Reproducible Analysis in Stata 16

with MarkDoc package

E. F. Haghish

University of Göttingen

Overview

- Reproducible research
    - Reproducible analysis
        - Literate programming
        - Automating statistical analysis using `markdoc` in Stata
        - Practicing reproducible analysis in Stata
- `markdoc` software
    - Installing `markdoc` from GitHub
    - Installing Pandoc and Wkhtmltopdf (optional)
- `markdoc` workflow
- Adding documentation to create automatic analysis report
    - Markdown notation
    - `markdoc` commands
    - `markdoc` additional markup notations
- Using `markdoc` in classroom
    - presentation slides within Stata
    - classroom documents
- Using `markdoc` for software documentation
    - Stata help files
    - Package vignette or supplementary web material

Reproducible research

- Replication vs reproducibility
- Statistical analyses are carried out to draw conclusions from data, gain knowledge, and communicate claims with the scientific community.
- Traditionally, this workflow has been manual. However, it has many flaws if we put it into perspective:
    - No syntax is written for data analysis (mouse and click approach)
    - Only a summery of the analysis and results is reported
    - Analysis details are not communicated in the manuscript
    - Analysis and results cannot be validated or reviewed

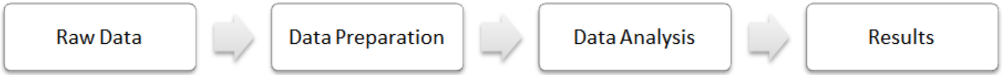


Figure 1: The procedure we are intending to automatize

Literate programming

- The big problem of software documentation
- The literate programming solution
- Adaption of the literate programming in statistics
    - Should ideally supports real-time documentation
    - Should examine the analysis
    - Should provide a restricted framework to improve the code development

markdoc package

- `markdoc` is a general purpose literate programming software
- developed particularly for Stata
- `markdoc` is versatile:
  - generate publication-ready analysis report in various document formats (PDF, Docx, ODT, HTML, LaTeX, etc.)
  - includes a syntax highlighter
  - generate dynamic presentation slides
  - generate dynamic Stata help files in STHLP format or create a package vignette
- Analysis documentation/interpretation is written within *do-files*, as usual
- It emphasizes code readability by keeping the documentation simple

MarkDoc features

- It works with the usual workflow of Stata do-files
    - It is easy to use
    - It underscores clean and readable documentation
- recognizes multiple markup languages
- has a built-in syntax highlighter
- supports several output documents
    - develops text documents
    - presentation slides
    - software documentation

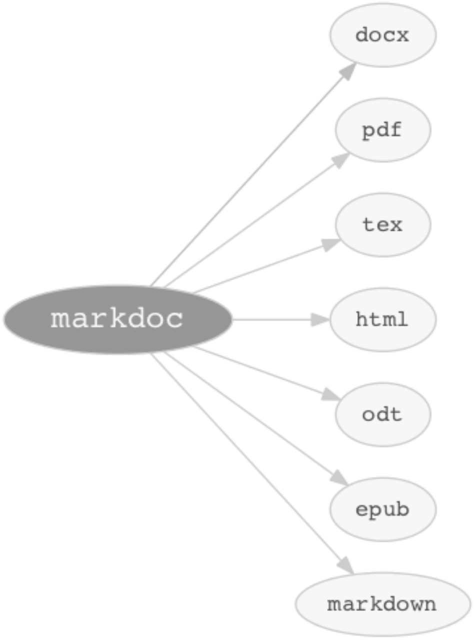


Figure 2: Supported document formats

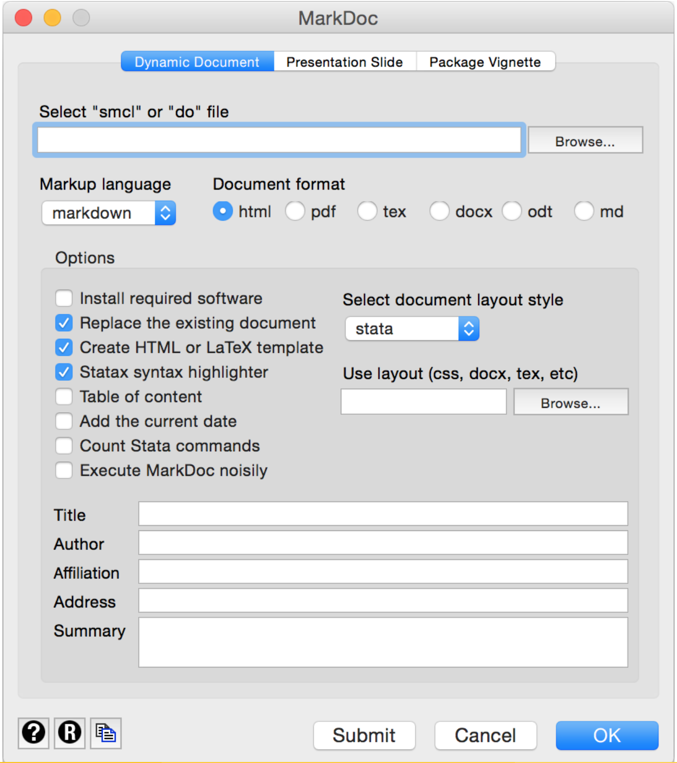
Dialog box

`markdoc` was designed to be a very user-friendly package. To further facilitate learning `markdoc`, a dialog box was programmed to visualize the main options and functionalities of the package.

- The dialog box includes three tabs, each specializes in a particular document format
    - dynamic document
    - presentation slide
    - package vignette

To lunch the dialog box type:

```
db markdoc
```


MarkDoc

Dynamic Document | Presentation Slide | Package Vignette

Select "smcl" or "do" file: [ ] Browse...

Markup language: markdown

Document format: [✓ html] pdf tex docx odt md

Options

- [ ] Install required software
- [✓] Replace the existing document
- [✓] Create HTML or LaTeX template
- [✓] Statax syntax highlighter
- [ ] Table of content
- [ ] Add the current date
- [ ] Count Stata commands
- [ ] Execute MarkDoc noisily

Select document layout style: stata

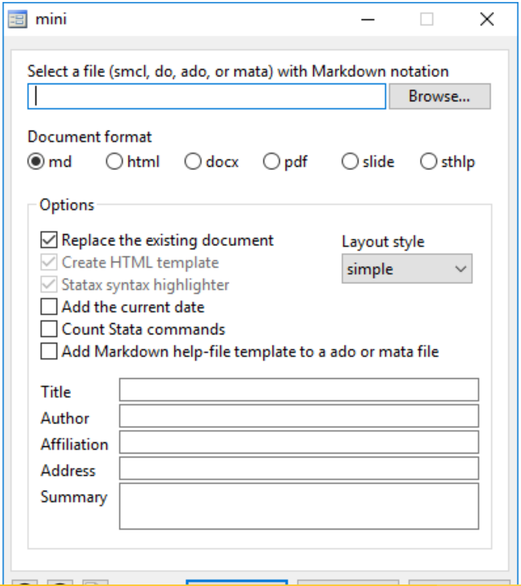
Use layout (css, docx, tex, etc): [ ] Browse...

