Title

stata.com

dyndoc — Convert dynamic Markdown document to HTML or Word (.docx) document

Description

`dyndoc` converts a dynamic Markdown document—a document containing both formatted text and Stata commands—to an HTML file or Word document. Stata processes the Markdown text and Stata dynamic tags (see [RPT] **Dynamic tags**) and creates the output 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 file or Word document, see [RPT] **markdown**. If you want to convert a plain text file containing Stata dynamic tags to a plain text output file, see [RPT] **dyntext**. If you want to convert an HTML file to a Word document, see [RPT] **html2docx**.

Quick start

Convert text file `myfile.txt` with Stata dynamic tags and Markdown formatting to an HTML file with Stata output saved as `myfile.html`

```
dyndoc myfile.txt
```

As above, but save the HTML file as `mydoc.html`

```
dyndoc myfile.txt, saving(mydoc.html)
```

As above, and overwrite the existing `mydoc.html`

```
dyndoc myfile.txt, saving(mydoc.html) replace
```

Convert `myfile.txt` to a Word document saved as `myfile.docx`

```
dyndoc myfile.txt, docx
```

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 or Word (`.docx`) document to be saved
`replace`	replace the target HTML file or Word (`.docx`) document if it already exists
`hardwrap`	replace hard wraps (actual line breaks) with the ` ` tag in an HTML file or with line breaks in a Word (`.docx`) document
`nomsg`	suppress message with a link to *targetfile*
`nostop`	do not stop when an error occurs
`embedimage`	embed image files as Base64 binary data in the target HTML file
`docx`	output a Word (`.docx`) document instead of an HTML file

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 or with the `.docx` extension if `docx` is specified. If the *targetfile* has the `.docx` extension, the `docx` option is assumed even if it is not specified.

`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 `<br>` tag in the HTML file or with a line break in the Word (`.docx`) document if the `docx` option is specified.

`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 Stata code executed within the tag.

`embedimage` allows image files to be embedded as data URI (Base64-encoded binary data) in the HTML file. The supported image file types are portable network graphics (`.png`), JPEG (`.jpg`), tagged image file format (`.tif`), and graphics interchange format (`.gif`). This option cannot be used to embed SVG and PDF image file types.

The image must be specified in a Markdown link; you cannot embed images specified by URLs. This option is ignored if `docx` is specified.

`docx` specifies that the target file be saved in Microsoft Word (`.docx`) format. If the target file has the `.docx` extension, the `docx` option is implied. The conversion process consists of first producing an HTML file and then using `html2docx` to produce the final Word document.

Remarks and examples

stata.com

A dynamic document contains both static narrative and dynamic tags. Dynamic tags are instructions for dyndoc to perform a certain action, 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, or include a link to the image file. 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: Converting a dynamic document to an HTML file

