

Package: Statamarkdown (via r-universe)

September 23, 2026

Title 'Stata' Markdown

Version 1.1.0

Description Settings and functions to extend the 'knitr' 'Stata' engine.

License MIT + file LICENSE

URL <https://github.com/Hemken/Statamarkdown/>,
<https://hemken.github.io/Statamarkdown/>

BugReports <https://github.com/Hemken/Statamarkdown/issues>

Imports knitr (>= 1.46), xfun (>= 0.39)

Suggests markdown, quarto, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder quarto

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

Encoding UTF-8

SystemRequirements Stata

Repository <https://remlapmot.r-universe.dev>

Date/Publication 2026-09-23 11:23:27 UTC

RemoteUrl <https://github.com/Hemken/Statamarkdown>

RemoteRef HEAD

RemoteSha 303ff2707c28f51321ec2cafade71c6f709ee5ae

Contents

Statamarkdown-package	2
find_stata	4
purl_stata	5
spinstata	7
stata_engine	9

Index	11
--------------	-----------

Statamarkdown-package *Statamarkdown: Settings and functions to extend the knitr Stata engine*

Description

To use these functions and settings, attach the **Statamarkdown** library from *within* the document to be knit. A typical preliminary code chunk in a document would be

```
```{r setup, include=FALSE}
library(Statamarkdown)
```
```

Details

Using the "Stata" language engine in **knitr** has a number of limitations. Each Stata code chunk is run as a separate batch file, and source code is part of the output returned to the document being knit. This package provides a language engine with code chunk options to overcome these limitations.

Multiple documents can be rendered from the same script or R session; the engine re-establishes the Stata executable path and the collectcode hook for each document. (In versions upto 0.9.7 you had to detach("package:Statamarkdown") in between documents.)

Code Block (Chunk) Options

Statamarkdown Chunk Options:

collectcode (logical):

A function here sets up a chunk hook, that silently repeats selected code chunks at the beginning of later code chunks. This allows the code in one chunk to use the results of a previous chunk. The user marks code chunks to be silently repeated with the chunk option collectcode=TRUE.

cleanlog (logical):

A second function here sets up an output hook. This removes Stata code from the output by default. To leave Stata commands in the output, specify the chunk option cleanlog=FALSE.

savedo (logical):

To save the code from a code block (as a "do" file) and also to save the Stata log file produced by that code block, specify chunk option savedo=TRUE. The filenames are the same as the chunk label.

Knitr Chunk Options:

eval (logical, numeric vector):

Whether or not to evaluate the code in the code block. Use eval=FALSE to show code to the reader without having it evaluated.

Selective evaluation by specifying a numeric vector (as for R code blocks) is also supported: the vector must be either all positive (evaluate only these lines) or all negative (evaluate all but these lines). Lines excluded from evaluation are commented out in the Stata do-file.

include (logical):

Whether or not any trace of this code block appears in your document. Use `include=FALSE` to evaluate code but suppress the source code echo and all output (including error messages). This is equivalent to `eval=TRUE`, `echo=FALSE`, `results="hide"`, `error=FALSE`.

echo (logical, numeric vector):

Whether or not to show the reader the source code. Use `echo=FALSE` to suppress the source code in your document.

If this is specified as a numeric vector, it indicates which source lines to show or suppress. For example, `echo=c(1,2)` shows only the first two lines of the code block in the document (while still evaluating the entire code block). Likewise, `echo=-1` hides just the first line of code from the reader.

results (character):

To suppress normal output while still showing error messages use `results="hide"`.

error (logical):

Whether or not to show error messages in your document. To suppress error messages use `error=FALSE`.

Error messages that Stata writes to the log will appear as normal output - they are not "errors" in this context. This option affects error messages returned to/by the operating system.

comment (character):

A prefix to use before lines of output. The default for R output is `comment="##"`

child (character):

Filename to be run and input in the document.

