a Bootstrap CSS file in the header (see <http://getbootstrap.com/>). *spec* is

`[theme][, jscript selfcontained]`

where *theme* is either equal to `default` (for the default Bootstrap CSS) or equal to the name (in lowercase letters) of a Bootswatch theme (such as `cerulean`, `cosmo`, `simplex`, `united`, etc.; see <http://bootswatch.com/> or <https://www.bootstrapcdn.com/bootswatch/> for the list of available themes). If *theme* is omitted, the default Bootstrap CSS is used. In addition to the Bootstrap CSS, `webdoc` will append a few additional CSS definitions to slightly modify the display of images and code. Furthermore, if you use the `bstheme()` option, you should consider specifying a maximum page width using the `width()` option or including the body of your page in a container, e.g. typing `<div class="container-fluid" style="max-width:800px">` on the first line and `</div>` on the last line. In general, for more information on Bootstrap, see <http://getbootstrap.com/>.

By default, `webdoc` does not load Bootstrap's JavaScript plugins. Specify suboption `jscript` if you want to use Bootstrap elements that require JavaScript. `webdoc` will then add code at the end of the document to load the relevant plugins (also see <http://getbootstrap.com/getting-started/#template>).

Unless suboption `selfcontained` is specified, `webdoc` includes the Bootstrap CSS and JavaScript plugins using links pointing to the minified files at <https://www.bootstrapcdn.com/>. Specify `selfcontained` to copy the (non-minified versions of the) files into your document (this will increase the file size of your document by about 150 KB or, if `jscript` is specified, by about 500 KB). For larger projects it may make sense to provide a copy of the CSS and JavaScript files at your website and include them in your HTML pages using local links.

If the `bstheme` option is omitted, a minimum set of CSS definitions resulting in a plain look will be included in the header of the document.

`include(filename)` adds the contents of *filename* to the HTML header. The contents of *filename* will be included within the `<head>` tag after the definitions requested by the `bstheme()` option.

`stscheme(stscheme_options)` specifies the look of the Stata output sections. This has only an effect on sections containing Stata output, not on sections containing Stata code. That is, sections created by the `cmdlog` option (see below) will not be affected

by `stscheme()`. Note that, currently, `webdoc` does not tag errors and links in the Stata logs, so that these elements will appear as regular output. *stscheme_options* are as follows.

`standard`, `studio`, `classic`, `desert`, `mountain`, `ocean`, or `simple` select one of Stata's built-in color schemes (see the preferences dialog of Stata's Results window; you can right-click on the Results window to open the dialog).

`bg(color)`, `fg(color)`, `rfg(color)`, `cfc(color)`, `rbf`, and `cbf` affect the appearance of the different elements in the Stata output, where *color* is a CSS color specification (see <http://www.w3schools.com/colors/default.asp>). These options override the corresponding settings from the built-in schemes. `bg()` specifies the background color, `fg()` the default foreground color (i.e. the color of standard output), `rfg()` the color of results (typically the numbers in the output), and `cfc()` the color of input (the commands). Furthermore, use `rbf` and `cbf` to request bold font for results and input/commands, respectively.

`lcom` italicizes and shades comments in the Stata output.

`[no]logall` specifies whether to include the output of all Stata commands in the output document. The default is `nologall`, that is, to include only the output selected by `webdoc stlog` (see section 2.5). Specify `logall` if you want to log all output. When `logall` is specified, `webdoc do` will insert appropriate `webdoc stlog` and `webdoc stlog close` commands automatically at each `/** */` block and at each `webdoc` command (but not at `webdoc stlog oom` and `webdoc 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 `webdoc stlog`. See section 2.5 for details.

`gropts(graph_options)` specifies default options to be passed through to `webdoc graph`. See section 2.6 for details. Updating `gropts()` in repeated calls to `webdoc 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 output document, using the name of the output 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 same name as the output 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 output document.

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

`dodir(path)` specifies an alternative subdirectory to be used by `webdoc stlog` for storing the do-files requested by the `dosave` option (see below), where *path* is a relative path

starting from the folder of the output document. The default is to store the do-files 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 output 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 `webdoc stlog` (see section 2.5). The suffix of the physical log files on disk is always “.log”.

`[no]stpath[(path)]` specifies how the path for linking files in the output document is to be constructed (`stpath()` has no effect on where the log files and graphs are stored in the file system). If `stpath` is specified without argument, then the path of the output document (to be precise, the path specified in *docname* when initializing the output document) is added to the include-path. Alternatively, specify `stpath(path)` to add a custom path. The default is `nostpath`.

2.3 Including HTML or Markdown code

After initializing the output document, use

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

to include a section of HTML or Markdown code. *text* can contain any text, including multiple lines and paragraphs. The opening tag of a HTML or Markdown section, `/***`, 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, `***`, 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. The provided text will be passed through to the output document as is, that is, without expanding Stata macros (although see section 3.10). However, you can use command `webdoc substitute` to define a set of substitutions that will be applied to the text. The syntax of `webdoc substitute` is:

```
webdoc substitute [from to [from to ...]][, add ]
```

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

A single line of HTML or Markdown code can also be written to the document using

`webdoc write textline`

or

`webdoc put textline`

Stata macros in *textline* will be expanded before writing the line to the output document. The difference between `webdoc write` and `webdoc put` is that `webdoc put` includes a new-line character at the end of the line, whereas `webdoc write` omits the new-line character so that more text can be added to the same line. Furthermore, to copy the contents of an external file to the output document, type

```
webdoc append filename [, substitute(from to [from to ...]) drop(numlist) ]
```

where *filename* is the name (and path) of the file to be added. The contents of *filename* will be copied into the output document as is, at the position where `webdoc 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")`. Option `drop()` causes the specified lines to be omitted when copying the file.

2.4 Adding a table of contents

An automatic table of contents from the headings in the document can be generated by `webdoc toc`. The syntax of `webdoc toc` is

```
webdoc toc [levels [offset]] [, toc_options ]
```

`webdoc toc` collects the HTML headings found in subsequent `/** */` blocks and constructs a corresponding table of contents (using `` lists). The table of contents will be inserted into the output document at the position where `webdoc toc` appears. The *levels* argument specifies the desired number of levels to be considered. For example `webdoc toc 3` will create a table of contents with three levels from `<h1>` to `<h3>`. Furthermore, use the *offset* argument to shift the highest level to be taken into account. For example, `webdoc toc 3 1` will use `<h2>`, `<h3>`, and `<h4>`; `webdoc toc 2 4` will use `<h5>` and `<h6>`. *offset* must be an integer between 0 and 5; the default is 0. *levels* must be an integer between 1 and 6 – *offset*; the default is 3. *toc_options* are as follows.

`numbered` causes section numbers be added to the headings and the entries in table of contents. The numbers added to the headings will be tagged by ``; the numbers in the table of contents will be tagged by ``.

`md` specifies that Markdown headings are to be taken into account. By default, only HTML headings, that is, lines starting with `<h1>` to `<h6>`, are collected. If `md` is specified, lines

starting with # to ##### are also treated as headings. In any case, a heading will only be detected if it starts at the beginning of the line (save white space in case of HTML tags). To construct an entry in the table of contents, only the text that follows on the same line will be taken into account.

2.5 Including Stata output

If the `logall` option is specified with `webdoc do` or `webdoc init`, output from all Stata commands will automatically be added to the HTML document. Alternatively, select the output to be included using the `webdoc stlog` command. The syntax of `webdoc stlog` is

```
webdoc stlog [name] [, stlog_options]
```

commands ...

```
webdoc stlog close
```

where `webdoc stlog` opens the log, *commands* are the Stata commands to be logged, and `webdoc 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

```
webdoc 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

