SOC385L – Lab 1: Intro to Stata Coding

TA: Nicholas Reith

22 January 2016

> This lab assignment serves as an introduction to basic data cleaning, coding, and analysis in Stata, It may be a refresher for students already familiar with Stata basics.

These lab notes will cover:

1. Importing data,
2. Naming, labeling, and organizing variables,
3. Looking for missing data,
4. Summarizing and Tabulating data (Univariate Statistics), and
5. Recoding, dummy coding, and transforming variables.

Stata Orientation [1]

[1] NOTE: It is highly recommended that you use version 14 or, at a minimum, version 13 of Stata.

Main Interface

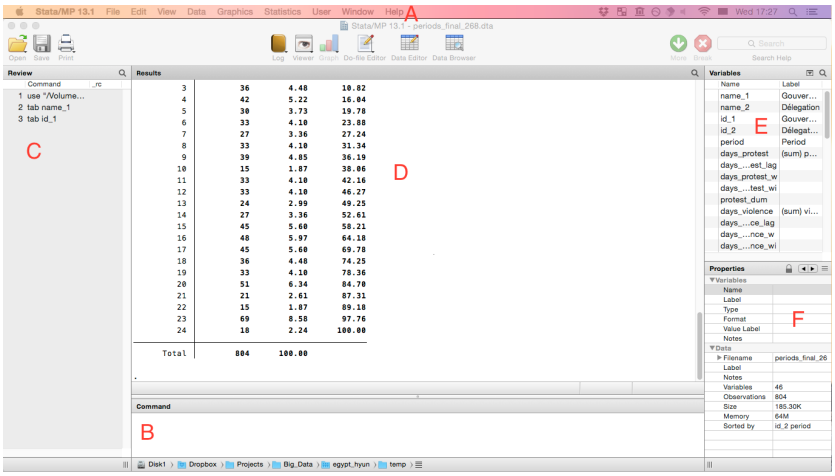

Figure 1: The Stata main interface window.

a. Menu bar: Do things manually with point-and-click.)

b. Command line: Enter code one line at a time.

c. Command history: A history of previous commands.

d. Results window: Output appears here.

e. Variables window: Lists all variables in dataset.

f. Properties window: More info on highlighted variable.

Other Windows

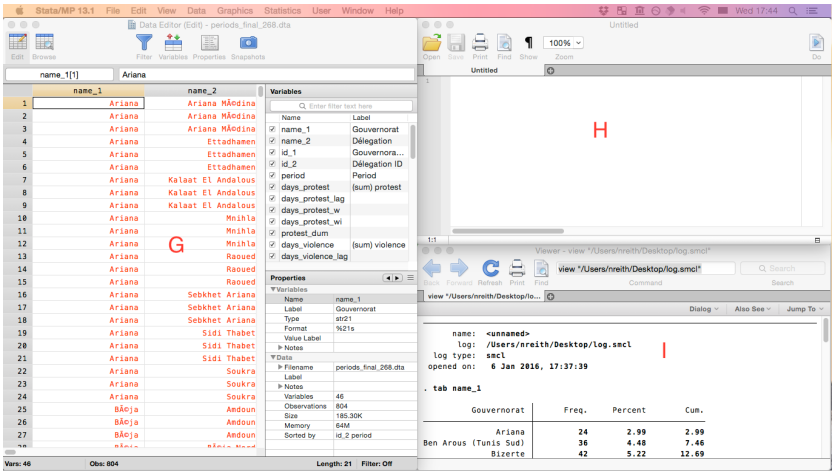

Figure 2: Additional Stata Windows.
g. Data Browser/Editor: View dataset.
h. DO File Editor: Edit and run Stata code.
i. LOG File Viewer: A record of logged output.

Folders and Files

- **Working directory:** Directory where all output is saved by default. Can be viewed with `pwd` (print working directory) command, or changed with `cd` (change directory) command.
- **LOG files:** `.smcl` or `.txt` files that record any logged output.
- **DO files:** `.do` files that save and run Stata code.
- **DTA files:** `.dta` files are Stata data files.