Title:
Author:
Affiliation:
Address:
Summary:

Dialog box for **markdoc** 5.0

Use the **mini** dialog box, if third-party software are not installed.

```
. db mini
```



mini

Select a file (smcl, do, ado, or mata) with Markdown notation

Browse...

Document format

[✓ md] html docx pdf slide sthlp

Options

- [✓] Replace the existing document
- [✓] Create HTML template
- [✓] Statax syntax highlighter
- [ ] Add the current date
- [ ] Count Stata commands
- [ ] Add Markdown help-file template to a ado or mata file

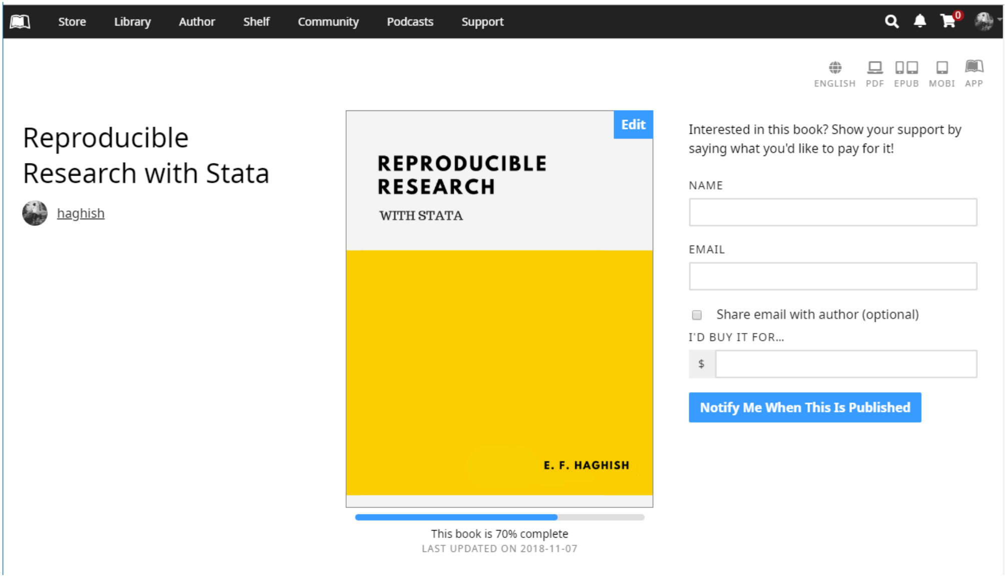
Layout style: simple

Title	
Author	
Affiliation	
Address	
Summary	

Who can use `markdoc`?

`markdoc` was designed having learners in mind. It offers a GUI, a syntax highlighter, and plenty of features to encourage beginners to use it.

1. Students - as early as introductory statistics courses - can use `markdoc` to actively take note inside Stata Do-file Editor
2. University lecturers who teach statistics using Stata, can use `markdoc` to generate PDF slides, educational materials
3. Statisticians can use `markdoc` for creating dynamic analysis reports
4. Finally, advanced users and Stata programmers can use `markdoc` to generate dynamic help files and package vignettes



Reproducible Research with Stata

haghish

Interested in this book? Show your support by saying what you'd like to pay for it!

NAME

EMAIL

☐ Share email with author (optional)

I'D BUY IT FOR...

$

This book is 70% complete

LAST UPDATED ON 2018-11-07

Figure 5: The presentation slides are based on a book

markdoc Installation

- markdoc is hosted on GitHub only https://github.com/haghish/markdoc
- markdoc has package dependencies which are:
  - weaver
  - datadoc
  - md2smcl
- The github command can install markdoc and its dependencies. You can install the github command as follows:

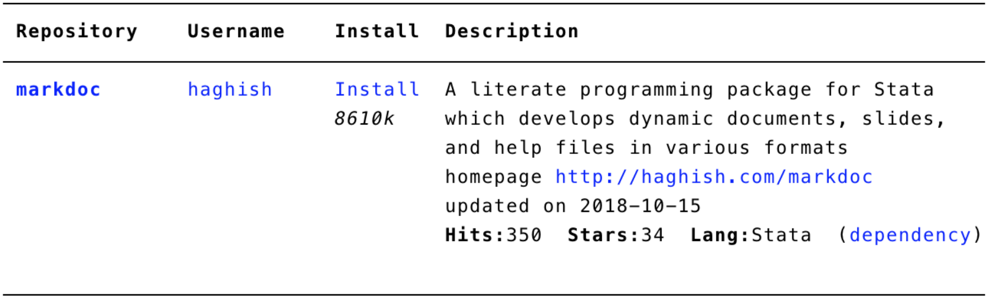
```
. net install github, from("https://haghish.github.io/github/")
```

- Once the `github` command is installed, installing any Stata package from GitHub would be easy
- The installation only requires the authors' GitHub `username` and the `repository` name, separated by a slash
- For example, to install or update `markdoc` and its dependencies type the following command:

```
. github install haghish/markdoc
```

Alternatively, you can use `github` command to search for `markdoc` package in GitHub by typing:

```
. github search markdoc
```



```
Repository   Username   Install  Description

markdoc      haghish    Install  A literate programming package for Stata
                        8610k    which develops dynamic documents, slides,
                                 and help files in various formats
                                 homepage http://haghish.com/markdoc
                                 updated on 2018-10-15
                                 Hits:350  Stars:34  Lang:Stata  (dependency)
```