```
webdoc stlog [name] [, stlog_options]: command
```

(note that `webdoc stlog close` is not needed after the using-form or the colon-form of `webdoc 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 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 output 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 **webdoc 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. Furthermore, **nolog** can be used to deselect output if the **logall** option has been specified.

[no] **cmdlog** specifies whether to print a plain copy of the Stata code 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** (see below). 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. Tag `<pre class="stcmd"><code>` will be used to start a **cmdlog** section in the output document. Other Stata output sections will be started by `<pre class="stlog"><samp>`. **cmdlog** is not allowed with the colon-form of **webdoc stlog**.

[no] **dosave** specifies whether to store a copy of the commands in an external do-file. The default is **nodosave**, that is, not to store a do-file. The name of the Stata output section is used as name for the do-file (with suffix “.do”). The do-files will be stored in the same location as the log files, unless an alternative location is specified using the **dodir()** option. All **webdoc** commands will be stripped from the do-file.

[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 **webdoc 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 `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.

[no] `gtstrip` specifies whether to strip continuation symbols from command lines in the Stata output. The default is `no``gtstrip`, 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. `gtstrip` has no effect if `cmdlog` is specified.

[no] `ltrim` specifies whether to remove indentation of commands (that is, whether to remove white space on the left of commands) before running the commands and creating the log. 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 `webdoc stlog using`.

`mark(strlist)` adds the `<mark>` tag 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] `plain` specifies whether to omit markup in the log file. The default is `no``plain`, that is, to annotate the log file with HTML tags. In particular, input (commands) will be tagged using ``, results will be tagged using ``, and comments will be tagged using `` (if `cmdlog` is specified, only comments will be tagged). Specify `plain` to omit the HTML tags.

[no] `raw` specifies whether to omit markup in the log file and retain special characters. The default is `no``raw`, that is, to annotate the log file with HTML tags (see the `plain` option above) and to replace characters `<`, `>`, and `&` by their HTML equivalents `<`, `>`, and `&`. Specify `raw` to omit the HTML tags and retain the special characters.

- [no]**custom** specifies whether to use custom code to include the log file in the output 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.
- [no]**keep** specifies whether the external log file will be kept. 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]**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, **webdoc** will stop execution and display an error message. **certify** has no effect if **nolog** or **cmdlog** is specified or if a help file is processed (see the **sthlp** option below).
- [no]**sthlp**[(*subst*)] specifies whether to treat the provided file as a Stata help file. This is only allowed with **webdoc stlog using**. By default, files with a **.hlp** or **.sthlp** suffix are treated as help files; all other files are treated as do-files. Type **nsthlp** or **sthlp** to override these defaults. Files treated as help files are translated by undocumented **log webhtml** (or, if **plain** or **raw** is specified, by **translate** with the **smcl2log** translator) and are not submitted to Stata for execution. Unless **plain** or **raw** is specified, text markup and help links are preserved. Internal help links (i.e. links pointing to the processed help file) will be converted to appropriate internal links in the output document; other help links will be converted to links pointing to the corresponding help file at <http://www.stata.com/>. In addition, you may provide custom substitutions in **sthlp**(*subst*), where *subst* is a space separated list (*from to [from to ...]*). The custom substitutions will be applied before converting the internal links and the stata.com links (unless **plain** or **raw** is specified, in which case no substitutions will be applied). The help links written by **log webhtml** are constructed as ``. Hence, you could, for example, type **sthlp(/help.cgi?mycommand mycommand.html)** convert the help links for **mycommand** to links pointing to the local page **mycommand.html**.

Options **nolog**, **cmdlog**, and **dosave** are not allowed in help-file mode. Furthermore, contents options such as **nooutput**, **cmdstrip**, or **matastrip** will have no effect. However, you may use **nodo** to prevent re-processing the help file or **custom** to use custom inclusion code. By default, the included help file will be wrapped by a `<pre class="sthlp">` tag.

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 **webdoc stlog using**.

Furthermore, among the commands to be logged, you may use

```
webdoc stlog oom command
```

to suppress the output of a specific command and display an output-omitted message instead,

```
webdoc stlog quietly command
```

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

```
webdoc stlog cnp
```

to insert a page break (page breaks are ignored in screen display of a HTML page, but they affect the print version of the page). The output-omitted message is produced by

```
<span class="stoom">(output omitted)</span>
```

and the page break is produced by

```
<span class="stcnp" style="page-break-after:always"><br/>(continued on next  
page)<br/></span>
```

The class attribute is set so that you can use stylesheets to affect the appearance of these messages. For example, including

```
<style type="text/css">  
  .stoom, .stcnp { font-style: italic; }  
  @media screen { .stcnp { display: none; } }  
</style>
```

in the header of the output document will use italic font for the messages and suppress the continued-on-next-page message in screen display (where page-breaks have no effect).

Within or after a Stata output section, you can use the `webdoc 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 `webdoc local` is

```
webdoc 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 `/** */` blocks up until the next `webdoc stlog` command. Alternatively, you can write the locals to your document using `webdoc put` or `webdoc write`. See the example in section 3.10 below.

2.6 Including graphs

`webdoc graph` exports the current graph and include appropriate code in the output document to display the graph. `webdoc graph` can be specified within a `webdoc stlog` section or directly after `webdoc stlog close`. If `webdoc graph` is specified within a `webdoc stlog` section, the graph is included in the output document before the Stata output; if `webdoc graph` is specified after `webdoc stlog close`, the graph is included after the Stata output. Furthermore, if `webdoc graph` is used outside a `webdoc stlog` section while `logall` is on,

the graph will be placed at the position in the output where the `webdoc graph` command occurs. In general, if `nodo` is on, no graph will be exported and only the include-code will be written to the output document. The syntax of `webdoc graph` is

```
webdoc graph [name] [, graph_options]
```

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

`as(fileformats)` sets the output format(s). The default is `as(png)`. See [G] **graph export** for available formats. A further, currently undocumented format available since Stata 14 is `svg` (Scalable Vector Graphics). Multiple formats may be specified as in, for example, `as(png pdf)`, in which case `webdoc graph` will create multiple graph files. The first format will be used for the image in the output document.

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

`width(#)` specifies the physical width of the graph (in pixels). The default is `width(500)` unless `height()` is specified. If `height()` is specified, the appropriate width is determined from the graph's aspect ratio. `width()` only has an effect if the output format is PNG or TIFF.

`height(#)` specifies the physical height of the graph (in pixels). The default height is determined from the graph's aspect ratio. `height()` only has an effect if the output format is PNG or TIFF.

`override_options` modifies how the graph is converted. See [G] **graph export** for details.

`alt(string)` provides an alternative text for the image to be added to the `` tag using the "alt" attribute. The default is to use the name of the graph as alternative text. The `alt()` option has no effect if embedding an SVG using the `hardcode` option.

`title(string)` provides a "tooltip" title for the image to be added to the `` tag using the "title" attribute.

`attributes(args)` provides further attribute definitions to be added to the `` tag. For example, to set the display width of the graph to 50%, type `attributes(width="50%")`.

`[no]link[(fileformat)]` specifies whether to add a link to the image pointing to the graph file. Clicking the image in the browser will then open the graph file. The default is `link`, that is, to add a link, unless `hardcode` is specified (see below), in which case `nolink` is the default. Argument *fileformat* may be used to select the file for the link if multiple output formats have been requested by the `at()` option. For example, specifying `link(pdf)` together with `as(svg pdf)` will display the SVG image and use the PDF for the link. The default is to use the first format for both the image and the link.

`[no]figure[id]` specifies whether to enclose the image in a `<figure>` environment. The default is `figure`, that is, to use the figure tag. Type `nofigure` to omit the figure tag. To add a custom ID to the figure tag, type `figure(id)`. If `id` is omitted, `webdoc` will add an automatic ID (constructed as `fig-name`, where `name` is the base name of the graph).

`caption(string)` provides a caption for the figure using the `<figcaption>` tag.

`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.

`[no]hardcode` specifies whether to embed the graph source in the output document. This is only supported for PNG and SVG. In case of PNG, the graph file will be embedded using Base64 encoding. In case of SVG, the SVG code will be copied into the output document. The default is `nohardcode`, that is, to include the graph using a link to the external graph file.

`[no]keep` specifies whether the external graph file (and its Base64 variant) will be kept. This is only relevant if `hardcode` has been specified. The default is `keep`, that is, to keep the graph files so that `nodo` can be applied later on. Type `nokeep` if you want to erase the external graph files.

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

2.7 Changing the HTML settings for Stata output and graphs

Parts of the HTML code written by `webdoc` can be customized by the `webdoc set` command. The syntax of `webdoc set` is

```
webdoc set [setname [definition]]
```

where `setname` is the name of the element you want to change. To restore the default settings for all elements, type `webdoc set` without argument. `webdoc set` only has an effect if applied within a do-file processed by `webdoc do`. Furthermore, all settings will be removed when `webdoc do` terminates. The elements you can modify, and their default definitions, are as in table 1.

