Lab 1: Review of basic Stata use

Andreas Beger

January 7, 2009

1 Overview

This first lab is a review of the basic use of Stata and some common Stata commands. You will learn how to:

- write do files
- keep log files
- merge datasets
- collapse datasets
- create new variables from existing variables
- recode variables
- issue conditional commands
- calculate descriptive statistics for variables and graph histograms
- run a bivariate regression
- generate and save graphs

The motivating example for this lab comes from international relations. I want to see historical trends for the number of states in the international system as well as for the number of ongoing wars and how many states were actively fighting in them at any given point. Preferably we would present that information in a nice graph. It does not really matter whether you care about this particular example, because producing something of use here will incidentally also teach you a lot about data management and basic Stata commands.

The Correlates of War project (COW Sarkees 2000) provides convenient data that we can use here. Particularly, I will use three datasets that are described in more detail below.

2 Setup

Before we actually start working on this problem, there are two things you should usually do when you work with Stata. First, in most situations it pays off to keep track of your work in what Stata calls a do file. Second, you will usually also want to keep a log of your work and the output Stata generates. On a general note, keeping track of your work with do files and log files makes it incredibly easier for others to replicate your work. Note that this goes both ways—not only will other people be able to replicate your work, but you yourself will have a much easier time figuring out what you did on that project you worked on two years ago, and of which you now have little concrete memory. Once you get familiar with doing Stata work in this manner, it can also be incredibly helpful to you by making it a *lot* easier to fix mistakes that you do not discover until later, or to make changes to your work afterwards. In other words, I would recommend that you *write do files and keep log files for everything you do in Stata*.

So, what is a do file? A do file, also known as a batch file, is a document that contains a list of Stata commands, and that you can call up in Stata to execute those commands. To take a step back, there are fundamentally three different ways to do work in Stata. You can use the menus that are on top of the window, you can type commands into the command prompt at the bottom, or you can type commands into a do file and then run the do file. Actually all three of those things really do the same thing in Stata, and even if you use the menus, the commands will show up in the output window. Of course using menus is easier, while typing commands in the command prompt or do files is somewhat more difficult.[1] Either way, once you have figured out how to properly do something that you want to be done, copy the command and put it in your do file.

There are different ways to create a do file. Stata has a built-in do file editor (which really is just a text editor like Notepad in Windows). There is a button for it in the top, or you can get to it through the menu *Window / Do-file Editor*. Alternatively, you can just use any third-party text editor (Notepad, WinEdt, etc.) to create a text file—when you save it just make sure that you save it as a ".do" file (i.e. the file extension should be ".do", not ".txt" or something like that).

As for keeping a log, there are again different options. There is a button for it on the top bar, or you can click *File / Log / Begin*, or you can type the command "log using *filename*", where *filename* is whatever you want to save the log file as. A log file copies all the output in Stata, i.e. everything that Stata displays in the output window. Once you are done, you can type "log close" to close the log. By default, Stata saves logs in it's own log format, ".smcl". You can use Stata to open those logs (*File / Log / View...*). I prefer saving logs as regular text files that you can open view them with any text editor. To do that save the file as a ".log" file rather than a ".smcl" file.[2]

Good do files do not just contain Stata commands, but also comments and other information to help the reader along. Comments help a reader understand what the purpose of the do file is, who the author is, what is going on, etc. Just as importantly, they will help you in the future to figure out just what it is that you did on that project you last worked on six months ago. To do that, you need to be able to write things in do files that Stata knows are not commands. There are different ways of marking comments in Stata—type `help comments` to see all of them. Anyways, my do files generally have the following structure:

```
set more off
capture log close
clear
log using "C:\abeger\Teaching\09-1 Methods III\lab1.log", replace

/*=========================================================================

    Andreas Beger
    Department of Political Science
    Florida State University
    abeger@fsu.edu
```

[1]Personally, I do not use the menus except to open files or open the data browser (the little button that shows you the data as a table). There is nothing you can do by using menus that you could not also do in the command prompt, but vice versa there are things that you can do with the command line that you could not do by using the menus.

