# Dynamic Documents in Stata

Bill Rising
StataCorp LLC

2018 Brazilian Stata Users Group meeting
São Paulo, SP
7 December 2018

## Contents



# 1 Introduction

## 1.1 Goals for Creating Documents

### The Good and Bad of Creating Documents

- Think of documents you've made in the past, good and bad
- Good:
  - ◇ Reused ideas from one project for another
  - ◇ Reused and polished lessons for teaching
- Bad:
  - ◇ Questions on methods for reaching particular numerical results
  - ◇ Updating analyses because of new or improved data
  - ◇ Producing repetitive reports

---

### General Idea

- What gets done once often gets done twice
  - ◇ Similar projects
  - ◇ Updated datasets
  - ◇ Datasets arriving over time or from various sources
  - ◇ Teaching
  - ◇ Production work, such as dreaded monthly reports
- The second and later repetitions should not start from scratch

---

### Dynamic Documents

- Needed: reproducible, reusable, and maintainable documents, aka dynamic documents
  - ◇ Documents should be reproducible at the push of a button
    - ⋆ No manual intervention!
  - ◇ Documents should be reusable
  - ◇ Documents should be easily maintained and improved
    - ⋆ This is especially necessary for teaching
- Both of these are easy for pure narratives
- Including computational results is trickier
- Making this nice for all collaborative parties is even trickier

---

### Best Possible Process

- One underlying file for producing a final document, including both narrative and computation
  - ◇ If not a single document, a single folder with easily-related files
- The final document can be reliably reproduced from scratch
- Drafts of the final document can be passed around to all collaborators
  - ◇ Topic experts as well as statistical experts as well as writers
  - ◇ Those comfortable with programmerish work and those who are not
- The final document could be in a variety of forms

---

### What We'll See Here

- Several tools for producing dynamic documents
- Some way of deciding between complexity, completeness, and comprehension

---

# 2 Dynamic Documents

## 2.1 General Needs

### Bare Necessities for Teaching

- Commands
- Results
- Graphs

---

### Bare Necessities for Reports

- Results without commands
- Inline results
  - ◇ Results often show up within the narrative
- Invisible commands

---

### Dream World

- Extremely readable documents
- Flexible formatting

---

# 3 Software Review

## 3.1 Overview

### A Sketch of What to Do

- Here is a basic outline of a small evaluation we'd like to do
  - ◇ This is in the data/shared/pseudo.txt file
- It has a few items of interest
  - ◇ Stata commands and output
  - ◇ Graphics
  - ◇ A table from `tabout`
  - ◇ An unnumbered list
  - ◇ Boldface, italics and fixed-width fonts
- We would like to realize this report (or something close to it) in different ways

---

### Included Software

- We will look at four and one half pieces of software
- Germán Rodríguez' `markstat` command
- Stata's official `dyndoc` command
  - ◇ As well as the unofficial `dynpandoc` command
- Stata's official `putdocx` command
- A wrapper to (possibly) make `putdocx` simpler, called `putwrap`
- A brief glance at StatTag, which is made to hide the Stata code from other

---

### Excluded Software

- The software below was covered in a similar talk in 2016:
  - ◇ `texdoc` for making documents which are like Stata Journal articles
    - ⋆ Still relevant
  - ◇ Markdoc for creating general-purpose documents in many formats
  - ◇ StatWeave for making general-purpose documents
  - ◇ A suite for producing lessons with handouts

---

### Terminology

- It will help to have some defined jargon here to refer to files
  - ◇ A *base* file gets processed by the software
  - ◇ The result of the processing is an *interim* file, if that file needs more processing
  - ◇ The document as it would be viewed will be called a *final* file
    - ⋆ This is not final as in "final draft"

---

### Working Through the Examples

- Much as something fully interactive would be nice, typing is dull
- We'll look at examples of files for each of the methods and then see if we can get them to turn into documents
- Most of the talk will be spent looking at these files
- When this talk is posted, all the example files will be in the file `repdoc.zip`
- Start by getting into the proper location

```
. cd "~/Desktop/2018_brazil_repdoc/repdoc"
```

---

## 3.2 markstat

### `markstat` Basics

- `markstat` was written and is maintained by Germán Rodríguez
- `markstat` is based on the **markdown** language
- `markstat` can produce most any document type you would like
  - ◇ Be sure to use `png` files for graphics if you want this
- `markstat` can be used in either simple markdown mode or in a strict mode
- Narrative and code are in the same file

---

### `markstat` Example, Basic Syntax

- Change into the `markstat` subdirectory

```
. cd markstat
```

- Take a look at `paper_simple.stmd` in your favorite text editor
  - ◇ Stata code is indented with a single tab character
  - ◇ Graphics are included with the odd `![alt-title]{source_file}` construction
- To typeset into a web page

```
. markstat using paper_simple
```

- To now produce a `docx` document

```
. markstat using paper_simple, docx nodo
```

- To now produce a pdf

```
. markstat using paper_simple, pdf nodo
```

---

### `markstat` Example, Strict Syntax

- Take a look at `paper_strict.stmd`
  - ◇ There are now real code fences for Stata code
  - ◇ It is possible to suppress Stata commands
- To typeset into a web page

```
. markstat using paper_strict
```

- To turn this into a `docx` document or pdf document, it would need to be edited
  - ◇ The `tabout` command explicitly saves as an html
  - ◇ This points to the disadvantage of trying to be too fancy!

