The Stata Journal (2020)
20, Number 2, pp. 336–362 DOI: 10.1177/1536867X20931000

Software documentation with markdoc 5.0

E. F. Haghish
Department of Medical Psychology and Medical Sociology
University of Göttingen
Göttingen, Germany
haghish@med.uni-goettingen.de

Abstract. `markdoc` is a general-purpose literate programming package for generating dynamic documents, dynamic presentation slides, Stata help files, and package vignettes in various formats. In this article, I introduce `markdoc` version 5.0, which performs independently of any third-party software, using the `mini` engine. The `mini` engine is a lightweight alternative to Pandoc (MacFarlane [2006, https://pandoc.org/]), completely written in Stata. I also propose a procedure for remodeling package documentation and data documentation in Stata and present a tutorial for generating help files, package vignettes, and GitHub Wiki documentation using `markdoc`.

Keywords: pr0072, markdoc, mini, Pandoc, statistics software, software documentation, literate programming, social coding

1 Introduction

Writing good documentation is one of the oldest recommendations of software development (Walsh 1969). However, despite the importance of software documentation (Vasilescu et al. 2014; Sousa and Moreira 1998) and the time needed to write and update it (de Souza, Anquetil, and de Oliveira 2005), no author earns any admiration or credits for the endeavor (Brown 1974). Therefore, programs that facilitate writing and updating documentation, such as Javadoc (Kramer 1999; Leslie 2002) and Doxygen (van Heesch 1997), are favorable. These programs implement a procedure called literate programming (Knuth 1992; Cordes and Brown 1991; Ramsey and Marceau 1991; Leisch 2002). In this approach, the documentation is written within code files using special comment signs. Subsequently, anytime the code is changed, the documentation can also be updated within the same file. Next, a program extracts and renders the documentation and updates the documents (Knuth 1983). For statistical software developed with R, `roxygen2` (Wickham, Danenberg, and Eugster 2019) mimics a similar approach to generate R documentation files. For Stata, `markdoc` offers similar capabilities (Haghish 2016b,c) that are addressed in this article.

For publishing statistical packages on the Comprehensive R Archive Network (CRAN) and the Boston College Statistical Software Components archive, you must document individual functions (R Core Team 2006; Leisch 2008; Baum 2011). However, a statistical package might include several help files, each corresponding to an individual function. Therefore, a long-form package vignette, which not only includes the help files but also adds a detailed description of the package with a tutorial, can be rewarding (Wickham 2015).

In this article, I provide a tutorial on how to use `markdoc` to write and update software documentation in various file formats (`.sthlp`, `.pdf`, `.html`, and `.docx`) from the same documentation source. Moreover, I demonstrate how individual help files can be organized and combined to generate a package vignette or GitHub Wiki documentation within the Stata command line. Finally, I introduce the new features of `markdoc` 5.0 that make it independent of any third-party software.

Proposing an agenda for remodeling software documentation implies that this article is chiefly aimed at advanced users, who are accustomed to Stata programming and package development. Moreover, to escape repetition, I assume the reader is familiar with the syntax and workflow of `markdoc` 4.0, as well as its journal article (Haghish 2016d). Supplementary documentation can also be found on the `markdoc` Wiki.[1]

2 markdoc 5.0

`markdoc` is a general-purpose literate programming package for Stata. It implements a literate programming procedure for writing documentation within Stata code and includes a multipurpose engine that supports generating Stata help files, dynamic documents, and dynamic presentation slides, using the same documentation source. `markdoc` 4.0 was published in 2016, with a tutorial for writing dynamic analysis documents in Stata. In this section, I introduce the new features of `markdoc` 5.0 and recap the essential points that are relevant to this article.

`markdoc` 5.0 is accompanied by a new lightweight engine—called `mini`—that will free the package of any third-party software such as Pandoc (MacFarlane 2006) or wkhtmltopdf (Kulkarni and Truelsen 2015). The `mini` engine is optional and requires Stata 15 or above. As shown in figure 1, the `mini` engine offers the same versatility and flexibility for generating dynamic documents, dynamic presentation slides, help files, and package vignettes in various formats. Therefore, it allows `markdoc` to be fully functional on restricted machines or servers, where the user lacks administrative privileges to install third-party software.

1. https://github.com/haghish/markdoc/wiki

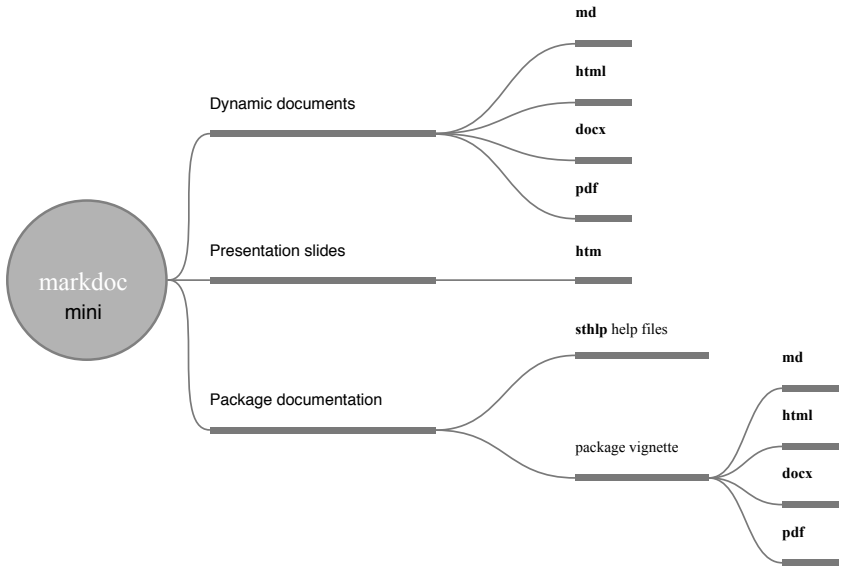


Figure 1. Documents supported by the `mini` engine

2.1 Supported markup languages

A full installation of `markdoc` with its dependencies supports three notation markup languages: Markdown (Gruber 2004), HTML, and LaTeX. The `mini` engine, however, supports only Markdown (see section 3.1 for exception).