Figure 6: `github search` output

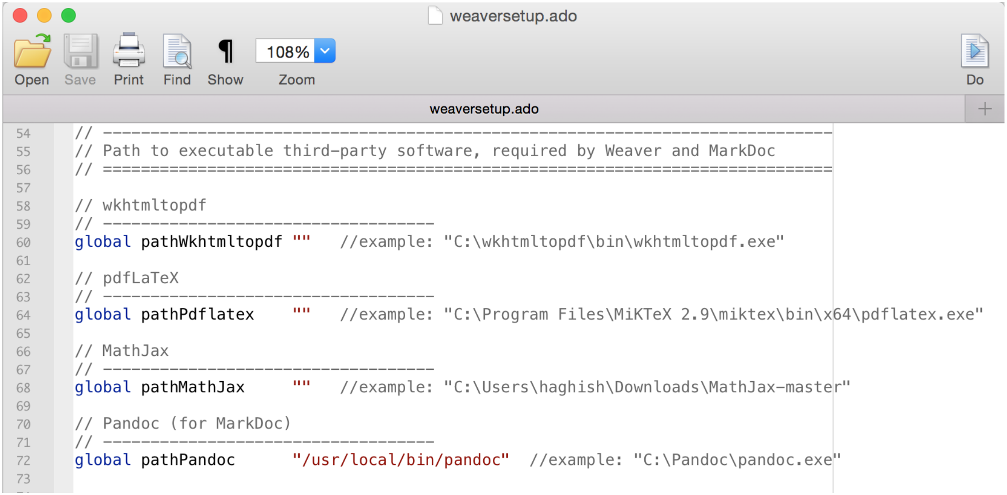
Third-party software installation (optional)

- Previous versions of `markdoc` required other software for generating Word and PDF documents. However, in the recent version of `markdoc`, **this is no longer a necessity**. The third-party software are particularly required for Stata version 14 and below.
- Throughout this presentation, I will use the **`mini`** engine that allows `markdoc` to run independent of any third-party software
- Nevertheless, installing the third-party software can enhance `markdoc`'s capabilities and is generally recommended
- The third-party software are
    - Pandoc for converting Markdown to other file formats
    - wkhtmltopdf for creating PDF documents from source written with Markdown or HTML
    - users who wish to write with LaTeX will require a LaTeX distribution
- The packages hosted on GitHub only include the Ado and help files
- The third-party software should be downloaded and installed manually. However, `markdoc` provides optional automatic installation for Pandoc and wkhtmltopdf, which maybe more convenient for many users

Manual installation of third-party software

- Pandoc software can be downloaded from `www.pandoc.org` website
  - Once Pandoc is installed, the path to executable Pandoc on the operating system can be provided to markdoc using the `pandoc(str)` option
- wkhtmltopdf software can be downloaded from `www.wkhtmltopdf.org`
  - Next, the path to the executable wkhtmltopdf file should be provided to `markdoc` using the `printer(str)` option
- For compiling LaTeX to PDF, a proper LaTeX distribution based on the operating system should be downloaded from `www.latex-project.org`
  - the path to executable pdfLaTeX compiler should be given to `printer(str)` option.
- The path to Pandoc, wkhtmltopdf, and pdfLaTeX can be permanently defined using the **`weave setup`** command. This command opens a script file that memorizes the path to each software within a particular global macro.

```
. weave setup
```



```
// ----------------------------------------------------------------------------
// Path to executable third-party software, required by Weaver and MarkDoc
// ============================================================================

// wkhtmltopdf
// -----------------------------------
global pathWkhtmltopdf ""    //example: "C:\wkhtmltopdf\bin\wkhtmltopdf.exe"

// pdfLaTeX
// -----------------------------------
global pathPdflatex    ""    //example: "C:\Program Files\MiKTeX 2.9\miktex\bin\x64\pdflatex.exe"

// MathJax
// -----------------------------------
global pathMathJax     ""    //example: "C:\Users\haghish\Downloads\MathJax-master"

// Pandoc (for MarkDoc)
// -----------------------------------
global pathPandoc      "/usr/local/bin/pandoc"  //example: "C:\Pandoc\pandoc.exe"
```

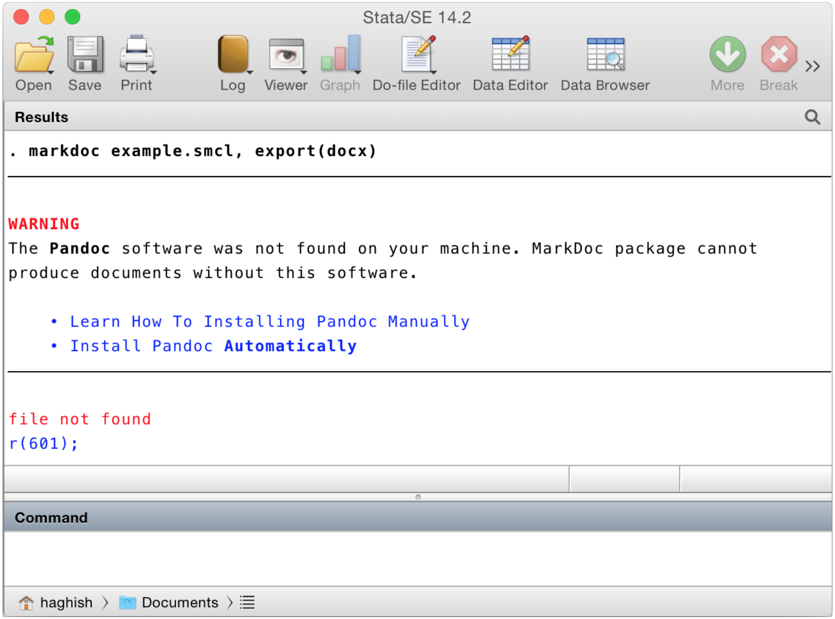
Figure 7: defining the paths to required software permanently

Automatic installation of third-party software

The markdoc command includes the install option which downloads Pandoc and wkhtmltopdf software automatically, if they are not already installed or cannot be accessed by markdoc. As shown in the example below, adding the install option will avoid any error regarding the required software and installs them on the fly:

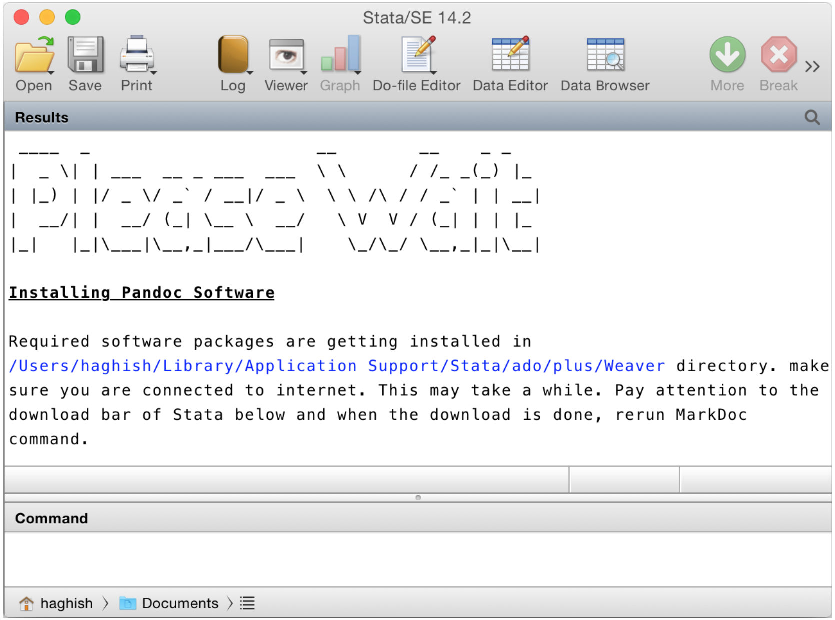
```
qui log using example.smcl, replace
display "If necessary, install the required software on the fly"
qui log c

markdoc example.smcl, export(pdf) install
```

