

Reproducible Research: Weaving with Stata and StatWeave

Bill Rising

StataCorp LP

Mexican Stata Users Group Meeting
April, 2008



Outline I

- 1 Introduction
 - Goals
 - Reproducible Research and Weaving
- 2 StatWeave
 - Implementation Basics
 - Options and Reusing Code
 - Examples Using LaTeX
 - Gotchas
- 3 Conclusion
 - What We've Seen
 - Just In Case



Goals

- Learn about reproducible research, or in its original name “literate programming”
- Show how this can be done using StatWeave
 - `http://www.stat.uiowa.edu/~rlenth/StatWeave/`

Concept

- Any analysis should be completely reproducible
- Reproduction of an analysis should be accessible

Typical Implementation in Stata

- In Stata, it is possible to have reproducible research by having
 - A series of do-files which reproduce the steps in the analysis
 - A document which somehow includes pieces of the log files produced by the do-files
 - The document could also include output as generated by ado-files
 - Inclusion is simple in something like \LaTeX , but is not very easy in typical word-processors
- This is only a partial solution, because this allows only listings and graphics, but not the direct use of computed quantities

Weaving

- Another approach is that of *weaving*, where the text and the analysis code are in the same document
 - It is analogous to writing computer programs which contain both the code (the analysis) and the documentation (the writeup)
 - Such documents *weave* together documentation and code
- Weaving has the advantage that there can be no separation between the statistics and the writeup
- Weaving has also been called *literate programming*
 - This was Knuth's original name for the idea of mixing code and documentation

Other Reasons for Weaving

- Clearly useful for documentation
- Weaving is fantastically useful when teaching courses which use software
 - Can remake documentation as the software is updated, making sure that all commands and output are up-to-date
 - Can make homework and test questions quite easily

Other Implementations

- Knuth wrote *WEB* for weaving C or C++ code
 - He also wrote T_EX, of course
- *docstrip* is another utility which can combine code and documentation
 - Really hard to use
- *Sweave* has been around for quite a while for S-plus and R

Today's Topic: StatWeave

- StatWeave is written by Russ Lenth at the University of Iowa
 - `http://www.stat.uiowa.edu/~rlenth/StatWeave`
- It is relatively new, but is quite useful
- Written in Java, so it is cross-platform
- It can support many different programming languages—we'll focus on Stata, of course



What Document Types are Allowed?

- StatWeave allows working with and creating \LaTeX and OpenOffice documents
 - Both have nice open formats
- The architecture of StatWeave allows other document types to be added

The Input Document

- Write an “swv” document which includes a mixture of text and code
- Include code in special blocks
 - Block definitions are specific to the type of document
- Add options which allow reuse or redisplaying of code or output

The Output Document

- Run the document through StatWeave
 - Currently implemented as a command-line application
- Open up the resulting document—the code blocks will be replaced by input, output, graphics, etc., depending on how they were defined
- Smile at the magic

Conceptual Model

- Each block of code is called a *code chunk*
- StatWeave looks through the document and pulls out each code chunk, keeping track of its position and optional label
- Each language (here: Stata only) runs its blocks of code as though they were sequential commands in one session unless specifically overridden
- We can reuse code or output by specifying options for the code chunks

Input and Output—Basic form

- Each block's input is gathered together
- Each block's output is gathered together
- The output is all displayed after the input
 - This is a bit of a shock when using Stata (or most other packages other than SAS)

Taking a Look at Some Examples

- We'll look primarily look at examples from OpenOffice
 - \LaTeX examples are not so easy to demonstrate interactively
 - Using OpenOffice will be the easiest way to see how the fine-tuning works, also

Working with OpenOffice

- This is most easily done by showing a document which already marked up, and by adding some code chunks
 - The final document will be available from Stata's website after the meeting
- Controlling what is being done by StatWeave is done by styles
- The `SWStyles.ott` file contains the styles needed to add code to a document
- The allowable options follow...