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: 2>>
<<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 the **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 weight") replace height(400)>>
<</dd_ignore>>
~~~~
This produces a graph of 400 pixels high.
<<dd_graph: sav("graph.svg") alt("scatter mpg weight") 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. This applies regardless of whether we are creating an HTML file or a Word document.

❑

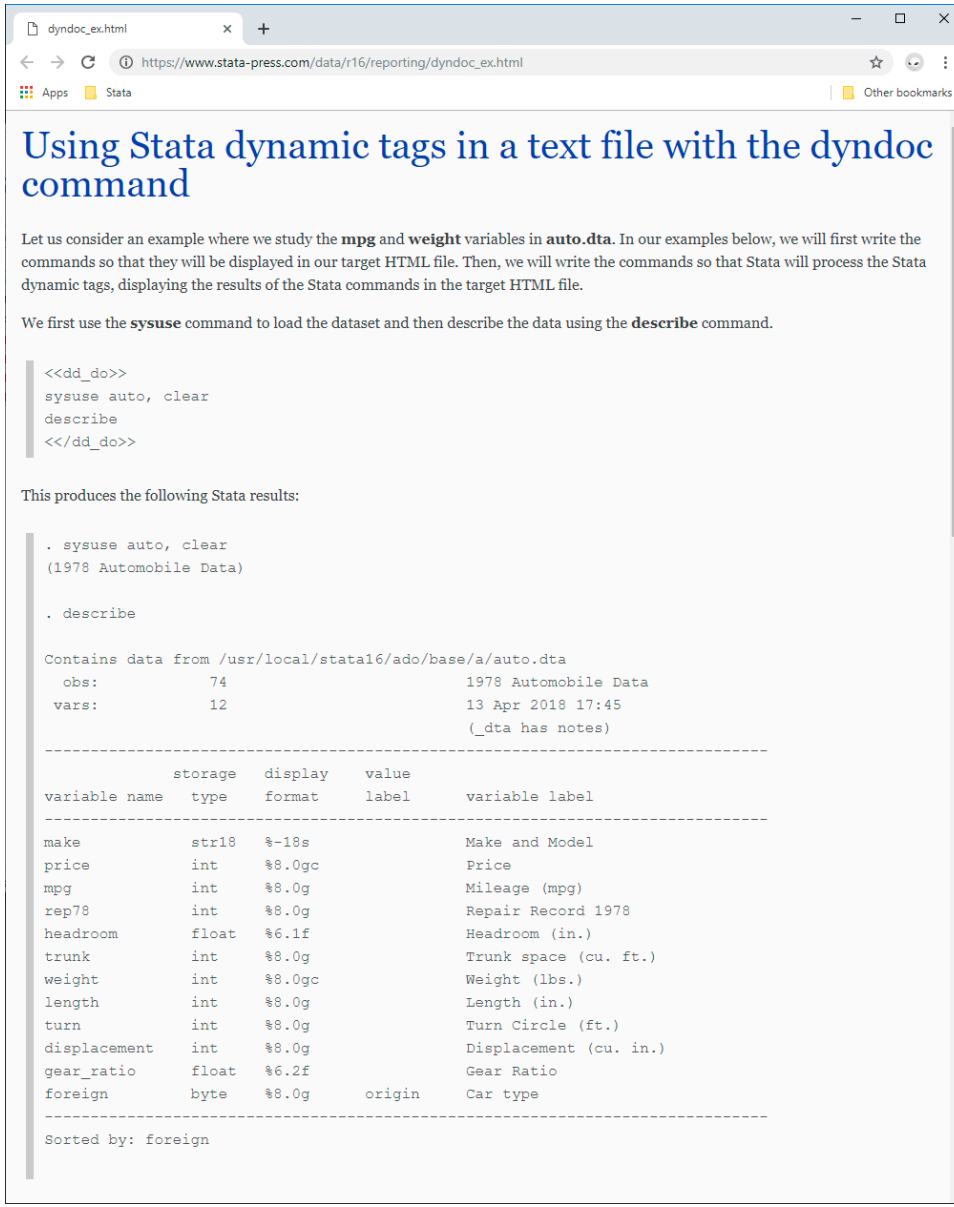
You will notice that we used the `<<dd_include>>` dynamic tag to include the `header.txt` file. The `header.txt` file contains HTML code to include at the top of our target HTML file. It refers to the `stmarkdown.css` file, which is a stylesheet that defines how the HTML document is to be formatted. This formatting would also apply if we were creating a Word document. We can copy these files and `dyndoc_ex.txt` to our working directory by typing

```
. copy http://www.stata-press.com/data/r16/reporting/header.txt .
. copy http://www.stata-press.com/data/r16/reporting/stmarkdown.css .
. copy http://www.stata-press.com/data/r16/reporting/dyndoc_ex.txt .
```

To generate the target HTML file in Stata, we type

```
. 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 https://www.stata-press.com/data/r16/reporting/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`.