Names without underscore refer to opening tags (or opening and closing tags), names with underscore refer to closing tags. As illustrated by the default settings, some of the elements make use of local macros, with a leading backslash for delayed expansion. An interesting additional macro that can be used in `stlog/_stlog` and `stcmd/_stcmd` is ‘`doname`’, containing the name of the do-file that is generated if the `dosave` option has been specified. For example, to provide a download link for the do-file in the upper right corner of each output section, you could type:

Table 1: HTML settings that can be changed by `webdoc set`

| Description | <i>setname</i> | Default definition |
|-----------------------|---|--|
| Stata output section | <code>stlog</code> <code>_stlog</code> | <code><pre id="\`id'" class="stlog"><samp></code> <code></samp></pre></code> |
| Stata code section | <code>stcmd</code> <code>_stcmd</code> | <code><pre id="\`id'" class="stcmd"><code></code> <code></code></pre></code> |
| Stata help section | <code>sthlp</code> <code>_sthlp</code> | <code><pre id="\`id'" class="sthlp"></code> <code></pre></code> |
| Stata input tag | <code>stinp</code> <code>_stinp</code> | <code></code> <code></code> |
| Stata result tag | <code>stres</code> <code>_stres</code> | <code></code> <code></code> |
| Stata comment tag | <code>stcmt</code> <code>_stcmt</code> | <code></code> <code></code> |
| Output-omitted tag | <code>stoom</code> | <code>(output omitted)</code> |
| Cont-on-next-page tag | <code>stcnp</code> | <code><span class="stcnp"</code> <code>style="page-break-after:always"> (continued</code> <code>on next page) </code> |
| Figure tag | <code>figure</code> <code>_figure</code> | <code><figure id="\`macval(id)'"></code> <code></figure></code> |
| Figure caption | <code>fcap</code> | <code><figcaption>\`macval(caption)'\</figcaption></code> |
| Figure link tag | <code>flink</code> <code>_flink</code> | <code></code> <code></code> |
| Image tag | <code>img</code> <code>_img</code> | <code><img alt="\`macval(alt)'" "\`macval(title)'" src="</code> <code>"\`macval(attributes)'" /></code> |
| Embedded SVG | <code>svg</code> <code>_svg</code> | <code></code> <code></code> |

```
. webdoc set stlog <pre id="\`id'" class="stlog" /*
    */style="position:relative;"><a href="\`doname'" /*
    */style="position:absolute;top:5px;right:5px">[code]</a><samp>
```

SVG images embedded in the output document using the `hardcode` option will be tagged by `svg/_svg`. For all other graphs, `img/_img` will be used.

2.8 Closing the output document and exiting the do-file

The syntax to stop writing to the output document is

```
webdoc close
```

`webdoc do` closes the output document automatically at the end of the do-file, so that `webdoc close` is usually not needed.

To cause `webdoc` to exit a do-file, type

```
// webdoc exit
```

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

2.9 Stripping webdoc commands from a do-file

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

```
webdoc 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. `webdoc strip` removes all `/** */` blocks and all `webdoc` commands from the do-file.

2.10 Stored results

`webdoc init` clears `s()` and `webdoc close` returns the following `s()` macros:

| | | | |
|--------------------------|--|---------------------------|--|
| <code>s(docname)</code> | name of output document (including absolute path) | <code>s(basename)</code> | base name of output document (excluding path) |
| <code>s(path)</code> | (absolute) path of output document | <code>s(md)</code> | md or empty |
| <code>s(logall)</code> | logall or empty | <code>s(linesize)</code> | specified line width or empty |
| <code>s(nodo)</code> | nodo or empty | <code>s(nolog)</code> | nolog or empty |
| <code>s(cmdlog)</code> | cmdlog or empty | <code>s(dosave)</code> | dosave or empty |
| <code>s(plain)</code> | plain or empty | <code>s(raw)</code> | raw or empty |
| <code>s(nooutput)</code> | nooutput or empty | <code>s(matastrip)</code> | matastrip or empty |
| <code>s(cmdstrip)</code> | cmdstrip or empty | <code>s(lbstrip)</code> | lbstrip or empty |
| <code>s(gtstrip)</code> | gtstrip or empty | <code>s(noltrim)</code> | ltrim or empty |
| <code>s(mark)</code> | contents of <code>mark()</code> option | <code>s(tag)</code> | contents of <code>tag()</code> option |
| <code>s(custom)</code> | custom or empty | <code>s(nokeep)</code> | nokeep or empty |
| <code>s(certify)</code> | certify or empty | <code>s(gropts)</code> | default graph export options |
| <code>s(logdir)</code> | subdirectory used for Stata log files | <code>s(grdir)</code> | subdirectory used for graphs (if unequal <code>s(logdir)</code>) |
| <code>s(dodir)</code> | subdirectory used for do-files (if unequal <code>s(logdir)</code>) | <code>s(prefix)</code> | prefix for automatic names |
| <code>s(stpath)</code> | include-path to be used in the output document | | |

`webdoc stlog close` and `webdoc 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(webname)</code> | name of log file with include-path for use in output document | <code>s(webname0)</code> | <code>s(webname)</code> without suffix |
| <code>s(id)</code> | id of the log in the output document | <code>s(doname)</code> | name (and include-path) of do-file |
| <code>s(linesize)</code> | line width used for the output log | <code>s(indent)</code> | size of indentation |
| <code>s(nodo)</code> | <code>nodo</code> or empty | <code>s(nolog)</code> | <code>nolog</code> or empty |
| <code>s(cmdlog)</code> | <code>cmdlog</code> or empty | <code>s(dosave)</code> | <code>dosave</code> or empty |
| <code>s(plain)</code> | <code>plain</code> or empty | <code>s(raw)</code> | <code>raw</code> or empty |
| <code>s(nooutput)</code> | <code>nooutput</code> or empty | <code>s(matastrip)</code> | <code>matastrip</code> or empty |
| <code>s(cmdstrip)</code> | <code>cmdstrip</code> or empty | <code>s(lbstrip)</code> | <code>lbstrip</code> or empty |
| <code>s(gtstrip)</code> | <code>gtstrip</code> or empty | <code>s(noltrim)</code> | <code>ltrim</code> or empty |
| <code>s(mark)</code> | contents of <code>mark()</code> option | <code>s(tag)</code> | contents of <code>tag()</code> option |
| <code>s(custom)</code> | <code>custom</code> or empty | <code>s(nokeep)</code> | <code>nokeep</code> or empty |
| <code>s(certify)</code> | <code>certify</code> or empty | | |

3 Examples

3.1 Basic usage

A simple do-file using `webdoc` might look as follows:

```

— example.do —
webdoc init example1, replace logall plain

/***
<html>
<head><title>Example 1</title></head>
<body>
<h2>Exercise 1</h2>
<p>Open the 1978 Automobile Data and run a regression of price on
milage using the <code>regress</code> command.</p>
***</

sysuse auto
regress price mpg

/***
</body>
</html>
***</
— end of file —

```

Option `logall` has been specified so that all Stata output is included in the HTML document. (In addition, option `plain` has been specified to omit HTML tags from the Stata output, so that the display of the HTML file below fits the page.) To process the file, type

```
. webdoc do example1.do
```

This will create file “example1.html” with the following contents:

```

— example1.html —
<html>
<head><title>Example 1</title></head>
<body>
<h2>Exercise 1</h2>
<p>Open the 1978 Automobile Data and run a regression of price on
milage using the <code>regress</code> command.</p>
<pre id="stlog-1" class="stlog"><samp>. sysuse auto
(1978 Automobile Data)

. regress price mpg

      Source |      SS          df           MS       Number of obs   =        74
-----+-----
      Model | 139449474          1    139449474   F(1, 72)          =       20.26
    Residual | 495615923         72    6883554.48   Prob > F          =       0.0000
-----+-----
      Total | 635065396         73    8699525.97   R-squared         =       0.2196
                                           Adj R-squared     =       0.2087
                                           Root MSE         =       2623.7