2.2 Installation

`markdoc` is hosted on GitHub and should be installed using the `github` command (Haghish 2016a, Forthcoming), which is a powerful tool for installing and managing Stata packages hosted on GitHub with their Stata dependencies. To install `github` modules, type

```
. net install github, from("https://haghish.github.io/github/")
```

Next, the latest stable release[2] of `markdoc` can be installed by typing the command below. Without specifying the `stable` option, you will install the development version (main branch) of the package:

```
. github install haghish/markdoc, stable
```

2. Stable releases are included on GitHub: https://github.com/haghish/markdoc/releases.

Dependencies

After you install the `markdoc` repository, the `github install` command installs three dependency packages, which are specified in a file named `dependency.do`.[3] The dependency packages are `weaver` (Haghish 2016e), `md2smcl` (Haghish 2016c), and `datadoc` (Haghish 2019a). The `weaver` package includes the `txt`, `img`, and `tbl` commands, used for writing dynamic text, capturing and adding figures, and creating dynamic tables, respectively. The `md2smcl` package, as the name suggests, converts Markdown to Stata Markup and Control Language (SMCL), which is needed for generating Stata help files. The `datadoc` command produces a help file template for a dataset that is currently loaded in Stata (see section 3.2).

2.3 Syntax

The syntax and procedure of `markdoc` 5.0 remain identical to those of version 4.0, which can be summarized as follows. Table 1 shows the options that are used throughout the examples of this article.

`markdoc` *filename* [`,` *options*]

Table 1. The essential options of the `markdoc` command

options	Description
`mini`	run `markdoc` independent of any third-party software
`install`	install Pandoc and wkhtmltopdf software, if not found
`export(name)`	format; *name* can be `docx`, `pdf`, `html`, `sthlp`, `slide`, `md`, or `tex`
`replace`	replace exported document, if exists
`statax`	activate `statax` (Haghish 2019b) syntax highlighter
`helplayout`	append a Markdown help layout template to a script file

2.4 Initiating the mini program

As noted earlier, `mini` is an alternative to Pandoc. There are a few ways to ask `markdoc` to use `mini` for document conversion instead of Pandoc. The easiest way is to launch the graphical user interface created for the lightweight engine by typing

```
. db mini
```

3. Declaring dependencies within the `dependency.do` file is the standard procedure for the `github` package.

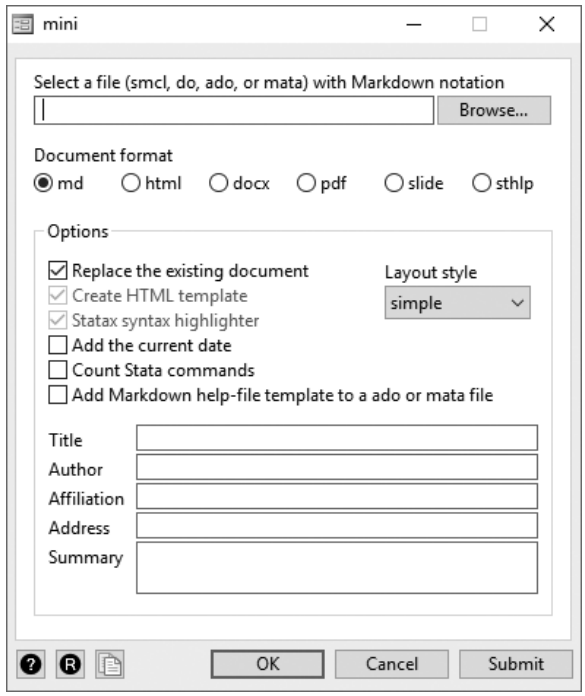


Figure 2. The `mini` graphical user interface

Alternatively, as shown in table 1, the `mini` option can be passed to `markdoc`:

`markdoc` *filename* [`, mini` ...]

The `mini` engine can also be called independently to convert a Markdown file to any of the supported formats (`.sthlp`, `.html`, `.docx`, `.pdf`, and `.slide`). This enables other Stata packages to call `mini` to convert a Markdown file to other document formats. Because `mini` was written for `markdoc`, it takes the same options as `markdoc` (see `help mini` for more details).

`mini` *filename* [`, export(`*name*`)` ...]

For example, if we have a Markdown file named `markdown.md`[4], we can convert it to a help file or PDF file within Stata:

```
. mini markdown.md, export(sthlp)
. mini markdown.md, export(pdf)
```

4. For example, see https://github.com/haghish/markdoc/wiki/markdown.

2.5 Package structure and workflow

To make `markdoc` a general-purpose package, I designed two separate workflows. The active procedure executes the do-file in a clean workspace[5] to examine the reproducibility of the code and generate a dynamic analysis document. In contrast, the passive procedure extracts only the documentation from a Stata file, converts the notation, and generates a document. The workflow, as shown in figure 3, is chosen automatically based on the given file extension.