Stata Coding Tips

- Always code in a `.do` file so that you can save and edit your code.
- If you are stuck, look for help:
  - Type `help _____` in Stata
  - Google for official and unofficial Stata help online
  - Browse the Stata menus and help manual in the program. Point-and-click a command you forgot, then save the code.
  - Ask a classmate, TA, or Professor. It is often just a matter of finding the right terminology.
- Stay organized!
  - Remember to always set your working directory, and keep it organized with subfolders if appropriate.

  - Include annotations in your DO files, by beginning a line with an asterisk:

    `*this line is a comment`

    or by `/*bracketing an entire section with star-slash, like this. No matter how many lines, it will all be commented out and Stata will ignore those lines or parts of lines when running code. Stata will not, however, insert line-breaks automatically, so you need to hit enter if a line is too long.*/`

  - Three back slashes like so `///`

    tells Stata to continue reading the next line of code as one continuous line. This is useful for keeping long sections of code visible on the screen.

- Be careful of your syntax!

  - Most Stata code uses the following syntax, although not each section is required:

    ```
    by var1: command var2 var3 var4 in 100-150 if ///
    var3==3, options
    ```

- Common symbols include:

  - `=` or `==` (equals): *different uses you shall see*
  - `>=` (greater than or equal to)
  - `>` (greater than)
  - `<=` (less than or equal to)
  - `<` (less than)
  - `~=` or `!=` (not equal to): *equivalent*
  - `|` (or)
  - `&` (and)

- Working on multiple computers with different file paths? Or have something you want to run just in case, but not have Stata stop? Try the `capture` command. [2]

  e.g.,

  ```
  capture noisily cd "/Users/nreith/Dropbox/Projects/"
  capture noisily cd "/Volumes/Disk2/Dropbox/Projects/"
  capture noisily cd "T:/Projects"
  ```

  or,

  ```
  capture noisily log close
  ```

[2] `capture` tells Stata to keep on running the do file, even if the command returns nothing or an error message. `noisily` tells Stata to display output including error messages. For example, it can check multiple working directories and only change to the one that exists, or close a log file if one is open.

Accessing Stata Outside of Lab

There are several ways students can access Stata for use outside of class time.

- On-campus computer labs, including classrooms can be used when not in session.
- A student license for Stata may be purchased for six months, one year, or perpetual home use on your own computer. For details, see: `https://www.stata.com/order/new/edu/gradplans/`
- PRC computer labs on the second floor of CLA may be used by students affiliated with the Population Research Center.
- Students affiliated with the PRC also can get access to the PRC Stats Server with Stata 14. For details, see: `http://www.utexas.edu/cola/prc/for-affiliates/computing-services/stats-server.php`
- Students unaffiliated with the PRC have access to an older server provided by the UT Division of Statistics. For details, see: `https://stat.utexas.edu/consulting/stat-apps-server` [3]
- There are other methods of obtaining software are discouraged by the University, but might be revealed by the grapevine.

[3] NOTE: This method is only recommended as a last resort, because the server is older (MS Server 2003) and only has an older version of Stata 12. Stata 12 is sufficient for many of our analyses, but there are limitations, compatibility issues, and possible coding changes in newer versions.

Lab 1 Coding Assignment

Setup [4]

1. Log into lab computer, and open Stata
2. Log into Canvas and download lab materials.
3. In Stata command line, type `pwd` to locate working directory.
4. Click on "file menu" and change working directory to your USB.
5. Note the text that appears in Stata's output window.

[4] NOTE: It is suggested to use a USB for lab assignments, and to back up your USB regularly in case it is lost or corrupted.)

Setting DO and LOG files [5]

6. Click "Do-File Editor" icon
7. Click "save" from the file menu to save your `.do` file to working directory, giving it a descriptive name like `Lab1.do`
8. Setup the first few lines of your `.do` file like this:

[5] NOTE: For the remainder of the semester, we will be completing our lab assignments in DO files to save our annotated code, and saving output to a log file. Graphs and Tables will be saved to our working directory. All these should be assembled in one orderly document for the final portfolio.

```
clear all [6]
set more off, permanently [7]
cd "F:/Lab1/"
capture noisily log close [8].
log using "F:/Lab1/Lab1LOG.smcl", smcl replace [9]
```

[6] Clears memory.

[7] Turns off scroll breaks.

[8] Just in case a log was open from a previous run. See sidenote #2 on `capture noisily`

[9] NOTE: You will want to log all of your lab assignments, because this provides a record of both inputs (code) and output (results). If you properly comment your `.do` file and complete the assignment correctly, you can just copy and paste your log into Word, and add charts, graphs and tables as needed.

There are a couple of options to consider with logs:
1) An `.smcl` file maintains proper formatting, but can only be opened with Stata, whereas saving the log as a `.txt` file makes it portable and universal, but it loses all formatting. If you don't have access to Stata at home, you should save it as a `.txt` so you can open them on your computer.
2) You may choose to `replace` or alternatively `append` your log file, depending on whether you wish to keep all previous runs, or just the latest run of your code.

Importing Data:

Though there are dedicated programs such as STAT TRANSFER for converting datasets, Stata is capable of importing or exporting in a number of common formats. You'll need this, since most datasets are not published in Stata format.

As with most things in STATA, there are multiple ways to import data, which include:

- Manually entering the data through the command line or in matrix language.
- Copying and pasting the data from a table (such as excel) into the data editor, provided rows and columns are in the correct direction.
- Importing using the point-and-click menu to choose the correct format and browse to the file.

- Importing via the command line or a do file with a line of code specifying the file type and location.
- Double-clicking a .dta file, assuming program preferences are set in Windows/Mac.
- Dragging and dropping a file onto the output screen or command line.

In general, you will find it easiest to import the first time with the point-and-click menu, or drag and drop. But thereafter, you should save the code that STATA writes and use this in your do file.

For the lab today and next week (Lab Assignment 1), the data has already been provided in STATA format. The name of the STATA dataset is: `Lab1data.dta`

For more help on how to import data into STATA, check some websites such as these:

- `http://www.usc.edu/its/stats/stata/import_excel.html`
- `http://www.stata.com/support/faqs/data/newexcel.html`
- `http://www.wiwi.uni-muenster.de/ioeb/Downloads/Forschen/Pfaff/Introduction_to_Stata_with_50+_Basic_Commands.pdf`

Stata Data Cleaning, Coding & Organization

Now, on to the assignment!

The variables included in this dataset contain within them a number of typical issues that you will often see in "raw" or "dirty" data. This handout will provide hints on the issues in question and the commands you can use, as well as samples of code to address these issues. You will work through some of it on your own.

The dataset is a subset of variables extracted from the World Values Survey, a publicly available dataset with survey questions in numerous countries and five waves from 1981 through 2008.

`http://www.wvsevsdb.com/wvs/WVSIntegratedEVSWVSvariables.jsp?Idioma=I`

Let's get started:

Exploring the dataset:

There are a few useful commands for getting a quick look at your dataset and variables. These include:

- `desc` or `describe var` – Descriptive Info about the variable
- `tab` or `tabulate var` – A one-way table of frequencies
- `tab` or `tabulate var1 var2` – A two-way table of frequencies

- `sum` or `summarize var` – A list of summary statistics (use the option `, d` for details.)
- `tabstats var` – A table of summary statistics

---

9. Use the `describe` command above to get an overview of the dataset. How many observations do we have? How many variables?
10. Next, `tabulate` a few variables: tab s002 for example. How many waves are in this survey? How many observations are in each wave? `tab` other variables to answer: How many countries? Note how many observations per country. How many religious categories (religcat)?
11. Now `tab x003` (age) and then `sum x003, d` (age). What is the mean age of our respondents? The Standard Deviation?