[2]Stata's log format adds some formatting to the text, and so it will not display properly in a regular text editor. Unless you particularly care about having commands bolded and things like that, you do not need to save log files in Stata's format.

```
    04 January 2009

    Methods III, Lab 1

    Log:    lab1.log
    Input:  ...
    Output: ...

    Notes:
    For this do file to work correctly:
    (1) Change the location in which the log file is saved (above).
    (2) Change the working directory (below).

==============================================================================*/

version 10.1
cd "C:\abeger\Teaching\09-1 TA Methods III\"

/*
//  1. Entering data ...
==============================================================================*/

Stata commands be here

...

log close
exit
```

The first four lines are there to ensure the do file does not crash and to start the log. The next big blurb of comments contains some basic information about the do file—who wrote it, date it was written, purpose, log file, etc. The command "version 10.1" after that tells Stata that this do file was written for Stata 10.1, so that it knows how to interpret commands. Just put whatever version of Stata you are using there.[3] Next I change the working directory, which is the default directory in which Stata saves files, looks for files, etc. At this point we can begin with the main body of the document, which contains the actual commands (and a liberal sprinkling of comments). At the end, you want to close the log and tell Stata the do file is done.

3 Entering and saving data

There are several different ways of entering data into Stata. Commonly you will have data that already is in Stata's data format (".dta"), and Stata can use these files without problem. Click *File / Open*, the appropriate button on the top, or type "use *filename*" to open a dataset.[4] Stata also has several commands for reading data in several common non-Stata file formats. Type "help infiling" to see a page summarizing all of them. Saving data is straightforward, and again you have the three options of menu, button, or command line.[5] In our case, we will use three

[3]Some Stata commands change between versions. By specifying which version of Stata you are using, future versions of Stata will know that they should interpret commands using syntax appropriate for the older version of Stata.

[4]Whichever way you open a dataset, note that filenames are typically enclosed in quotations marks. This is so that Stata can deal with filenames that have spaces in them. In other words, always enclose filenames in quotations marks when you write do files.

[5]Data formats have changed between Stata 9 and Stata 10. Stata 10 will read old files, but Stata 9 will not recognize Stata 10 files. You can save data in the old Stata 9 format by selecting the appropriate option when saving or by using the `saveold` command.

datasets to construct a new dataset containing the information we want:

1. system2008.csv — This is a comma-separated variable (csv) data file that contains all country-years between 1816 and 2008. It is consistent with the COW project's data on state system membership (e.g. all proper states in the world at any given moment in time).
2. warpartcy.dta — Contains all country-years during which a state actively participated in a war. This dataset includes a variable for the number of wars a state was participating in a given year.
3. wary.dta — A dataset that lists all the years between 1816 and 2008 as well as the number of wars ongoing during any given year.

The second and third datasets are already in the Stata format, but the first is not. One could use another program like StatTransfer to convert the file to the Stata data format, but we can also directly open it in Stata with the appropriate command. Looking at the help file for infiling data ("help infiling"), we can gather that *insheet* is the simplest command that will let us read this data into Stata:

```
. insheet using "lab1\system2008.csv", comma
(3 vars, 14393 obs)
```

There are several commands that you can use to examine the data we just opened in Stata: "describe" (shows you data types for variables, how they are formatted, and labels), "summarize" (basic summary statistics for variables), or "tab" (tabulate values of a variable). On a side note, *stateabb* is a three letter abbreviation for state names, *ccode* is a unique identifier for each state, and *year* is the year of the current observation. Since all three of these variables really are only there to identify observations, it does not make much sense to get summary statistics for them (e.g. if you had a list of credit card numbers, would you really care about the mean or median of those numbers?). But we can type "summarize" anyways:[6]

```
. summarize

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    stateabb |         0
       ccode |     14393    391.5753    249.6369          2        990
        year |     14393    1947.341    51.89074       1816       2008
```

At least we now know that there are 14,393 observations, and that our data cover the time period from 1816 to 2008. Tabulating year might be more useful (it will produce a long list):

```
. tab year


       year |      Freq.     Percent        Cum.
------------+-----------------------------------
       1816 |         23        0.16        0.16
       1817 |         23        0.16        0.32
```

[6]You could also type "su" or "sum" instead of "summarize". A lot of Stata commands have abbreviations. If you look at the help file for any Stata command, the underlined portion of the command is the minimum abbreviation for that command.

```
...
       2007 |        193        1.34       98.65
       2008 |        194        1.35      100.00
------------+-----------------------------------
      Total |     14,393      100.00
```

This shows us all unique values of the variable *year* and the frequency with which they occur (note that it hence makes little sense to use tabulate with continuous variables). In fact, what this tells us is that there were 23 states in 1816, 194 in 2008, etc.

