# **Do-file Templates & Do-File Basics**

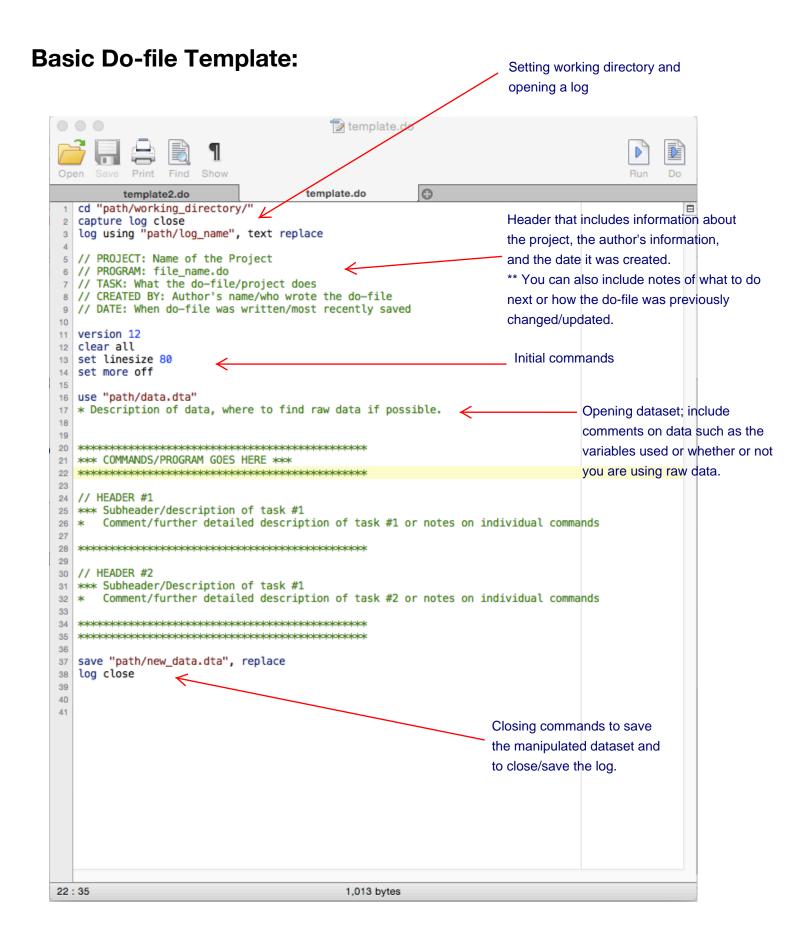
Do-files provide an effective way to write, edit, and share code, but do-files are most effective when they are clearly organized and well annotated. It is considered good practice to have your own standard template to use in your do-files. This post provides and discusses an example of a basic dofile template that you can use in your own coding practice. The template provided here is not a universal, standardized do-file template (such a thing doesn't exist as there are various approaches to writing and documenting code); however, this template includes key basic commands and introduces a simple organizational structure for your code and comments. Feel free to modify and incorporate this template into your own practice as you see fit.

# Why standardize and annotate your do-files?

- 1. A well organized and consistent do-file is easier to read and understand
- 2. Makes it easier for you and others to catch and edit errors in your code
- 3. Consistent and annotated do-files make your code/research replicable.
- 4. Comments can keep track of your work as you update your code and continue working on the project. I.e. comments orient you on where each do-file fits in your project.
- 5. Comments explain your code and why you employed certain data analysis/manipulation. This prevents confusion for others/collaborators or yourself when you return to your work.
- 6. Standardizing and annotating your do-files allow your do-files to be more **robust** and **legible.** Robust and legible do-files are self-contained, replicable, and easy to read and understand. For more information on how to make your work legible and robust, see Dr. Jenn Earl's notes on do-files here:

http://jearl.faculty.arizona.edu/sites/jearl.faculty.arizona.edu/files/Jan2 7\_2014\_material\_wiki.pdf

The following template is constructed with these goals in mind...



## **Commands Explained**

- 1.cd "/[path]/[working directory]/"
  - a. This sets your working directory; where you retrieve your data and save your work/files.
- 2. capture log close
  - a. log close- closes an already open log; if a log is already open, the code log using will break/fail to run.
  - b. capture- prevents an error message and allows do-file to continue running if a log is already closed. If a log is NOT opened, the log close command will fail without using capture.
- **3.** version [12]
  - a. Version control; sets your stata command interpreter to a certain version.
  - b. You cannot version control to a newer version than the version installed on your machine/computer.
  - c. This allows you to run the do-file in the version the code was written, therefore any discrepancies in stata versions will not break your code.
- 4. clear all
  - a. Removes data, labels, and stored results while closing open files, windows, and dialog boxes. This allows you to clear all previous work done in stata to run a new do-file.

```
5. set linesize [80]
```

- a. Sets/restricts the number of characters per line (maximum width) of Stata output
- 6.set more off
  - a. Prevents Stata from pausing and displaying the ---more--message.
- 7.Use "[pathway]/[filename]"
  - a. Opens/retrieves data from working directory
- 8. Save "[pathway]/[filename]"
  - a. Saves data to working directory

## Comments/Notes Explained

These notes/annotations explain code and make do-files easier to read. When comments are preceded by the following symbols, they are not interpreted as Stata commands and instead are highlighted in green.