Tabbing the Volunteering Variables (a081-a096): We will be working with variables that measure volunteering. Let us check if the relevant variables are available in both waves.

Sort the data by wave, in order to be able to explore this variable better.

```
sort s002
```

Now, we can check the "volunteer" variables by wave with `tab`, and the `by` clause.

`by s002: tab a081, m` [10]

[10] We add the `, m` option here, which is an abbreviation of "missing" to include missing observations in the tabulation.

---

12. Now you try a few more. Using the same code, tab some of the other volunteer variables (a081-a096) by wave.
13. What do you conclude? If we were to choose just one of the two waves (4 or 5) to work with, which should it be?

Dropping/Keeping Observations by Wave:

In order to keep observations, we simply use the keep command, carefully specifying which groups to keep with the if clause and `|` or `&` to separate multiple groups.

`keep if s002==x` [11]

In this case, we only wish to keep one of two waves. Using the code above, replace "x" with the number of the wave you have decided to keep based on the preceding tab you performed. Then tab the waves variable again to make sure it worked.

Dropping/Keeping Variables:

Now that we are finished keeping only the wave we want, we can look at some other variables to drop. Today we will focus on individual level variables, so we can drop some of the survey variables that are extraneous. These are `s001` (study) and `s008` (Interview number).

This time keep probably doesn't make sense, since there are only a few things we are dropping and many we are keeping.

The drop command follows the same syntax as the keep command, but this time we don't need any if clauses. We just write:

```
drop var1 var2 var3
```

---

14. Go ahead and write a bit of code to drop the two variables mentioned: `s001` and `s008`. Check the variables window and the output window to confirm it worked.

Combining/Generating Variabls

Coding volunteering:

Now that we have trimmed our dataset a bit, we need to begin thinking about how we would like to code some of our key variables. Coding should be thoughtful and informed by, or linked to, our theory and the design of our analysis. We may code variables based on how past researchers have coded them. But don't assume that previous research has coded a variable optimally. You may think of a way to code a variable better.

The questions `a081-a096` ask whether someone did "unpaid work" for the association. The text of the question in the survey is as follows:

Please look carefully at the following list of voluntary organizations and activities and say...
And for which, if any, are you currently doing unpaid voluntary work?
A081-A096 – List of Voluntary Organizations
0 'Not mentioned'
1 'Unpaid Work'

[11] Note the use of the double equal sign above (`==` rather than `=`).

15. Consider the wording, original coding and content of this group of questions. Thinking about our ultimate goal of analyzing, suggest some ways these variables might be recoded.

Creating a "count" variable:

Alone the variables each capture unpaid work for specific types of associations.

One way to recode them might be to group them into sub-types of associations such as secular vs. religious, or by thematic areas.

For our purposes we are interested in volunteering in general, so we will create a count variable of how many associations a respondent reports doing unpaid work for.

For this, we use the `egen` command with `rowtotal` extension (formerly `rsum` in older versions of STATA) and the `missing` option to ensure that we account for missing observations as well. The code to include in your do file is below.

```
egen volcount = rowtotal (a081-a096), m
```

`egen` is one possible command you can use in STATA to create a new variable, and the rowtotal part (previously called "rsum" in older versions of STATA) means that our new variable is equal to the sum of the number of non-zero answers in the group of variables specified for each respondent.

With this code, each individual respondent is given a number corresponding to the number of types of organizations for which they volunteer. The missing values generated are for those individuals who did not answer any of the 16 possible questions. [12]

[12] Note: For anyone missing only on some of the questions, these missing responses are considered to be zero in our new count variable.

16. Type `help egen` and take a moment to explore the help documentation for the `egen` command. As you can see, `rowtotal` is only one of many options that `egen` can perform.

17. Now `tab` the new variable `volcount` to see how it looks. What is the "mode" of this distribution?

18. Next, summarize the variable with the `sum` command. What is the mean number of types of organizations for which people volunteer?