If the `install` option is not specified and `markdoc` does not detect the required software on your machine, a message will be returned on your machine to indicate that the required software was not found.



```
. markdoc example.smcl, export(docx)

WARNING
The Pandoc software was not found on your machine. MarkDoc package cannot
produce documents without this software.

    • Learn How To Installing Pandoc Manually
    • Install Pandoc Automatically

file not found
r(601);
```

Clicking on the `install pandoc automatically` will install Pandoc on your machine:

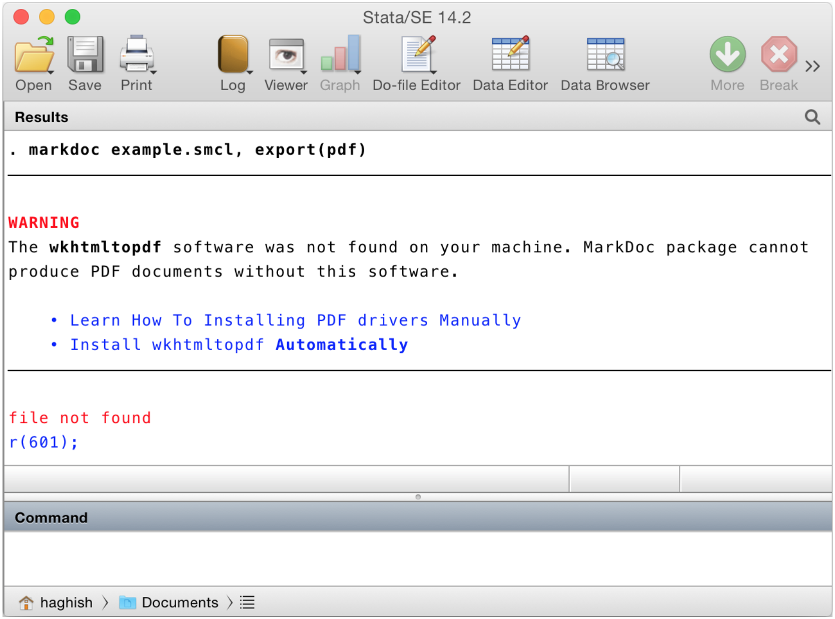


```
 ____  _                       __        __    _ _   
|  _ \| | ___  __ _ ___  ___   \ \      / /_ _(_) |_ 
| |_) | |/ _ \/ _` / __|/ _ \   \ \ /\ / / _` | | __|
|  __/| |  __/ (_| \__ \  __/    \ V  V / (_| | | |_ 
|_|   |_|\___|\__,_|___/\___|     \_/\_/ \__,_|_|\__|

Installing Pandoc Software

Required software packages are getting installed in
/Users/haghish/Library/Application Support/Stata/ado/plus/Weaver directory. make
sure you are connected to internet. This may take a while. Pay attention to the
download bar of Stata below and when the download is done, rerun MarkDoc
command.
```

Figure 9: installing Pandoc automatically

A similar massage is displayed if you are exporting a PDF document and `markdoc` does not access wkhtmltopdf

```
markdoc example.smcl, export(pdf)
```



```
. markdoc example.smcl, export(pdf)

WARNING
The wkhtmltopdf software was not found on your machine. MarkDoc package cannot
produce PDF documents without this software.

    • Learn How To Installing PDF drivers Manually
    • Install wkhtmltopdf Automatically

file not found
r(601);
```

Workflow

- `markdoc` has 2 separate modes
    - Passive mode (allows real-time documentation)
        - Takes a log-file / script file (.ado, .mata, etc.)
        - It does **NOT** evaluate the code nor reproduce the analysis
        - It produces a document very fast
    - Active mode (for testing the whole code in a fresh environment)
        - Takes a do-file
        - Executes the analysis
        - Evaluates its reproducibility
        - It is much slower than the passive mode, because it repeats the analysis

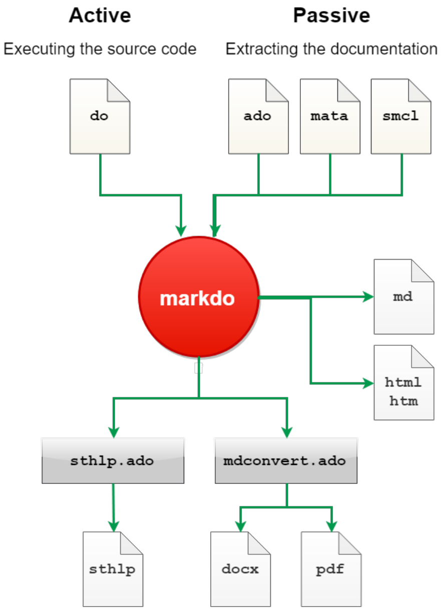


Figure 11: `markdoc` workflow

Active documentation

the do-file must be examined in a clean workspace, where no data is loaded in Stata. `markdoc` takes care of such a test, when executed actively.

- using a single command to convert a smcl log-file to various document formats is convenient, but it does not ensure the reproducibility of the source code
- For example, users might have made changes to the data that are not included in the do-file, but are registered in the log.
- There are markers for temporarily deactivating the log file. . .
- Active documentation is more strict, although time-consuming because every time `markdoc` is executed, the whole project is computed again.

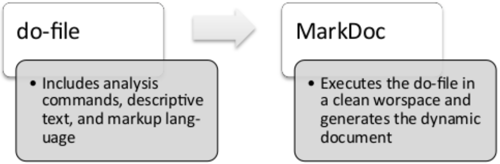

do-file
Includes analysis commands, descriptive text, and markup language
MarkDoc
Executes the do-file in a clean worspace and generates the dynamic document

Figure 12: The process of producing dynamic documents with `markdoc`

Let's assume that we have a do-file that only in displays the hello world text in a do-file named *example1.do*:

```
. display "Hello World"
```

Then the dynamic document can be produced by actively executing the do-file as shown below:

```
. markdoc example.do, mini export(docx)
```

Let's have a closer look. We will load a data set in Stata. Then we execute the command related to the loaded data set with `markdoc`. We would expect an **error**, because in the workspace that `markdoc` is using to test the reproducibility of the code, there is no information about the loaded data set.