-----+-----
      price |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-----+-----
      mpg | -238.8943   53.07669    -4.50   0.000   -344.7008   -133.0879
    _cons | 11253.06   1170.813     9.61   0.000   8919.088   13587.03
-----+-----
</samp></pre>
</body>
</html>
— end of file —

```

Displaying the file in a browser would look about as shown in figure 1.

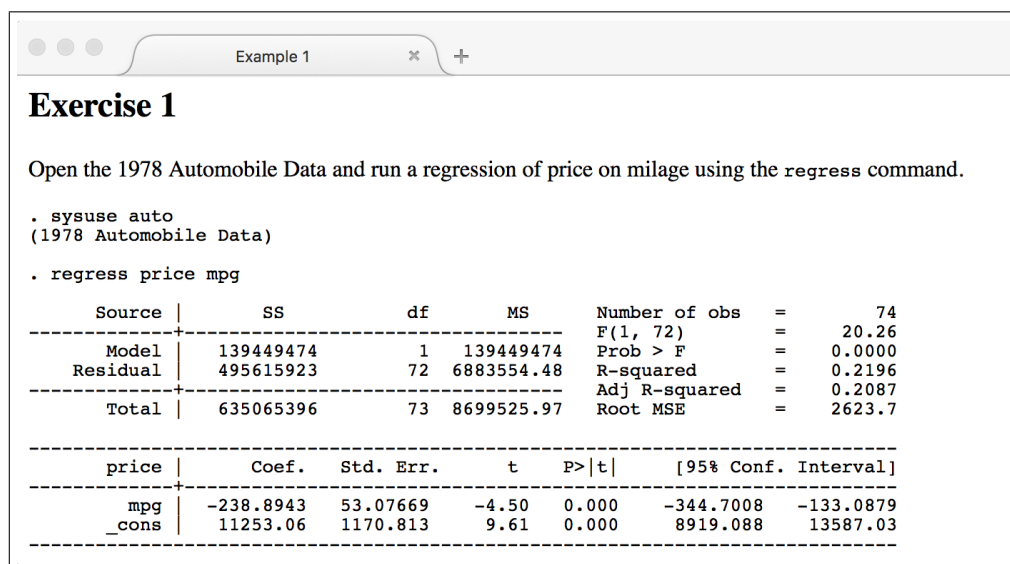


Figure 1: Output file “example1.html” displayed in a browser

3.2 Using Markdown

For simplified typing, you could also omit the HTML tags and use Markdown instead. An example do-file might look as follows:

```
— example1-md.do —
webdoc init example1, replace logall plain md

/***
## Exercise 1

Open the 1978 Automobile Data and run a regression of price on milage
using the `regress` command.

***/

sysuse auto
regress price mpg
— end of file —
```

Typing

```
. webdoc do example1-md.do
```

will create file “example1.md”, which can then be converted to HTML using a Markdown converter. For example, if Pandoc is installed on your system (see <http://pandoc.org/>), you could type

```
. shell pandoc example1.md -s -o example1.html
```

to create a HTML file from “example1.md”. The `-s` option has been specified so that Pandoc produces a standalone HTML file including a header and footer. The resulting file will look about the same as the file show in figure 1.

3.3 Changing the look of the HTML file

Use stylesheet specifications in the header of the HTML file to change the look of the document in the browser. For example, using the following header definition would create a file that displays about as shown in figure 2.

```
— example2.do —
webdoc init example2, replace logall plain

/***
<!DOCTYPE html>
<html>
<head>
  <title>Example 2</title>
  <style>
    body {
      font-family: sans-serif; padding: 0 15px; max-width: 700px;
    }
  </style>
</head>
<body>
  Exercise 1
</body>
</html>
***/
```

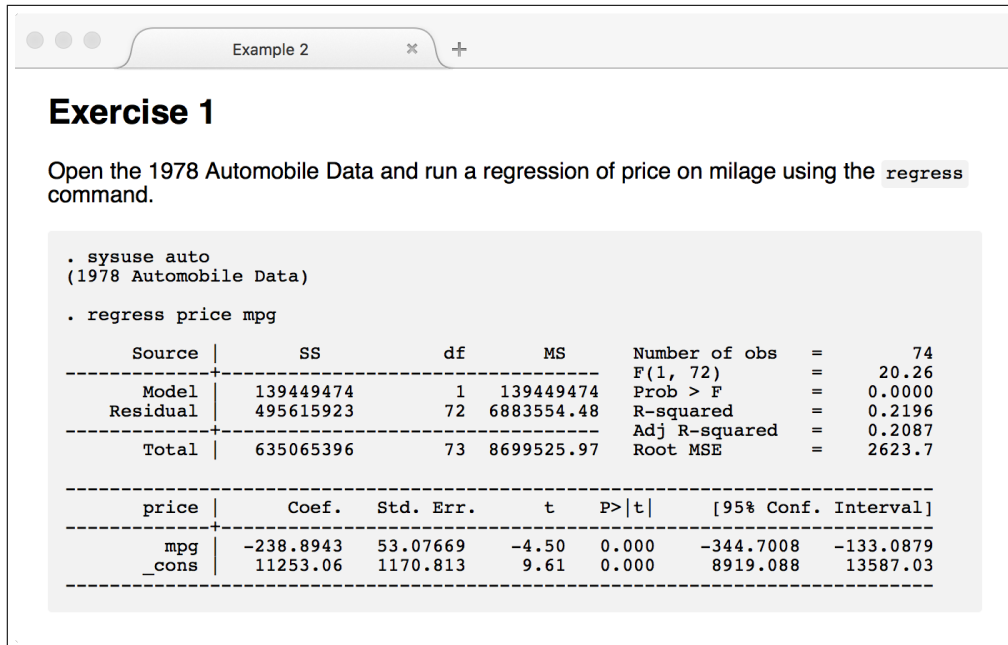
```

code {
    background-color: #f2f2f2; border-radius: 3px; padding: 3px;
}
pre {
    background-color: #f2f2f2;
    border-radius: 3px; padding: 12px;
}
pre code {
    background: transparent; padding: 0;
}
</style>
</head>
<body>
<h2>Exercise 1</h2>
<p>Open the 1978 Automobile Data and run a regression of price on
milage using the <code>regress</code> command.</p>
***</

sysuse auto
regress price mpg

/****
</body>
</html>
****</
— end of file —

```



Exercise 1

Open the 1978 Automobile Data and run a regression of price on milage using the `regress` command.

```

. sysuse auto
(1978 Automobile Data)

. regress price mpg

```

| Source | SS | df | MS | Number of obs | = | 74 |
|----------|-----------|----|------------|---------------|---|--------|
| Model | 139449474 | 1 | 139449474 | F(1, 72) | = | 20.26 |
| Residual | 495615923 | 72 | 6883554.48 | Prob > F | = | 0.0000 |
| | | | | R-squared | = | 0.2196 |
| | | | | Adj R-squared | = | 0.2087 |
| Total | 635065396 | 73 | 8699525.97 | Root MSE | = | 2623.7 |

| price | Coef. | Std. Err. | t | P> t | [95% Conf. Interval] |
|-------|-----------|-----------|-------|-------|----------------------|
| mpg | -238.8943 | 53.07669 | -4.50 | 0.000 | -344.7008 -133.0879 |
| _cons | 11253.06 | 1170.813 | 9.61 | 0.000 | 8919.088 13587.03 |

Figure 2: Output file “example2.html” displayed in a browser

If you do not want to put together your own header (and footer), you can use the `header` option of `webdoc init` to generate an automatic header, as in the following example.

```

— example3.do —
webdoc init example3, replace logall ///
    header(title(Example 3) width(700px) stscheme(classic))

/****
<h2>Exercise 1</h2>
<p>Open the 1978 Automobile Data and run a regression of price on
milage using the <code>regress</code> command.</p>
****/

sysuse auto
regress price mpg
— end of file —

```

In the example, `title()` specifies the text for the `<title>` tag in the document header, `width()` sets the maximum page width, and `stscheme(classic)` selects the “Classic” color scheme for the Stata output (see figure 3).

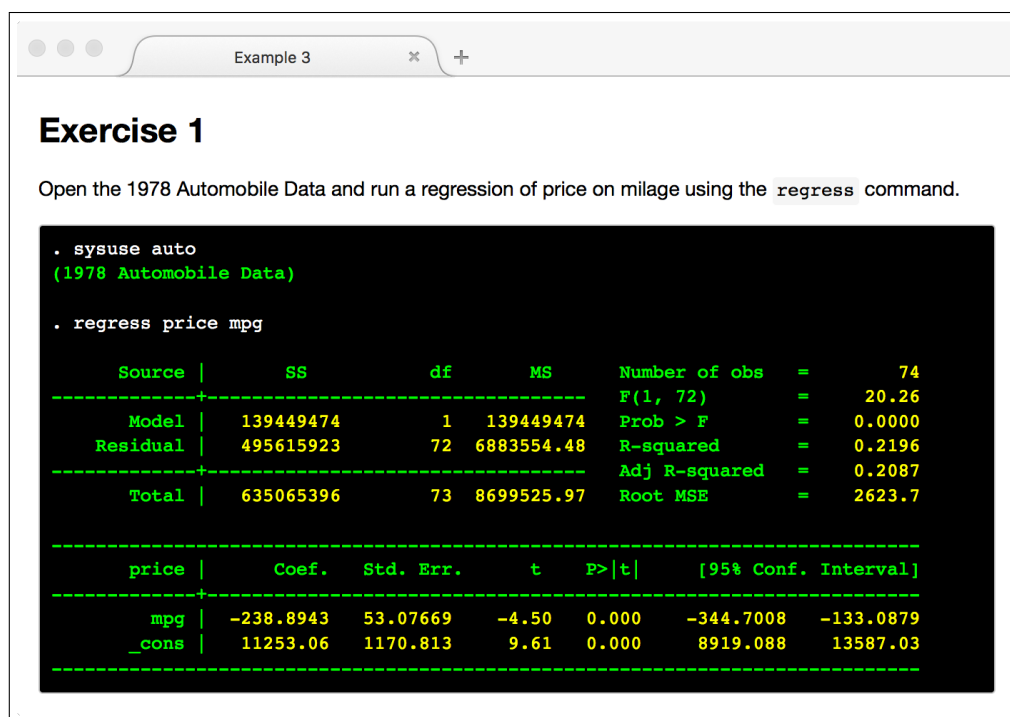


Figure 3: Output file “example3.html” displayed in a browser

By default, if the `header()` option is specified, `webdoc` writes a minimal header so that the page displays well on computer screens and mobile devices. Alternatively, you can use the `bstheme()` suboption to include a Bootstrap CSS file (see <http://getbootstrap.com/>). For example, the following code includes the “United” theme from <http://bootswatch.com/> and picks Stata’s “Desert” scheme for the output (see figure 4 for the result).