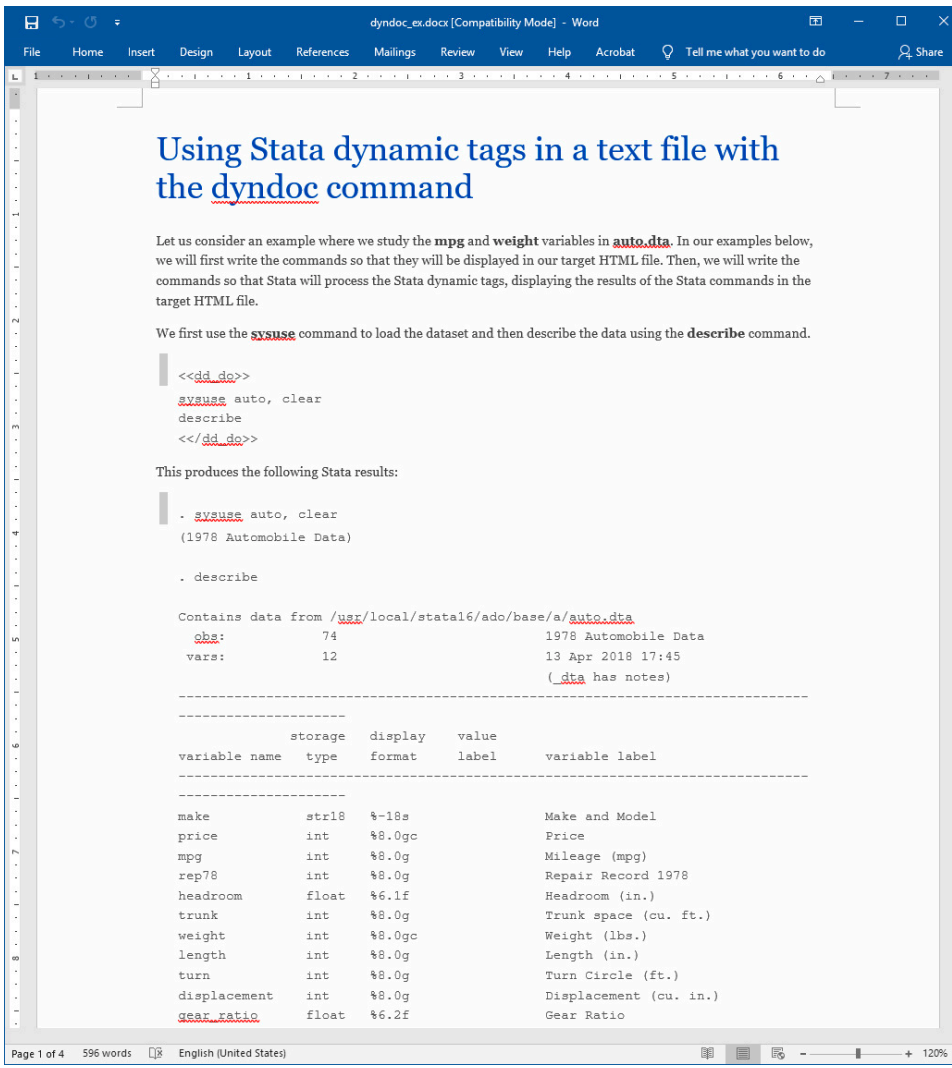
❑

▷ Example 2: Converting a dynamic document to a Word document

As above, we could easily create a Word document from the dynamic text file used in example 1 by specifying the `docx` option.

```
. dyndoc dyndoc_ex.txt, docx
```

This creates `dyndoc_ex.docx`, shown below:



◁

Specifying arguments in dyndoc allows even more flexibility. For instance, dyndoc_ex.txt a b c passes a, b, and c in local macros ‘1’, ‘2’, and ‘3’, which can be used in the text file. Below, we demonstrate how to use arguments with dyndoc to create multiple HTML files from a single text file.

Example 3: Specifying arguments

Suppose that we create histograms on a regular basis. To reduce our workload, we create a general text file that creates histograms for pairs of variables, which we specify as arguments to dyndoc. This way we do not create a new text file every time we create new histograms. To generalize the text file, we refer to macro 1 when loading the dataset, so that we may create histograms using any dataset we specify. We refer to macros 2 and 3 in the <<dd_do>> tags, instead of referring to actual variable names, because these will change. We use the <<dd_display>> dynamic tag to display the contents of the macros in the output HTML file, while suppressing the command lines with the nocommands attribute. Our text file is shown below:

begin dyndoc_ex2.txt

```
<<dd_version: 2>>
The distribution of <<dd_display: "‘2’">>  and <<dd_display: "‘3’">>
==================================
Let’s look at the histograms for <<dd_display: "‘2’">> and <<dd_display: "‘3’">>.
~~~~
. use <<dd_display: "‘1’">>, clear
~~~~
~~~~
<<dd_do: nocommands>>
use ‘1’.dta, clear
<</dd_do>>
~~~~
~~~~
. histogram <<dd_display: "‘2’">>, freq
~~~~
~~~~
<<dd_do: nocommands>>
histogram ‘2’, freq
<</dd_do>>
~~~~
<<dd_graph>>
~~~~
. histogram <<dd_display: "‘3’">>, freq
~~~~
~~~~
<<dd_do: nocommands>>
histogram ‘3’, freq
<</dd_do>>
~~~~
<<dd_graph>>
```