Anyways, before moving on it might be useful to save this data. We could keep it in the format it was in, but it would be more convenient if we saved it in the Stata format. No problem, we can use the "save" command:

```
. save "lab1\warsovertime.dta"
file lab1\warsovertime.dta saved
```

4 Merging data: match

By itself what we have right now is next to useless. It would be nice to know which states fought in a war and when. The second dataset, "warpartcy.dta", has all country-years in which a state fought in a war, but it does not have country-years in which a state did not fight a war. To get the best of both worlds we can combine the two using the "merge" command to merge "warpartcy.dta" (using dataset) into "warsovertime.dta" (master dataset). Note that in terms of observations, all observations in the using dataset should correspond to an observation in the master dataset, but there are going to be at least some observations in the master dataset that do not correspond to an observation in the using dataset. The key is that there should not be any observations in the using dataset that do not have a corresponding observation in the master dataset.

You can look at the help file for the merge command for a full syntax. In our case, we want to merge country-years, which are uniquely identified by the variables *ccode* and *year* in both datasets.[7] When merging data you need to sort both the master and using datasets by the variables you match accross ("sort ccode year" will do that). Both of the datasets are already sorted in this case, and we already have the master dataset in Stata's memory, so we can merge right away:

```
. merge ccode year using "lab1\warpartcy.dta"
```

Nothing happened after typing the merge command. That generally is a good thing, since Stata will tell you about annoying things like the variables not uniquely identifying observations in one or both datasets (i.e. there are duplicate observations). More useful is the variable that the merge command generated, *_merge*. For each observation, this variable contains information about whether that observation occurred in both the master and using data (*_merge*=3), or only in the master or using dataset (1 and 2 respectively). Given our prior expectations, there should be 1's and 3's, but not 2's. You can tabulate *_merge* to see if that is actually the case:

```
. tab _merge

     _merge |      Freq.     Percent        Cum.
------------+-----------------------------------
```

[7]You will want to actually check that that is true. In this case it is, and both datasets are already sorted as well.

```
          1 |     13,752       95.55       95.55
          3 |        641        4.45      100.00
------------+-----------------------------------
      Total |     14,393      100.00
```

Wonderful, this seems to indicate that we merged successfully. Values of 2 (or more generally unexpected values given the datasets you are merging) would have indicated that something went wrong or was wrong with the data to being with (i.e. more investigation would have been necessary). We can drop the variable now since we do not need it anymore:

```
. drop _merge
```

5 Collapsing data

The dataset we now have contains all country-years as well as a variable identifying whether a state participated in 1 or more wars in a given year. If you browse the data ("browse") however, two problems become apparent. First, the variable *warfight* is missing for country-years in which the state did not fight a war, but we probably want it to be zero instead. Second, we do not care if a state fought in 1 or 2 or 3 wars in a given year. All that matters is that it fought in at least 1 war. Both of these issues can be solved at the same time with the "recode" command. We want to (1) recode missing values as 0, and (2) recode anything larger than 0 (in practice 1's, 2's, and 3's) as 1:[8]

```
. recode warfight (. = 0) (1/3 = 1)
(warfight: 13778 changes made)
```

The expression "1/3 = 1" in the second rule does not mean one-third equals one, but rather indicates to state that we want to recode all values between 1 and 3 (i.e. 1, 2, 3) to 1. At this point summarizing or tabulating *warfight* will confirm that it only has the values 0 or 1.