```

— example4.do —
webdoc init example4, replace logall ///

```

```

header(title(Example 4) width(700px) stscheme(desert) bstHEME(united))

/****
<h2>Exercise 1</h2>
<p>Open the 1978 Automobile Data and run a regression of price on
milage using the <code>regress</code> command.</p>
****/

sysuse auto
regress price mpg
— end of file —

```

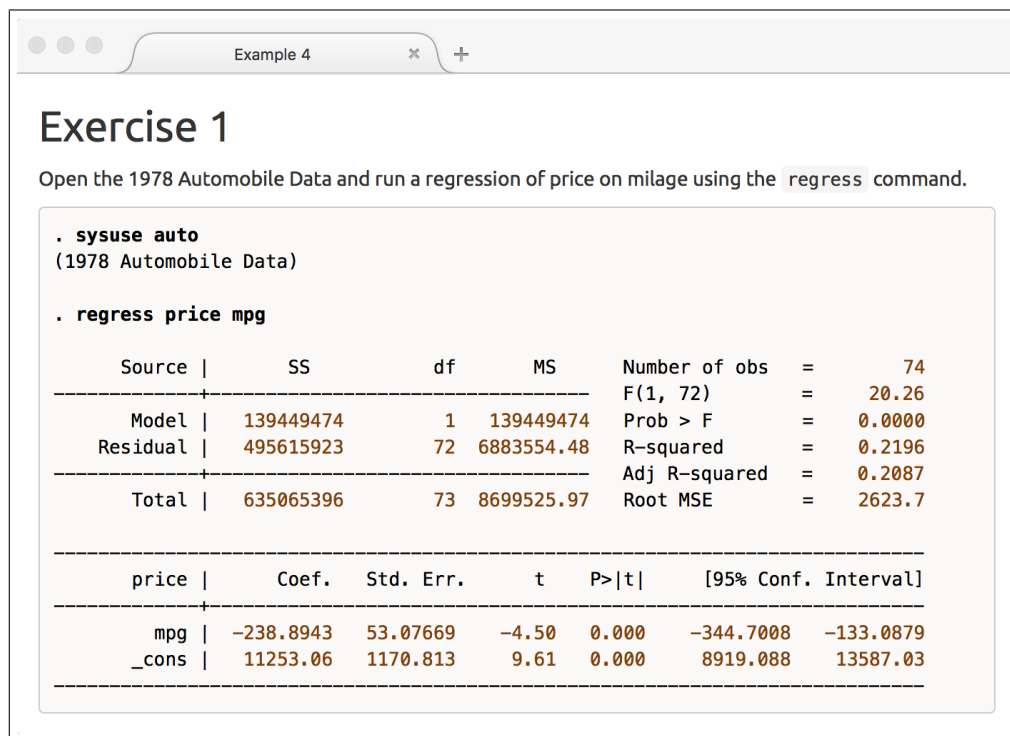


Figure 4: Output file “example4.html” displayed in a browser

3.4 Contents of output sections

In the examples above, the `logall` option was specified to create output sections from all Stata commands in the do-file. Alternatively, or in addition, you can use the `webdoc stlog` command to select the output to be included. For example, if the `logall` option has been specified, you could type

```

webdoc stlog, nolog
commands
webdoc stlog close

```

to omit creating an output section from *commands*. Furthermore, the `webdoc stlog` command is useful if you want to apply different options to specific output sections. The following example illustrates some of the available options (see figure 5 for the result). Note that all options can also be specified with `webdoc do` or `webdoc init` to set the default behavior. Furthermore, you can apply `webdoc init` repeatedly within a do-file (without specifying an output document) to change the defaults between different parts of the do-file.

```
— example5.do —
webdoc init example5, replace logall ///
    header(title(Example 5) width(700px) stscheme(studio) bstHEME)

/****
<h4>Options of webdoc stlog</h4>
<ul><li><p>Default: input (commands) and output</p>
****/

webdoc stlog
display as txt "sqrt(2) = " /// this is a comment
    as res sqrt(2)
webdoc stlog close

/****
</li><li><p><code>cmdstrip</code>: output without input</p>
****/

webdoc stlog, cmdstrip
display as txt "sqrt(2) = " /// this is a comment
    as res sqrt(2)
webdoc stlog close

/****
</li><li><p><code>nooutput</code>: input without output</p>
****/

webdoc stlog, nooutput
display as txt "sqrt(2) = " /// this is a comment
    as res sqrt(2)
webdoc stlog close

/****
</li><li><p><code>lbstrip</code> and <code>gtstrip</code>: remove line-break
comments and continuation symbols</p>
****/

webdoc stlog, lbstrip gtstrip
display as txt "sqrt(2) = " /// this is a comment
    as res sqrt(2)
webdoc stlog close

/****
</li><li><p><code>cmdlog</code>: display code instead of results</p>
****/

webdoc stlog, cmdlog
display as txt "sqrt(2) = " /// this is a comment
```

```

        as res sqrt(2)
webdoc stlog close

/***
</li><li><p><code>matastrip</code>: remove Mata begin and end commands</p>
***</li>

webdoc stlog, matastrip
mata:
sqrt(2)
end
webdoc stlog close

/***
</li></ul>
***</li>
— end of file —

```



Figure 5: Output file “example5.html” displayed in a browser

Note that `webdoc stlog` distinguishes between Stata output and Stata code. By default, `webdoc stlog` displays Stata output, tagged by `<pre class="stlog"><samp>`. However, if the `cmdlog` option is specified, `webdoc stlog` displays Stata code, tagged by `<pre class="stcmd"><code>`. The color scheme chosen in `header(stscheme())` only applies to sections of Stata output, not to code. Code is displayed using standard settings, with shaded comments.¹

3.5 Generating do-files from output sections

`webdoc stlog` has a `dosave` option that stores a do-file from the commands in the logged output section. This is useful if you want to provide the commands in a downloadable file. Here is a somewhat advanced example in which a “Code” button (with an arrow icon from <http://glyphicons.com/>) is placed in the upper right corner of the Stata output box (see figure 6):

```
— example6.do —
webdoc init example6, replace header(title(Example 6) width(700px) bstheme)

/**
<h2>Exercise 1</h2>
<p>Open the 1978 Automobile Data and run a regression of price on
milage using the <code>regress</code> command.</p>
***/

