Title

stata.com

dyndoc — Convert dynamic Markdown document to an HTML file

Description

`dyndoc` converts a dynamic Markdown document—a document containing both formatted text and Stata commands—to an output file in HTML format. Stata processes the Markdown text and Stata dynamic tags (see [P] **dynamic tags**) and creates the output HTML file. Markdown is a simple markup language with a formatting syntax based on plain text. It is easily converted to an output format such as HTML. Stata dynamic tags allow Stata commands, output, and graphs to be interleaved with Markdown text.

If you want to convert a Markdown document without Stata dynamic tags to an HTML document, see [P] **markdown**. If you want to convert a plain text file containing Stata dynamic tags to a plain text output file, see [P] **dyntext**.

Syntax

`dyndoc` *srcfile* [*arguments*] [`,` *options*]

srcfile is a plain text file containing Markdown-formatted text and Stata dynamic tags.

arguments are stored in the local macros `'1'`, `'2'`, and so on for use in *srcfile*; see **[U] 16.4.1 Argument passing**.

You may enclose *srcfile* and *targetfile* in double quotes and must do so if they contain blanks or other special characters.

options	Description
`saving(`*targetfile*`)`	specify the target HTML file to be saved
`replace`	replace the target HTML file if it already exists
`hardwrap`	replace hard wraps (actual line breaks) with the HTML tag ` `
`nomsg`	suppress message of a link to *targetfile*
`nostop`	do not stop when an error occurs

Options

`saving(`*targetfile*`)` specifies the target file to be saved. If `saving()` is not specified, the target filename is constructed using the source filename (*srcfile*) with the `.html` extension.

`replace` specifies that the target file be replaced if it already exists.

`hardwrap` specifies that hard wraps (actual line breaks) in the Markdown document be replaced with the HTML line break tag `<br>`.

nomsg suppresses the message that contains a link to the target file.

nostop allows the document to continue being processed even if an error occurs. By default, dyndoc stops processing the document if an error occurs. The error can be caused by either a malformed dynamic tag or by executing Stata code within the tag.

Remarks and examples

stata.com

A dynamic document contains both static narrative and dynamic tags. Dynamic tags are instructions for dyndoc to perform certain actions, such as run a block of Stata code, insert the result of a Stata expression in text, export a Stata graph to an image file and include a link to the image file, etc. Any changes in the data or in Stata will change the output as the document is created. The main advantages of using dynamic documents are

- results in the document come from executing commands instead of being copied from Stata and pasted into the document;
- no need to maintain parallel do-files; and
- any changes in data or in Stata are reflected in the final document when it is created.

▷ Example 1

Let us consider an example. Suppose that we have dyndoc_ex.txt with the following Markdown-formatted text that includes Stata dynamic tags.