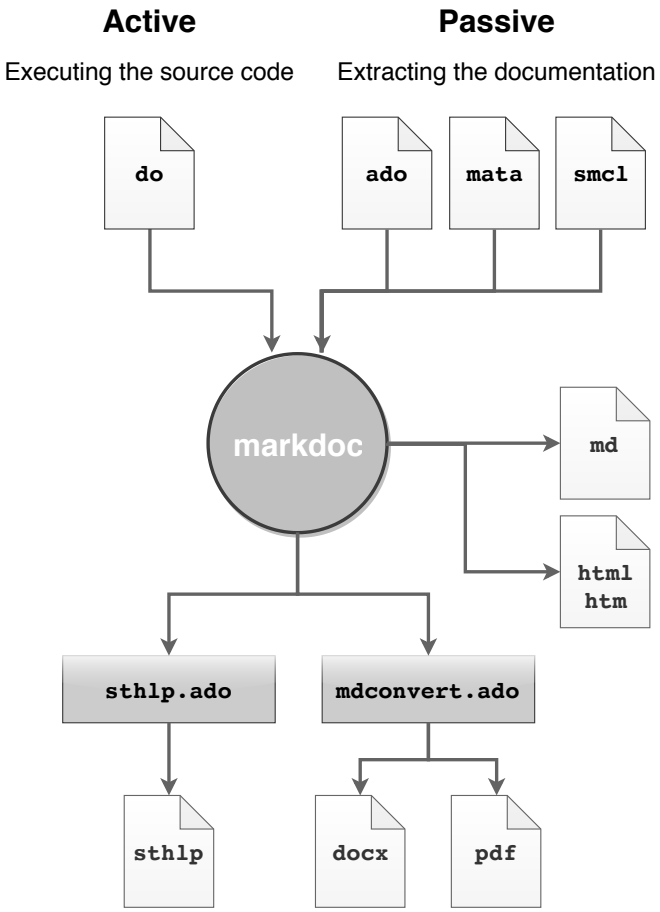


Figure 3. Structure and workflow of the `mini` engine

5. When a do-file is executed, `markdoc` ignores the currently loaded dataset, ensuring that the script file is reproducible.

2.6 Example

The example below is borrowed from `markdoc`'s (Haghish 2016b) publication and is executed with the `mini` engine. The example demonstrates using dynamic text—specified within the `<!`*scalar*`!>` marker—and a dynamic table.

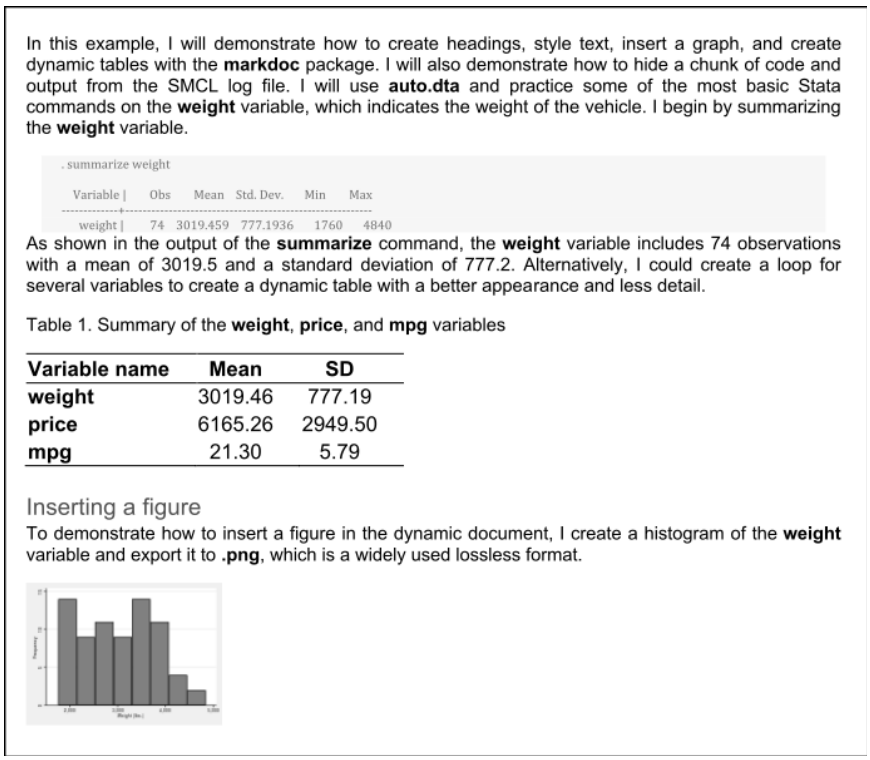
▷ Example

```
//--------------------  beginning of the example1.do  --------------------
/***
In this example, I will demonstrate how to create headings, style text, insert
a graph, and create dynamic tables with the __markdoc__ package. I will also
demonstrate how to hide a chunk of code and output from the SMCL log file.
I will use __auto.dta__ and practice some of the most basic Stata commands
on the __weight__ variable, which indicates the weight of the vehicle. I
begin by summarizing the __weight__ variable.
***/

//OFF
sysuse auto, clear
histogram weight, frequency scheme(sj)
quietly graph export graph.png, width(130) replace
//ON

summarize weight

/***
As shown in the output of the __summarize__ command, the __weight__ variable
includes <!r(N)!> observations with a mean of <!r(mean)!> and a standard
deviation of <!r(sd)!>. Alternatively, I could create a loop for several
variables to create a dynamic table with a better appearance and less detail.
***/

//OFF
foreach var of varlist weight price mpg {
      summarize `var´
      local `var´_mean : display %9.2f r(mean)
      local `var´_sd   : display %9.2f r(sd)
}
//ON

tbl ("Variable Name", "Mean", "SD" \                                    ///
     "__weight__", `weight_mean´, `weight_sd´ \                         ///
     "__price__", `price_mean´, `price_sd´ \                            ///
     "__mpg__", `mpg_mean´, `mpg_sd´),                                  ///
     title("Table 1. Summary of __weight__, __price__, and "            ///
     "__mpg__ variables")

/***
Inserting a figure
------------------

To demonstrate how to insert a figure in the dynamic document, I create
a histogram of the __weight__ variable and export it to __.png__,
which is a widely used lossless format.

![Figure 1. Distribution of the __weight__ variable](graph.png)
***/
//--------------------  end of the example1.do  -------------------------
```

◁

This example is prepared for the active procedure; that is, `markdoc` executes a do-file to generate the analysis report. If the example is saved in `example1.do`, typing the command below will execute the do-file, test its reproducibility, and generate a Word document, independent of any third-party software.

```
. markdoc example1.do, mini export(docx) replace
```



In this example, I will demonstrate how to create headings, style text, insert a graph, and create dynamic tables with the **markdoc** package. I will also demonstrate how to hide a chunk of code and output from the SMCL log file. I will use **auto.dta** and practice some of the most basic Stata commands on the **weight** variable, which indicates the weight of the vehicle. I begin by summarizing the **weight** variable.

```
. summarize weight

    Variable |     Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
      weight |      74    3019.459    777.1936       1760       4840
```

As shown in the output of the **summarize** command, the **weight** variable includes 74 observations with a mean of 3019.5 and a standard deviation of 777.2. Alternatively, I could create a loop for several variables to create a dynamic table with a better appearance and less detail.