---

### `markstat` Installation & Dependencies

- Getting `markstat` itself is simple

```
. ssc install markstat
```

- It does require another piece of Stata software

```
. ssc install whereis
```

- It also requires Pandoc (http://pandoc.org)
- If you want to use LaTeX, you need to install the package for your OS
  - ◇ You also need to get Stata's style file
  - ◇ Instructions for this are at the site (http://data.princeton.edu/stata/markdown)

---

### `markstat` Process

- `markstat` processes a markdown file to produce the end document
- `markstat` produces many small files containing code and output
  - ◇ By default these get deleted, but they can be kept
- It is possible to regenerate the document without running the Stata commands
  - ◇ While dangerous in general, this is useful when fixing typos in the narrative
  - ◇ Germán credits taking this idea from Ben Jann's `texdoc`

---

### `markstat` Advantages

- Can be quite simple
  - ◇ Simplicity can lose some important features
- Can be made more complex
  - ◇ The added complexity reduces the readability of the base file
- Has the ability to include external files as the markdown gets processed
  - ◇ This is not possible in vanilla markdown

---

### `markstat` Disadvantages

- Markdown has some limitations
- Unfortunately, markdown doesn't have some hidden rarely-used constructions which allow extra complexity

---

## 3.3 dyndoc

### `dyndoc` Basics

- `dyndoc` is an official Stata command
- `dyndoc` uses markdown for its formatting language
- `dyndoc` makes web pages (HTML)
- Narrative and code are in the same file
- Rather than indentation or code fences, `dyndoc` use its own `dyndoc` tags

---

### `dyndoc` Example

- First move to the proper location

```
. cd ../dyndoc
```

- Take a look at `paper.md`
  - ◇ You can see that the tags/code fences are more complex
  - ◇ Including a graph is downright byzantine
- You need to take some care about whitespace in some instances (as noted)
- Typesetting is simple

```
. dyndoc paper.md, replace
```

  - ◇ The extension is needed, as it is not assumed
  - ◇ The `replace` option is needed to replace the old webpage

---

### `dyndoc` Process

- `dyndoc` takes a markdown + Stata file and turns it into an html file
- There are no interim files

---

### `dyndoc` Advantages

- There are extra dyndoc tags which allow for conditional processing
  - ◇ This can be useful in dreadful monthly reports for calling out rare evants
- Has the ability to include external files as the markdown gets processed
- It's built in to Stata

---

### `dyndoc` Disadvantages

- The tags can look a bit cluttered
  - ◇ The clutter is not as bad when the file is viewed as a Stata do-file in your text editor
- Adding graphs seems very odd

---

### `dyndoc` Dependencies

- None, of course

---

## 3.4 dynpandoc

### `dynpandoc` Basics

- `dynpandoc` is an unofficial extension of `dyndoc`
- This is an unofficial Stata command which extends `dyndoc` to be able to use other formats, by using Pandoc
- It will need to be told each time where Pandoc has been installed
  - ◇ `markstat` gets around this by using the `whereis` command

---

### `dynpandoc` Example

- We are already in the proper location as this has been combined with `dyndoc`
- Take a look at `paper_no_tabout.md`
  - ◇ The `tabout` example was removed so that docx and pdf could be used
- Typesetting is similar to `dyndoc`

```
. dynpandoc paper_no_tabout.md, replace ///
    path(/usr/local/bin/pandoc)
```

  - ◇ The extension is needed, as it is not assumed
  - ◇ The `replace` option is needed to replace the old webpage
  - ◇ The path to Pandoc is needed

---

### `dynpandoc` Example, cont.

- Here is how you can make a docx file
- Making pdf files appears to be buggy on a Mac

---

### `dynpandoc` Advantages

- Same as for `dyndoc`
- More output types

---

### `dynpandoc` Disadvantages

- Same as for `dyndoc`
- Specifying the path to Pandoc every time is painful

---

### `dynpandoc` Dependencies & Installation

- This must be installed via

```
net install https://github.com/huapeng01016/StataMarkdown/blob/master/dynpandoc
```

  - ◇ Hua has this as a github site, so it is possible to download everything and make your own version: https://github.com/huapeng01016/StataMarkdown