At this point we have a country-year dataset that contains all proper states in the international system in any given year, as well as a variable that indicates which ones fought a war in any given year. What I really want to know though is how many states existed in any given year, and how many of them participated in wars. So I am really interested in a dataset in which years are the unit of observation, and where there are two variables indicating the number of states and the number of states that participated in a war. The "collapse" command allows us to do that:

```
collapse (count) states = warfight (sum) warfight, by(year)
```

The last part of the command specifies that I want Stata to collapse the current data by year, which implies that year will be the new unit of observation in my dataset. The first expression in the command itself, "(count) states = warfight", says that I want a new variable, *states*, the values of which will equal a count of nonmissing observations for warfight. This will give me the number of states in the international system in a given year. The second expression, "(sum) warfight", creates a variable called *warfight* that measures the number of states participating in a war in any given year. You can browse the data at this point to make sure that everything looks alright.

[8]You can summarize *warfight* to verify that in fact there are no states that participated in more than 3 wars in a given year.

6 Merging data: one to one

Now I almost have the data I want. I am only missing a variable that captures the number of wars ongoing in a given year, but I conveniently already have a dataset that has this variable—"wary.dta". We just need to merge datasets again. Because both datasets should contain exactly the same observations (all years between 1816 and 2008) but different variables, we will be doing one to one matching. I still use the same command, but now if everything went right all observations should have been present in both datasets. If there are some observations there were not, then there is a problem. Anyways, we can merge just as we did before:[9]

```
. merge year using "lab1\wary.dta"
```

Again tabulate *_merge*, and since all observation have the value 3, drop the variable and move on. Now I have the dataset I want. Before saving it though, it might be useful to label variables so I and others have some more indication about what they are other than a short and cryptic variable name:

```
. label var states "Number of states"

. label var warfight "No. States involved in war"

. label var wars "No. Ongoing wars"
```

At this point I can also save the dataset ("save "lab1\warsovertime.dta", replace").

7 Conditional commands

Many Stata commands can be made conditional (i.e. do something only if something else is true). Say I want to know the average number of states actively engaged in war during the 20th century. The summarize command will give me the mean for a variable in the whole dataset, but I only want the 20th century. Issuing the "summarize" command with an if condition will do that:[10]

```
. sum warfight if year<1900

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    warfight |        84    1.892857    2.866583          0         19
```

The "year<1900" part is called an expression in Stata language. Expressions that you use with "if" can involve relational and also logical operators ("help operator"). Note that if we only wanted to do something if the year was exactly 1900, you would write "year == 1900 " (two-equal signs) rather than "year = 1900". If you want to do something if two or more conditions hold, you can use logical operators to combine relational expressions. So if I only want to do something involving observations in the 19th century and only during years in which there was at least one war ongoing, I could add "if (year<1900) & (wars >= 1)" to my command (I added the parentheses for convenience, you do not need them).

Table 1 contains a list of logical and relations operators in Stata.

[9]After ensuring that both datasets contained a variable called "year" and sorting by *year*.

[10]If you want more summary statistics, like the median, add the option "detail" to the "summarize" command, e.g. "sum warfight, detail".

Table 1: Logical and relational operators in Stata

Logical		Relational	
&	and	>	greater than
|	or	<	less than
!	not	>=	greater than or equal
~	not	<=	less than or equal
		==	equal
		!=	not equal
		~=	not equal

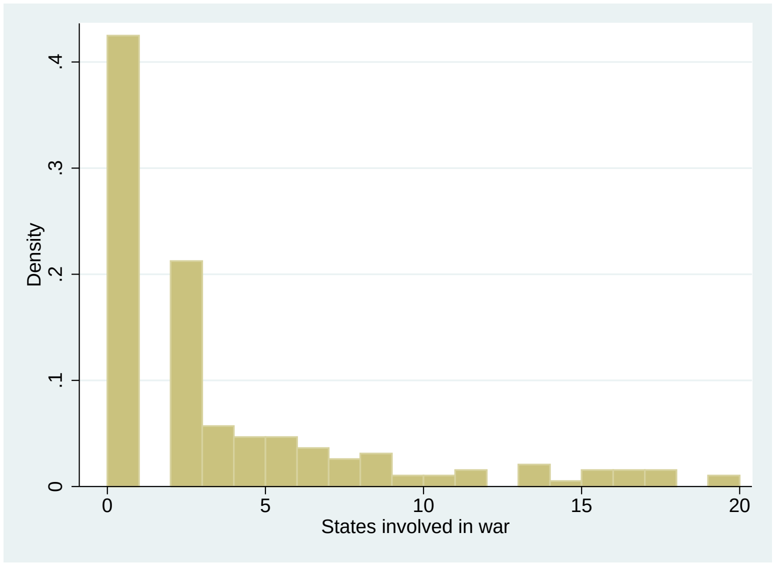
8 Graphing

Graphs can get fairly complex very quickly in Stata, but creating simple graphs is very easy. In particular, two kinds of graphs might be of interest here: (1) histograms, and (2) graphs of variables in your dataset.