Options for Fine Control

- Options are split by their scope
 - Whole document/following chunks
 - Entire code chunks
 - Input, and output
 - There are also special options which pertain to graphics
- Most options are boolean
 - *option* is the same as *option* = true
 - !*option* is the same as *option* = false

Rest-of-Document Options

- These are formatting options which are put into non-code blocks, which typically would be used at the top of a document
 - In OpenOffice, these are the SWopts style
 - In \LaTeX , these are `\weaveOpts{ }` commands
- These can also be made language-specific by including the language name
 - In OpenOffice, using `Stata:` in the options block
 - In \LaTeX , using `\StataweaveOpts{ }`

Code Chunk Options

- `label` is string, and defaults to *"lastchunk"*
 - Used for labeling the chunk for later reuse—often worthwhile
- `eval` is boolean, and defaults to *true*
 - If false, the code is displayed but not evaluated
- `restart` is boolean, and defaults to *false*
 - If true, a new session is started, so the previous state of the package is discarded

Common Input Options

- `echo` is boolean, and defaults to *true*
 - If false, the code is not displayed
- `savecode` is boolean and defaults to *false*
 - If true, the code is saved but is not displayed, sadly enough
 - Main conceptual use is for default setups for following code
- `codestyle` is string
 - For the document as a whole, it defaults to `Wininput`
 - It can be a style in OpenOffice, or a `FancyVerbatim` environment in \LaTeX

Common Output Options

- `hide` is boolean and defaults to *false*
 - If true, the output is not displayed
- `saveout` is boolean and defaults to *false*
 - If true, the output is saved, but not displayed
- `outstyle` is string, and is similar to `codestyle`
 - For the document as a whole, it defaults to `woutput`
 - It can be a style in OpenOffice, or a `FancyVerbatim` environment in \LaTeX

Common Graphics Options

- `fig` is boolean and defaults to *false*
 - It *must* be specified if a figure is produced by the codeblock
 - There can be only one figure per code block
- `figfmt` is string and specifies the type of output
 - `eps` is a common type, though StatWeave seems to like `png`, which is good for visual, but not printed, materials
- `scale` is numeric and defaults to 1.0
- `disph` and `dispw` are both numeric control the displayed height and width
 - These can be given in cm, in, pt, etc.
 - Scale overrides `disph` and `dispw`

Working with Expressions

- StatWeave claims it can evaluate Stata expressions
 - This is badly overstated, but should be easily fixed
- As it stands now, all it understands for expressions are `egen` functions(!)

Simple Stuff

- Since this is not interactive, it will be simple with a little explanation
- Rest assured that all output displayed below is a part of this \LaTeX document

Building Blocks

- Stata code is enclosed in blocks:

```
\begin{Statacode}  
    some code here  
\end{Statacode}
```

- There are options for including and hiding code

A First Example

- Opening the ubiquitous `auto` dataset and running a regression:

```
. sysuse auto
```

```
(1978 Automobile Data)
```

```
. regress mpg weight displacement headroom
```

Source	SS	df	MS
Model	1597.77483	3	532.59161
Residual	845.684629	70	12.081209
Total	2443.45946	73	33.4720474

```
Number of obs =      74
F( 3,      70) =    44.08
Prob > F       =    0.0000
R-squared      =    0.6539
Adj R-squared  =    0.6391
Root MSE      =    3.4758
```

mpg	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
weight	-.0064885	.0011863	-5.47	0.000	-.0088545	-.0041225
displacement	.005754	.0099834	0.58	0.566	-.0141573	.0256652
headroom	-.2444638	.5525116	-0.44	0.660	-1.346413	.8574858
_cons	40.48554	2.224643	18.20	0.000	36.04863	44.92245