- we load the *Auto* data set
  ```
  . sysuse auto, clear
  ```
- we create a do file that simply displays the first line of the data. we name the file *example2.do* and execute it in Stata:

```
. do example2.do

. list in 1

     +-------------------------------------------------------+
  1. | make          | price | mpg | rep78 | headroom | trunk |
     | AMC Concord   | 4,099 |  22 |     3 |      2.5 |    11 |
     |-------------------------------------------------------|
     | weight  | length  |  turn  |  displa~t  |   gear_r~o  |
     |  2,930  |    186  |    40  |       121  |       3.58  |
     |-------------------------------------------------------|
     |                       foreign                          |
     |                      Domestic                          |
     +-------------------------------------------------------+
```

But if we examine it with `markdoc`, we get the following error. `markdoc` says it can't find the data!

```
. markdoc example2.do, mini export(pdf)

. list in 1
observation numbers out of range
r(198);

end of do-file
r(198);
```

Passive documentation

- Is used for generating help files, package vignettes, or quick analysis documents from a log-file
- the SMCL log-file registers every entry in Stata including comments, commands, and text-based output, `markdoc` can produce a dynamic document passively from the SMCL log-file.
- This workflow is indeed convenient, but not recommended for generating analysis documents
- the log-file – which is updated in real-time during the analysis session – can be used to generate the document in real-time too

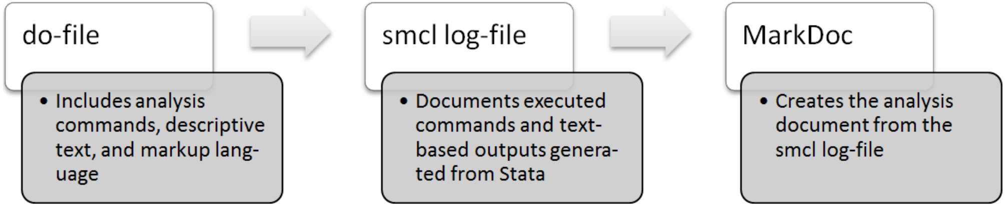

do-file
• Includes analysis commands, descriptive text, and markup lang-uage
smcl log-file
• Documents executed commands and text-based outputs genera-ted from Stata
MarkDoc
• Creates the analysis document from the smcl log-file

Figure 13: The process of producing dynamic documents with `markdoc`

Example

Create a do file with this code and generate a PDF document with syntax highlighter. name the example *example3.do*. let's also use a few of the `markdoc` options to create the title of the document.

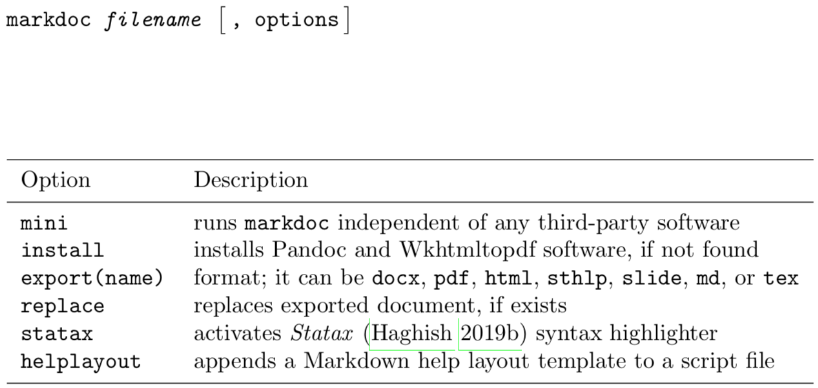
```
. quietly log using example, replace smcl

. display "Hello World"
Hello World

. qui log c
. markdoc example.smcl, mini export(pdf) statax
```

Syntax

To produce a dynamic document, the *filename* of the documentation source should be given to `markdoc`

- **PASSIVE MODE**: a smcl log file with `.smcl` file extension or a script file with `.ado` or `.mata` extension
- **ACTIVE MODE**: a do-file with `.do` extension
- If the file extension is not specified, **SMCL log file is assumed**
    - Specifying the file extension is recommended to provide further clarity

Essential syntax



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

Option	Description
`mini`	runs **`markdoc`** independent of any third-party software
`install`	installs Pandoc and Wkhtmltopdf software, if not found
`export(name)`	format; it can be **`docx`**, **`pdf`**, **`html`**, **`sthlp`**, **`slide`**, **`md`**, or **`tex`**
`replace`	replaces exported document, if exists
`statax`	activates *Statax* (Haghish 2019b) syntax highlighter
`helplayout`	appends a Markdown help layout template to a script file

Figure 14: `markdoc`'s essential syntax

Markup Languages

- `markdoc` supports
  - LaTeX (requires third-party software)
  - HTML
  - Markdown
- In this lecture we will focus on Markdown, which is the simplest. The following links, from its developer's site, can provide a good background about Markdown:
  - `https://daringfireball.net/projects/markdown/`
  - `https://daringfireball.net/projects/markdown/syntax`
  - `https://daringfireball.net/projects/markdown/dingus`
- Markdown is
  - minimalistic and clean
  - simple to read and write
  - helps to focus on the content
  - can be converted to many formats
  - it has become the standard documentation markup language on coding sites as well as statistical software

- in `markdoc` the documentation (Markdown, html, or LaTeX) are written within a special comment signs.

```
/***

   ...

***/
```

- There is no limit in how many times you can place these signs in a do-file.
- These signs can appear anywhere in the analysis document, but not inside loops
    - I will introduce the `txt` command later on which can be included inside loops and programs

Markdown syntax	Result
Heading 1 =========	**Heading 1**
Heading 2 ---------	**Heading 2**
###Heading 3	**Heading 3**
####Heading 4	**Heading 4**
plain text paragraph	plain text paragraph
> text	block quote
Bold or __Bold__ text	**Bold** or **Bold** text
Italic or _Italic_ text	*Italic* or *Italic* text
`monospace` text	monospace text
superscript^2^	superscript^2
---	horizontal rule

1. Ordered item1
  A. Sublist 1
    a. Subsublist 1
2. Ordered item2
	1. Ordered item1
	  A. Sublist 1
	    a. Subsublist 1
	2. Ordered item2

* Unordered item1
  * Sublist 1
    * Subsublist 1
* Unordered item2
	• Unordered item1
	  - Sublist 1
	    * Subsublist 1
	• Unordered item2

![Text](*filename*)	Insert an image with description
[Text](http://url)	Insert a hyperlink

Figure 15: Markdown syntax

Additional Markup Notations

- Additional markers further organize documents prepared for `markdoc` software
- additional markers fall into two categories
  - Passive markers, used for writing static text and styling
  - Active markers, used for interpreting Stata macros in the document
- The Active markers only work if `markdoc` is executed in the active mode

Passive markers

- annotate Stata “commands” and “outputs”
- they help to write a clean analysis report
- By default, MarkDoc includes all of the do-files and text outputs that appear in the Stata results windows. The additional notations allows you to be more selective about what to include:
  - Hiding Stata commands
  - Hiding Stata output
  - Hiding a part of a do-file
  - Importing external files

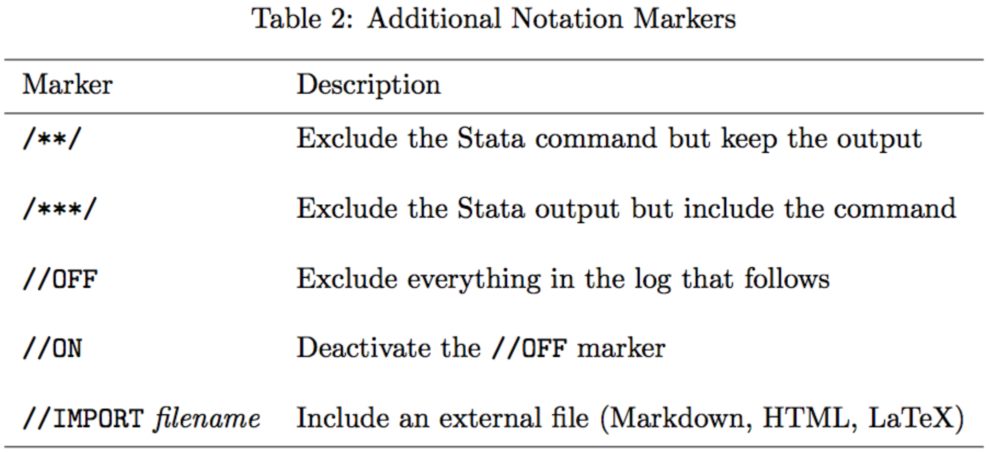


Table 2: Additional Notation Markers

Marker	Description
`/**/`	Exclude the Stata command but keep the output
`/***/`	Exclude the Stata output but include the command
`//OFF`	Exclude everything in the log that follows
`//ON`	Deactivate the `//OFF` marker
`//IMPORT` *filename*	Include an external file (Markdown, HTML, LaTeX)

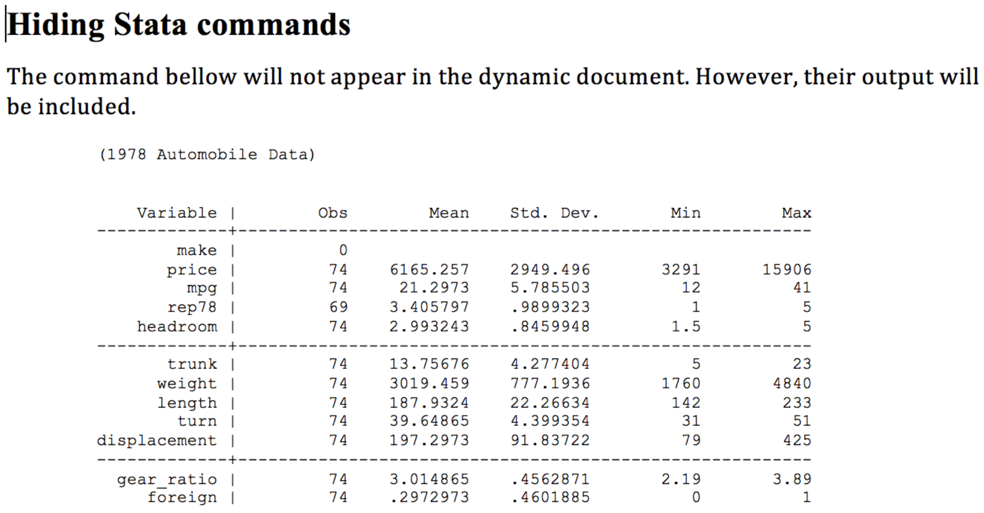
Figure 16:

Hiding Stata commands

```
// -------------- Beginning additional_hide.do ---------------

/***

Hiding Stata commands
---------------------

The command bellow will not appear in the dynamic document.
However, their output will be included.

***/

/**/ sysuse auto, clear
/**/ summarize
```

Executing the `markdoc` command will results in the following output:

```
. markdoc additional_hide.do, export(docx)
```



Hiding Stata commands

The command bellow will not appear in the dynamic document. However, their output will be included.

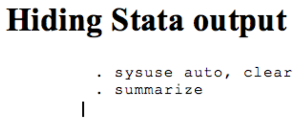
```
(1978 Automobile Data)


    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
        make |          0
       price |         74    6165.257    2949.496       3291      15906
         mpg |         74     21.2973    5.785503         12         41
       rep78 |         69    3.405797    .9899323          1          5
    headroom |         74    2.993243    .8459948        1.5          5
-------------+---------------------------------------------------------
       trunk |         74    13.75676    4.277404          5         23
      weight |         74    3019.459    777.1936       1760       4840
      length |         74    187.9324    22.26634        142        233
        turn |         74    39.64865    4.399354         31         51
displacement |         74    197.2973    91.83722         79        425
-------------+---------------------------------------------------------
  gear_ratio |         74    3.014865    .4562871       2.19       3.89
     foreign |         74    .2972973    .4601885          0          1
```

Figure 17:

Hiding Stata output

```
// --------------- Beginning additional_hide2.do ---------------

/***

Hiding Stata output
---------------------

***/

/***/ sysuse auto, clear
/***/ summarize
```

```
. markdoc additional_hide2.do, export(docx)
```



Hiding Stata output

```
. sysuse auto, clear
. summarize
```

Figure 18:

Hiding a part of a do-file

- MarkDoc also allows hiding a section of the do-file, without influencing the code execution

```
// --------------- Beginning additional_hide3.do ---------------

/***

Hiding Stata commands and output
--------------------------------


***/

//OFF

sysuse auto, clear
summarize

//ON
```

Importing external files

- A convenient feature for producing sophisticated documents
  - Slides
  - Handouts
  - eBook!
- It reads other files (tables, documents, etc) into the main document
- This is the feature you are most-likely looking for writing publication-ready documents

Example

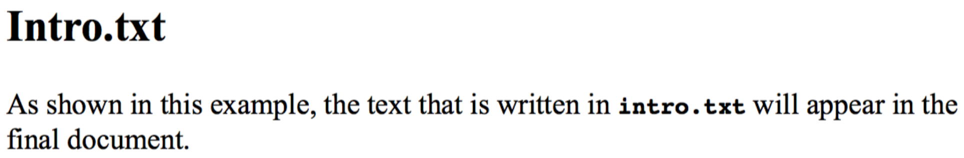
- create a text file and name it *Intro.txt*
- Import the text file passively into a do-file
- execute `markdoc` and create a PDF file

```
Intro.txt
------------

As shown in this example, the text that is written in
__`intro.txt`__ will appear in the final document.
```

The

```
// -------------- additional_import2.do ---------------

//IMPORT intro.txt
```

```
. markdoc additional_import2.do, export(pdf)
```



Intro.txt

As shown in this example, the text that is written in **`intro.txt`** will appear in the final document.

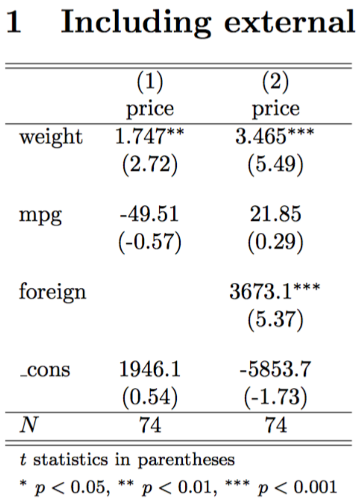
Figure 19: Preview of the output document

estout package for exporting LaTeX tables

- LaTeX also has a command for including external tex files.
- we will use the **estout** package for generating a publication-ready better table
  . ssc install estout
- In the next example, first a LaTeX table is exported from Stata
- Then we write a simple LaTeX document and allow markdoc to complete the LaTeX layout automatically

```
// -------------- Beginning additional_import.do ---------------

//OFF

sysuse auto, clear
sysuse auto
eststo: quietly regress price weight mpg
eststo: quietly regress price weight mpg foreign
esttab using table.tex, replace
eststo clear

//ON

/***

\section{Including external file}

\input{table.tex}

***/
```

```
. markdoc additional_import.do, markup(latex) export(pdf) master replace
```



1 Including external file

	(1)	(2)
	price	price
weight	1.747**	3.465***
	(2.72)	(5.49)
mpg	-49.51	21.85
	(-0.57)	(0.29)
foreign		3673.1***
		(5.37)
_cons	1946.1	-5853.7
	(0.54)	(-1.73)
N	74	74

t statistics in parentheses

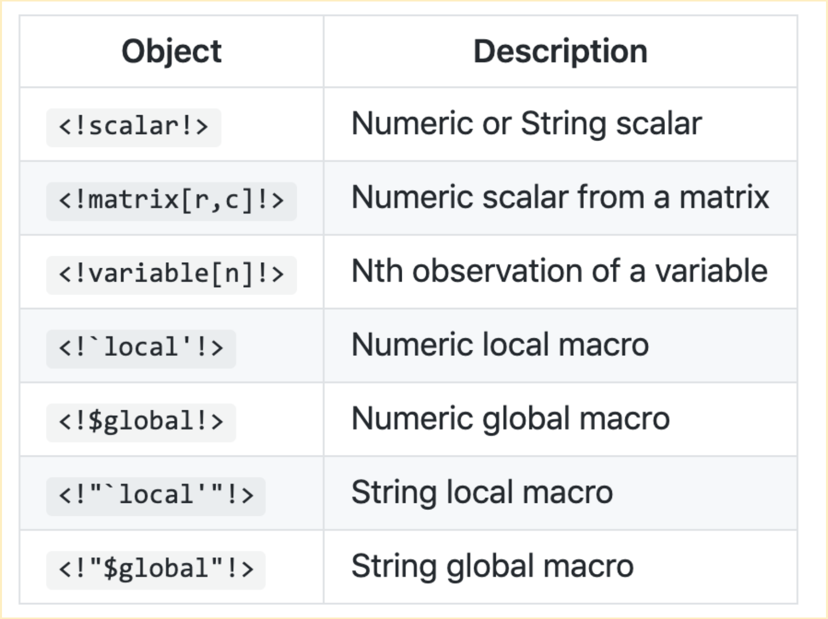
* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Figure 20: Preview of the PDF document

Active markers

- used for writing dynamic text, which includes *scalars* or *macros* that should be automatically interpreted int he text
- only work in the Active mode
- can show the values of
  - scalars
  - variable observations
  - local macro
  - global macros
- values should be placed within `<!*!>` marker

ACTIVE MARKERS TABLE



Object	Description
`<!scalar!>`	Numeric or String scalar
`<!matrix[r,c]!>`	Numeric scalar from a matrix
`<!variable[n]!>`	Nth observation of a variable
``<!`local'!>``	Numeric local macro
`<!$global!>`	Numeric global macro
``<!"`local'"!>``	String local macro
`<!"$global"!>`	String global macro

Figure 21: Preview of the PDF document

Example

```
local a = 1
scalar b = 2
matrix define A = (20,30\40,50)

/***
This is heading <!`a'!>
=======================

The values of a matrix can be displayed within the text. For example,
you can write <!A[1,1]!> which shows the scalar of the first row and
first column of the matrix in your documentation. This feature makes
writing dynamic text much more convenient compared to the previous procedure.

This is heading <!b!>
---------------------

REMEMBER, that this procedure only works if you execute a do-file with
markdoc, that is, using the `markdoc filename.do, export(format)` syntax.
***/
```

Additional commands

- these commands are borrowed from `weaver` package
  - they are installed automatically as a dependency
- They come very handy when the document is generated by a program dynamically or within a loop
- They allow more details for styling a document, compared to Markdown
  - Adding a figure Automatically
  - adding a dynamic table
  - adding dynamic text

Adding figures dynamically

- we previously used Markdown to include an image in the document
- The process was:
    1. saving a graph from Stata to the disk
    2. including the graph to the dynamic document
- This procedure can be further simplified, using the **`img`** command
    1. Automatically capture the current graph from Stata and include it in the dynamic document
    2. Include a figure from the disk/internet in the dynamic document
    3. Resize the width and the height of the image in the dynamic document
    4. Align the image to the left (default) or center of the document
    5. Add a graph description

