Introduction to Stata

Training Workshop on the Commitment to Equity Methodology
CEQ Institute, Asian Development Bank, and The Ministry of Finance
Dili
May-June, 2017
What is Stata?

• A programming language to do statistics
• Strongly influenced by economists
• Open source, sort of
  • You can see how Stata codes many of its commands
  • You can add your own commands to Stata
  • You can publish commands for others to use
• An acceptable way to manage data
How Stata Works

• You can work interactively, through a user interface
• For serious work, it is much better to write programs
  • Stata calls these “do files”
  • Allows reproduction of your results
  • Allows identification and rectification of errors
  • We still run these “do files” interactively, mostly

• Stata does everything in RAM, except reading data (from other places) and saving data (to other places)
  • To do anything, you must load data into RAM
Where Your Computer Stores “Data” (Stuff)

• RAM (random access memory)
  • very fast
  • “forgets” what it had when the power goes out

• Disks (hard drives, usb drives, dvd’s, etc)
  • Much slower
  • But stable – they remember what’s recorded on them when the power goes out

• Internet (“the cloud”, file servers, etc)
  • Slower still
  • But vast

• To work in Stata, you must “load” or “read” data stored on a disk or the internet into the RAM
Three Topics for Today

• Statistics
  • Stata is really good at generating sophisticated statistical analyses
  • But we will mostly skip this

• Data management
  • This is about getting data into the RAM (and Stata) and then getting (other) data out to permanent storage
  • And about manipulating data – creating new variables, modifying existing ones
  • And about manipulating datasets
    • Mostly, merging two different datasets

• Programming
  • How to keep a permanent record of what you’re doing
  • How to manipulate data efficiently
How to Load Data into Stata

• Many options
  • Type them in by hand (ugh)
  • Manually copy from Excel or Word or a text processor and paste into Stata’s data editor window
  • Read them from a comma- or space-delimited file
  • Read them from an Excel spreadsheet
  • Read them from a Stata dataset (Stata extension is .dta)

• We will focus on the last two, which are the most common

• Stata has commands for each
  • use <dataset path and filename> for Stata datasets
  • Import excel <spreadsheet path and filename>
An Aside on (Sub-)Directories and Paths

- Windows (or Mac OS) must organize its many files stored on a disk. In Windows, it’s like this:
An Aside on (Sub-)Directories and Paths

• The sub-directories, or folders, help you (and Windows) keep files organized

• To read a file, you need to tell Stata where it is
  • Requires a path (subdirectory) ...
  • ... and filename

• For example:
  • use c:\CEQ_Timor_training\stata\data\??\.dta
    • This is a Stata command to read the Stata dataset ??\.dta into the RAM so Stata can work on it
  • import excel using "c:\CEQ_Timor_training\info_for_examples\inc_dist.xlsx", sheet("Gini") cellrange(B4:F15) firstrow
    • This is a Stata command to read part of an Excel spreadsheet
Loading Data from Excel – Let’s Try It

• First, check that the Excel file is on your disk:
  • dir "c:\CEQ_Timor_training\info_for_examples\"
• Now go look at that spreadsheet (with Excel)
• Import the data:
  • import excel using "c:\CEQ_Timor_training\info_for_examples\small_data.xlsx", sheet(“HH_1") cellrange(A3:E8) firstrow
• See what you imported:
  • list *, clean
  • describe
• Put some labels on the variables
  • label var hhid “Unique household id”
  • etc
Labeling and Saving Data

• Label the dataset:
  • label data “Practice dataset #1, household data”

• Sort the data:
  • important for us to be able to merge later
  • sort hhid

• Describe the data again

• Save the data:
  • first, check the default (sub)directory:
    • pwd (“present working directory”)
  • now save:
    • save “c:\CEQ_Timor_training\info_for_examples\HH_1”

• And load the data again (now a Stata dataset)
  • use “c:\CEQ_Timor_training\info_for_examples\HH_1”
Manipulating Data

