

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

[Description](#)      [Quick start](#)      [Syntax](#)      [Options](#)  
[Remarks and examples](#)      [References](#)      [Also see](#)

## Description

`markdown` converts a Markdown document to an HTML file or a Word document.

## Quick start

Convert Markdown document `myfile.txt` to an HTML file saved as `myfile.html`  
`markdown myfile.txt, saving(myfile.html)`

Same as above, and overwrite the existing `myfile.html`  
`markdown myfile.txt, saving(myfile.html) replace`

Convert Markdown document `myfile.txt` to a Word document saved as `myfile.docx`  
`markdown myfile.txt, saving(myfile.docx) docx`

## Syntax

`markdown srcfile, saving(targetfile) [options]`

*srcfile* is the Markdown document to be converted.

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

<i>options</i>	Description
* <u>saving</u> ( <i>targetfile</i> ) <u>replace</u>	HTML file or Word (.docx) document to be saved replace the target HTML file or Word (.docx) document if it already exists
<code>hardwrap</code>	replace hard wraps (actual line breaks) with the <code>&lt;br&gt;</code> tag in an HTML file or with line breaks in a Word (.docx) document
<code>nomsg</code>	suppress message with a link to <i>targetfile</i>
<code>embedimage</code>	embed image files as Base64 binary data in the target HTML file
<code>basedir</code> ( <i>string</i> )	specify the base directory for relative links in <i>srcfile</i>
<code>docx</code>	output a Word .docx document instead of an HTML file

\* `saving(targetfile)` is required.

## Options

`saving(targetfile)` specifies the target file to be saved. If the *targetfile* has the .docx extension, the docx option is assumed even if it is not specified. `saving()` is required.

`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) file if the docx option is specified.

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

`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.

`basedir(string)` specifies the base directory for the relative links in the *srcfile*. This option only applies when specifying either the docx option or the embedimage option; otherwise, this option is ignored.

`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](http://www.stata.com)

`markdown` converts a Markdown document to an HTML file or a Word (.docx) document. A Markdown document is written using an easy-to-read, plain-text, lightweight markup language. For a detailed discussion and the syntax of Markdown, see the [Markdown Wikipedia page](#).

Stata uses Flexmark's Pegdown emulation as its default Markdown document processing engine. For information on Pegdown's flavor of Markdown, see the [Pegdown GitHub page](#).

See [\[RPT\] dyndoc](#) and [\[RPT\] dyntext](#) for a full description of Stata's dynamic document-generation commands. `markdown` is used by `dyndoc` but may also be used directly by programmers.

If you are not familiar with Markdown, we recommend that you first review [example 1](#) below for a brief introduction to this markup language. If you have used Markdown before, see [example 2](#) for details on embedding Stata graphs in your HTML file.

### ► Example 1: Create a basic HTML file with text

Suppose that we want to create a webpage with tips on working with Stata. We have created `markdown1.txt`, shown below, using Markdown-formatted text. We include comments to describe the formatting.

begin markdown1.txt

```

Tips on working in Stata
=====
<!-- To create a heading, we underline text with equal signs. -->
<!-- Text enclosed in these arrows will be ignored. -->
This webpage will provide useful tips on working with Stata.
---
<!-- The three dashes above create a horizontal line. -->
## Working with strings
<!-- We create a sub-heading with two pound signs. -->
We begin by demonstrating when to use destring and encode.
<!-- We use two asterisks around each word we want to format as bold. -->
You should only use destring if a variable contains numeric values.
For example, in the following dataset income is stored as a string:
'''
webuse destring1, clear
destring income, replace
'''
<!-- We use three back-ticks for blocks of code. -->

```

end markdown1.txt

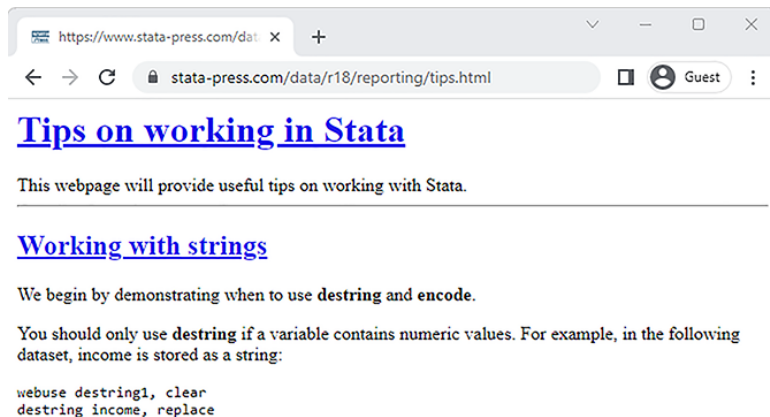
You can copy this file into your working directory by typing

```
. copy https://www.stata-press.com/data/r18/reporting/markdown1.txt .
```

To convert this text file to an HTML file, we type the following command:

```
. markdown markdown1.txt, saving(tips.html)
```

This creates `tips.html`, which looks like the following:



`tips.html` is also available at <https://www.stata-press.com/data/r18/reporting/>.

## ► Example 2: Create an HTML file with Stata graphs

The do-file below creates a couple of graphs using Stata.

---

```
begin markdown.do
sysuse auto, clear
tway lfitci mpg weight || scatter mpg weight, title(MPG as a function of weight)
graph export mpg1.png
regress mpg weight
rvfplot, yline(0) title(Residuals versus fitted values)
graph export diagplot.png, replace
```

---

end markdown.do

Suppose we wish to create a webpage (an HTML file) with the instructions on how to produce these graphs. First, we write `markdown2.txt` containing Markdown-formatted text and the Stata code to create the graphs above.

