Econometric Softwares: Stata

Jérémy Do Nascimento Miguel

Class 2 Spring 2024

Session 2 outline

- Data structure
- Duplicates
- Transforming variables to numeric/string format, adding value labels
- Generating group-level/aggregate variables
- Reshaping data: long vs. wide format data
- Combining data: merge vs. append

Data structures

- **Cross-sectional**: data with observations at a single point in time
- **Time-series**: sequence of data points observed over time for a single unit (e.g. individual, country etc.)
- **Panel**: data with many units (individuals, countries etc.) observed over time

How to identify data structure?

- Prior knowledge about data in hand
- Data documentations
- Checking identification numbers (unique number assigned to each unit of observation)

Data Structure

Cross-sectional:

ID	X	Y
1	100	2
2	250	4
3	150	3
4	200	7
5	300	9
6	400	5

Time-series:

time	X	Y
2007	100	2
2008	250	4
2009	150	3
2010	200	7
2011	300	9
2012	400	5

Panel:

ID	time	X	Y
1	2007	100	2
1	2008	250	4
1	2009	150	3
2	2007	200	7
2	2008	300	9
2	2009	400	5

Data structures I

isid checks whether specified variables uniquely identify the observations in the data

- isid *varlist*
- For instance, your data is panel, and the variable id is identification number and time indicates year
- If these variables uniquely identify observations, then following code should give no error/message: isid id year

Data Structures

To enable Stata commands specific to time-series or panel data, sometimes data structure has to be declared

- tsset declares data as time-series: tset *timevar*
- xtset declares data as panel. xtset *panelvar timevar*

Duplicates

duplicates offer set of commands that report, tag, list, or drop duplicate observations in the data

- duplicates report shows the total number of observations and the number of duplicates
- duplicates report varlist checks for duplicates by variables specified in varlist (similar to isid?)
- duplicates tag, generate(newvar) generates a new variable called newvar which is equal to 0 for unique observations or the number of duplicates observations
- duplicates drop drops overall duplicates (i.e. across values of all variables)
- duplicates drop varlist, force force dropping observations with duplicates by varlist variables

Exercise 1: 10 min

There are three data files: data1.csv, data2.csv, data3.csv. First, set/create a working directory for this session and store the data files in the same folder. Do following tasks for **each data**:

1. Import data in Stata and try to figure out whether it is cross-section, time-series or panel.
2. Check if there are any duplicates across all variables. Drop them if any.
3. Check whether there are any ID variable(s) that uniquely identify observations. Hint: you may want to use isid or duplicates commands.
4. After finding unique identifier(s), declare data structure if necessary (only for time-series and panel data).
5. Save data with a name corresponding to its data structure.

Transforming variables

Converting string variable to numeric: destring varlist, replace

- Option replace replaces the variable, but possible to generate new variable using option gen(newvarlist)
- If there are non-numeric characters in the variable, they can be specified in option ignore("chars ") to ignore them
- Option force sets observations with non-numeric characters to missing

Converting numeric variable to string: tostring varlist, replace

- similarly, possible to generate new variable using gen(newvarlist)
- helpful to generate unique ID variables that may contain non-numeric characters, e.g.:

```
tostring clustcode, replace
tostring hhnumber, replace
gen hhid = clustcode+hhnumber
```

Value Labels

Assigning labels to values:

- Define value label: label define labelname # "label " [# "label "]
- Assign defined value label to variable: label values varlist labelname E.g.: label define genderlab 1 "Female" 2 "Male" or label values gender genderlab

Self-study: check what do Stata commands encode and decode do and try to run it on any data you like

Function-based Variables

- 'Extension' to generate: egen varname1 = function(varname2)
- Functions can be mean(), max(), sum(), count() etc. See full list with help egen
- Functions can do 'row-wise' or 'column-wise' calculations
- 'Row-wise': across variables for each observations (names of these functions start with **row*()**, e.g. rowmax(), rowmean())
- 'Column-wise': across observations for each variable (those which do not start with **row*()**, e.g. max(), mean())
- egen often used with by or bysort to generate group-level variables (e.g. mean by group)
- Usually requires data to be sorted, so better to use bysort by default
- E.g. to generate mean wage across regions: bysort regionID: egen regionwage = mean(wage) Or to generate the number of people aged below 17 by gender bysort gender: egen aged17below = sum(age<=17)