Histograms show the distribution of values for a variable, and are roughly the graphical equivalent of what you get when you tabulate a (non-continous) variable. To create a histogram of the number of state fighting in a war in any given year, type “hist warfight”. Because *warfight* is an integer variable with fairly low values, it might make more sense in this case to force Stata to calculate and graph the frequency of individual integer values, rather than non-integer ranges, which you can do by specifying the “width()” option:

```
. hist warfight, width(1)
```

Figure 1: Histogram of *warfight*



Stata also has a myriad of other commands you can use to make various graphs. They all begin with “graph” and if you type “help graph” you will get to a good starting place to figure out what you need. Alternatively, you can also use the menus to produce whatever graph you need, and then copy the entire graph command from the output window.[11] Anyways, I want a graph that shows me the number of states in the international system, the number that fought in wars, and the number of ongoing wars for the time period my data cover:

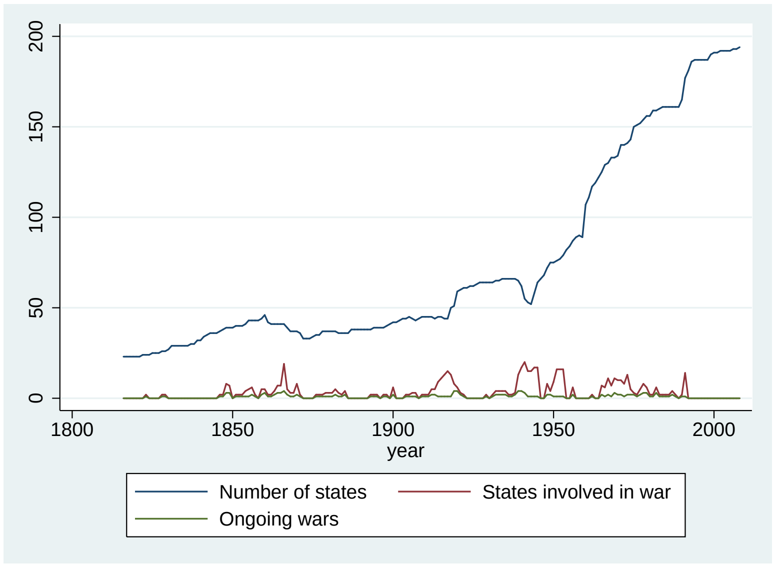
```
. graph twoway line states warfight wars year
```

There are a couple of things you can do with a graph other than look at it in Stata. At some point you may want to include it in a document you are writing. Stata saves graphs in it's own format that will not work with anything else, so you will have to convert the graph to something useful first using the "graph export" command. If you look at the relevant helpfile, you can see the file formats supported by Stata. I use eps here:[12]

```
. graph export "percentfighting.eps", replace
(note: file percentfighting.eps not found)
(file percentfighting.eps written in EPS format)
```

The resulting graph is shown in figure 2.

Figure 2: States in the international system and war



Rather than looking at the number of states involved, we may also care about the percentage of states fighting in a war at any particular time. To do that we need to create a new variable that captures that percentage:

```
. gen percfight = warfight/states

. label var percfight "Percent states engaged in war"
```

[11]One of the few cases where the menus are potentially useful in my opinion.