webdoc put <div style="position:relative">
webdoc stlog, dosave
    sysuse auto
    regress price mpg
webdoc stlog close
webdoc put /*
    */<a href="`s(doname)'" class="btn btn-default btn-sm"/*
    */ style="position:absolute; top:10px; right:10px">/*
    */<span class="glyphicon glyphicon-arrow-down" aria-hidden="true"></span>/*
    */ Code</a>
webdoc put </div>
— end of file —
```

If the user clicks the “Code” button, a file containing the Stata commands opens. The `webdoc put` command is used here to write the necessary code to generate the button (an alternative would be to use `webdoc set`; see section 2.7). Note that the Stata output box is included in a `<div style="position:relative">` tag so that the button can be positioned relative to the box. For information on the code generating the button, see <http://getbootstrap.com/css/#buttons>; for the code to display the arrow icon see <http://getbootstrap.com/components/#glyphicons>.

¹To omit the shading of comments in code display, you can specify option `plain` with `webdoc stlog`. To apply comment shading in output display, add option `lcom` in `header(stscheme())`.

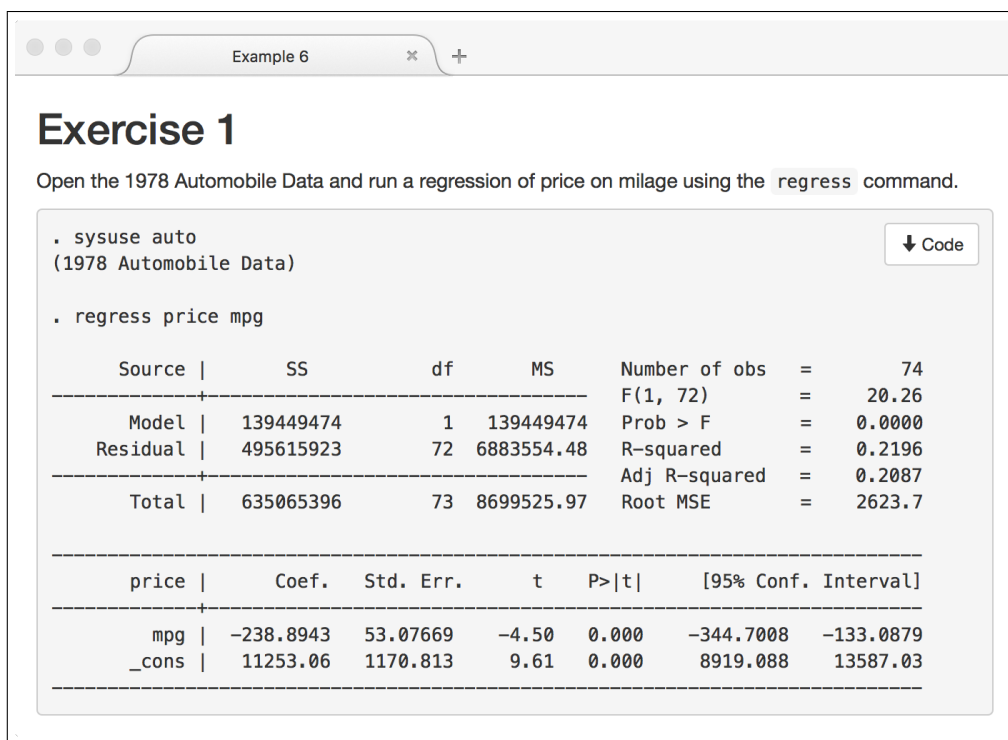


Figure 6: Output file “example6.html” displayed in a browser

`webdoc stlog close` returns the name (and relative path) of the do-file in macro `s(doname)`, from where it can be provided to `webdoc put`. By default, the do-file is placed in the same folder as the output document. Specify `dodir(path)` with `webdoc do` or `webdoc init` to request a different location. Furthermore, if you want the do-file to have a specific name, specify a name with `webdoc stlog`. For example, type

```
. webdoc stlog exercise1, dosave
```

to use name “`exercise1.do`” for the do-file (the suffix will always be “`.do`”).

3.6 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. `webdoc` 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

```
webdoc stlog, nodo
```

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

```
webdoc stlog, do
```

Note that you can also turn commands on and off between different parts of the document by applying the `webdoc init` command with the `do` or `nodo` option repeatedly within the do-file.

Be aware that `webdoc` 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.² Hence, if a specific Stata output section contains time consuming commands it is always a good idea to assign a fixed name (i.e. type `webdoc stlog name`).

3.7 Graphs

To include a graph in the output document, simply type `webdoc graph` after the graph has been created. `webdoc graph` will store the graph on disk and place an appropriate `` in the output document to display the graph. By default, a PNG image with a width of 500 pixels is produced. There are various options to change how the graph is exported and how it is integrated into the output document. The following example sets the physical width of the graph to 1000 pixels, sets the display width to 100%, provides a caption for the graph, and also sets a tooltip title.

```
— example7.do —
webdoc init example7, replace logall header(title(Example 7) width(700px))

/***
<h2>Exercise 1</h2>
<p>Open the 1978 Automobile Data and draw a scatter plot of price against
milage using the <code>twoway</code> command and include a linear fit.</p>
***//

sysuse auto
twoway (scatter price mpg) (lfit price mpg)
webdoc graph, caption(Figure 1: Twoway plot of price by milage) cabove ///
      width(1000) title(price by mpg) attributes(width="100%")
— end of file —
```

Figure 7 displays the resulting file as it looks in a browser. If the user moves the pointer to the graph, a tooltip containing “price by mpg” will be shown. Furthermore, if the user clicks the graph, the graph file will be opened. Note that `webdoc graph` automatically creates a name for the graph (based on the name of the relevant Stata output section). If you want your graph to have a specific name, you can type `webdoc graph name`.

If, as in the example above, the `logall` option is specified, `webdoc` will stop the Stata output section at the position of the `webdoc graph` command, insert the graph, and then continue with a new output section. If you want to display a graph that has been produced

²An exception are `cmdlog` output sections (see section 3.4 above), as the log files of these sections will always be updated irrespective of whether `nodo` is specified or not.

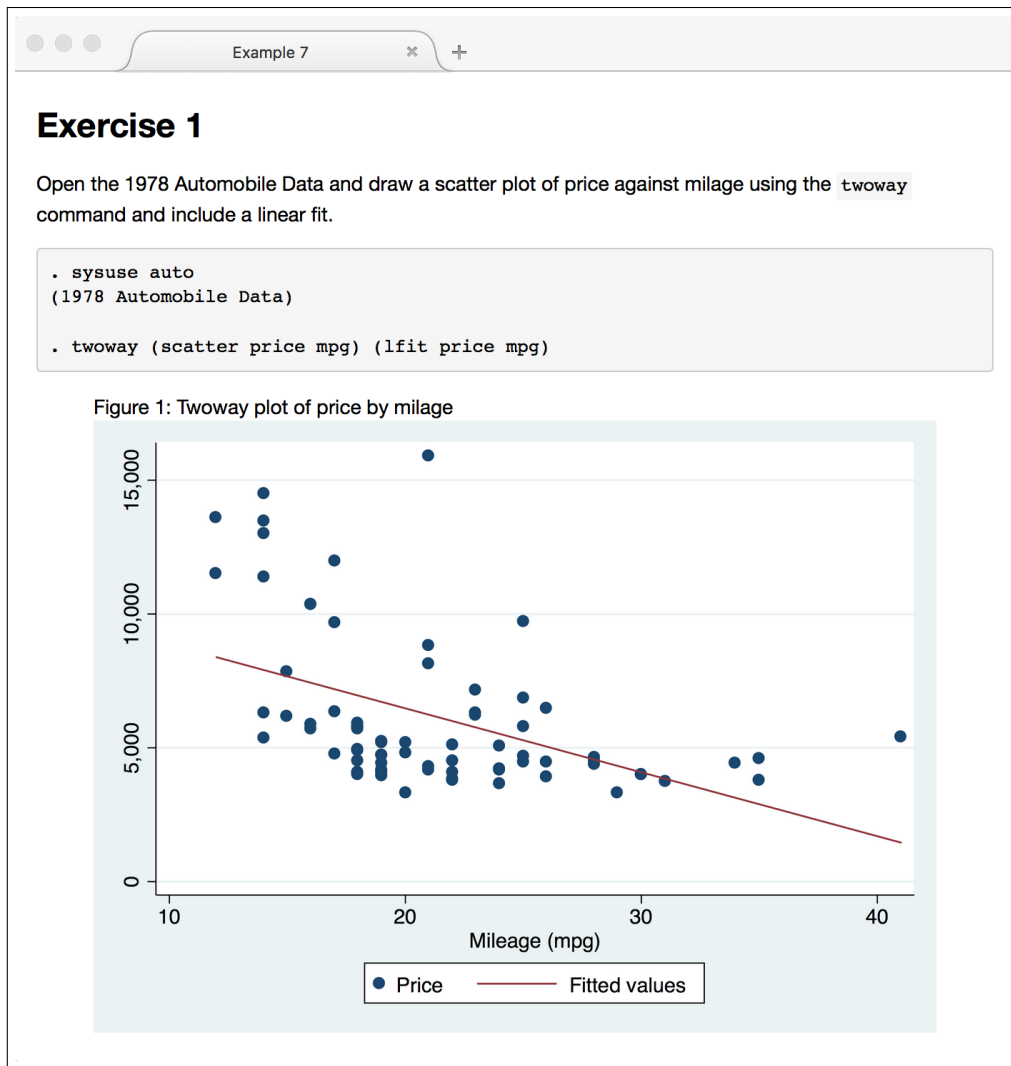


Figure 7: Output file “example7.html” displayed in a browser

within an explicit `webdoc stlog` section, it is usually better to call `webdoc graph` after the section has been closed. That is, type:

```
webdoc stlog
  sysuse auto
  twoway (scatter price mpg) (lfit price mpg)
webdoc stlog close
webdoc graph
```

Typing `webdoc graph` within a `webdoc stlog` section is allowed, but it will cause the graph to be included in the HTML document before the output box.

The default for `webdoc graph` is to provide the image source in form of a link to the external graph file. However, you can also specify the `hardcode` option to directly embed

the image in the HTML document (this only works if the requested graph format is PNG or SVG). The `hardcode` option is useful if you want to share your HTML file without having to copy around multiple files. Another use might be if you want to embed a low resolution PNG in the HTML document and, at the same time, provide a link to an external high resolution graph file. This could be achieved by typing

```
webdoc graph, hardcode width(200) link
webdoc graph, custom width(1000)
```

The first `webdoc graph` command embeds a low-resolution graph (200 pixels wide) in the HTML document and also includes a link to the external graph file. The second `webdoc graph` command overwrites the external graph file with a high-resolution variant (1000 pixels wide), but does not include any code in the HTML document (due to option `custom`). If the user clicks the image in the browser, the high-resolution graph will be opened.

3.8 Tables

`webdoc` does not provide specific tools for producing tables. However, you can use other programs such as `listtex` by Newson (2001) or `esttab` by Jann (2007) to write a table in HTML format and then add the result to your HTML document using `webdoc append`. Below is an example based on `esttab` (see figure 8 for the result). The procedure for `listtex` or other commands would be similar.

```
— example8.do —
webdoc init example8, replace header(title(Example 8) width(700px))

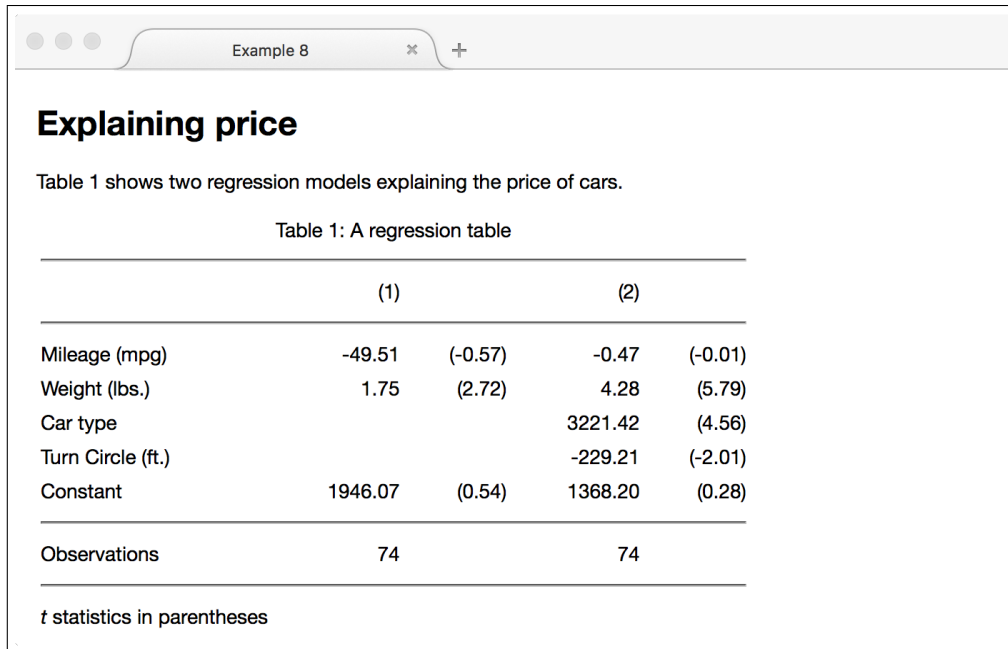
/****
<h2>Explaining price</h2>

<p>Table 1 shows two regression models explaining the price of cars.</p>
****/

webdoc stlog, nolog
    sysuse auto
    regress price mpg weight
    estimates store m1
    regress price mpg weight foreign turn
    estimates store m2
    esttab m1 m2 using example8_tab1.html, replace label wide nomtitle ///
        nostar b(2) align(right) width(500) title(Table 1: A regression table)
webdoc stlog close
webdoc append example8_tab1.html
— end of file —
```

3.9 Table of contents

To generate a (clickable) table of contents (TOC) from the headings in your HTML document, you can use the `webdoc toc` command. Simply include the `webdoc toc` command at



Explaining price

Table 1 shows two regression models explaining the price of cars.

Table 1: A regression table

| | (1) | | (2) | |
|-------------------|---------|---------|---------|---------|
| Mileage (mpg) | -49.51 | (-0.57) | -0.47 | (-0.01) |
| Weight (lbs.) | 1.75 | (2.72) | 4.28 | (5.79) |
| Car type | | | 3221.42 | (4.56) |
| Turn Circle (ft.) | | | -229.21 | (-2.01) |
| Constant | 1946.07 | (0.54) | 1368.20 | (0.28) |
| Observations | 74 | | 74 | |

t statistics in parentheses

Figure 8: Output file “example8.html” displayed in a browser

the position in the file where you want the TOC to appear. All relevant headings from this position on will be collected to construct the TOC. By default, `webdoc toc` collects three levels of headings, from `<h1>` to `<h3>`. To collect, say, four levels from `<h2>` to `<h5>`, you could type `webdoc toc 4 1`. The first number specifies the number of desired levels, the second specifies the offset (i.e. how many upper levels to skip). To add automatic section numbers to the headings and the entries in the TOC you can specify the `numbered` option. The numbers will be tagged (as class `toc-secnum` in the TOC and as class `heading-secnum` in the headings), so they can be styled by CSS. Likewise, use CSS definitions for the `` tag to affect the look of the TOC. To prevent the definitions from being applied to other instances of `` in the document, it is a good idea to wrap the TOC in an own class or include it in a `<nav>` tag and make the definitions conditional on that. A somewhat advanced example is as follows (for the result see figure 9).

```
— example9.do —
webdoc init example9, replace header(title(Example 9) width(700px) bstheme)

/**
<style>
.toc ul { padding-left:0; list-style:none; font-weight:bold; }
.toc ul ul { font-weight:normal; }
.toc-secnum, .heading-secnum { float:left; min-width:45px; }
</style>
***/

/**
<h1>The Title</h1>
<p>Some leading text.</p>
```

```

<h4>Contents</h4>
<div class="toc">
***/

webdoc toc 3 1, numbered

/***
</div>

<h2>A first section</h2>
<p>Some text.</p>
***/

/***
<h2>A second section</h2>
<p>Some text.</p>

<h3>A first subsection to the second section</h3>
<p>Some text.</p>

<h3>A second subsection to the second section</h3>
<p>Some text.</p>

<h4>A first subsection to the second subsection of the second section</h4>
<p>Some text.</p>
***/

/***
<h2>A final section</h2>
<p>Some text.</p>
***/
— end of file —

```

3.10 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 `webdoc put` or `webdoc 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 `webdoc 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 `webdoc local` will be expanded in subsequent `/*** ***/` blocks (up until the next `webdoc stlog` command, which causes the macro library to be reset). An example is as follows (see figure 10 for the compiled result):

```

— example10.do —
webdoc init example10, replace header(title(Example 10) width(700px))

