Reproducible Research: Weaving with Stata

Bill Rising

StataCorp LP

Italian Stata Users Group Meeting
October, 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
Goals

- Learn about reproducible research, or in its snobby name “literate programming”
- Show how this can be done using StatWeave
  - web address goes here
Any analysis should be completely reproducible
Reproduction of an analysis should be accessible
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.
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 using 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 \texttt{TEX}, 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
StatWeave allows working with and creating \LaTeX{} and OpenOffice documents

- Both have nice open formats which allow

The architecture of StatWeave allows other document types to be added
Building Blocks

- Write a document
- 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
Creating the Document

- Run the document through StatWeave
  - Currently implemented as a command-line application
- Open up the resulting document
- Smile and nod knowingly
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. The language(s) (here: Stata) run their 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 at examples from both OpenOffice and \texttt{\LaTeX}.
- They’ll be similar, so that you can see how they work.
- 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 the course website
- 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 $option = false$
Whole Document Options

- These are formatting options which are put into non-code blocks
  - In OpenOffice, these are the SWopts style
  - In \LaTeX, these are \texttt{\weaveOpts{}} commands

- They pertain to all the following blocks, so they truly are from-here-on options

- These can also be made language-specific by including the language name
  - In OpenOffice, using \texttt{Stata: } in the options block
  - In \LaTeX, using \texttt{\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 is 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 `Winput`
  - It can be a style in OpenOffice, or a `FancyVerbatim` environment in `\LaTeX`
Less Common Input Options

- **prompt**, **prom** and **ompt** are all string, and control the look of the prompt
  - None work for Stata

- **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
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
Less Common Output Options

- **results** is a 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 **\LaTeX** only and is similar to **codefmt**.
- **beforeout** and **afterout** are just like their counterparts for code.
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 materials only

- **scale** is numeric and defaults to 1.0

- **disphe** and **dispew** are both numeric control the displayed height and width
  - These can be given in cm, in, pt, etc.
  - Scale overrides **disphe** and **dispew**
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.
Besides code chunks, there are other tags:

- `coderef` will reuse code by its label:
  - The code is executed once again.

- `recall thing` will recall saved chunks using the chunk’s label:
  - The `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
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
  - I will include the outline of the document on the conference website—it will reproduce the output, but not the formatting
    - It takes some work to integrate this so that it displays nicely on both the slides and the handouts
    - I also have many customizations for putting together blocks of lectures
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`  
  `. regress mpg weight displacement headroom`  

(1978 Automobile Data)

<table>
<thead>
<tr>
<th>Source</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>Number of obs = 74</th>
<th>F( 3, 70) = 44.08</th>
<th>Prob &gt; F = 0.0000</th>
<th>R-squared = 0.6539</th>
<th>Adj R-squared = 0.6391</th>
<th>Root MSE = 3.4758</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>1597.77483</td>
<td>3</td>
<td>532.59161</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Residual</td>
<td>845.684629</td>
<td>70</td>
<td>12.081209</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>2443.45946</td>
<td>73</td>
<td>33.472047</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 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  |
The code for the above block is just:
\begin{Statacode}
sysuse auto
regress mpg weight displacement headroom
\end{Statacode}

For short blocks, all is quite simple
Here is an example of a graph:

```
. twoway (scatter mpg weight if !foreign) ///
.   (scatter mpg weight if foreign), ///
.   legend(order(1 "US" 2 "Non-US"))
```

A Graph
Ideally, the code for the graph is also simple:

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

In reality, life is not so simple when working with pdftex

- We’ll see a workaround in a bit
Here is an example from the Mata manual

```mata
. mata
. X = (76, 53, 48 \\ 53, 88, 46 \\ 48, 46, 63 )
. Xi = invsym(X)
. Xi
.

[ symmetric ]
+-----------------+-----------------+-----------------+
1 | .0298458083    |                 |
2 | -.0098470272   | .0216268926    |
3 | -.0155497706   | -.0082885675   | .0337724301
+-----------------+-----------------+-----------------+
```

Bill Rising

Reproducible Research: Weaving with Stata
This continues the last computation and quits Mata

. $X_i * X$
. end

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>-1.11022e-16</td>
<td>-1.11022e-16</td>
</tr>
<tr>
<td>2</td>
<td>-1.11022e-16</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

+----------------------------------------------+
Note on the Last Two Slides

- Even though we were working in Mata, the input was split across the two slides.
- This illustrates that each block starts where the previous block stopped.
- So... no worry about losing track of where you are.
Working with Graphs and pdftex

- Statweave is a bit heavy-handed when trying to make pdf documents—it will not allow making an eps file
  - pdftex doesn’t understand eps files, but there is a built-in epstopdf converter with all modern \LaTeX\ distributions
- The trick is to make an eps file, and then use Stata to use internal tools to convert it
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 Stata code environment.

- Not nice, but workable.
All together

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Obs</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpg</td>
<td>74</td>
<td>21.2973</td>
<td>5.785503</td>
<td>12</td>
<td>41</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Car type</th>
<th>Summary of Mileage (mpg)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
</tr>
<tr>
<td>Domestic</td>
<td>19.826923</td>
</tr>
<tr>
<td>Foreign</td>
<td>24.772727</td>
</tr>
<tr>
<td>Total</td>
<td>21.297297</td>
</tr>
</tbody>
</table>
Splitting the lines into separate chunks

```
. sum mpg

<table>
<thead>
<tr>
<th>Variable</th>
<th>Obs</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpg</td>
<td>74</td>
<td>21.2973</td>
<td>5.785503</td>
<td>12</td>
<td>41</td>
</tr>
</tbody>
</table>

. tab foreign, sum(mpg)

<table>
<thead>
<tr>
<th>Car type</th>
<th>Summary of Mileage (mpg)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
</tr>
<tr>
<td>Domestic</td>
<td>19.826923</td>
</tr>
<tr>
<td>Foreign</td>
<td>24.772727</td>
</tr>
<tr>
<td>Total</td>
<td>21.297297</td>
</tr>
</tbody>
</table>
```
What We’ve Seen

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