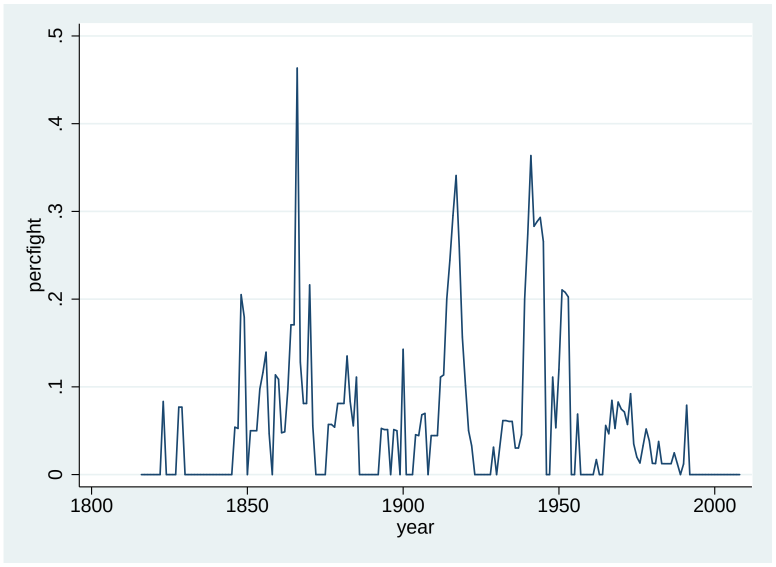
[12]If you use something like MS Word or whatever the equivalent in Mac is, you will probably want to save your graphs as png or pdf respectively. In the case any of you particularly care, I save figures as eps, convert to pdf, and then use them in LaTeX documents that I typeset with pdf2latex.

```
. graph twoway line percfight year

. graph export "percentfighting.eps", replace
(file percentfighting.eps written in EPS format)
```

This graph is shown in figure 3.

Figure 3: Percent of states fighting



9 Bivariate regression in Stata

Let's assume that I had some sort of argument that led me to believe that there was a relationship between the number of states in the international system and the percentage of states involved in war. To empirically evaluate that assertion, you could estimate an ordinary least squares (OLS) regression of the percentage of states involved in war. The Stata command for estimating and OLS regression is "regress y x", where y is the dependent variable and x is the set of independent variables. Stata by default includes a constant term as well, although you can specify an option to remove it. We could thus estimate an OLS regression like this:

```
. reg percfight states

      Source |       SS       df       MS              Number of obs =     193
-------------+------------------------------           F(  1,   191) =    8.64
       Model |  .055852214     1  .055852214           Prob > F      =  0.0037
    Residual |  1.23506601   191  .006466314           R-squared     =  0.0433
-------------+------------------------------           Adj R-squared =  0.0383
       Total |  1.29091823   192  .006723532           Root MSE      =  .08041

------------------------------------------------------------------------------
   percfight |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      states |  -.0003182   .0001083    -2.94   0.004    -.0005318   -.0001047
       _cons |   .0804616   .0099353     8.10   0.000     .0608646    .1000586
------------------------------------------------------------------------------
```

You will learn how to interpret this output later, but as a cautionary note let me point out that what I just did, other than showing you how to use the "regress" command, is potentially inappropriate for a number of reasons.

10 Remarks

Chances are that you are at least somewhat confused by now. I would recommend that you attempt to replicate the example I have here if you want to better understand what is going on. These notes give you the answers and most of the commands you need already, and you can always ask me for help.

A few more general comments on using Stata:

1. Always, always use do files and log files.

2. On a related matter, do use do files in situations where you create a new dataset from other people's data. As a rule of thumb, use do files every time you modify data that is not based on your own research.

3. Be liberal in commenting your do files and structuring them in ways that make it easier to read them. A blob of 50 lines of dense code is hard to follow. This is as much for your own, future benefit as for others'.

4. There probably is an easier solution to most potentially time-consuming problems in data management or analysis. You should not have to sit on Excel for hours to make changes to some existing data set, unless you are entering data.[13]

5. To find said solutions, and for general help on Stata, there are several useful resources: (1) Stata help files, (2) Stata manuals, (3) google, and (4) other graduate students. Do not underestimate the power of google, and if you do don't be surprised if you annoy graduate students by asking them questions to which a quick online search would have provided the answer.

References

Sarkees, Meredith Reid. 2000. "The Correlates of War Data on War: An Update to 1997." *Conflict Management and Peace Science* 18(1): 123–144.

[13]Ask me for an example—20 hours of coding versus 30 minutes of figuring out *one* Stata command.