end dyndoc_ex2.txt

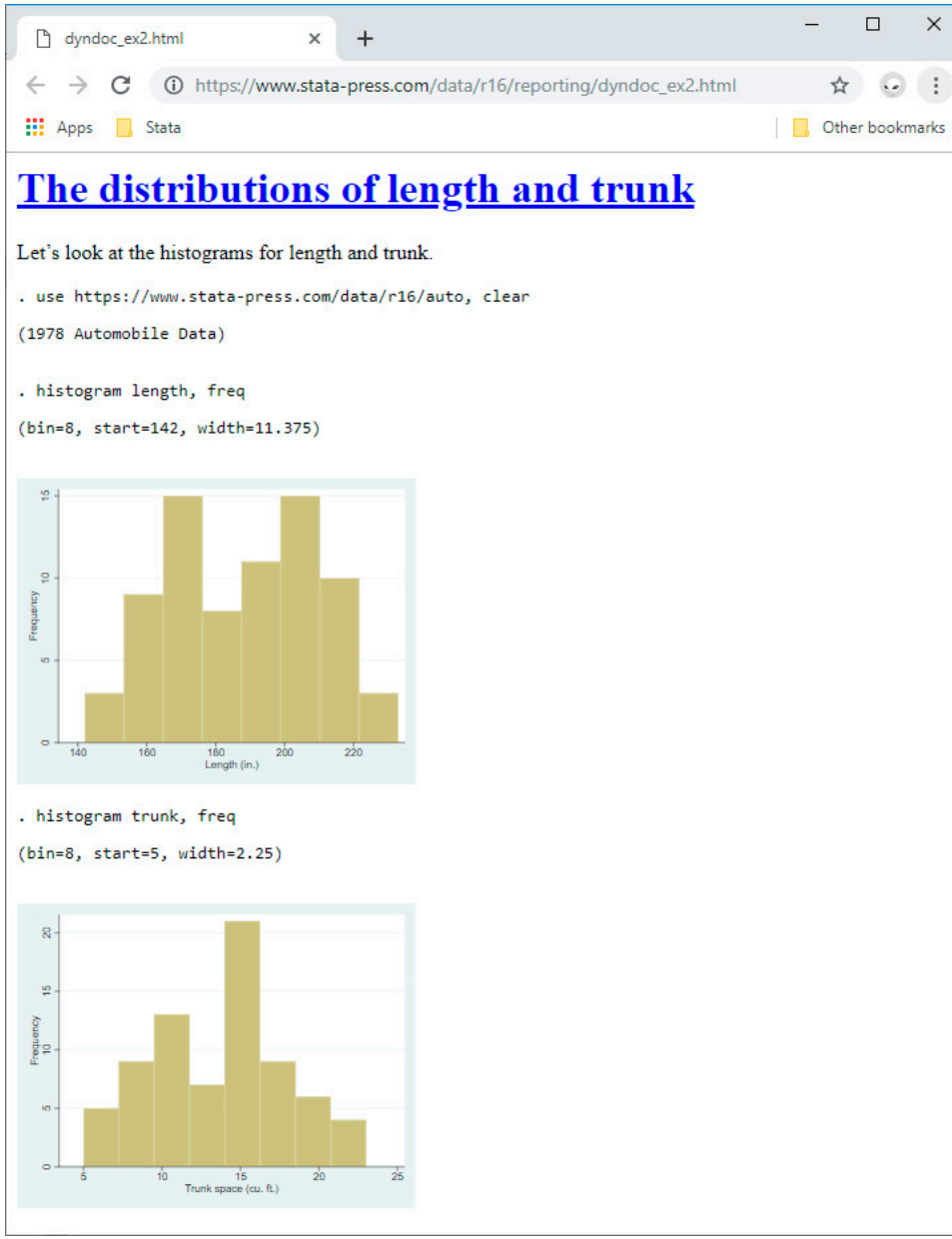
You can copy this file to your working directory by typing

```
. copy http://www.stata-press.com/data/r16/reporting/dyndoc_ex2.txt .
```

Now we issue dyndoc with three arguments: a link to auto.dta followed by variable names length and trunk.

```
. dyndoc dyndoc_ex2.txt "https://www.stata-press.com/data/r16/auto" length trunk
```

This creates `dyndoc_ex2.html`, shown below:

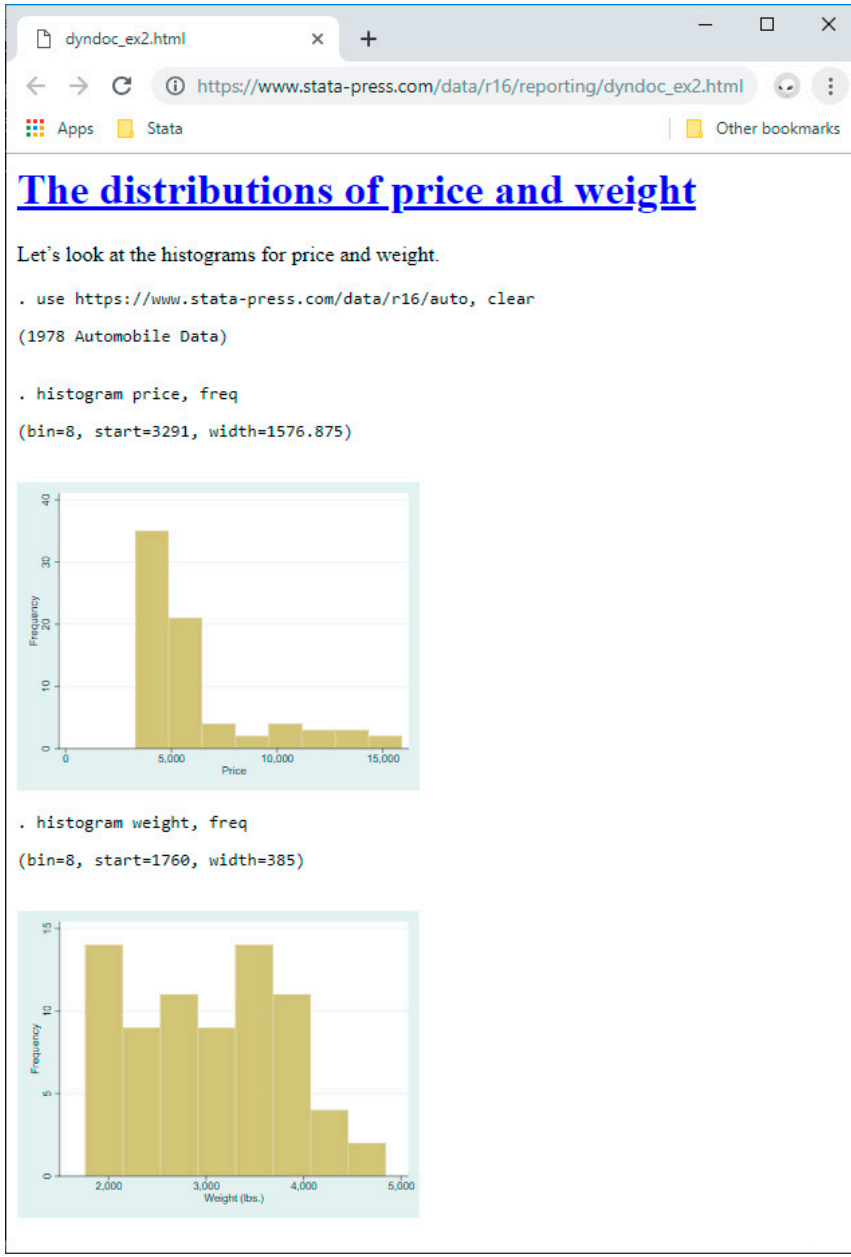


We optionally displayed the information on bin size, width, and starting values that `histogram` reports, but we could suppress that as well by specifying the `quietly` attribute instead.

If we want to create histograms with other variables, we do not have to edit our text file. We can simply issue `dyndoc` with the corresponding dataset and variable names to create this file. This is as easy as

```
. dyndoc dyndoc_ex2.txt "https://www.stata-press.com/data/r16/auto" price weight,
> replace
```