Table 1. Summary of the **weight**, **price**, and **mpg** variables

Variable name	Mean	SD
weight	3019.46	777.19
price	6165.26	2949.50
mpg	21.30	5.79

Inserting a figure

To demonstrate how to insert a figure in the dynamic document, I create a histogram of the **weight** variable and export it to **.png**, which is a widely used lossless format.

Figure 4. The `example1.do` output from the Word file

The passive procedure begins by initiating a SMCL log file. Then, the log file is given to `markdoc` to convert it to any format. This procedure is shown in the example below, which produces HTML-based slides within Stata. In addition to the `mini` option, `markdoc` also utilizes the `statax` syntax highlighter for Stata code.

Example

```
//--------------------  beginning of the example.do  --------------------
quietly log using "passive", replace

/***
Passive procedure
=================

The passive procedure does not ensure reproducibility of the code. However, it
provides a convenient procedure to examine your report as you write it interactively.

---

To add a new slide, apply the Markdown´s horizontal line syntax, as shown below.

---

Writing dynamic content
=======================

The __txt__ and __tbl__ commands can be used to write dynamic text or create
dynamic tables respectively. The __img__ command or Markdown syntax can also
be used to add a figure in the report. The example below demonstrates using the
__txt__ command to write dynamic text.
***/

summarize price
return list
txt "There are " r(N) " rows in the dataset. "

// calling markdoc to generate the slides from the log file
qui log c
markdoc passive, mini export(slide) statax replace
//--------------------  end of the example.do  --------------------------
```

Limitations of mini

For generating the Word documents and PDF files, the `mini` program uses two sets of Stata commands, which are `putdocx` and `putpdf`. Therefore, it is bound by their limitations as well. For Stata 15, the following limitations can restrict Word documents and PDF files:

1. generating a hyperlink,
2. styling options within each cell of the table,
3. creating an ordered or unordered lists, and
4. drawing a horizontal line.

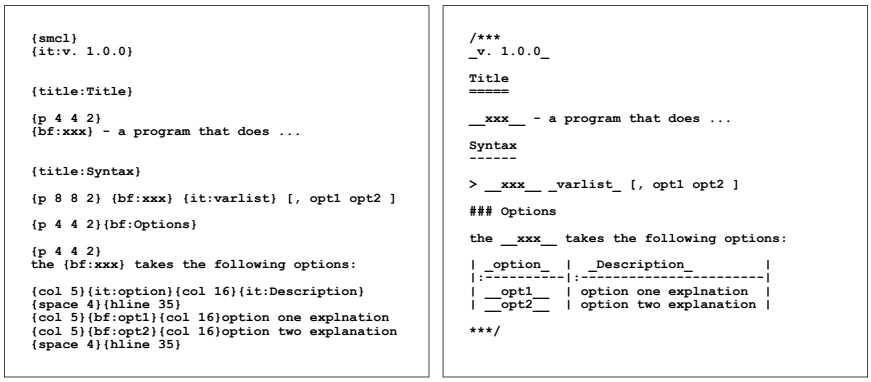
In Stata 16, however, these limitations are no longer relevant. Consequently, the `mini` program was updated to take advantage of the new features of Stata 16. For Stata 15 users, the `html` and `docx` formats are recommended.

3 Writing package documentation

When applying the literate programming paradigm, you can write software documentation within Stata script files using simplified notations such as Markdown (see section A.1 in the appendix for Markdown syntax reference). Compared with writing software documentation with SMCL, writing it with Markdown offers three main advantages:

1. Writing the documentation within the script files allows updating the documentation as soon as a change is made in the program, which simplifies updating the documentation.

2. Compared with Markdown, the SMCL markup looks rather complex (see figure 5), which makes reading and writing the documentation difficult.



```
{smcl}
{it:v. 1.0.0}


{title:Title}

{p 4 4 2}
{bf:xxx} - a program that does ...


{title:Syntax}

{p 8 8 2} {bf:xxx} {it:varlist} [, opt1 opt2 ]

{p 4 4 2}{bf:Options}

{p 4 4 2}
the {bf:xxx} takes the following options:

{col 5}{it:option}{col 16}{it:Description}
{space 4}{hline 35}
{col 5}{bf:opt1}{col 16}option one explnation
{col 5}{bf:opt2}{col 16}option two explanation
{space 4}{hline 35}
```

```
/***
_v. 1.0.0_

Title
=====

__xxx__ - a program that does ...

Syntax
------

> __xxx__ _varlist_ [, opt1 opt2 ]

### Options

the __xxx__ takes the following options:

| _option_  |  _Description_          |
|:----------|:------------------------|
| __opt1__  | option one explnation   |
| __opt2__  | option two explanation  |

***/
```

Figure 5. From left: SMCL documentation (`example2.sthlp`) and its Markdown source notation (`example2.ado`)

3. With Markdown notation, the documentation can be converted not only to Stata help files but also to a variety of other formats, facilitating the project dissemination.