webdoc stlog

```

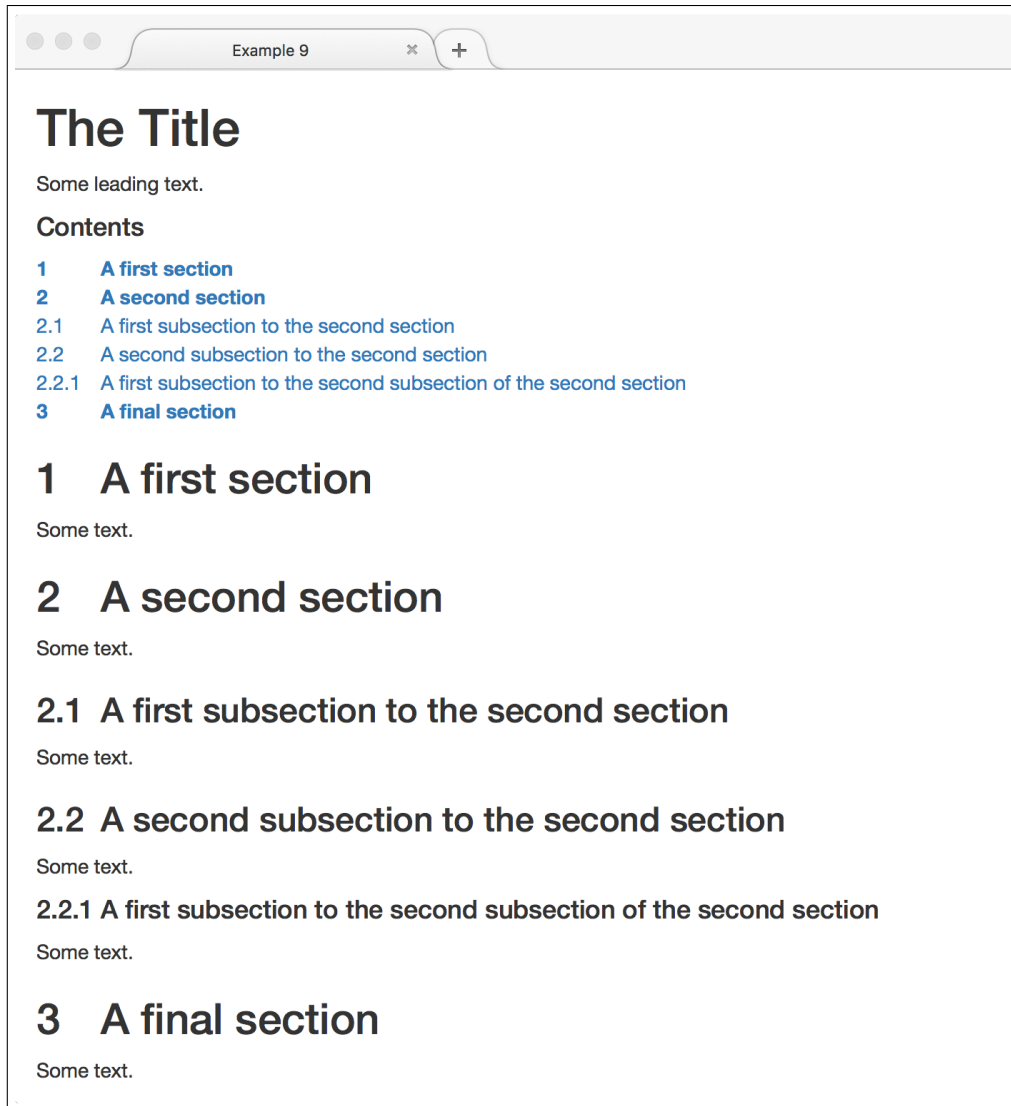


Figure 9: Output file “example9.html” displayed in a browser

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

/***
<p> 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').</p>
***/
— end of file —
```

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

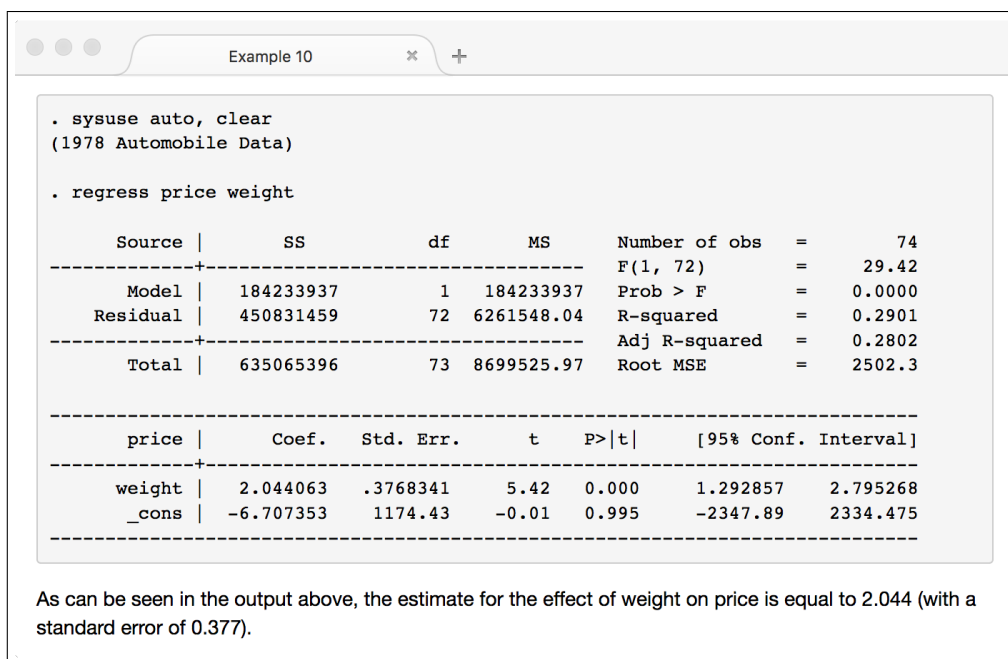


Figure 10: Output file “example10.html” displayed in a browser

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

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

4 Limitations

In general, you can work on a do-file containing `webdoc` commands in the same way as you would work on another do-file. For example, if you submit the do-file to Stata without applying `webdoc do`, Stata will process the do-file like any other do-file; the `/** */` blocks containing HTML code will be ignored and the `webdoc` commands will do nothing. However, there are some limitations and technical issues that should be kept in mind when working with `webdoc`:

- The `$` character is used for global macro expansion in Stata. If you use `webdoc write` or `webdoc put` to write text containing `$`, type `\$` instead of `$`.
- `webdoc do` only provides limited support for the semicolon command delimiter (see [P] `#delimit`). For example, do not use semicolons to delimit `webdoc` commands.

However, the semicolon command delimiter should work as expected as long as it is turned on and off between `/** */` blocks and between `webdoc` commands.

- `webdoc` commands should always start on a new line with `webdoc` being the first (non-comment) word on the line. For example, do *not* type

```
. quietly webdoc ...
```

or similar.

- `webdoc stlog` cannot be nested. Furthermore, do not use `webdoc do` or `webdoc init` within a `webdoc stlog` section.
- When processing a do-file, `webdoc 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, for example, `/** */` blocks in such a file will be ignored. Use `webdoc do` instead of `do` to include such a do-file.
- `webdoc` 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 `webdoc`.

References

- Baum, C. F., N. J. Cox, and B. Rising. 2001. LOG2HTML: Stata module to produce HTML log files. Statistical Software Components, Boston College Department of Economics. Available from <https://ideas.repec.org/c/boc/bocode/s422801.html>.
- Bruun, N. H. 2016a. LOG2MARKUP: Stata module to transform a Stata text log into a markup document. Statistical Software Components, Boston College Department of Economics. Available from <https://ideas.repec.org/c/boc/bocode/s458147.html>.
- . 2016b. MATRIXTOOLS: Stata module to build, present and style Stata matrices. Statistical Software Components, Boston College Department of Economics. Available from <https://ideas.repec.org/c/boc/bocode/s458201.html>.
- Haghighi, E. 2014a. MARKDOC: Stata module for literate programming. Statistical Software Components, Boston College Department of Economics. Available from <https://ideas.repec.org/c/boc/bocode/s457868.html>.
- . 2014b. WEAVER: Stata module to produce dynamic reports in HTML, LaTeX and PDF. Statistical Software Components, Boston College Department of Economics. Available from <https://ideas.repec.org/c/boc/bocode/s457878.html>.

- Jann, B. 2007. Making regression tables simplified. *The Stata Journal* 7(2): 227–244.
- . 2016. Creating L^AT_EX documents from within Stata using texdoc. *The Stata Journal* 16(2): 245–263.
- Jeanty, P. W. 2010. HLP2HTML: Stata module to translate a list of Stata help files to HTML. Statistical Software Components, Boston College Department of Economics. Available from <https://ideas.repec.org/c/boc/bocode/s457209.html>.
- Newson, R. 2001. LISTTEX: Stata module to list variables as rows of a TeX, HTML or word processor table. Statistical Software Components, Boston College Department of Economics. Available from <https://ideas.repec.org/c/boc/bocode/s423201.html>.
- . 2015. HTMLUTIL: Stata module to provide utilities for writing Hypertext Markup Language (HTML) files. Statistical Software Components, Boston College Department of Economics. Available from <https://ideas.repec.org/c/boc/bocode/s458085.html>.
- Quintó, L., S. Sanz, E. De Lazzari, and J. J. Aponte. 2012. HTML output in Stata. *The Stata Journal* 12(4): 702–717.
- Watson, I. 2004. TABOUT: Stata module to export publication quality cross-tabulations. Statistical Software Components, Boston College Department of Economics. Available from <https://ideas.repec.org/c/boc/bocode/s447101.html>.