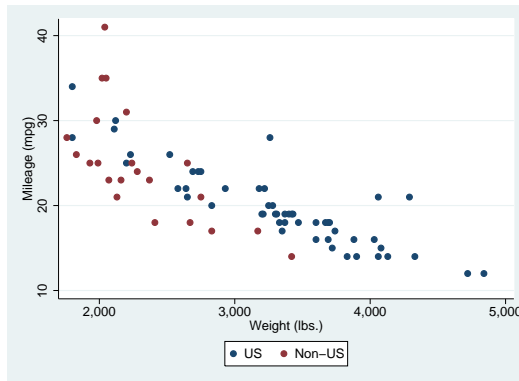
Code for the First Example

- The code for the above block is just:

```
\begin{Statacode}  
    sysuse auto  
\end{Statacode}  
\begin{Statacode}  
    regress mpg weight displacement headroom  
\end{Statacode}
```

A Graph

- Here is an example of a graph:



Ideal Code for the Graph

- Ideally, the code for the graph is also simple:

```
\begin{Statacode}[fig, figfmt=eps]
twoway (scatter mpg weight if !foreign) ///
      (scatter mpg weight if foreign), ///
      legend(order(1 "US" 2 "Non-US"))
\end{Statacode}
```

- This doesn't quite work in Stata on a Mac, because StatWeave won't allow `figfmt=eps` when using `pdftex`

Input and Output

- As mentioned above, the behavior of StatWeave is much more SAS-like than Stata-like, because it gathers all output from a code chunk together
- The workaround is simple, though unfriendly: simply enclose each line in its own Statacode environment
 - Not nice, but workable

Input and Output Example, Part I

- All together

```
. sum mpg  
. tab foreign, sum(mpg)
```

Variable	Obs	Mean	Std. Dev.	Min	Max
mpg	74	21.2973	5.785503	12	41

Car type	Summary of Mileage (mpg)		
	Mean	Std. Dev.	Freq.
Domestic	19.826923	4.7432972	52
Foreign	24.772727	6.6111869	22
Total	21.297297	5.7855032	74

Input and Output Example, Part II

- Splitting the lines into separate chunks

```
. sum mpg
```

Variable	Obs	Mean	Std. Dev.	Min	Max
mpg	74	21.2973	5.785503	12	41

```
. tab foreign, sum(mpg)
```

Car type	Summary of Mileage (mpg)		
	Mean	Std. Dev.	Freq.
Domestic	19.826923	4.7432972	52
Foreign	24.772727	6.6111869	22
Total	21.297297	5.7855032	74

What We've Seen

- Embedding code in documents
- Being able to rerender output quite simply
- A few rough edges—but these are fixable

That's It

Less Common Input Options

- `prompt`, `prom` and `ompt` are all string, and control the look of the prompt
 - These work for Stata in OpenOffice but not \LaTeX .
- `showref` is boolean and defaults to *false*
 - If there is recalled code in a block and this is true, the recalled code is displayed
- `codefmt` is \LaTeX only, and requires some knowledge of the `fancyvrb` package
- `beforecode` and `aftercode` are also \LaTeX only, and cause \LaTeX code to be placed before and after every code block

Less Common Output Options

- `results` is string, and is used for using a package to insert document-type specific code
- `loose` and `tight` change how series of blank lines are displayed (not too useful in Stata)
- `outfmt` is $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ only and is similar to `codefmt`
- `beforeout` and `afterout` are just like their counterparts for code

Less Common Graphics Options

- There are also `height` and `width` options, but they do not preserve the aspect ratio
 - These would make for smaller bitmap files, such as `png`
- `savefig` holds the figure for later display
- `beforefig` and `afterfig` are \LaTeX only

Referring to Code

- Besides code chunks, there are other tags
- `coderef` will reuse code by its label
 - The code is executed once again
- `recall \textit{thing}` will recall saved chunks using the chunk's label
 - The \textit{thing} can be `code`, `out`, or `fig`

Special Tricks

- StatWeave understands code substitution for numbered arguments
 - This can be used for defining code chunk templates which get reused
- This provides a very primitive programming interface