begin dyndoc_ex.txt

```
<<dd_version: 1>>
<<dd_include: header.txt >>

Using Stata dynamic tags in a text file with the dyndoc command
===============================================================

Let us consider an example where we study the **mpg** and **weight** variables
in **auto.dta**.  In our examples below, we will first write the commands so
that they will be displayed in our target HTML file.  Then, we will write the
commands so that Stata will process the Stata dynamic tags, displaying the
results of the Stata commands in the target HTML file.

We first use the **sysuse** command to load the dataset and then describe
the data using the **describe** command.
~~~~
<<dd_ignore>>
<<dd_do>>
sysuse auto, clear
describe
<</dd_do>>
<</dd_ignore>>
~~~~
This produces the following Stata results:
~~~~
<<dd_do>>
sysuse auto, clear
describe
<</dd_do>>
~~~~

Now, we want to check if **mpg** is always greater than 0 and less than 100.
We use the **assert** command to perform the check.  In this case, we do not
want to include any output in the target HTML file, so we use the **quietly**
attribute to modify the behavior of the **dd_do** Stata dynamic tag.
```

```
~~~~
<<dd_ignore>>
<<dd_do:quietly>>
assert mpg > 0 & mpg < 100
<</dd_do>>
<</dd_ignore>>
<<dd_do:quietly>>
assert mpg > 0 & mpg < 100
<</dd_do>>
~~~~
If the data do not satisfy the conditions, **dyndoc** will fail with an error
message, which will occur if we run the same **assert** command in a do-file.
Next, we want to summarize the **weight** variable:
~~~~
<<dd_ignore>>
<<dd_do>>
summarize weight
<</dd_do>>
<</dd_ignore>>
~~~~
This produces the following in the target HTML file:
~~~~
<<dd_do>>
summarize weight
<</dd_do>>
~~~~
We want to use the minimum and maximum values of **weight** in a sentence.
Instead of copying and pasting the numbers from the **summarize** output, we can
use the **dd_display** Stata dynamic tag with the **r(min)** and **r(max)**
stored results:
~~~~
<<dd_ignore>>
The variable weight has minimum value <<dd_display: %4.2f `r(min)'>> and
has maximum value <<dd_display: %4.2f `r(max)'>>.
<</dd_ignore>>
~~~~
This produces the following in the target HTML file:
~~~~
> The variable weight has minimum value <<dd_display: %4.2f `r(min)'>>
and has maximum value <<dd_display: %4.2f `r(max)'>>.
~~~~
The **dd_display** dynamic tag uses Stata's **display** command to evaluate
expressions.  It can be used as a calculator. For example, if we want to
include the $$range = max - min$$ in a sentence, instead of calculating the
number and then copying and pasting it, we can use
~~~~
<<dd_ignore>>
The variable weight has range <<dd_display: %4.2f `r(max)'-`r(min)'>>.
<</dd_ignore>>
~~~~
which produces the following in the target HTML file:
~~~~
> The variable weight has range <<dd_display: %4.2f `r(max)'-`r(min)'>>.
~~~~
Now, we want to graph **mpg** and **weight** using a scatterplot.  We use the
**dd_do** tag with the **nooutput** attribute to generate the scatterplot
first.  The **nooutput** attribute leaves the command in the output only,
```

```
~~~~
<<dd_ignore>>
<<dd_do:nooutput>>
scatter mpg weight, mcolor(blue%50)
<</dd_do>>
<</dd_ignore>>
~~~~
which generates a scatterplot of **mpg** and **weight** with 50% opacity
color markers.
~~~~
<<dd_do:nooutput>>
scatter mpg weight, mcolor(blue%50)
<</dd_do>>
~~~~
Now, we want to export the graph to a file and include an image link to the
file.
~~~~
<<dd_ignore>>
<<dd_graph: sav("graph.svg") alt("scatter mpg price") replace height(400)>>
<</dd_ignore>>
~~~~
This produces a graph of 400 pixels high.
<<dd_graph: sav("graph.svg") alt("scatter mpg price") replace height(400)>>
```

end dyndoc_ex.txt

❑ Technical note

We use four tildes in a row, `~~~~`, in our source file around parts of the document that we want to appear in plain text, such as Stata commands and output. Without the `~~~~`, Stata's output would be interpreted as HTML in the final document and would not look as it should.