19. Lastly, make a "histogram" of the new variable with the command `hist volcount` and comment on its distribution visually. Is it skewed? In which direction? What is the range of the distribution from min to max?

Creating and recoding a dummy variable:

We could code volunteering differently. For example, we might decide that, theoretically, what matters is whether someone volunteers for any organization, versus not volunteering at all.

So we could create a dummy variable for *"0=does not volunteer"*, and *"1=volunteers"*. This time we use the `gen` command to create a new variable equal to `volcount`, called `volany`, and then recode it accordingly.

```
gen volany=volcount
recode volany 2=1 3=1 4=1 5=1 6=1 7=1 8=1 9=1 10=1 11=1 ///
12=1 13=1 14=1 15=1
```

Another way to write this is:

```
recode volany 2/15=1
```

The recode above leaves missing and zero the same as the count variable, but changes all other values to one.

---

20. After running the code above, explore the new `volany` variable with the `tab` command. What do you conclude about most people from this quick summary? Hint: Pay attention to missing.

Cross-Tabs of Volany

A cross-tab is a two-way tabulation that includes two variables. This is performed with the same syntax, for example:

```
tab s003 volany, row
```

to see volunteering by country. The `row` option gives us percentages across the rows in addition to numbers of observations. `col` would do the same for columns.

---

21. Using cross-tabs, explore `volany` and other variables, one at a time, and try to answer these questions:

22. Which country has the highest percentage of people who volunteer? Which has the lowest?

23. Given that x001=1 if male and 2 if female, do men or women volunteer more?

Dropping original "volunteer" variables:

Now that we are done with creating our `volcount` and `volany` variables, let's clean up our variables list and get rid of the originals that are cluttering our screen.

`drop a0*` [13]

[13] We could write this as `drop a081-a096` but the asterisk in the code tells STATA to perform the command on all variables that begin with that prefix, which in this case are our volunteer questions. If we had variables with a suffix, we would say drop *suffix.

More Recoding and Reverse Coding:

Gender Equality:

We will now turn to variables `c001` (jobs scarce/men have more right), `d058` (equal income contribution of husband and wife), `d059` (men better political leaders).

---

24. Take a moment to look at the original coding and text of the questions from the World Values Survey Codebook below. Think about measurement and suggest some ideas for how to recode these.

C001.- Jobs scarce: Men should have more right to a job than women
Do you agree or disagree with the following statements?
When jobs are scarce, men should have more right to a job than women
1 'Agree'
2 'Disagree'
3 'Neither'

D058.- Husband and wife should both contribute to income
For each of the following statements I read out, can you tell me how much you agree with each. Do you agree strongly, agree, disagree, or disagree strongly?
Both the husband and wife should contribute to household income
1 'Agree strongly'
2 'Agree'
3 'Disagree'
4 'Strongly disagree'

D059.- Men make better political leaders than women do
For each of the following statements I read out, can you tell me how much you agree with each. Do you agree strongly, agree, disagree, or disagree strongly?
On the whole, men make better political leaders than women do
1 'Agree strongly'
2 'Agree'
3 'Disagree'
4 'Strongly disagree'

Recoding/Reverse Coding of Variables:

If you are writing a paper about gender egalitarian values, then you may want to consider reversing the coding of one or more of these variables. But, you have a choice. The first and third questions, `c001` and `d059` are coded so that misogynist values are coded more highly. The second question, `d058` makes a statement about gender equality, with higher values indicating agreement that men and women should both contribute equally to income in the household.

Scaling the variables in one direction or the other isn't wrong, but it might be easier for readers of your work to be consistent in the direction of coding. If the paper is largely about gender equality, we may choose to have higher values indicate agreement with gender egalitarianism. But in this case, we will scale them so that higher numbers equate to more patriarchal attitudes.