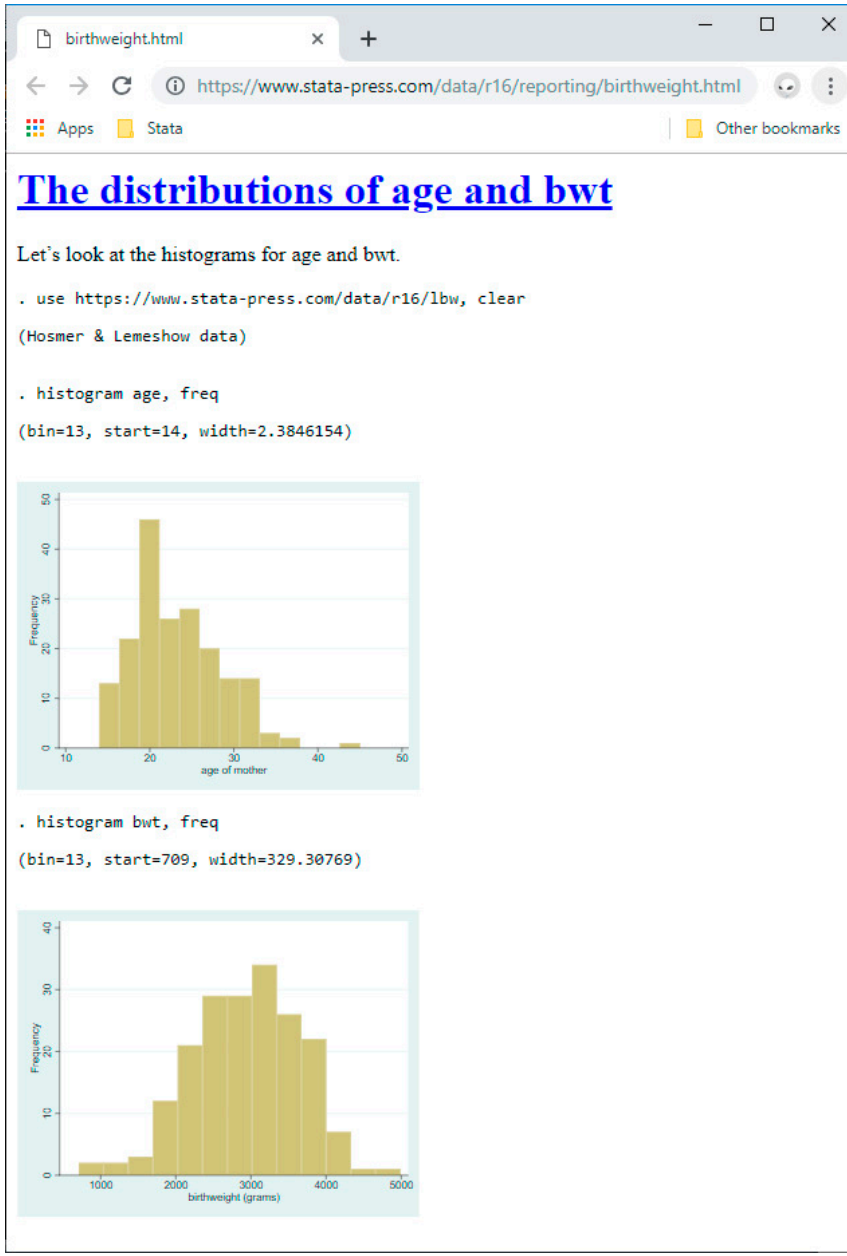
which produces the following:



Next we use the data from Hosmer, Lemeshow, and Sturdivant (2013, 24) to create histograms for mother's age and the baby's birthweight, bwt.

```
. dyndoc dyndoc_ex2.txt "https://www.stata-press.com/data/r16/lbw" age bwt,
> saving(birthweight.html)
```

This produces the following:



Whether you are creating the same graphics for different sets of variables or you need to create the same reports using different datasets, you can use arguments with `dyndoc` to streamline your work process.

◁

References

Gillman, M. S. 2018. Some commands to help produce Rich Text Files from Stata. *Stata Journal* 18: 197–205.

Hosmer, D. W., Jr., S. A. Lemeshow, and R. X. Sturdivant. 2013. *Applied Logistic Regression*. 3rd ed. Hoboken, NJ: Wiley.

Jakubowski, M., and A. Pokropek. 2019. piaactools: A program for data analysis with PIAAC data. *Stata Journal* 19: 112–128.

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

Rodríguez, G. 2017. Literate data analysis with Stata and Markdown. *Stata Journal* 17: 600–618.

Also see

[RPT] **Dynamic tags** — Dynamic tags for text files

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

[RPT] **markdown** — Convert Markdown document to HTML file or Word (.docx) document