Package Vignette and Help Files Using MarkDoc

E. F. Haghish

February 2016

Difficulties in software documentation

- Writing package documentation is necessary in order to teach the community how to use our statistical software. Their value is not any less than the computational code, if not more. However, they are often ignored, except when you face poor documentation for a package you really need to use.
- software documentation is boaring and time consuming. The more detailed and comprehensive the documentation, the better it is learned.
- An actively maintained statistical package evolves rapidly and the documentation should be updated frequently.
- Updating the documentation of the package is often secondary, i.e. first the code is updated and the the documentation is changed in the help files. In this presentation, I suggest a more efficient approach to package documentation in Stata.

Limits of package documentation in Stata

Stata help-files are written in smcl markup language, which imposes a number of limitations for because the package documentation:

- can only be written in smcl markup
- can only be rendered by Stata
- cannot create heading and subheading, which requires different font sizes
- cannot render graphical files (graphs, images)
- cannot render mathematical notation

Literate programming solution

- The documentation is written in the same script file that the code is written
- Any time a change is made, the documentation is updated as well
- The documentation also work as a guideline for those who are willing to review the code or update the package
- A literate programming software - i.e. `markdoc` will generate the documentation dynamically

What is `markdoc`?

- A general-purpose literate programming package for Stata
- It supports several markup languages such as SMCL, Markdown, LaTeX, and HTML
- It has several features such writing **dynamic analysis reports** in HTML, LaTeX, Microsoft Word docx, PDF, and Markdown.
- It can also produce **presentation slides** in PDF and HTML
- It also produce dynamic package documentation in STHLP, SMCL, HTML, and PDF (on-going)
- It can be imagined as a combination of **r-markdown**, **r-sweave**, **r-presentation**, and **roxygen2**

Package documentation with `markdoc` package

- `markdoc` supports both **ado** and **mata** environments for package documentation
- For each script file (ado or mata), a Stata help file can be generated
- The documentation is written using **Markdown**
- The documentation can be combined with **smcl** for writing the syntax of the package
- `markdoc` processes the documentation and generates **sthlp** or **smcl** files
- `markdoc` can use the same source to generate **PDF package vignettes** in

Installation

`markdoc` requires two other packages which are `weaver` and `statax`. `weaver` provides commands for creating dynamic tables, inserting figures, etc., and `statax` is a Stata syntax highlighter for HTML and LaTeX. Al three packages are also available on GitHub.

- **`ssc install markdoc`**
- **`net install markdoc, force`** /// `from`(https://raw.githubusercontent.com/haghish/markdoc/master/)
- **`ssc install weaver`**
- **`ssc install statax`**

Installation

In addition, `markdoc` requires third-party software for generating PDF and converting markup languages to one another. For writing PDF package vignettes, there are 2 conversions that take place:

- Markdown to smcl for creating Stata help files (using `markdoc`)
- smcl to Markdown to include documentations written for Stata help file in the package vignette (using `markdoc`)
- Markdown to HTML for creating PDF package vignette (using `pandoc`)
- HTML to PDF (using `wkhtmltopdf`)

Therefore 2 additional software are required which are **Pandoc** and **ekhtmltopdf**. Both are open-source freeware available for all common operating systems.

- `markdoc` provides optional automatic installation of the third-party software. although they can also be installed manually and wired to `markdoc`.

Workflow

- `markdoc` changes its behavior based on the given script file. It process the file based on the file extension. Therefore a `.do`, `.ado`, or `.mata` file is processed diferently from a `.smcl` file.
- The package documentation can be written as a combination of smcl and Markdown
- The documentation is written in the same manner as writing a dynamic document using /*** and ***/ markers.

```
/***

Heading 1
=========

Heading 2
---------

### Heading 3

### Heading 4

- - -

- - - Tab

> Indented paragraph (quotation)

: Paragraph

    Indention changes the font to monospace

***/
```

Figure 1:

`markdoc` will add a header to the script file that includes the

- package version
- packagename
- packagedescription

Using the header is optional, but recommended. The documentation can be written in separate chunks, throughout the script file.