---

25. Using the sample code on page 10 that we used for `volany`, recode `d059` to reverse its scale and make it align with `d058`. Be sure to make an annotation in your do file to remember later that we have reverse coded this variable, and what the new values represent.

We would also like to code `c001` so that it corresponds to the direction of the other two. But we must consider another issue: the scale is not currently an increasing ordinal scale.

---

26. Recode `c001` so that `0=disagree`, `.5=neither` and `1=agree`. Make a note in your do file.

Transforming Town Size into "Urban/Rural":

Consider variable `x049`, which is town size. The scale of this variable is discrete, not continuous, but contains a number of town size ranges from "under 2000" to "over 500,000". See the original text of the survey question below.

X049.- Size of town
Size of town:
1 '2,000 and less'
2 '2,000-5,000'
3 '5,000-10,000'
4 '10,000-20,000/10,000-25,000; EVS81:10M-25M'
5 '20,000-50,000'
6 '50,000-100,000'
7 '100,000-500,000'
8 '500,000 and more'

---

27. Pause here for a moment and think about this scale. An inexperienced researcher might just throw this variable into an analysis. It is, after all, an increasing scale. But do the numbers 1-8 mean anything in terms of population or size of town? What coding solution would you suggest to make this more meaningful as a variable?

There are a number of ways to recode this variable. We will suggest two. First, we could recode each value to be the midpoint of the stated range.

---

28. First, use the `gen` command to create a new variable called townsize. Recode this variable so that each value 1-8 instead corresponds to the midpoint. For example, 2,000-5,000 would be 3,500. For 2,000 or less, code it as 1,000 and for 500,000 or more, code 750,000.

Another way to code the variable would be as a dummy variable – urban vs. rural. In this case, you would be theoretically more interested in the difference between urban and rural dwellers, not in population per se. Unfortunately, the United Nations does not agree on the precise population size definition of "urban," and countries vary greatly in their own definitions. So we'll make our own cutoff at 100,000.

---

29. Using the `gen` command you learned above, create a new variable from the original `x049` called urban_setting. Then recode it to be a dummy variable where `urban=1`.

Recoding Religious Attendance:

From Durkheim and Weber until today, many sociologists have looked at religion and religiosity as predictors of a number of outcomes. Religious attendance is one of a number of measures of religious commitment.

---

30. Look at the survey question below for the variable `f028` and consider its current coding. How would you propose to code it differently so that the time dimension in question is more meaningfully represented?

F028.- How often do you attend religious services
Apart from weddings, funerals and christenings, about how often do you attend religious services these days?
1 'More than once a week'
2 'Once a week'
3 'Once a month'
4 'Only on special holy days/Christmas/Easter days'
5 'Other specific holy days'
6 'Once a year'
7 'Less often'
8 'Never practically never'

If we think about the time dimension of religious attendance, the maximum possible might be daily and the least would be never. Since there are 365 days in a year, we can code the variable according to the number of days of attendance.

---

31. Create a new variable from `f028` with `gen` called `days_relig_service` and recode it so that `0=never`, `.5=less than once a year`, `1=once a year`, `2=special holy days`, `4=other specific holy days`, `12=once a month`, `52=once a week`, and `104=more than once a week`.

Renaming & Labelling Variables and Values:

This final part covers basic coding techniques to rename, reorganize, and label your variables.

Renaming variables

There are several ways to rename variables in STATA.

- The least efficient is with the variable manager through clicking and manually changing the names of variables one at a time.
- The most common is with the `ren` or `rename` command, which renames one variable.
- But when working with a lot of data, it is often useful to rename multiple variables at the same time by using the `ren` command with parentheses.

---

32. Run this code to rename one variable.

    ```
    ren x001 gender
    ```

33. Then try it again for at least one other variable.
34. Then try this code to rename multiple variables.

    ```
    renames (s003 x003) (country age)
    ```

35. Write another line of code to rename variables `x007`, `x011a` and `x047r`.

Labeling Variables Values

There are two main things to label, 1) Variables, and 2) Values of variables. The comprehensive method.

