STATA - Create the Sample for Analysis: Part 2

Prof. Tzu-Ting Yang

Institute of Economics, Academia Sinica

April 12, 2022

Create the Sample for Analysis: Part 2

- Please see **C5_create_sample_part2.do**

mvdecode:Replace all the missing value quickly for many variables

STATA Command: mvdecode

Replace all the missing value quickly for many variables

```
1 mvdecode ftotval inctot incwage, mv(9999 9999999)
```

- In some datasets, missing values are denoted by specific numbers (e.g. 99999999)
  - Not denoted by one of Stata's reserved values for missings (i.e. a dot)
- **mvdecode**: replace all the missing value quickly for many variables
  - Replace 9999 or 9999999 to missing (e.g. a dot)

STATA Command: mvdecode

Replace all the missing value quickly for many variables

	ftotval	inctot	incwage
1	207803	101500	15000
2	207803	77300	42000
3	29003	29003	29000
4	29003	99999999	9999999
5	29003	99999999	9999999

STATA Command: mvdecode

Replace all the missing value quickly for many variables

	ftotval	inctot	incwage
1	207803	101500	15000
2	207803	77300	42000
3	29003	29003	29000
4	29003	.	.
5	29003	.	.

Pay Attention to Missing Values

Pay Attention to Missing Values

```
1 gen incwage_test = incwage
2 replace incwage_test = . if incwage==0
3 gen largewage=(incwage_test>=30000) if incwage_test
      !=.
4 gen largewage1=(incwage_test>=30000)
5 tab largewage
6 tab largewage1
```

- STATA treat **missing value .** as large number so we should pay attention to this issue when we create a variable

Pay Attention to Missing Values

```
. tab largewage
```

largewage	Freq.	Percent	Cum.
0	12,388	28.85	28.85
1	30,553	71.15	100.00
Total	42,941	100.00	

```
. tab largewage1
```

largewage1	Freq.	Percent	Cum.
0	12,388	20.65	20.65
1	47,612	79.35	100.00
Total	60,000	100.00	

keep: Keep the variables or observations that satisfy if statement

STATA Command: keep

Keep the variables or observations that satisfy if statement

```
keep serial year ftotval inctot incwage
keep if year>=2015
keep if inctot != .
keep if (incwage>=30000 & incwage~=50000) | (inctot
    >=30000 & inctot~=50000)
```

- The original dataset may contain variables you are not interested in or observations you don't want to analyze
- **keep**: keep the variables or observations that satisfy if statement
- | means "or"
- & means "and"

drop: Drop the variables or observations that satisfy if statement

STATA Command: drop

Drop the variables or observations that satisfy if statement

```
1 drop region-county
2 drop if year==2015
3 drop if ftotval == .
```

- **drop**: drop the variables or observations that satisfy if statement
- Note that the hyphen sign (-) is a useful shortcut
  - **Line 1:** indicates all the variables between "region" and "county" are to be dropped

STATA Command: drop

Drop the variables or observations that satisfy if statement

```
1 tostring year,gen(year1)
2 drop if year1 ~= 2015
3 drop if year1 ~= "2015"
```

- Note, with **string variables**, you must enclose the observation reference in double quotes " "
- Otherwise, Stata will think that 2015 refers to a variable and claim not to be able to find what you are referring to.

STATA Command: _n / _N

```
1 keep if _n<=50000
2 drop if _n==_N
```

- Both _n and _N are in-built system variables
- The upper case N refers to the last number of observation
- Lower case n always refers to the number of each observation

STATA Command: _n / _N

```
1 keep if statefip[_n] != statefip[_n-1]
2 keep if statefip != statefip[_n-1]
```

- A variable name followed by square brackets means that we want to refer to a certain observation
  - [1] would mean the first observation
  - [_N] would mean the last observation
  - A relative index [_n] means the current and [_n-1] the observation before the current observation

STATA Command: _n / _N

```
1 keep if statefip[_n] != statefip[_n-1]
2 keep if statefip != statefip[_n-1]
```

- You may want to keep only a single occurrence of a specific observation type
  - For example, just the first observation of each state code
- Stata starts at observation number one and applies the command
- Then moves onto observation two and applies the command again, then onto three and so on.