The procedure for writing Markdown documentation for help files in `markdoc` is identical to writing dynamic documents. The documentation text is written within special comment blocks in the ado-file or Mata file, starting with `/***` and ending with `***/` signs, each on a separate line. There is no limit on how many times such notation blocks can be used throughout the script file, although writing the documentation at the outset of the script file is recommended, as shown in `example2.ado` in figure 5. The `markdoc` command can extract the documentation from `example2.ado` and generate a help file:

```
. markdoc "example2.ado", mini export(sthlp)

              v. 1.0.0


              Title
              ─────

                  xxx - a program that does ...


              Syntax
              ──────

                      xxx varlist [, opt1 opt2 ]

                  Options

                  the xxx takes the following options:

                  option     Description
                  ──────────────────────────────────
                  opt1       option one explnation
                  opt2       option two explanation
                  ──────────────────────────────────
```

Figure 6. The help file extracted from `example2.ado`

3.1 Writing with Markdown and SMCL

`markdoc` is capable of distinguishing Markdown from SMCL. Thus, if needed, the SMCL markup language can be used with the Markdown language concurrently. Nevertheless, when the documentation is rendered to another document format, such as HTML, the SMCL markup will dissolve into plain text. Therefore, writing documentation with a combination of SMCL and Markdown is not encouraged.

3.2 Help file templates

Talking about how documentation should be written is simpler than saying what should be included in software documentation (Briand 2003; de Souza, Anquetil, and de Oliveira 2005). Reviewing Stata's and CRAN's guidelines (R Core Team 2006) for documenting functions and programs, I present a structured documentation template in this article. The template not only serves as an example of Markdown documentation but also reminds the user of a few structural key points to consider. The template is organized in several sections, as shown below. The complete template is included in the appendix (see section A.2).

1. Declaration of the version of the software at the top of the help file
2. Title of the software, with a short description
3. Syntax of the program
4. Table summarizing the options
5. Detailed description of the command
6. Detailed description of the options
7. Technical remarks about the program, if any
8. Examples
9. Description of the scalars, matrices, etc., returned by the program
10. Acknowledgment or acknowledgments
11. Author information
12. Software license
13. Reference or references

`markdoc` can append the program documentation template to the outset of a script file, using the `helplayout` option. For example, if `example3.ado` is an empty script file, running the commands below will append the program documentation templates to the file and also render the help file, creating `example3.sthlp` (see figure 7).

```
. markdoc "example3.ado", mini export(sthlp) helplayout
```

```
v. 1.0.0

Title
─────

    commandname - explain your command briefly.  You can use simplified
    syntax to make text italic, bold, emphasized, or add hyperlink

Syntax
──────

        XXX varlist =exp [if] [in] [weight] using filename [, options]

    option              Description
    ─────────────────────────────────────────
    minabbrev           whatever does yak yak
    breakline           whatever does yak yak
    export(arg)         whatever does yak yak
    ─────────────────────────────────────────

    by is allowed; see [D] by
    fweight is allowed; weight


Description
───────────

    XXX does ... (now put in a one-short-paragraph description of the
    purpose of the command)

Options
───────

    whatever does yak yak

        Use > for additional paragraphs within and option description to
        indent the paragraph.

    2nd option etc.

Remarks
───────

    The remarks are the detailed description of the command and its nuances.
    Official documented Stata commands don't have much for remarks, because
    the remarks go in the documentation.

Example(s)
──────────

    explain what it does
        . example command
```

Figure 7. The help file template, completely written with Markdown

3.3 Data documentation template

In contrast with the Boston College Statistical Software Components archive, CRAN requires R packages to document the datasets and offers a structured template, indicating how a dataset should be documented. Such a template can be adopted for Stata as well; however, Stata offers several additional features for data documentation that are lacking in R. For example, a dataset can include a label and multiple notes; furthermore, each variable can also include several notes.

The `datadoc`[6] (Haghish 2019a) command, which is automatically installed as a dependency of `markdoc`, merges Stata documentation features within the template suggested by R and generates a documentation layout and a help file for the data loaded in Stata. The file is named after the data's name, with a `.do` extension. If no data are loaded in the memory, `datadoc` creates a data documentation template named `example.do`. After you update the created Markdown document, `markdoc` or `mini` can update the Stata help file. The Markdown documentation template for datasets is included in the appendix (section A.3). The data documentation template includes the following:

1. Title, the label of the dataset, and where it was published (package name)
2. Description
3. Format, including a table summarizing the variables' types and labels
4. Notes attached to the dataset or the variables
5. The source of the data; that is, where they are coming from
6. References, if any
7. Examples, if needed

In the example below, a few notes are added to `auto.dta` and its variables. Next, the `datadoc` command is called, which generates a do-file named `auto.do` and a help file named `auto.sthlp` (see figure 9 in the appendix).

```
. sysuse auto, clear
. notes : this data set is included in Stata 15
. notes : add another second note to the dataset
. notes price : add a note to the price variable
. notes make : add a note to the make variable
. notes weight : add a note to the weight variable

// generates auto.do template and auto.sthlp help file
. datadoc, replace

// edit the template, and then replace the Stata help file with mini
. mini "auto.do", export(sthlp) replace
```

6. See `help datadoc` for details.

4 Example

To demonstrate how a Stata package can be documented using Markdown, I created the `echo` repository on GitHub. The repository includes one ado-program named `echo.ado`, which displays the given string character in Stata and includes a few styling options to print the text in red color or to display it as bold or italic. You may fork the repository[7] as well as its Wiki[8] to inspect the documentation and follow the example. Below, the documentation of `echo.ado` is shown, which is written within the script file.

```
// documentation written for markdoc software; visit github.com/haghish/markdoc

/***
_version 1.0_

echo
====

a program that displays the given string in Stata

Syntax
------

> __echo__ _"character string"_ [, _red_ _bf_ _it_ ]

### Options

| _option_  |  _Description_              |
|:----------|:----------------------------|
| __red__   | print the text in red color |
| __bf__    | bold face text              |
| __it__    | italic face text            |

Description
-----------

__echo__ is a simple Stata program that is documented using Markdown format, in order to
facilitate software documentation, particularly on social coding sites such as GitHub.
the documentation can be extracted as _Markdown_ file for GitHub wiki or as _STHLP_ file
using [__markdoc__](https://github.com/haghish/markdoc) software.

Example
-------

Display "Hello World" in red color

     . echo "Hello World", red

Author
------

E. F. Haghish
_haghish@med.uni-goesttingen.de_
[https://github.com/haghish/echo](https://github.com/haghish/echo)

License
-------

MIT License
***/
```

7. https://github.com/haghish/echo.git
8. https://github.com/haghish/echo/wiki

Calling `markdoc` can extract the Markdown documentation and convert it to SMCL, generating a help file named `echo.sthlp`.

```
. markdoc "echo.ado", mini export(sthlp) replace
```

version 1.0

echo

a program that displays the given string in Stata

Syntax

echo *"character string"* [, *red bf it*]

Options

option	*Description*
red	print the text in red color
bf	bold face text
it	italic face text

Description

echo is a simple Stata program that is documented using Markdown format, in order to facilitate software documentation, particularly on social coding sites such as GitHub. the documentation can be extracted as *Markdown* file for GitHub wiki or as *STHLP* file using **markdoc** software.

Example

Display "Hello World" in red color

```
. echo "Hello World", red
```

Author

E. F. Haghish
haghish@med.uni-goesttingen.de
https://github.com/haghish/echo

License

MIT License

Figure 8. The `echo.sthlp` generated by `markdoc`

5 Package vignette and GitHub Wiki

Statistical packages often include several script files, which are documented separately. A holistic overview of the package, known as a vignette, can provide a fruitful overview of the package in a single document. In this section, I demonstrate how to use `markdoc` to generate a package vignette, as well as GitHub Wiki documentation.

5.1 Wiki

GitHub is a site for hosting not only source code but also software documentation called Wiki. The Wiki documentation is written with Markdown. We can use `markdoc` to generate Markdown files. For instance, in the example above, one can export the documentation written in `echo.ado` to a Markdown file by executing

```
. markdoc "echo.ado", mini export(md)
```

Next, we can move the generated Markdown files to the Wiki repository to update the documentation. To improve the Wiki repository, you should organize the generated Markdown files within a single document named `Home.md`, which is the homepage of Wiki repositories. The `Home.md` document can index and link the generated Markdown files, serving as a convenient start page for the documentation. GitHub uses double square brackets to link to pages uploaded to the Wiki repository, as shown below.

```
//----------------------  beginning of the Home.md  ----------------------
Documentation files
===================

- [[echo]]
//----------------------  end of the Home.md  --------------------------
```

5.2 Vignette

`markdoc` provides several possibilities for producing a package vignette. The easiest procedure would be the following:

1. Export Markdown documentation from each ado-file, as shown above.
2. Create a do-file that imports the generated Markdown files.
3. Typeset the prepared do-file to generate the package vignette.

The `mini` engine is capable of typesetting such a document in HTML, Word, or PDF format. A full installation of `markdoc` and its third-party dependencies would provide greater flexibility for styling the package vignette using LaTeX, with Markdown. As shown in the example below, `markdoc` distinguishes LaTeX from Markdown notations and allows additional LaTeX markup to be added to the Markdown documentation. This will allow the user to keep the LaTeX markup to the bare minimum and write most of the documentation with Markdown.

In the example below, `vignette.do` is created to write the vignette documentation. The file includes LaTeX notation for adding page breaks and partitioning the vignette.

```
//--------------------  beginning of the vignette.do  --------------------

/***
\part{Introduction}
\newpage

Summary
=======

The __echo__ package is a tutorial repository for documenting
and hosting Stata package on GitHub. ...

Installation
------------

To install the package, first install __github__ module. If
[github](https://github.com/haghish/github) is not installed,
type the following command:

. net install github, from("https://haghish.github.io/github/")

Next, install the latest stable release of the __echo__ repository:

. github install haghish/echo, stable


\newpage
\part{Documentation files}
\newpage
***/

//IMPORT echo.md

/***
\part{Tutorial}
\newpage
***/


//--------------------  end of the vignette.do  -------------------------
```

Next, the vignette can be exported by typing

```
markdoc "vignette.do", export(tex) toc replace master       ///
        title("ECHO package vignette")                      ///
        author("E. F. Haghish")                             ///
        affiliation("University of Goettingen")             ///
        address("haghish@med.uni-goettingen.de")
```

The resulting vignette PDF file[9] includes a title page and table of contents, and the vignette is ready to be included within the repository for a quick review of the entire package documentation.

9. https://github.com/haghish/echo/blob/master/vignette.pdf

❑ **Technical note**

In the example above, `echo.md` was included in the document using the `//IMPORT` *filename* command. This is one of the markers[10] recognized by `markdoc`, and it appends a text file to the main document.

❑

6 Discussion

In this article, I touched on two main topics about `markdoc`. On the one hand, I introduced new features of `markdoc` 5.0. On the other, I prepared a tutorial demonstrating how to use the package for documenting Stata programs and datasets, as well as for generating package vignettes or GitHub Wiki documentation. Below, I discuss the main points of the article.

6.1 Using markdoc without third-party software

Lab computers or laptops provided by universities often have restrictions for installing new software, which could make installing and using `markdoc` problematic. With the new release of `markdoc` and its lightweight `mini` engine, this problem is completely solved. Nevertheless, the `mini` engine is not a replacement for Pandoc, and I recommend that users install the binary dependencies when possible. A full installation of `markdoc` and its third-party dependencies provides heartwarming features, for example, including mathematical notations, changing the Word template by providing an example Word file, generating highly customizable dynamic PDF presentation slides, etc.[11]

6.2 Software documentation

As long as software is intended to be used by someone other than its programmer, there is a need for a user manual. However, writing such a document, and particularly keeping it updated, is labor intensive. In this article, I presented a detailed tutorial of how `markdoc` can be used for generating Stata help files, package vignettes, and Wiki documentation.

10. https://github.com/haghish/markdoc/wiki/markers
11. Type `db markdoc` to review the features within the dialog box.