If you want to be comprehensive about it, you can give a description of the variable with the variable label, and then give exact labels to each value of the variable. Here is some sample code where we label `religcat` with this comprehensive method.

---

36. First, `tab religcat` in order to see how the values are currently labeled only with numbers.
37. Now, we use `label var` to give the variable a name.

    ```
    label var religcat "denomination"
    ```

38. Then we define the labels for the values with `label define`. I usually use the variable's name with an "l" for label at the end.

    ```
    label define religcatl 0 "No Religion" 1 "Catholic" 2
    "Evangelical" 3 "Protestant" 4 "Orthodox" 5 "Jewish"
    6 "Muslim" 7 "Hindu" 8 "Buddhist" 9 "Other"
    ```

39. And finally, we attach those new value labels to our variable with label values.

    ```
    label values religcat religcatl
    ```

40. Now, `tab religcat` again in order to see the change with the labels. [14]

41. Next, `tab X007` (or whatever you have now renamed it above) to see the current non-labeled values for Marital Status. Then, using the technique above, create labels for Marital Status from the original question below, and assign them to the variable. Tab it again to verify that your changes were successful.

[14] NOTE: For future reference, if variable values DO HAVE labels, and you would like to view the variable without them, you can always type the `, nolab` option when performing a tab. For example,

`tab religcat, nolab`

shows the numbers only.

X007.- Marital status
Are you currently ...
1 'Married'
2 'Living together as married'
3 'Divorced'
4 'Separated'
5 'Widowed'
6 'Single/Never married'
7 'Divorced, Separated or Widow'
8 'Living apart but steady relation (married,cohabitation)'

The Short-Cut Method

Personally, I like the above comprehensive method because I'm a bit of a perfectionist and like to see everything neatly labeled. But not everyone does and there are times when you need to save time in coding. The short-cut method is to skip labeling values and instead of having a description for a variable label, we put a note on the coding.

---

42. For example, try this code to recode gender to 0,1 instead of 1,2 and label it.

    ```
    recode gender 1=0 2=1
    label var gender "0=male, 1=female"
    ```

Compressing and Alphabetizing:

Here are some final things to do with your newly recoded dataset.

43. After all of this recoding, renaming, and labeling of values and variables, please type two small commands in order to finalize our organization of the dataset.

    `compress` [15]
    `aorder` [16]

[15] This stores all variables in their smallest possible form without losing data and helps keep our dataset footprint small, especially after we have created a number of new variables.
[16] This puts all of your variables in alphabetical order.

LAST SECTION! YAY!... PUTTING IT ALL TOGETHER

By now, you are all quite proficient coders and have a nicely cleaned up, recoded, organized, named, and labeled dataset.

In the following lab sessions we will be delving a bit more into analysis, but for now we will leave you with a few exploratory questions that you can accomplish with the skills you already have. Please think about these and we can discuss them during the beginning of the next lab session.

Explore your newly cleaned dataset using some `tab` and `sum` commands on various combinations of variables, with `by` and `if` clauses.

44. Of all of the categories of religion (including "No Religion" and excluding "missing"), which group has the highest percentage of members who volunteer? Which group has the lowest?

45. On the three gender questions, compare men only, by country. In which country do men have the most patriarchal views? In which country do they most strongly support gender equality?

46. Earlier, you labeled marital status. But would you use this variable as it is currently coded? Think about how you might recode this variable. And how might your coding choice depend on your research question? For example, how might you code this variable if "mental health" was your outcome? What if "trust in others" was your outcome?

47. Finally, end your session with the following commands to close your log and save the new data: [17]

    ```
    log close
    save "Lab1Data_Revised.dta"
    ```

[17] NOTE: Now that you have perfected all of your code in the DO file, and have made sure to also save your do file, if you wish, you can run it again from the beginning in order to produce a clean log without all of the error messages that you may have gotten while experimenting the first time.