• Create a new variable, income per capita
  • generate income_pc = income/hhsize
  • label var income_pc “HH Income per capita”

• Create a new variable, conditional on some criterion
  • generate income_pa = income/hhsize if income>500
  • list what you got
  • generate poor = (income_pc<500)
  • list what you got

• Label the values of a variable
  • label define poorstatus 0 “Non-Poor” 1 “Poor”
  • label values poor poorstatus

• Save the data again ...
Structure of (Almost) All Stata Commands

• verb variable(s) <if ...> [weights], options
• verb is the command
• variable(s) are the variables to operate on
• if ... is to subset the command to only some observations
• [weights] are to apply different weight to each observation
  • Stata has several types of weighting schemes
• options are command-specific, and always come after a comma, at the end
Merging Data: one-to-one merges

- Sometimes we merge datasets that have one record (row of data) for each value of the variable we are merging on (for example, hhid):

<table>
<thead>
<tr>
<th>Dataset 1</th>
<th>Dataset 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>income</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>80</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
</tr>
<tr>
<td>5</td>
<td>70</td>
</tr>
</tbody>
</table>

- command syntax is:
  - merge 1:1 <merge variable> using <name of dataset 2>
  - to work dataset 1 must be loaded into RAM
  - to work, both datasets must be sorted by the merge variable
Merging Data: one-to-one

• This merge also works as a one-to-one merge:
  • merge 1:1 id using dataset2

<table>
<thead>
<tr>
<th>Dataset 1</th>
<th>Dataset 2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>income</td>
<td>id</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>80</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>70</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>

• Note that Stata leaves missing value codes where it found no data
Merging Data: one-to-one

- This merge does not work:
  - merge 1:1 id using dataset2

<table>
<thead>
<tr>
<th>Dataset 1</th>
<th>Dataset 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>income</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>80</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
</tr>
<tr>
<td>5</td>
<td>70</td>
</tr>
</tbody>
</table>

- What went wrong?
Merging Data: one-to-one

• This merge works, but is wrong:
  • merge using dataset2

<table>
<thead>
<tr>
<th>Dataset 1</th>
<th>Dataset 2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>income</td>
<td>id</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>80</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>70</td>
<td></td>
</tr>
</tbody>
</table>

• What went wrong?
• This is the most dangerous merge mistake!
  • Avoid it by always using 1:1 or n:1 or 1:n in your merge command
• Note what happens to “id” – no overwrite
Merging Data: one-to-one

• This merge does not work:
  • merge 1:1 id using dataset2

<table>
<thead>
<tr>
<th>id</th>
<th>income</th>
<th>id</th>
<th>HH size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>80</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>70</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

• What went wrong?
Merging Data: one-to-one and n-to-one

• Some datasets may have multiple observations for each unique observation in another dataset
  • household-level data vs. individual-level data
  • Stata will do an “n-to-one” or “one-to-n” merge here

• Load the individual-level data from the spreadsheet
  • import excel using "c:\CEQ_Timor_training\info_for_examples\small_data.xlsx", sheet("indiv") cellrange(A3:D20) firstrow
  • sort hhid pid
  • save "c:\CEQ_Timor_training\info_for_examples\indiv"

• Now merge in household data
  • merge n:1 hhid using "c:\CEQ_Timor_training\info_for_examples\HH_1"
Merging Data – Practice

• Load and save the data in tabs HH_1 and HH_2
  • import excel using "c:\CEQ_Timor_training\info_for_examples\small_data.xlsx", sheet("HH_1") cellrange(A3:E8) firstrow
  • save "c:\CEQ_Timor_training\info_for_examples\HH_1"
  • import excel using "c:\CEQ_Timor_training\info_for_examples\small_data.xlsx", sheet("HH_2") cellrange(A3:C8) firstrow
  • save "c:\CEQ_Timor_training\info_for_examples\HH_2"
  • use "c:\CEQ_Timor_training\info_for_examples\HH_1", clear