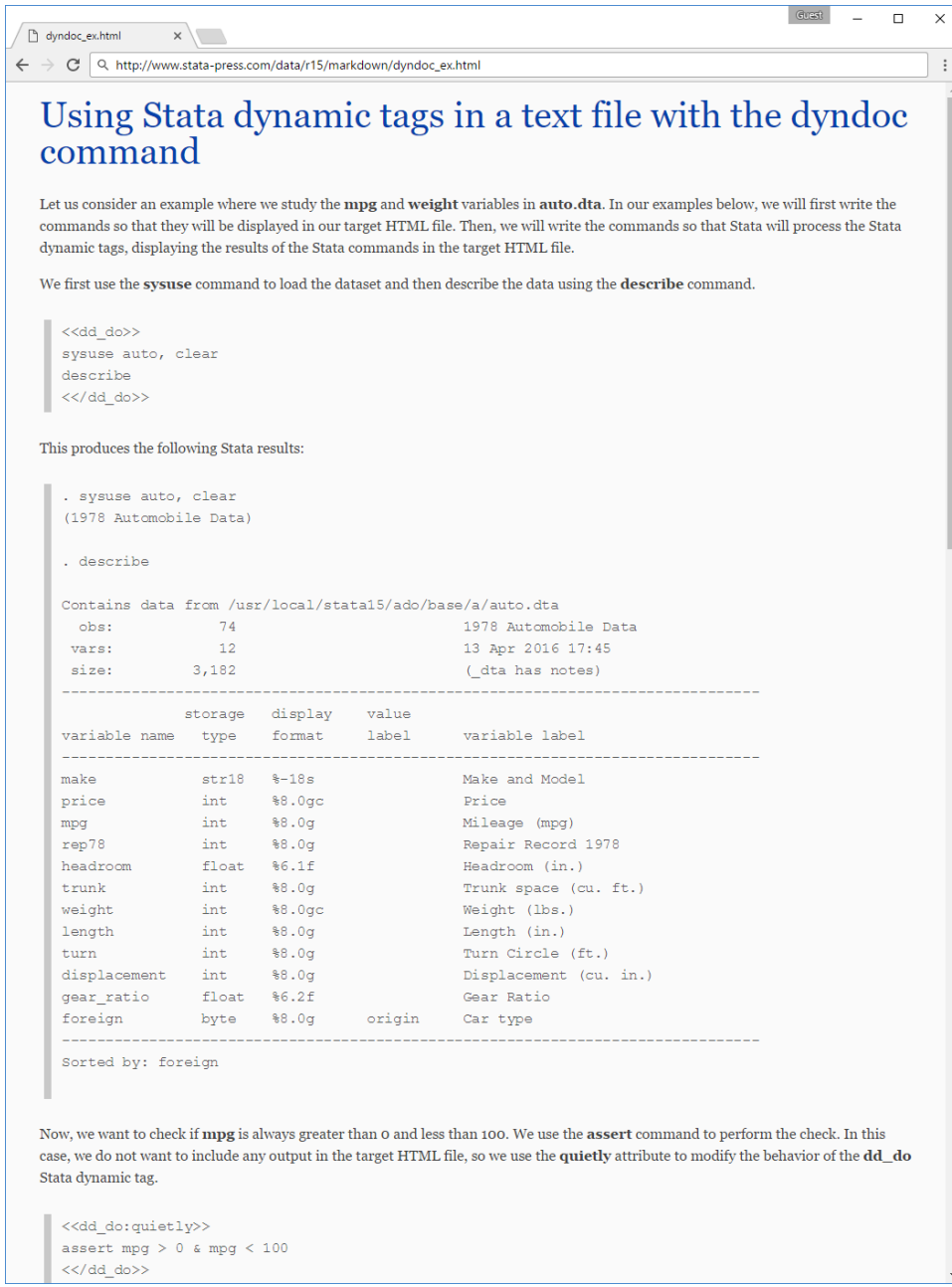
❑

You will notice that we used the `<<dd_include>>` dynamic tag to include the `header.txt` file. The `header.txt` file contains HTML code to include at the top of our target HTML file. It refers to the `stmarkdown.css` file, which is a stylesheet that defines how the HTML document is to be formatted. Both of these files and `dyndoc_ex.txt` are available at http://www.stata-press.com/data/r15/markdown/.

With these three files in our working directory, we generate the target HTML file in Stata by typing

```
. dyndoc dyndoc_ex.txt
```

The HTML file `dyndoc_ex.html` is saved. Here is a portion of this file:



You can see the whole file at http://www.stata-press.com/data/r15/markdown/dyndoc_ex.html.

❑ Technical note

Because `quietly` and `capture` suppress the results of the command from being produced, you should not use these prefix commands with Stata code to be converted by `dyndoc`.

❑

Reference

Jann, B. 2017. Creating HTML or Markdown documents from within Stata using webdoc. *Stata Journal* 17: 3–38.

Also see

[P] **dynamic tags** — Dynamic tags for Markdown documents

[P] **dyntext** — Process Stata dynamic tags in text file

[P] **markdown** — Convert Markdown document to an HTML file