header example

- In order to create the template, pass the filename to `markdoc` and use the `export(sthlp)` option.
- I also added the `install` option for auto-installing the third-party software, if they are not already installed

```
. markdoc example.ado, replace export(sthlp) date install
(MarkDoc created example.sthlp)
```

- This will result in the following slide

```
/*** DO NOT EDIT THIS LINE ----------------------------------------------------
Version: 1.0.0
Title: packagename
Description: __explain__ _your_ ___function___ briefly. For more
information visit [MarkDoc](http://www.haghish.com/markdoc) homepage.
---------------------------------------------------- DO NOT EDIT THIS LINE ***/

program myprogram
    display "`0'"
end

/***
Example
=================

    explain what it does
        . example command

    second explanation
        . example command
***/
```

Figure 2:

sthlp view

version 1.0.0, 10 Jun 2016

Title

packagename —— **explain** *your* function briefly. For more information visit MarkDoc homepage.

Example

explain what it does
. example command

second explanation
. example command

Figure 3:

Editing the header

- Using the header is optional
- If you remove the text from each parameter, `markdoc` will ignore that parameter in the help file
- The header is **at the moment** ignored in the HTML and PDF package vignettes

```
/*** DO NOT EDIT THIS LINE ----------------------------------------------------
Version: 1.0.0
Title:
Description:
------------------------------------------------------ DO NOT EDIT THIS LINE ***/

program myprogram
    display "`0'"
end

/***
Example
================

    explain what it does
        . example command

    second explanation
        . example command
***/
```

Figure 4:

sthlp view

```
                                                    version 1.0.0, 10 Jun 2016

Example

    explain what it does
        . example command

    second explanation
        . example command
```

Figure 5:

Exporting package vignette

To export a package vignette in **HTML** or **PDF**, that can easily be shared on the internet or your website, use the same command that you use for generating the

```
. markdoc example.ado, replace export(pdf) date install ///
title("packagename vignette") author("E. F. Haghish") style(stata)
```

packagename vignette

E. F. Haghish

10 Jun 2016

Example

```
explain what it does
        . example command

second explanation
        . example command

when you tab the code, __Markdown does not work__ but smcl works fine
```

Figure 6:

Writing with smcl

- Converting smcl markup to other formats turns it into simple monospace font.
- However, the markdown code will appear nice in both smcl and the package vignette
- The markdown is **easier to read, write, and update**

```
/***
{title:Syntax}

{phang}
{opt commandname} {depvar} [{indepvars}] {ifin} using
[{it:{help filename:filename}}] [{cmd:,} {it:options}]


### Syntax by markdown

> __`commandname`__ _depvar_ [_indepvars_] [_if_] [_in_] using [_filename_] [, _options_]

***/
```

Figure 7:

STHLP view

Syntax

```
commandname depvar [indepvars] [if] [in] using [filename] [, options]
```

Syntax by markdown

```
commandname depvar [indepvars] [if] [in] using [filename] [, options]
```

Figure 8:

HTML and PDF view

Syntax

```
commandname depvar [indepvars] [if] [in] using [filename] [, options]
```

Syntax by markdown

> **commandname** *depvar* [*indepvars*] [*if*] [*in*] using [*filename*] [, *options*]

Figure 9:

Markdown tables

- Stata help files can represent tables, written in smcl markup
- Writing smcl tables has a bit of learning curve
- `markdoc` supports Markdown tables, and it renders them to smcl using the **`asciitable`** option
- TheMarkdown table can be converted to other formats when the package vignette is generated

```
. markdoc example.ado, replace export(sthlp) date install asciitable
```

- example of Markdown table

```
#### Markdown Table

Table Header  | Second Header | Third Header
------------- | ------------- | -------------
Cell 1        | Cell 2        | Cell 5
Cell 3        | Cell 4        | Cell 6
```

STHLP view

Markdown Table

Table Header	Second Header	Third Header
Cell 1 Cell 3	Cell 2 Cell 4	Cell 5 Cell 6

Figure 10:

HTML and PDF view

Markdown Table

Table Header	Second Header	Third Header
Cell 1	Cell 2	Cell 5
Cell 3	Cell 4	Cell 6

Figure 11: