

**Dynamic tags** — Dynamic tags for text files[Description](#)[Remarks](#)[Also see](#)

## Description

Dynamic tags are instructions used by Stata's dynamic documents commands, [dyndoc](#) and [dyntext](#), 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.

## Remarks

Remarks are presented under the following headings:

*Descriptions of dynamic tags*

*Version control*

*Execute and include output from a block of Stata code*

*Include strings and values of scalar expressions in text*

*Include values of scalar expressions and formatted text in a .docx file*

*Export and include a Stata graph*

*Include a text file*

*Disable dynamic text processing*

*Process contents based on condition*

*Skip contents based on condition*

*Remove contents*

## Descriptions of dynamic tags

Here is a list of available dynamic tags and a short description for each. The tag may be abbreviated; the minimum abbreviation is indicated by the underlined letters.

## 2 Dynamic tags — Dynamic tags for text files

---

Dynamic tag	Description
<code>&lt;&lt;dd_version&gt;&gt;</code>	specify the minimum version required to convert the dynamic document
<code>&lt;&lt;dd_do&gt;&gt;</code>	execute a block of Stata code and optionally include its output
<code>&lt;&lt;/dd_do&gt;&gt;</code>	end <code>&lt;&lt;dd_do&gt;&gt;</code>
<code>&lt;&lt;dd_display&gt;&gt;</code>	include output of Stata expression as shown by Stata's <code>display</code> command
<code>&lt;&lt;dd_docx_display&gt;&gt;</code>	include output of Stata expression in a <code>.docx</code> file as shown by Stata's <code>display</code> command and format text within a block
<code>&lt;&lt;dd_graph&gt;&gt;</code>	export a Stata graph and include a link to the file
<code>&lt;&lt;dd_ignore&gt;&gt;</code>	disable processing of dynamic tags except <code>&lt;&lt;dd_remove&gt;&gt;</code>
<code>&lt;&lt;/dd_ignore&gt;&gt;</code>	end <code>&lt;&lt;dd_ignore&gt;&gt;</code>
<code>&lt;&lt;dd_include&gt;&gt;</code>	include the contents of a text file
<code>&lt;&lt;dd_remove&gt;&gt;</code>	remove the following text until <code>&lt;&lt;/dd_remove&gt;&gt;</code> is specified
<code>&lt;&lt;/dd_remove&gt;&gt;</code>	end <code>&lt;&lt;dd_remove&gt;&gt;</code>
<code>&lt;&lt;dd_if&gt;&gt;</code>	process text based on condition
<code>&lt;&lt;dd_else&gt;&gt;</code>	process text based on condition
<code>&lt;&lt;dd_endif&gt;&gt;</code>	end <code>&lt;&lt;dd_if&gt;&gt;</code> block
<code>&lt;&lt;dd_skip_if&gt;&gt;</code>	skip text based on condition
<code>&lt;&lt;dd_skip_else&gt;&gt;</code>	skip text based on condition
<code>&lt;&lt;dd_skip_end&gt;&gt;</code>	end <code>&lt;&lt;dd_skip_if&gt;&gt;</code> block

---

`<<dd_docx_display>>` is only for use with `putdocx` `textblock` commands in a do-file.

Some tags must start at the beginning of a line, and the text in the same line after the tag is simply ignored. Other tags can be written in the middle of a line. The following table lists the required position in text for all tags.

Dynamic tag	Description
<<dd_version>>	beginning of a line, recommended at the start of a file
<<dd_do>>	beginning of a line
<</dd_do>>	beginning of a line
<<dd_display>>	within a line
<<dd_docx_display>>	within a line
<<dd_graph>>	within a line
<<dd_ignore>>	beginning of a line
<</dd_ignore>>	beginning of a line
<<dd_include>>	beginning of a line
<<dd_remove>>	within a line
<</dd_remove>>	within a line
<<dd_if>>	beginning of a line
<<dd_else>>	beginning of a line
<<dd_endif>>	beginning of a line
<<dd_skip_if>>	beginning of a line
<<dd_skip_else>>	beginning of a line
<<dd_skip_end>>	beginning of a line

Tags can have attributes. Attributes are modifiers of a tag's behavior. Attributes can be repeated, and the last one will take effect. For example, if you specify <<dd\_do: *commands nocommands*>>, the commands will not be displayed because the attribute *nocommands* supersedes the previously specified attribute *commands*. This is useful when you experiment with the behavior of attributes for the best output. Some attributes have values; for example, *graphname()* requires the name of the graph to be exported. If a tag has only one attribute and that attribute requires a value, then the attribute name is omitted and only the value is required; for example, the *dd\_version* tag is used as <<dd\_version: *an integer number*>>.

## Version control

```
<<dd_version: version_number>>
```

The <<dd\_version>> tag specifies the minimum version required to convert the source file. The version number is independent of Stata's [version](#) command. The tag must be at the beginning of a new line. We recommend that the tag be placed at the beginning of the *srcfile*.

The current version, and the default, is 2, and it is introduced as of the release of Stata 16. The current version number is also stored in *c(dyndoc\_version)*.

## Execute and include output from a block of Stata code

```
<<dd_do: attribute>>
block of Stata code ...
<</dd_do>>
```

The <<dd\_do>> tag runs the block of Stata code, replacing the lines between <<dd\_do>> and <</dd\_do>> with Stata output. Both the start tag, <<dd\_do>>, and the end tag, <</dd\_do>>, must be at the beginning of new lines.

<i>attribute</i>	Description
<u>quietly</u>	suppress all output
<u>nocommands</u>	suppress printing of command
<u>nooutput</u>	suppress command output
<u>noprompt</u>	suppress the dot prompt

---

## Include strings and values of scalar expressions in text

```
<<dd_display: display_directive>>
```

The `<<dd_display>>` tag executes Stata's `display` command and then replaces the tag with its output. The tag cannot contain a line break or `>>`. Use `>>` (with a space in between) instead if you need to include `>>` in the *display\_directive*.

The `<<dd_display>>` tag can be used multiple times inside a line of text. For example, say that we want to display the circumference of a circle of radius 1 up to the two digits after the decimal. Instead of computing the number and then copying and pasting the result into the text, we can write

```
2*1*<<dd_display:%4.2f c(pi)>> = <<dd_display:%4.2f 2*1*c(pi)>>
```

which produces

```
2*1*3.14 = 6.28
```

## Include values of scalar expressions and formatted text in a .docx file

```
<<dd_docx_display text_options: display_directive>>
```

This tag includes expressions and formatted text within a block of text in a `.docx` file. It can only be used with text enclosed in `putdocx textblock` commands, as follows:

```
putdocx textblock begin
... text <<dd_docx_display directive>> text ...
putdocx textblock end
```

The `<<dd_docx_display>>` tag executes Stata's `display` command and then replaces the tag with its output. The output is formatted according to the *text\_options* available with `putdocx text`. The tag cannot contain a line break or `>>`. If you need to include `>>` in the *display\_directive*, use the symbols with a space in between (`>>`).

The `<<dd_docx_display>>` tag can be used multiple times inside a line of text. For example, say that we want to display the circumference of a circle with radius 1 up to the two digits after the decimal. Instead of computing the number and then copying and pasting the result into a block of text, we can write

```
putdocx textblock begin
2*1*<<dd_docx_display bold:%4.2f c(pi)>> = <<dd_docx_display bold:%4.2f 2*1*c(pi)>>
putdocx textblock end
```

which formats the value of  $\pi$  and the product in bold and produces the following in the `.docx` file being created.

```
2*1*3.14 = 6.28
```

For another example demonstrating the use of this dynamic tag, see [Working with blocks of text](#) in [\[RPT\] putdocx paragraph](#).

## Export and include a Stata graph

```
<<dd_graph: attribute>>
```

The `<<dd_graph>>` tag exports a Stata graph and then includes a link to the exported image file in the target file.

<i>attribute</i>	Description
<code>saving(filename)</code>	export graph to <i>filename</i>
<code>replace</code>	replace the file if it already exists
<code>graphname(name)</code>	name of graph to be exported
<code>svg</code>	export graph as SVG
<code>png</code>	export graph as PNG
<code>pdf</code>	export graph as PDF
<code>eps</code>	export graph as EPS
<code>ps</code>	export graph as PS
<code>html</code>	output an HTML link
<code>markdown</code>	output a Markdown link; default is <code>html</code>
<code>pathonly</code>	output the path of the file; default is <code>html</code>
<code>alt(text)</code>	alternative text for the graph to be read by voice software; ignored if <code>pathonly</code> in effect
<code>height(#)</code>	height in pixels of the graph in HTML; ignored if <code>markdown</code> or <code>pathonly</code> in effect
<code>width(#)</code>	width in pixels of the graph in HTML; ignored if <code>markdown</code> or <code>pathonly</code> in effect
<code>relative</code>	use file path relative to the <i>targetfile</i> path specified in <code>dyndoc</code> or <code>dyntext</code> ; this is the default
<code>absolute</code>	use absolute path in the link; default is <code>relative</code>
<code>basepath(path)</code>	use <i>path</i> as base directory where graph files will be exported; default is the current working directory if it is not specified
<code>nourlencode</code>	do not encode the path to a percent-encoded URL; ignored if <code>html</code> or <code>markdown</code> in effect

If `graphname(name)` is not specified, the topmost graph is used. You can use the default name “Graph” to export the graph without the name.

For paths specified in the `saving()` or `basepath()` attributes, a single backslash (\) is interpreted as an escape character rather than as the directory separator character. When working on Windows, we recommend using a forward slash (/) as the directory separator character (for example, `C:/mypath/myfile`); otherwise, you must use a double backslash (for example, `C:\\mypath\\myfile`).

If `saving(filename)` is not specified, a filename will be constructed based on the graph name.

If none of `.svg`, `.png`, or `.pdf` is specified, the `saving(filename)` is checked first; if the name specified in `saving(filename)` has the extension of `.svg`, `.png`, or `.pdf`, then the graph will be exported in the format corresponding to the extension. For example, the dynamic tag

```
<<dd_graph:saving(gr1.png) graphname(gr1)>>
```

produces

```

```

Otherwise, the type `.svg` will be used as in

```
<<dd_graph:saving(gr1.pgg) graphname(gr1)>>
```

which produces

```

```

If `markdown` is specified, a Markdown link will be produced. For example, the dynamic tag

```
<<dd_graph:saving(gr1.svg) graphname(gr1) markdown>>
```

produces

```

```

You may use `pathonly` if you want an HTML link with more attributes than `html` or `markdown` can provide or if you want to use the path in a different target file type such as `LATEX`.

By default, the path is outputted as a percent-encoded URL. For example, the dynamic tag

```
<<dd_graph:saving("gr 1.svg") graphname(gr1) pathonly>>
```

produces

```
gr%201.svg
```

You may use `nourlencode` to disable the encoding process as in

```
"<<dd_graph:saving("gr 1.svg") graphname(gr1) pathonly nourlencode>>"
```

which produces

```
"gr 1.svg"
```

The `<<dd_graph>>` tag can be used inside a line of text.

### Include a text file

```
<<dd_include: filename>>
```

The `<<dd_include>>` tag replaces the tag with the contents of the specified text file. The text file is included as is. The tag must be at the beginning of a new line. The *filename* itself may contain Stata macros, but not the file contents.

### Disable dynamic text processing

```
<<dd_ignore>> and <</dd_ignore>>
```

The `<<dd_ignore>>` tag causes `dyntext` and `dyndoc` to ignore the dynamic tag processing, starting from the next line until the line right before a `<</dd_ignore>>` tag. Both the beginning and ending tags must be at the beginning of a line. The only tag it does not affect is the `<<dd_remove>>` tag.

## Process contents based on condition

```
<<dd_if: Stata expression>>
lines of text ...
<<dd_endif>>
```

or

```
<<dd_if: Stata expression>>
lines of text ...
<<dd_else>>
lines of text ...
<<dd_endif>>
```

<<dd\_if: *Stata expression*>> evaluates the *Stata expression*; if it evaluates to true (anything but 0 or "0"), the lines before the next <<dd\_endif>> are processed. If there is a <<dd\_else>>, the lines before <<dd\_else>> are processed, and the lines between <<dd\_else>> and <<dd\_endif>> are skipped.

If the *Stata expression* evaluates to false (0 or "0"), the lines before the next <<dd\_endif>> are skipped. If there is a <<dd\_else>>, the lines before <<dd\_else>> are skipped, and the lines between <<dd\_else>> and <<dd\_endif>> are processed.

## Skip contents based on condition

```
<<dd_skip_if: Stata expression>>
lines of text ...
<<dd_skip_end>>
```

or

```
<<dd_skip_if: Stata expression>>
lines of text ...
<<dd_skip_else>>
lines of text ...
<<dd_skip_end>>
```

<<dd\_skip\_if: *Stata expression*>> evaluates the *Stata expression*; if it evaluates to true (anything but 0), the lines before the next <<dd\_skip\_end>> are skipped. If there is a <<dd\_skip\_else>>, the lines before <<dd\_skip\_else>> are skipped, and the lines between <<dd\_skip\_else>> and <<dd\_skip\_end>> are processed as usual.

If the *Stata expression* evaluates to false (0), the lines before the next <<dd\_skip\_end>> are not skipped. If there is a <<dd\_skip\_else>>, the lines before <<dd\_skip\_else>> are not skipped, and the lines between <<dd\_skip\_else>> and <<dd\_skip\_end>> are skipped.

## Remove contents

```
... <<dd_remove>>text to remove ...
lines of text to remove ...
text to remove ... <</dd_remove>> ...
```

The <<dd\_remove>> and <</dd\_remove>> tags remove all the contents between the two tags from the resulting target file. The tags can be used inside a line of text.

<<dd\_remove>> is a postprocessing tag, which means it is processed after all other tags.

## Also see

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

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

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