Author(s)

Maintainer: Tom Palmer <remlapmot@hotmail.com> ([ORCID](#)) (MacOS, linux)

Authors:

- Tom Palmer <remlapmot@hotmail.com> ([ORCID](#)) (MacOS, linux)
- Doug Hemken <d_hemken@yahoo.com> (SSCC, Univ. of Wisconsin-Madison (retired))

Other contributors:

- Philipp Lepert [contributor]

References

More documentation and examples: <https://www.ssc.wisc.edu/~hemken/Stataworkshops/stata.html#stata-and-r-markdown>

See Also

The package that this extends: [knitr](#).

| | |
|------------|------------------------------------|
| find_stata | <i>Locate the Stata executable</i> |
|------------|------------------------------------|

Description

A helper function that seeks to locate your Stata executable. Ordinarily this is run automatically when **Statamarkdown** is loaded.

Usage

```
find_stata(message = TRUE)
```

Arguments

| | |
|---------|--|
| message | (logical) Whether or not to print messages: where Stata was found, or that no Stata executable could be found. |
|---------|--|

Details

This function searches for recent versions of Stata ($\geq$ Stata 11), in some of the usual default installation locations.

If Stata is not found, you will have to specify its correct location yourself.

Value

A character string with the path and name of the Stata executable.

Author(s)

Doug Hemken

See Also

[Statamarkdown-package](#)

Examples

```
indoc <- '
# An R console example
## In a first code chunk, set up with
```{r}
library(Statamarkdown)
```

## Then mark Stata code chunks with
```{stata}
sysuse auto, clear
generate gpm = 1/mpg
summarize price gpm
```

```

...
,

if (nzchar(Statamarkdown::find_stata(message = FALSE)) &&
 requireNamespace("rmarkdown", quietly = TRUE)) {
 # To run this example, remove tempdir().
 frmd <- file.path(tempdir(), "test.Rmd")
 fhtml <- file.path(tempdir(), "test.html")

 # Knit and render in a fresh R process, so that stale knitr state in a
 # long-running session (e.g. from RStudio's "Run examples" button)
 # cannot interfere with how the document text is parsed.
 xfun::Rscript_call(
 function(indoc, frmd, fhtml) {
 writelines(indoc, frmd)
 rmarkdown::render(frmd, "html_document", fhtml)
 },
 args = list(indoc, frmd, fhtml)
)
 message("HTML output created at: ", fhtml)
 if (interactive()) {
 # Show in the RStudio Viewer pane if available, otherwise the browser
 viewer <- getOption("viewer", default = utils::browseURL)
 viewer(fhtml)
 }
}

```

purl\_stata

*Extract Stata code from a dynamic document***Description**

The Stata analogue of `knitr::purl()`: extracts the code from the Stata code chunks of an R Markdown or Quarto document and writes it to a Stata do-file.

**Usage**

```
purl_stata(input, output = NULL, text = NULL, documentation = 1L)
```

**Arguments**

input	A character string with the name of the input document.
output	A character string with the name of the do-file to write. Defaults to the name of the input document with its extension changed to <code>.do</code> .
text	A character string with the document text to use in place of a file.
documentation	How much documentation to carry into the do-file, following <code>knitr::purl()</code> : <code>0</code> (or <code>FALSE</code> ) extracts the code only; <code>1</code> (or <code>TRUE</code> , the default) precedes the code of each chunk with a Stata comment giving the chunk's header (its label and

options), in the `*** label ----` form knitr uses when it tangles a script in another language; 2 also includes the document's text as Stata comments (the code of non-Stata chunks is not included).

## Details

Chunks are recognised with knitr's own chunk patterns, so indented chunks and fences of more than three backticks are handled. A chunk is extracted when its header engine is `stata`, or when it uses the older `r` chunk form with an `engine = "stata"` option. Chunks with the `purl = FALSE` or `eval = FALSE` options (either in the chunk header or in option comments) are skipped. Option comments are split from the code by `knitr::partition_chunk()`, so the forms knitr itself recognises are honoured: `*|` in a `stata` chunk and `#|` in a chunk using the older `engine='stata'` form. They are never copied into the do-file as code, but with `documentation >= 1` they are recorded as plain Stata comments below the chunk header line.

## Value

If a do-file is written, the path to the do-file, invisibly. If text is given and output is `NULL`, a character vector of the extracted lines.

## See Also

[knitr::purl\(\)](#), [Statamarkdown-package](#)

## Examples