- a. Section break- indicates a new section of the do-file.
- b. It's a good idea to use this to organize coding/data cleaning sections of the do-file from data analysis sections (especially in a master.do)
- c. Unspecified number of asterisks.

2. //

a. Section and/or do-file header; Title or brief description of each section.

3. \*\*\*

- a. Subheader and description of a task or lines of code.
  - Can include number of lines this comment applies to. (e.g. "\*\*\* The next 3 lines recode the gender variable")

4. \*

- a. A note or explanation of what a line of code is doing
- b. A note to self

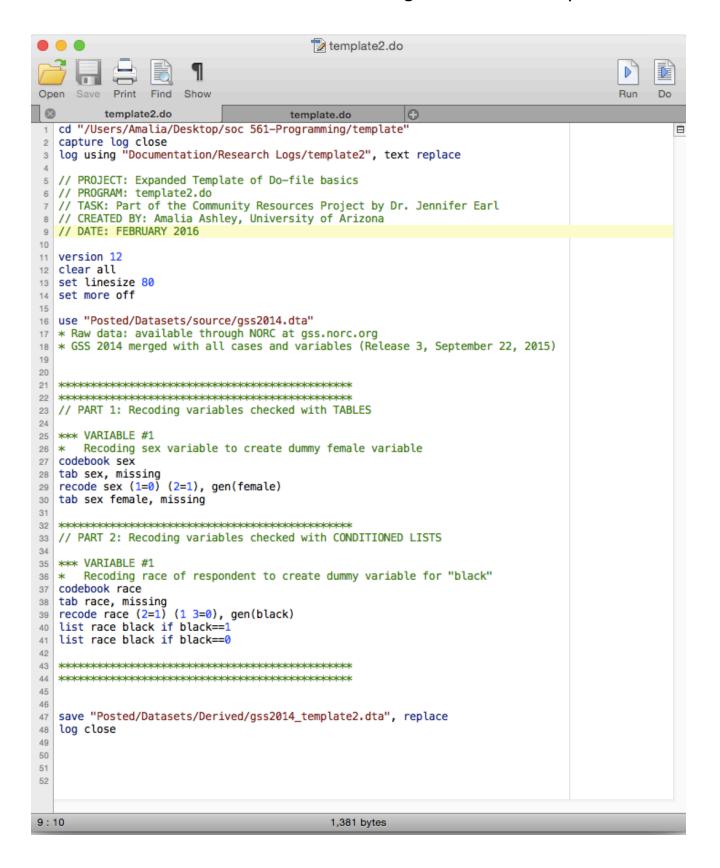
Other symbols for notes in Stata:

1. / 2. /\*

Feel free to use these indicators of notation as you see fit according to your own preferred method of organization.

#### **Expanded Do-file Template: An Example**

This do-file uses the same template presented above. This example provides expanded commands and notes so you can see how your own code, comments, directories might look in a complete do-file.



# **Bugs and Debugging**

#### Common Do-file Errors

- 1. Mistyping variable names
  - a. Use Stata's autocomplete function
  - b. OR copy variable names from the variable window and pasting it into command in the do-file.
- 2. Mistyping commands
  - a. The font of commands is dark blue in Stata do-files. If Stata does not recognize a command (because it is misspelled or otherwise incorrect) it will NOT be blue in the do-file.
  - **b.E.g.** recode sex (1=0) (2=1), gen(female)
- 3. Using log close at the beginning of a program when a log is already open
  - a. Use capture log close to close log at beginning of a program.
- 4. Forgetting to add , replace to save and log commands.
  - a. Add , replace to commands to save logs and new data sets to overwrite existing files.
- 5. Option error
  - a. Use help [operation] to check if you are using a valid option and syntax.
  - b. Make sure you use a comma [,] between the operation and the option.
- 6. Putting by in the wrong place
  - a. Use help [operation] command to check the correct syntax and order.
- 7. Loop Errors
  - a. Use set trace on to see how Stata interprets/evaluates each step of the loop.
- 8. Confusing assignment [=] and Evaluation [==]
  - a. <u>Assignment</u>
    - i. ex:gen var1=5
    - ii. "assign a new value to ... "
  - b. Evaluation (a test of equivalence)
    - i. **ex.** replace var1=6 if evnmth==12
    - ii. "is the same as, is equal to"

#### **Other Troubleshooting Methods**

If you can't identify the error or debug your do-file, try these methods to get your do-file running.

- 1. Break the program down and run it a few lines at a time to locate where the error is.
- 2. Run clear all and macro drop \_all
- 3. Restart Stata
- 4. Reboot your computer
- 5. Google the error message or command
- 6. Check Statalist, an online stata forum: http://www.statalist.org/
- 7. Copy and paste a similar command above it to proof by eye
- 8. Share your code with another pair of eyes- maybe they can find the error
- 9. Post your question/problem to statalist as a LAST resort
  - a. Search stataforum first.

## Don't forget-

Your programs and do-files should be created with the intention to make your work efficient, replicable, and legible. Incorporating standardized do-files and detailed comments into your regular coding practice will help you work efficiently and facilitate collaboration.

Best of luck!