STATA Command: _n / _N

region	statefip	asecflag	hflag	county
new eng…	maine	asec	5/8 file	0
new eng…	new ham…	asec	5/8 file	0
new eng…	vermont	asec	5/8 file	0
new eng…	massach…	asec	5/8 file	0
new eng…	rhode i…	asec	5/8 file	0

order: Do some cosmetic changes to the order of your variable list

STATA Command: order

```
1 use "$rawdata\cps_2014_16.dta",replace
2 order *, alphabet
3 order incwage year, first
4 order incwage, after(year)
```

- **order**: This command can be used to do some cosmetic changes to the order of your variable list in the variables window
- **Line 1:** To sort variables alphabetically, simply use the **alphabetic** option
- Sort them by hand the **first**, **last**, **before** and **after** options are used
- **Line 3:** sort "incwage" "year" in the first two variable
- **Line 4:** sort "incwage" after "year"

sort: Sort observations by specific variables

STATA Command: sort

Sort observations by specific variables

```
1 sort serial year statefip
```

- **sort**: sort observations by specific variables
- **Line 1:** This command first sorts the data by "serial" (ID), and then within each "serial" code it sorts the data by "year"
- Then within each "serial" and "year", it sort the data by "statefip"
- Note that sorting is in **ascending order** (A,B,C or 2014, 2015,2016).

gsort: Sort observations in more complicated way

STATA Command: gsort

Sort observations in more complicated way

```
1 gsort -year
2 gsort -year statefip
```

- **gsort**: sort observations in more complicated way
- **Line 1:** Note that you need to **place a minus sign** before every variable you want to sort in **descending order**
- **Line 2:** This command allows you to sort in complicated ways, e.g. to sort "year" in descending order but then "statefip" (state code) in ascending order

bysort: Sort observations within a subgroup

STATA Command: by/bysort

Sort observations within a subgroup

```
1 sort year
2 by year: sum incwage
3 bysort year: sum incwage
```

- **Line 1-2:** You can re-run a command for different subsets of the data using the **by** prefix
- **Line 3:** Note that you have to either sort the data first or use the **bysort** prefix

STATA Command: by/bysort

```
. sort year

. by year: sum incwage
```

-> year = 2014

Variable	Obs	Mean	Std. Dev.	Min	Max
incwage	**19,999**	**2156954**	**4083863**	**0**	**9999999**

-> year = 2015

Variable	Obs	Mean	Std. Dev.	Min	Max
incwage	**20,000**	**2324854**	**4199207**	**0**	**9999999**

-> year = 2016

Variable	Obs	Mean	Std. Dev.	Min	Max
incwage	**20,001**	**2436293**	**4267486**	**0**	**9999999**

STATA Command: by/bysort

```
. bysort year: sum incwage
```

-> year = 2014

Variable	Obs	Mean	Std. Dev.	Min	Max
incwage	**19,999**	**2156954**	**4083863**	**0**	**9999999**

-> year = 2015

Variable	Obs	Mean	Std. Dev.	Min	Max
incwage	**20,000**	**2324854**	**4199207**	**0**	**9999999**

-> year = 2016

Variable	Obs	Mean	Std. Dev.	Min	Max
incwage	**20,001**	**2436293**	**4267486**	**0**	**9999999**

STATA Command: by/bysort

```
1 bysort statefip : keep if _n == 1
```

- As mentioned above, we asked Stata to keep only the first observation for each state.
- The **bysort** command makes this selection a lot easier
- This will keep the first observation of each subset, i.e. the first observation for each state

STATA Command: by/bysort

```
1 bysort statefip (age): keep if _n == 1
```

- **Line 1:** we want to select the youngest observations in each state
- The parentheses tell Stata to sort age within state

STATA Command: bysort/egen

```
1 bysort statefip: egen state_inc = mean(incwage)
2 bysort statefip: egen state_inc_sd = sd(incwage)
```

- **bysort** can easily combine with many other commands, such as **egen**
- **Line 1-2:** generate average wage income and standard deviation of wage income by state

collapse: Converts the data into a dataset of summary statistics

STATA Command: collapse

Converts the data into a dataset of summary statistics

- **collapse** This command converts the data into a dataset of summary statistics, such as sums, means, medians, and so on
- This command is useful if you want to **create aggregate level dataset from individual level dataset**