With the `mini` engine, `markdoc` enables writing documentation with Markdown and exporting it to Stata help files or other document formats. This is already a considerable improvement for documenting Stata software given that, to date, the only possible markup language for Stata help files was SMCL. Comparatively, Markdown has a simpler syntax, is easy to read and write, and is more versatile. Studies have shown that using Markdown for documentation can also improve the quality of the documents, allowing the author to focus on the content (Voegler, Bornschein, and Weber 2014). This is particularly important if the documentation is written within the script file, which not only makes the code file look better but also provides readable documentation at the outset of the file for anyone who wishes to understand or update the code.

I also proposed a Markdown template for documenting Stata programs that can be appended to an ado-file or a Mata file. Using Markdown instead of SMCL and writing the documentation within script files is a considerable shift from the common practice of writing help files in Stata. However, learning this approach is not time consuming and, more importantly, reduces the time and effort needed for writing and updating software documentation. As a bonus, of course, `markdoc` uses the same procedure for generating dynamic analysis documents and dynamic presentation slides. This makes `markdoc` a general and intuitive literate programming package for Stata users at any level.

6.3 Real-world examples

The examples of section 4 are based on an elementary ado-file with a few options. For the interested reader, more complex packages can serve as more intricate examples of documenting Stata software with Markdown and generating package vignettes and Wiki documentation. The `datadoc`[12], `md2smcl`[13], `weaver`[14], `github`[15], `markdoc`[16], and `rcall`[17] packages—sorted by their ascending level of complexity—are fully documented using the procedure explained in this article and are real-world examples of software documentation with `markdoc`. Each of these repositories has a file named `make.do`—as recommended by the `github` package (Haghish Forthcoming)—that includes the code for not only building the package installation files but also generating the Stata help files and the package vignettes.

12. https://github.com/haghish/datadoc
13. https://github.com/haghish/md2smcl
14. https://github.com/haghish/weaver
15. https://github.com/haghish/github
16. https://github.com/haghish/markdoc
17. https://github.com/haghish/rcall

7 References

Baum, C. F. 2011. Submitting and retrieving materials from the SSC Archive. Department of Economics, Boston College. http://repec.org/bocode/s/sscsubmit.html.

Briand, L. C. 2003. Software documentation: How much is enough? In *2003 IEEE Seventh European Conference on Software Maintenance and Reengineering*, 13–15. Benevento, Italy: IEEE. https://doi.org/10.1109/CSMR.2003.1192406.

Brown, P. J. 1974. Programming and documenting software projects. *ACM Computing Surveys* 6: 213–220. https://doi.org/10.1145/356635.356637.

Cordes, D., and M. Brown. 1991. The literate-programming paradigm. *IEEE Computer* 24: 52–61. https://doi.org/10.1109/2.86838.

Gruber, J. 2004. Markdown: Syntax. http://daringfireball.net/projects/markdown/syntax.

Haghish, E. F. 2016a. github: A module for building, searching, installing, and managing Stata packages from GitHub. GitHub. https://github.com/haghish/github.

———. 2016b. markdoc: A general-purpose literate programming package for Stata. GitHub. https://github.com/haghish/markdoc.

———. 2016c. md2smcl: A Stata module that converts Markdown to SMCL language. GitHub. http://github.com/haghish/md2smcl.

———. 2016d. markdoc: Literate programming in Stata. *Stata Journal* 16: 964–988. https://doi.org/10.1177/1536867X1601600409.

———. 2016e. Rethinking literate programming in statistics. *Stata Journal* 16: 938–963. https://doi.org/10.1177/1536867X1601600408.

———. 2019a. datadoc: Automatic data documentation help file generator. GitHub. https://github.com/haghish/datadoc.

———. 2019b. On the importance of syntax coloring for teaching statistics. *Stata Journal* 19: 83–86. https://doi.org/10.1177/1536867X19830892.

———. Forthcoming. github: A better place for developing, maintaining, and hosting statistical software. *Stata Journal*.

Knuth, D. E. 1983. The WEB system of structured documentation. Technical Report STAN-CS-83-980, Department of Computer Science, Stanford University. http://infolab.stanford.edu/pub/cstr/reports/cs/tr/83/980/CS-TR-83-980.pdf.

———. 1992. *Literate Programming*. Stanford, CA: CSLI Lecture Notes.

Kramer, D. 1999. API documentation from source code comments: A case study of Javadoc. In *Proceedings of the 17th Annual International Conference on Computer Documentation*, 147–153. New Orleans, LA: SIGDOC. https://doi.org/10.1145/318372.318577.

Kulkarni, A., and J. Truelsen. 2015. WK<html>TOpdf. http://wkhtmltopdf.org/.

Leisch, F. 2002. Sweave: Dynamic generation of statistical reports using literate data analysis. In *COMPSTAT 2002*, ed. W. Härdle and B. Rönz, 575–580, 575–580. Physica Verlag: Heidelberg.

———. 2008. Creating R packages: A tutorial. In *Compstat 2008 Proceedings in Computational Statistics*. Heidelberg, Germany: Physica Verlag. https://cran.r-project.org/doc/contrib/Leisch-CreatingPackages.pdf.

Leslie, D. M. 2002. Using Javadoc and XML to produce API reference documentation. In *Proceedings of SIGDOC 02*, 104–108. Toronto, Canada: SIGDOC. http://xml.coverpages.org/LeslieSIGDOC2002-DITA.pdf.

MacFarlane, J. 2006. Pandoc: A universal document converter. https://pandoc.org.

R Core Team. 2006. *Writing R extensions*. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org.

Ramsey, N., and C. Marceau. 1991. Literate programming on a team project. *Software: Practice and Experience* 21: 677–683.

Sousa, M. J. C., and H. M. Moreira. 1998. A survey on the software maintenance process. In *Proceedings of the International Conference on Software Maintenance (Cat. No. 98CB36272)*, 265–274. Bethesda, MD: IEEE. https://doi.org/10.1109/ICSM.1998.738518.

de Souza, S. C. B., N. Anquetil, and K. M. de Oliveira. 2005. A study of the documentation essential to software maintenance. In *Proceedings of the 23rd Annual International Conference on Design of Communication: Documenting & Designing for Pervasive Information*, 68–75. Coventry, UK: ACM. https://doi.org/10.1145/1085313.1085331.

van Heesch, D. 1997. Doxygen: Source code documentation generator tool. http://www.doxygen.nl.