Syntax of `img` command

Import graphical files in the dynamic document

```
img [using filename] [, markup(str) title(str) width(int) height(int) left center
```

Automatically include the current graph from Stata in the dynamic document

```
img [, markup(str) title(str) width(int) height(int) left center ]
```

Examples

- create a do-file and execute it with `markdoc` actively or passively

```
. sysuse auto
. histogram price
. img
```

In this example, **`img`** has stored the current graph in a directory called **Weaver-figure**

Adding text dynamically

- the `txt` command is somehow like the `display` command, but it's used for writing text in the dynamic document
- it can be used to write text within loops or programs and interpret scalars, global, or local macros within
- try typing `txt 1+1`

```
. sysuse auto
. summarize price
. txt "the mean of Price variable is " r(mean)
```

Syntax of the `txt` command

```
txt [code] [display_directive [display_directive [...]]]
```

where the `display_directive` can be:

```
"double-quoted string"
`"compound double-quoted string"'
[%fmt] [=]exp
_skip(#)
_column(#)
_newline[(#)]
_dup(#)
,
,,
```

Writing dynamic tables

- `tbl` simplifies writing and styling dynamic tables
- The default markup language is Markdown, but it also support LaTeX and HTML
- It can align the content of each column to the left, center, or right
- It creates a table somehow similar to the way a matrix is defined in Stata

tbl Syntax

The syntax of the command is:

```
tbl (*[,*...] [\ *[,*...] [\ [...]]]) ///
    [, markup(str) title(str) width(int) height(int) center left ]
```

where the * represents a display directive, which is:

```
"double-quoted string"
`"compound double-quoted string"'
[%fmt] [=]exp
,
{l}
{c}
{r}
```

Examples

- creating a simple 2x3 table with string and numbers

```
tbl ("Column 1", "Column 2", "Column 3" \ 10, 100, 1000 )
```

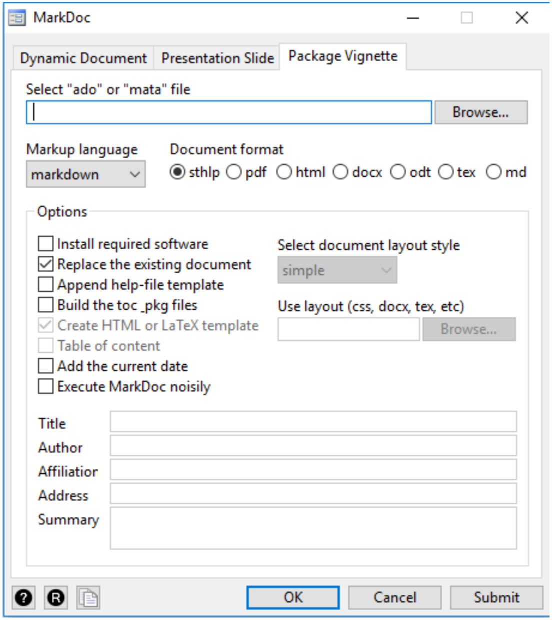
- creating a table that includes scalars and aligns the columns to left, center, and right respectively

```
tbl ({l}"Left", {c}"Centered", {r}"Right" \ c(os),  c(machine_type), c(username))
```

Stata help files

- Stata has it's own markup language
    - *Stata Markup and Control Language* (SMCL)
- All help files as well as default log files are written in this markup language
- Writing documentation with SMCL is not appealing:
    1. smcl is difficult
    2. somehow messy to write
    3. difficult to read, write, and comprehend
- literate programming with smcl is difficult and makes the script file too complex to read

- `markdoc` can generate Stata help files from Ado and Mata files
- The software documentation can be written in Markdown, using the same procedure
- If the documentation can be exported to Stata help files or package vignette
- Type `db markdoc` and navigate to the **Package Vignette** tab:



Select "ado" or "mata" file

Markup language: markdown

Document format: [✓ sthlp] pdf html docx odt tex md

Options

- [ ] Install required software
- [x] Replace the existing document
- [ ] Append help-file template
- [ ] Build the toc _pkg files
- [x] Create HTML or LaTeX template
- [ ] Table of content
- [ ] Add the current date
- [ ] Execute MarkDoc noisily

Select document layout style: simple

Use layout (css, docx, tex, etc)

Title

Author

Affiliatior

Address

Summary

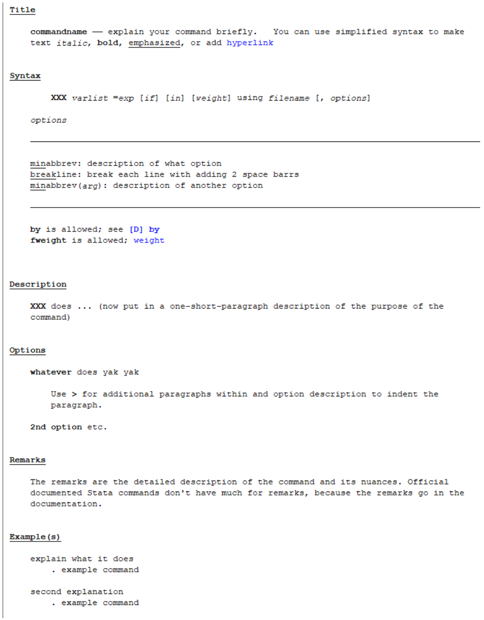
Figure 22: Using markdoc GUI for generating software documentation

Example

- Let's make an Ado file, and use some simple Markdown syntax to write in it
- let's write a:
    - Header 1
    - Header 2
    - style some text
    - Indent text
    - add a line
    - add a link

```
/***
Title
======

__commandname__ - explain your command briefly. You can use simplified
syntax to make text _italic_, __bold__, ***emphasized***, or
add [hyperlink](http://www.haghish.com/markdoc)

Syntax
------

> __XXX__ _varlist_ [, _options_]

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

    explain what it does
        . example command

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

- execute this example with `markdoc` GUI and generate:
    - a `sthlp` file
    - a `html` vignette
    - a `docx` vignette
- In the GUI, there is an option for appending documentation to an Ado file
- Apply the **Append help-file template** to see an example documentation template
- generate a sthlp and html file from the template



```
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]

    options

    ________________________________________________________________________________

    minabbrev: description of what option
    breakline: break each line with adding 2 space barrs
    minabbrev(arg): description of another option

    ________________________________________________________________________________

    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

    second explanation
        . example command
```

Figure 23: Example help file template