---

## 3.5 putdocx

### `putdocx` Basics

- `putdocx` is an official Stata command
- `putdocx` makes docx documents
  - ◇ The documents are based on the open standard for docx
  - ◇ So... `putdocx` works best with Open Office and its relatives
  - ◇ `putdocx` also works well with Microsoft Office

---

### `putdocx` Example

- First, get into the right place

```
. cd ../putdocx
```

- Next, take a look at `putdocx.do`
  - ◇ Pretty difficult to read
  - ◇ There is no split between narrative and code
  - ◇ Every font change requires an entire command
  - ◇ If you would like commands to appear, you must repeat them as code
  - ◇ This is quite different than the other dynamic document commands
- Do the dofile

```
. do putdocx
```

---

### `putdocx` Process

- `putdocx` allows writing text, tables and graphs
- It does not write Stata commands or their output directly
  - ◇ It is made more for reports than for reporting on Stata
- It is always in Stata mode

---

### `putdocx` Advantages

- Easy to push out estimation tables
- Very flexible table generation
  - ◇ Can write line by line to update a table rather than needing to write one single massive command
- Has a lot of user interest, so there are a slew of community-contributed aids

---

### `putdocx` Disadvantages

- `putdocx` documents look like pure code
  - ◇ Tough on collaborators
- Changing small pieces can take some effort
  - ◇ Reduces maintenance or changing of documents to nil

---

### `putdocx` Dependencies

- None, of course

---

## 3.6 putwrap

### `putwrap` Basics

- `putwrap` attempts to allow `putdocx` to have a narrative mode and a Stata mode
- Otherwise it is `putdocx`

---

### `putdocx` Example

- The `putwrap` example is already in the `putdocx` folder
- Use `putwrap` on the basic file

```
. putwrap using putwrap.wrap, replace
```

- Then use `putdocx` on the resulting file

```
. do putwrap
```

---

### `putwrap` Process

- By default, it is assumed that the do-file is in narrative mode (i.e. writing the document)
- To go into Stata mode, use `putdocx pause`
- To go back to narrative mode, use `putdocx resume`
  - ◇ Adding two subcommands to an official Stata command breaks all the rules for community-contributed software
  - ◇ Be forewarned!
- `putwrap` takes a file using these commands, and created a do-file which has all the requisite paragraph and text commands

---

### `putwrap` Advantages

- It should make documents with long narrative sections easier to read

---

### `putwrap` Disadvantages

- If there are a lot of font changes, it is still necessary to break up the narrative
- It is not a clever program, so it can get fooled if lines start with special contructions (like inline macro expansions)

---

### `putwrap` Dependencies

- Needs `putdocx`, of course

---

## 3.7 StatTag

### StatTag Basics

- StatTag is much different from the above Stata-centric tools
- You need to have an MS-Word document with one or more separate do-files
- You then use StatTag to create tags
  - ◇ These end up being special comments within the do-file
- You then put the tags into the MS-Word document
- The results get filled in by replacing the tags in a way similar to a mail merge

---

### StatTag Example

- This example is a bit painful to show, because StatTag's dialog boxes do not respect scaling properly (at least in Windows 10)
- StatTag allows 4 types of tags:
  - ◇ **Value**, which must be the result of a `display` command
  - ◇ **Table**, which must be the result of a `matrix list` command
  - ◇ **Figure**, which must be the result of a `graph export` command
  - ◇ **Literal** which can be any command
- There are no Stata commands to show; just the do-file and the MS Word document

---

### StatTag Process

- Create a do-file which will have all the results you need in the paper
  - ◇ Be sure to observe the above restrictions about how values and tables work
- Write the paper
- Define and insert the tags
- Update the tags
  - ◇ From my testing, StatTag can be finicky and might allow updating only one tag at a time

---

### StatTag Advantages

- You can show the finished paper to a colleague, and (s)he does not need to look at any Stata code
- The results are really dynamic

---

### StatTag Disadvantages

- StatTag ignores all formatting in Stata commands
  - ◇ This means that you must do the formatting in StatTag itself
- Each tag update requires rerunning the entire do-file
  - ◇ This can get tedious for large do-files
- Debugging is difficult
  - ◇ It seems that StatTag can throw spurious errors when running do-files
  - ◇ The errors cause it to exit without leaving Stata open for inspection
  - ◇ Keeping logs can cause other errors
- The Mac version crashes in Mojave

---

### Getting StatTag

- It requires software from http://sites.northwestern.edu/stattag/
- Downloading the software requires that you register with Northwestern
  - ◇ Probably so that they can count the downloads and justify the grant money
- The documentation tells you how to get it running from there

---

# 4 Conclusion

## 4.1 Conclusion

### Conclusion

- There are plenty of packages out there for making dynamic documents
- The quality of the packages has greatly increased in the past couple of years
- There is still plenty to iron out, but the basics work fine
- You should really give this a try

---

# Index