Vasilescu, B., S. van Schuylenburg, J. Wulms, A. Serebrenik, and M. G. J. van den Brand. 2014. Continuous integration in a social-coding world: Empirical evidence from GitHub. In *Proceedings of the International Conference on Software Maintenance and Evolution*, 401–405. Washington, DC: IEEE. https://doi.org/10.1109/ICSME.2014.62.

Voegler, J., J. Bornschein, and G. Weber. 2014. Markdown—A simple syntax for transcription of accessible study materials. In *Computers Helping People with Special Needs*. International Conference on Computers for Handicapped Persons. Lecture notes in computer science, vol. 8547, ed. K. Miesenberger, D. Fels, D. Archambault, P. Peňáz, and W. Zagler, 545–548. Cham, Switzerland: Springer. https://doi.org/10.1007/978-3-319-08596-8_85.

Walsh, D. A. 1969. *A Guide for Software Documentation*. New York: Advanced Computer Techniques.

Wickham, H. 2015. *R Packages: Organize, Test, Document, and Share Your Code*. Sebastopol, CA: O'Reilly.

Wickham, H., P. Danenberg, and M. Eugster. 2019. *roxygen2: In-Line Documentation for R: Version 6.1.1*. R package version ≥3.1. https://rdrr.io/cran/roxygen2/.

About the author

E. F. Haghish is a statistician at the Department of Medical Psychology and Medical Sociology, University of Göttingen, Göttingen, Germany.

A Appendix

A.1 Markdown reference

Table 2. A subset of Markdown syntax supported in SMCL format

Markdown syntax	Results
`Heading 1` `=========`	**Heading 1**
`Heading 2` `---------`	**Heading 2**
`###Heading 3`	**Heading 3**
`####Heading 4`	**Heading 4**
`plain text paragraph`	plain text paragraph
`> text` `> > text`	block quote nested block quote

`__Bold__ text`
`_Italic_ text`
`**Underline** text`
	Bold text
	Italic text
	Underline text *

`- - -` or `---`	horizontal rule
`1. Ordered item1` `2. Ordered item2`	1. Ordered item1 2. Ordered item2
`- Unordered item1` `- Unordered item2`	• Unordered item1 • Unordered item2
`~~~` *text* `~~~`	preserving white space
`[Text](http://url)`	Inserting a hyperlink
a line beginning with four white spaces a line ending with two white spaces	preserving white space line break

NOTE: * only in Stata help files. In other documents, it will be rendered as **bold**.

A.2 Complete help file template

```
/***
_version 1.0.0_

XXX
===========

explain the XXX command briefly...

Syntax
------

> __XXX__ _varlist_ =_exp_ [_if_] [_in_] [_weight_] using _filename_ [, _options_]

### Options

| _option_           |  _Description_          |
|:-----------------|:-----------------------|
| **em**phasize     | explain whatever does   |
| **opt**ion(_arg_) | explain whatever does   |

Description
-----------

describe __XXX__ in more details ...

Options
-------

describe the options in details, if the options table is not enough

Remarks
-------

discuss the technical details about __XXX__, if there is any

Example(s)
----------

explain the example

        . XXX example command

second explanation

        . XXX example command

Stored results
--------------

describe the Scalars, Matrices, Macros, stored by __XXX__, for example:

### Scalars

> __r(level)__: explain what the scalar does

### Matrices

> __r(table)__: explain what it includes

Acknowledgments
---------------

If you have thanks specific to this command, put them here.

Author(s)
---------

Your Name
Your affiliation
Your email address, etc.

License
-------

Specify the license of the software

References
----------

Author Name (year), [title & external link](https://github.com/haghish/markdoc/)
***/
```

A.3 Data documentation template

The template below mimics CRAN's criteria for data documentation. I recommend that you load the dataset in Stata, update the variable labels, and then apply the `datadoc` command to generate a customized template that already includes the intended Markdown table.

```
/***
YYY.dta
=======

data label and to which package it belongs

Description
-----------

The __YYY__ data is about ...

Format
------

The __YYY__ dataset includes ??? observations on ??? variables.

### Summary of the variables

| _Variable_  |  _Type_  | _Description_          |
|:------------|:---------|:-----------------------|
| __var1__    | numeric  | explain var1           |
| __var2__    | string   | explain var2           |

Notes
-----

### Dataset

1. list the data notes

### Variables

| _Variable_  |  _Notes_                          |
|:------------|:----------------------------------|
| __var1__    | note of var1                      |
| __var2__    | note of var2                      |

Examples
--------

if there is a necessity to provide examples about the dataset usage ...

Source
------

cite the source ...

References
----------

cite the reference ...
***/
```

A.4 Data documentation example

auto.dta

1978 Automobile Data ... included in XXX package

Description

The **auto.dta** dataset is about ...

Format

auto.dta data set includes *74* observations and *12* variables.

Summary of the variables

Variable	Type	Description
make	str18	Make and Model
price	int	Price
mpg	int	Mileage (mpg)
rep78	int	Repair Record 1978
headroom	float	Headroom (in.)
trunk	int	Trunk space (cu. ft.)
weight	int	Weight (lbs.)
length	int	Length (in.)
turn	int	Turn Circle (ft.)
displacement	int	Displacement (cu. in.)
gear_ratio	float	Gear Ratio
foreign	byte	Car type

Notes

Dataset

1. from Consumer Reports with permission
2. this data set is included in Stata 15
3. add another second note to the data set

Variables

Variable	*Note*
make	add a note to the make variable
price	add a note to the price variable
weight	add a note to the weight variable

Source

cite the source ...

References

cite the references ...

Figure 9. The results of the `datadoc` command for `auto.dta`