• Merge the data
  • merge using "c:\CEQ_Timor_training\info_for_examples\HH_2"
  • oops
  • merge hhid using "c:\CEQ_Timor_training\info_for_examples\HH_2"
Merging Data - Practice

• Load the second dataset again
  • use "c:\CEQ_Timor_training\info_for_examples\HH_2", clear
  • sort hhid
  • save "c:\CEQ_Timor_training\info_for_examples\HH_2", replace

• Now load the first dataset again
  • use "c:\CEQ_Timor_training\info_for_examples\HH_1", clear
  • merge hhid using "c:\CEQ_Timor_training\info_for_examples\HH_2"

• Check results
  • list, clean
  • desc
Aggregating or “collapsing” data

• Sometimes we would like to add up several rows of data for each household, like this:

• the “collapse” command can do this
Collapsing Data - Practice

• Load the third dataset
  • use "c:\CEQ_Timor_training\info_for_examples\HH_3", clear
  • sort hhid

• list what you have

• collapse (sum) cons, by(hhid)

• list what you have

• try it again, after reloading the data, using
  • collapse cons, by(hhid)

• try it again, after reloading the data, using
  • collapse (sum) cons
Programming – Writing “do files”

• It is very bad form to do research with interactive or point-and-click commands

• Programs (do files):
  • keep a record of what you have done
  • allow you (and others) to cross-check your work
  • make it very easy to make small changes to your research

• Goal: Let’s write a do-file to do this:
  • read all the data, both HH and individual, in the spreadsheet
  • clean the error in hhid
  • merge them together
  • create HH income per capita and per adult equivalent
  • tabulate average HH income per capita by area of residence
Programming – Writing “do files”

• Stata has an internal text editor, like a word processor
  • start it with ctrl-9, or the “window” menu
• Enter the commands we have learned, in order
• run them: ctrl-D or the “tools” menu
• Add comments
  • very important for good programming
  • help you remember what you are doing
• Locals and globals – place-keepers
  • for example, use a global for the path
  • or a local for a specific value
  • or a local with a list of variables
Programming – locals and globals

• Locals and globals are “place-keepers” you can use in your do-files
  • globals stay active until you close Stata
  • locals stay active only until your do-file finishes running
  • for example, use a global for the path
    • global datadir c:\CEQ_Timor_training\info_for_examples\n    • then this: use ${datadir}HH_1
    • is the same as: use c:\CEQ_Timor_training\info_for_examples\HH_1
    • in general, programmers do not like to use globals
  • or a local for a specific value
    • local schoolfee 500
    • Then these two are the same:
      • generate cost = in_school*500
      • generate cost = in_school`schoolfee’
  • or a local with a list of variables
Programming – locals and globals

• Use a local for a list of variables
  • local vnames “ hhid income hhsize ”
  • then summarize ‘vnames’ is the same as summarize hhid income hhsize

• There are other uses for locals, to come later
Programming – Looping

• Looping is when you ask the computer to do the same operation many times.
• Stata has several ways to loop, but the foreach command is easiest

• Looping with foreach
  • foreach <local> in <list> {
    • ... do something to every item in the list ...
    • }

For example:

```stata
foreach nn in hhsize eqscale {
    generate income_`nn' = income/`nn'
}
```

Or:

```stata
local namelist " hhsize eqscale "
foreach nn of local namelist {
    generate income_`nn' = income/`nn'
}
```
Exercise

• Write a do-file to:
  • read and merge all the data, both HH and individual, in the spreadsheet: c: \CEQ_Timor_training\info_for_examples\small_data.xlsx
  • clean the error in hhid
  • merge them together
  • allow adding an arbitrary value to HH income if the HH is rural
  • create HH income per capita and per adult equivalent
  • tabulate average HH income per capita by area of residence
  • find out how many secondary graduates there are per HH

• Comment it nicely
• Use a local and a global
• Use a loop when you can