STATA Command: collapse

Converts the data into a dataset of summary statistics

```
1 use "$rawdata\cps_2014_16.dta",replace
2 collapse (mean) incwage, by(statefip year)
3 use "$rawdata\cps_2014_16.dta",replace
4 collapse (max) max_wage=incwage, by(statefip year)
5 use "$rawdata\cps_2014_16.dta",replace
6 gen num=1
7 collapse (sum) samplesize = num, by(statefip year)
```

- **Line 2:** converts the data into mean of "incwage" (wage) by state and year - state and year mean and use the original variable name **incwage**
- **Line 4:** converts the data into maximum of "incwage" (wage) by state and year and generate a new variable **max_wage**
- **Line 7:** converts the data into sum of "num" by state and year, and generate a new variable **samplesize**

STATA Command: collapse

Converts the data into a dataset of summary statistics

	year	statefip	incwage
1	2014	connect…	2149647.498
2	2016	delaware	2268460.189
3	2015	illinois	2362452.85
4	2015	indiana	2503870.14
5	2016	iowa	2671571.387

STATA Command: collapse

Converts the data into a dataset of summary statistics

	year	statefip	max_wage
1	2014	connect…	9999999
2	2016	delaware	9999999
3	2015	illinois	9999999
4	2015	indiana	9999999
5	2016	iowa	9999999
6	2016	kansas	9999999
7	2014	maine	9999999

STATA Command: collapse

Converts the data into a dataset of summary statistics

	year	statefip	samplesize
1	2014	connect…	2799
2	2016	delaware	2113
3	2015	illinois	4393
4	2015	indiana	2137
5	2016	iowa	2486
6	2016	kansas	1984
7	2014	maine	1794

STATA Command: collapse

```
1 use "$rawdata\cps_2014_16.dta",replace
2 collapse (mean) incwage, by(statefip year) fast
```

- Note, if you are running this command on a large dataset, it may be worthwhile to use the **fast** option
- option **fast**: this option lets Stata not restore the original dataset

append: Add extra observations (rows) using other dataset

STATA Command: append

Add extra observations (rows) using other dataset

```
1 use "$workdata\state_wage_2015.dta",replace
2 append using "$workdata\state_wage_2016.dta"
```

- **append**: Add extra observations (rows) using other dataset
- **Line 2:** Add extra observations from `state_wage_2016.dta`

STATA Command: append

Add extra observations (rows) using other dataset

	year	statefip	incwage
1	2015	illinois	2362452.85
2	2015	indiana	2503870.14
3	2015	michigan	2458451.324
4	2015	new jer…	2145208.016
5	2015	ohio	2336920.091
6	2015	pennsyl…	2212914.227
7	2015	wiscons…	2147006.916

STATA Command: append

Add extra observations (rows) using other dataset

	year	statefip	incwage
1	2015	illinois	2362452.85
2	2015	indiana	2503870.14
3	2015	michigan	2458451.324
4	2015	new jer…	2145208.016
5	2015	ohio	2336920.091
6	2015	pennsyl…	2212914.227
7	2015	wiscons…	2147006.916
8	2016	delaware	2268460.189

STATA Command: append

```
1 use "$workdata\state_wage_2015.dta",replace
2 append using "$workdata\state_wage_2016.dta"
```

- If the two datasets you want to append have stored their variables in different formats (meaning string vs. numeric)
- In this case, **Stata converts the data in the file to be appended to the format of the original file** and in the process **replaces all values to missing!**
- It is thus very important to check via command **describe** that the two files you intend to append have stored all variables in the same broad data categories (string/numeric)

merge: Add extra variables using other dataset with common variables

STATA Command: merge

Add extra variables (columns, horizontal adding) using other dataset with common variables

- **merge**: Add extra variables (columns, horizontal adding) using other dataset with common variables
- Merge in Stata is for adding new variables from a second dataset to the dataset you're currently working with
  - Current active dataset = **master dataset**
  - Dataset you'd like to merge with master = **using dataset**