```
indoc <- '
Some text.

```{r}
library(Statamarkdown)
```

```{stata first-Stata, collectcode=TRUE}
sysuse auto, clear
generate gpm = 1/mpg
```

```{stata second-Stata}
regress price gpm
```
'
purl_stata(text = indoc)
```

spinstata

*Convert a specially marked up Stata "do" file to Markdown and HTML***Description**

This function takes a Stata file containing special markup in its comments, and converts it to Markdown and HTML documents (or one of several other formats).

**Usage**

```
spinstata(statafile, text = NULL, keep = FALSE, ...)
```

**Arguments**

|           |                                                                                                                                        |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------|
| statafile | A character string with the name of a Stata "do" file, containing markup in its comments.                                              |
| text      | A character string in place of a file.                                                                                                 |
| keep      | Whether to save intermediate files.                                                                                                    |
| ...       | options passed to <code>knitr::spin()</code> , for example format (the output format, such as "Rmd" or "qmd"), knitr, report or envir. |

**Details**

This function takes a Stata file containing special markup in its comments, and converts it into knitr's "spin" format. This is in turn sent to `knitr::spin()`, and converted to Markdown and HTML (or one of several other formats).

Special Markup:

- `"/* ' "` - Begin document text, ends with `" ' */`
- `"/*+ "` - Begin chunk header, ends with `"*+ /`
- `"/*R "` - Begin a chunk of R code, ends with `"R* /`
- `"/** "` - Dropped from document, ends with `"* /`

Code in a chunk is taken to be Stata code, and the chunk header is given the `engine='stata'` chunk option, unless the code is marked as R code with the `"/*R ... R* /` markup, or the chunk header sets an engine option itself. Writing `engine='stata'` in the chunk headers of a "do" file is therefore no longer necessary, but it is still honoured.

With knitr  $\geq$  1.53, Stata is also used for code which is not preceded by a chunk header at all, in a document which has no chunks of R code.

**Value**

The path to the output file.

If given text instead of a file, returns the compiled document as a character string.

**Author(s)**

Doug Hemken

**See Also**

[knitr::spin\(\)](#), [Statamarkdown-package](#)

**Examples**

```
indoc <- "/*'
Statamarkdown Example

This is a special Stata script which can be used to generate a report.
You can write normal text in command-style comments.

First we load Statamarkdown.
'*/

/*+ setup +*/
/*R
library(Statamarkdown)
R*/

/*' The report begins here. '*/

/*+ example1 +*/
sysuse auto
/* Stata comment */
summarize

/*' You can use the ***usual*** Markdown to mark up text.'*/
"

if (nzchar(Statamarkdown::find_stata(message = FALSE)) &&
 requireNamespace("markdown", quietly = TRUE)) {
 # To run this example, remove tempdir().
 fhtml <- file.path(tempdir(), "test.html")
 # Spin in a fresh R process, so that stale knitr state in a
 # long-running session (e.g. from RStudio's "Run examples" button)
 # cannot interfere with how the document text is parsed.
 x <- xfun::Rscript_call(
 function(indoc) Statamarkdown::spinstata(text = indoc),
 args = list(indoc)
)
 writeLines(x, fhtml)
 message("HTML output created at: ", fhtml)
 if (interactive()) {
 # Show in the RStudio Viewer pane if available, otherwise the browser
 viewer <- getOption("viewer", default = utils::browseURL)
 viewer(fhtml)
 }
}
```

---

|              |                                        |
|--------------|----------------------------------------|
| stata_engine | <i>Define a Stata engine for knitr</i> |
|--------------|----------------------------------------|

---

## Description

This function creates a modified Stata engine.

## Usage

```
stata_engine(options)
```

## Arguments

|         |                                                                                                |
|---------|------------------------------------------------------------------------------------------------|
| options | Chunk options, passed to the engine function when it is actually invoked within <b>knitr</b> . |
|---------|------------------------------------------------------------------------------------------------|

## Details

Set up once per session (i.e. document). Ordinarily this is run automatically when **Statamarkdown** is loaded.

`stata_engine(options)` is a language engine that returns Stata log output. The end user should not need to use the language engine function directly. This is the workhorse function that actually calls Stata and returns output.

## Value

The language engine function returns Stata code and output internally to **knitr**.

## Including Stata graphs

Setting the chunk option `stata.fig=TRUE` exports the graph drawn by the chunk (Stata's current graph) to a figure file, and includes it in the output document. The figure is laid out by knitr's usual plot machinery, so the standard figure chunk options apply, including `fig.cap` (the figure caption), `fig.alt` (the alternative text, for accessibility; falling back to `fig.cap` if unset), `out.width`, `out.height`, `fig.align`, `fig.link` and `fig.path`.

The export format is controlled with the `stata.fig.format` chunk option, and defaults to "svg", which Stata can export on all platforms in batch mode (including console Stata on Linux, which cannot export PNG). For PDF/LaTeX output set, for example, `stata.fig.format="pdf"`.

The hyphenated option spellings `stata-fig` and `stata-fig-format` are also accepted, matching Quarto's option naming convention (as in `fig-cap` and `fig-alt`). These work in YAML-style option comments (`#|` or `*|` lines at the start of the chunk, in either R Markdown or Quarto documents), but not in the chunk header's comma-separated syntax, where a hyphenated name is not valid R.

Note that knitr's `fig.width`, `fig.height` and `dpi` options control R's graphics devices and have no effect on Stata graphs; set the graph size in Stata, for example with the `xsize()` and `ysize()` options to `graph display`. One graph is exported per chunk; to include several graphs, draw them in separate chunks (using `collectcode=TRUE` to carry the data over).

**Author(s)**

Doug Hemken

**See Also**[knitr::knit\\_engines](#)**Examples**

```

indoc <- '
An R console example
In a first code chunk, set up with
```{r}
library(Statamarkdown)
```

Then mark Stata code chunks with
```{stata}
sysuse auto, clear
generate gpm = 1/mpg
summarize price gpm
```

,

if (nzchar(Statamarkdown::find_stata(message = FALSE)) &&
 requireNamespace("rmarkdown", quietly = TRUE)) {
 # To run this example, remove tempdir().
 frmd <- file.path(tempdir(), "test.Rmd")
 fhtml <- file.path(tempdir(), "test.html")

 # Knit and render in a fresh R process, so that stale knitr state in a
 # long-running session (e.g. from RStudio's "Run examples" button)
 # cannot interfere with how the document text is parsed.
 xfun::Rscript_call(
 function(indoc, frmd, fhtml) {
 writeLines(indoc, frmd)
 rmarkdown::render(frmd, "html_document", fhtml)
 },
 args = list(indoc, frmd, fhtml)
)
 message("HTML output created at: ", fhtml)
 if (interactive()) {
 # Show in the RStudio Viewer pane if available, otherwise the browser
 viewer <- getOption("viewer", default = utils::browseURL)
 viewer(fhtml)
 }
}

```

# Index

`find_stata`, [4](#)

`knitr`, [3](#)

`knitr::knit_engines`, [10](#)

`knitr::partition_chunk()`, [6](#)

`knitr::purl()`, [5](#), [6](#)

`knitr::spin()`, [7](#), [8](#)

`purl_stata`, [5](#)

`spinstata`, [7](#)

`stata_engine`, [9](#)

`Statamarkdown (Statamarkdown-package)`, [2](#)

`Statamarkdown-package`, [2](#), [4](#), [6](#), [8](#)