---

```
begin markdown2.txt

Creating graphs in Stata
=====

Below we review some diagnostic plots available in Stata, and we demonstrate
how to overlay plots. We use 'auto.dta', which contains pricing and mileage
data for 1978 automobiles.

---
<!-- In the previous example, we used three back-ticks for a block of code.
Here we use a single back-tick for inline code. -->
## Plotting predictions

We are interested in modeling the mean of mpg, miles per gallon, as a function of
weight, car weight in pounds. We can use tway lfitci to graph the predicted
miles per gallon from a linear regression, as well as the confidence interval:
'''
sysuse auto, clear
tway lfitci mpg weight
'''

To see how these predictions compare to our data, we can overlay a scatterplot
of the actual data
'''
tway lfitci mpg weight || scatter mpg weight, title(MPG as a function of weight)
'''

which produces the following graph:
![[Graph of mpg](mpg1.png)]
<!-- We previously used graph export to save the graph as a .png
file, which we now embed in the document. -->
We could have also created separate graphs for domestic and foreign
cars with the by() option. See
[graph tway lfitci](https://www.stata.com/manuals/g-2graphtwaylfitci.pdf)
in the Stata Graphics Reference Manual for details.

---
## Diagnostic plot

There are multiple diagnostic plots available for use after regress. Here,
we use rvfplot to graphically check for a relationship between the
residuals and fitted values from our model. We regress mpg on
weight and then issue rvfplot.
'''
regress mpg weight
rvfplot, yline(0) title(Residuals versus fitted values)
'''

The commands above produce the following graph:
![[Diagnostic plot](diagplot.png)]

---
```

---

end markdown2.txt

We can copy both of these files to our current working directory by typing

```
. copy https://www.stata-press.com/data/r18/reporting/markdown.do .  
. copy https://www.stata-press.com/data/r18/reporting/markdown2.txt .
```

We now type the following to execute the commands in `markdown.do` that create the graphs and export them to PNG files. We then convert `markdown2.txt` to an HTML file by using the `markdown` command.

```
. do markdown.do  
. markdown markdown2.txt, saving(graphs.html)
```

This creates `graphs.html`, which looks like the following:

https://www.stata-press.com/data/r18/reporting/graphs.html

## Creating graphs in Stata

Below we review some diagnostic plots available in Stata, and we demonstrate how to overlay plots. We use `auto.dta`, which contains pricing and mileage data for 1978 automobiles.

### Plotting predictions

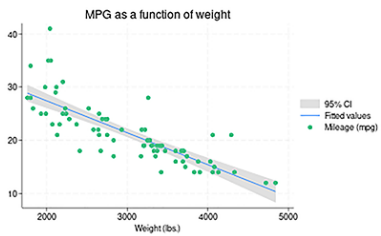
We are interested in modeling the mean of `mpg`, miles per gallon, as a function of `weight`, car weight in pounds. We can use `twoway lfitci` to graph the predicted miles per gallon from a linear regression, as well as the confidence interval:

```
sysuse auto, clear
twoway lfitci mpg weight
```

To see how these predictions compare to our data, we can overlay a scatterplot of the actual data

```
twoway lfitci mpg weight || scatter mpg weight, title(MPG as a function of weight)
```

which produces the following graph:



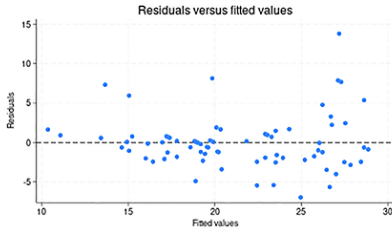
We could have also created separate graphs for domestic and foreign cars with the `by()` option. See [graph twoway lfitci](#) in the Stata Graphics Reference Manual for details.

### Diagnostic plot

There are multiple diagnostic plots available for use after `regress`. Here, we use `rvfplot` to graphically check for a relationship between the residuals and fitted values from our model. We regress `mpg` on `weight` and then issue `rvfplot`.

```
regress mpg weight
rvfplot, yline(0) title(Residuals versus fitted values)
```

The commands above produce the following graph:



You can view the HTML file at <https://www.stata-press.com/data/r18/reporting/graphs.html>.

In this example, we ran the Stata commands and created the HTML file in two separate steps. However, we could do both in a single step with `dyndoc`, which processes Stata code to embed Stata output and graphs in the destination HTML file, and calls `markdown` to process the Markdown-formatted text (like we did above). See [\[RPT\] dyndoc](#) for information on how to create an HTML or Word (.docx) file with Stata output from a Markdown text file.

## References

- Haghighi, E. F. 2016. [markdoc: Literate programming in Stata](#). *Stata Journal* 16: 964–988.
- . 2020a. [Software documentation with markdoc 5.0](#). *Stata Journal* 20: 336–362.
- . 2020b. [Developing, maintaining, and hosting Stata statistical software on GitHub](#). *Stata Journal* 20: 931–951.
- Jann, B. 2017. [Creating HTML or Markdown documents from within Stata using webdoc](#). *Stata Journal* 17: 3–38.

## Also see

- [RPT] [Dynamic tags](#) — Dynamic tags for text files
- [RPT] [dyndoc](#) — Convert dynamic Markdown document to HTML or Word (.docx) document
- [RPT] [dyntext](#) — Process Stata dynamic tags in text file

Stata, Stata Press, and Mata are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow and NetCourseNow are trademarks of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2023 StataCorp LLC, College Station, TX, USA. All rights reserved.



For suggested citations, see the [FAQ on citing Stata documentation](#).