STATA Command: merge

Add extra variables (columns, horizontal adding) using other dataset with common variables

- Several different ways that you might be interested in merging data
- Merge in Stata is for adding new variables from a second dataset to the dataset you're currently working with
  - Two datasets with same observations (sample), one row per observation (1:1)
  - A dataset with one observation per row with a dataset with multiple rows per observation (1:many or many:1)

STATA Command: merge

Add extra variables (columns, horizontal adding) using other dataset with common variables

```
1 use "$workdata\state_wage_2015.dta",replace
2 merge m:1 year using "$workdata\state_size.dta"
3 tab _merge
```

- **Line 2:** The master dataset is a state-year data and using dataset is a year data
- We merge two datasets using common variable "year" and the relationship is m:1

STATA Command: merge

Add extra variables (columns, horizontal adding) using other dataset with common variables

	year	statefip	incwage	samplesize1	_merge
1	2015	illinois	2362452.85	20000	matched (3)
2	2015	indiana	2503870.14	20000	matched (3)
3	2015	michigan	2458451.324	20000	matched (3)
4	2015	new jer…	2145208.016	20000	matched (3)
5	2015	ohio	2336920.091	20000	matched (3)
6	2015	pennsyl…	2212914.227	20000	matched (3)
7	2015	wiscons…	2147006.916	20000	matched (3)

STATA Command: merge

- Stata automatically creates a variable called `_merge` which indicates the results of the merge operation
- It is crucial to tabulate this variable to check that the operation worked as you intended
- The variable can take on the values:
  - 1 : observations from the **master dataset that did not match** observations from the using dataset
  - 2 : observations from the **using dataset that did not match** observations from the master dataset
  - 3 : observations from the **both datasets that are matched**

STATA Command: merge

- **update** and **replace** options:
  - In standard merge, the master dataset is the authority and WON'T CHANGE
  - What if your master dataset has missing data and some of those values are not missing in your using dataset?
  - Specify **update**: it will fill in missing without changing nonmissing
- What if you want data from your using dataset to overwrite that in your master?
  - Specify **replace**: it will replace master data with using data UNLESS the value is missing in the using dataset

reshape: Change data to wide or long format

STATA Command: reshape

- Datasets may be laid out in wide or long formats
  - Long format
  - Wide format

STATA Command: reshape

To go from long to wide

Syntax: To go from long to wide

```
reshape wide stub, i(varlist) j(varname)
```

Example:

```
reshape wide incwage, i(statefip) j(year)
```

- **reshape**: Convert data from wide to long form and vice versa
- **i(varlist):** use varlist as the ID variables
- **j(varname):** long->wide,*varname*, existing variable

STATA Command: reshape

To go from long to wide

	year	statefip	incwage
1	2014	connect…	2149647.498
2	2016	delaware	2268460.189
3	2015	illinois	2362452.85
4	2015	indiana	2503870.14
5	2016	iowa	2671571.387
6	2016	kansas	2478861.048
7	2014	maine	2150442.11

STATA Command: reshape

To go from long to wide

	statefip	incwage2014	incwage2015	incwage2016
1	connect…	2149647.498	.	.
2	delaware	.	.	2268460.189
3	illinois	.	2362452.85	.
4	indiana	.	2503870.14	.
5	iowa	.	.	2671571.387
6	kansas	.	.	2478861.048
7	maine	2150442.11	.	.
8	maryland	.	.	2195578.841

STATA Command: reshape

Syntax: To go from wide to long

```
reshape long stub, i(varlist) j(varname [values])
```

Example:

```
reshape long incwage, i(statefip) j(year)
```

- **reshape**: Convert data from wide to long form and vice versa
- **i(varlist):** use varlist as the ID variables
- **j(varname [values]):** wide->long, *varname*, new variable optionally specify values to subset *varname*

STATA Command: reshape

To go from wide to long

	statefip	year	incwage
1	connect…	2014	2149647.498
2	connect…	2015	.
3	connect…	2016	.
4	delaware	2014	.
5	delaware	2015	.
6	delaware	2016	2268460.189
7	illinois	2014	.
8	illinois	2015	2362452.85
9	illinois	2016	.
10	indiana	2014	.
11	indiana	2015	2503870.14