Exercise 2

- Import data file ”somedirtydata.csv”.
- Quickly check for data structure (by browsing), duplicates and any identifiers.
- Convert variables **hhcons** and **zipcode** to numeric variables. Remove non-numeric characters from their values (i.e. don’t use force option)
- Assign value labels to variable **poor**, where 1s should be labelled as ”poor” and 0s should be ”non-poor”.
- Generate variable measuring household size (= sum of the number of adults and children in the household)
- Generate poverty rate by zip code (using bysort and egen commands)
- Generate maximum HH consumption (hhcons) by zip code.
- Save your data as ”somecleandata.dta”.

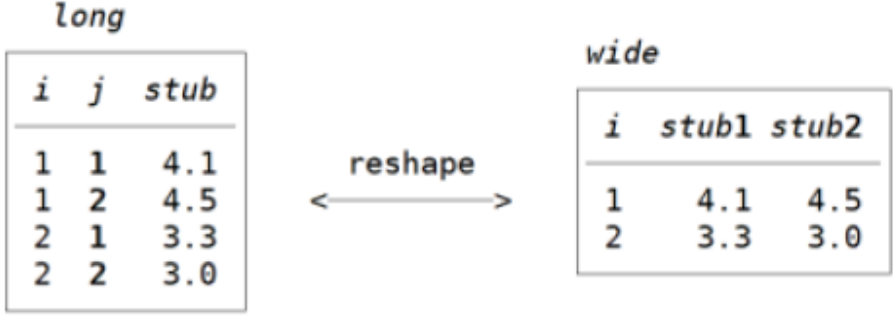
Reshaping Data

Before combining datasets or data analysis, make sure it is in the correct format

- **Data in wide-format**: e.g. if country-year panel data, separate 'columns' for each year, and rows for country
- 'Reshape' to long format → country X year combinations in rows, variables in columns
  - → reshape long varlist, i(countryid) j(year)
  - → Years are put at the end of variable names, command will automatically generate variable year based on variable names
  - → or you have to specify where is j-identifier (year) in the name of variables using @: E.g.: reshape long gdp@ fdi@, i(countryid) j(year)
- Opposite scenario: **long-to-wide reshaping**
  - → reshape wide varlist, i(countryid) j(year)
  - → here variable year already exists, and the command will put year values at the end of variable names of varlist

Reshaping' data

From help menu



Self-study: check what do Stata command xpose does and try to run it on any data

Combining datasets

Adding variables from other data ('using' data) for the observations in the data in memory ('master' data): merge [1:1 / m:1 / 1:m] IDvars using filename

- 1:1 → for every observation in the master data, there is an observation in the using data
- m:1 → many observation in the master data are matched to one observation in the using data
- 1:m → every observation in the master data is matched to many observations in the using data
- IDvars should uniquely identify observations in both datasets (1:1), in the using data (m:1), or in the master data (1:m)
- merge command generates variable named $_merge$ that records matching outcomes (1=master only, 2=using only, 3=matched)
- Options → explore help menu

Adding new observations (with the same variables), i.e. appending data: append using filename

Exercise 3 I

1. Reshaping:
   - → Import data file ”cpi zipcode.csv”. This (hypothetical) data contains consumer price index by zip codes for 2015, 2016 and 2017.
   - → Reshape it to long format, so that for each zip code there should be three rows with each row for one year. Note that you should also have a new variable year after reshaping.
   - → Now reshape it back to wide format ;)
   - → Save your data as ”cpi zipcode.dta”.
2. Merging:
   - → Open ”cpi zipcode.dta” and merge it with the data from previous exercise ”somecleandata.dta” using zip codes. Note that ”somecleandata.dta” should contain zip codes in numeric format.
   - → Self-study. Try merging other way around: first open ”somecleandata.dta” and merge ”cpi zipcode.dta”. Make sure that the result is the same as in previous merging.

Exercise 3 II

3. Appending:
   - → Import data files ”data2015.csv” and ”data2016.csv” and save them as DTA data files.
   - → Combine these data files so that the result will be