

read and manipulate:
data reading/saving (formats/conversion)
and manipulation

adam okulicz-kozaryn
`adam.okulicz.kozaryn@gmail.com`

this version: Wednesday 19th September, 2018 10:55

outline

misc

data types

Stata

import/export

manipulating data

outline

misc

data types

Stata

import/export

manipulating data

outline

misc

data types

Stata

import/export

manipulating data

data basics

- dataset is a matrix
- columns are variables (var), rows are observations (obs)
- obs are also often referred to as U/A
- variables are characteristics of observations
- e.g., 'education', 'age', and 'income' are variables and persons are observations; each row is a separate person

paths

- a location of a file on hard drive
- e.g. `C:\Documents and Settings\myfile.txt`
- if there is a blank in path, as above, stata needs quotes
 `"C:\ Documents and Settings\ myfile.txt"`
- avoid blanks: computers understand blank as a character
- and avoid special characters: everything that is not a letter or a number, say `$ % &`
- special characters have special meaning for a computer
- linux/unix (this lab) uses `"/"` instead of win `"\"`

finding the path

- Windows: to find the path right-click the file— > properties
- Mac: ctrl-left-click the file — > get info
- linux/unix: easy! in file explorer/cabinet, the path appears at the top address bar

paths

- remember that you write code that should run on other computers
- and remember to `cd` first to desired directory, so you can say
- `cd ?`
- and then log using `ps1.txt`
- as opposed to:
log using `C:\Users\Documents\ASTATA\ps1.txt`
- that won't run, because I do not have these dirs!
- and it is messy to repeat path for each reading/writing

putting data online

- usually the biggest issue was to put data online!
- eg for google sites i often get error:
 - “You need permission”
- so the file you’ve put up online was not made public
- maybe better try wordpress.com, or dropbox.com, or sth else
- make sure it works! (ideally, make sure on other PC, too)
 - say try it on apps.rutgers.edu or some other computer
 - it is important it runs out of the box!
 - i will be picky about it

data for today

- data we use is a subset of general social survey:
<http://www.norc.org/gss+website/>
- probably the most comprehensive social science data for the US
- whatever you study you are likely to find it in gss
- we will look today at income, education and gender across US regions

outline

misc

data types

Stata

import/export

manipulating data

make comments in your code

- for each class we will have dofile with Stata code
- make comments in the electronic code files – you will run electronic files not the printout
- if you do not make comments, you will forget...
- use very handy keywords like “LATER” and “FIXME”

get the goodies: packages/user-written commands

- to get them either google or findit;
- say we want to load spss data eg findit spss
and then help usespss

commenting

- have preamble (notes, install packages, etc)

- `*comment`

```
/*comment
```

```
block */
```

-

```
net install usespss,
```

```
from(http://fmwww.bc.edu/RePEc/bocode/u)
```

stata command syntax and getting help

- `<command> <variables> , <options>`
`sum var1 var2, detail`
- `<variables>` and `<options>` are optional
- command specific syntax is in help files,
e.g. `help describe`
- `help` if you know command name, eg `help use`
- `esp options, examples, full pdf help`

getting help using gui and google

- gui, eg to load/save, edit data, graphs, etc
- google: "stata" + "what you want to do"
- eg "stata read excel"
- use google a lot! extremely useful!

tips

- if you did something wrong, load data again and start over
 - (replication: you have dofile and can always start over)
- page -up and -down to get previous/next command in command window
- don't memorize commands but reuse and share code
- learn (naturally) abbreviations, e.g. d for describe
 - (they are underlined in help files)

navigating

- you can navigate in stata:
change, list/make/rm dirs and preview files
dofile has the commands

outline

misc

data types

Stata

import/export

manipulating data

excel

- many people use it and you may need to import from there
- can save as csv and then insheet
- or just use gui to generate the code you need
- in some cases (as here) gui is useful to generate code
- File-Import-Excel Spreadsheet
- Worksheet: Cell Range: Import first row as variable names
- more commands in the dofile

saving

//good

```
use data1.dta
```

...

```
save data2.dta
```

//bad

```
use data1.dta
```

...

```
outsheet data1.tab //loosing var/val labels,notes
```

//ugly

```
use data1.dta
```

...

```
save, replace //loosing code in between
```

import/export

outline

misc

data types

Stata

import/export

manipulating data

general idea, intuition

- data management is mostly about manipulating data:
 - generating, recoding, labeling etc
 - today's class covers what you'll be doing most of the time with your data
- it's pretty easy—no complicated code, no fancy things
 - but also little boring, unexciting, and tedious, but necessary!
 - we'll be doing exciting and difficult things with programming and visualizing in few weeks

basic coding rules

- simplicity, clarity, efficiency:
 - drop everything that is not necessary
 - drop the clutter and be clean
- have “tight” code:
 - as few lines as possible that do as much as possible
- be lazy (copy from others, not 100% !)
 -
- more rules later

operators

- ◇ == equal to (status quo)
- ◇ = use for assigning values
- ◇ != not equal to
- ◇ > greater than
- ◇ >= (<=) greater (smaller) than or equal to
- ◇ & and (shift+7)
- ◇ | or
- ◇ replace happy=1 if(educ>10 | inc>=10) &
(unemp!=1 & div!=1)

basics

- ◇ most standard variables manipulation (e.g. generating, transforming, and recoding variables) can be done with:
 - ◇ `gen` and `replace`
 - ◇ or:
 - ◇ `recode`
- `recode` is often (not always) cleaner and better
- better use `gen` and `replace`
 - if it is complicated, multistage process to `gen` a var
 - say based on many other vars (as on previous slide)
- ◇ `dofile`

egen

- ◇ `egen` means “extended generate”
- ◇ powerful, difficult, and confusing
(typically these adjectives go together)
- ◇ for details: `help egen`; examples:
- ◇ `egen maxInc=rowmax(husInc wifInc)`
- ◇ `egen avgInc=mean(inc)`
- ◇ `gen devInc=inc-avgInc` $(x - \bar{x})$

by, sort, egen

- ◇ **by:** runs command by some group
- ◇ you always need to sort the group first
- ◇ so always use **by sort:** or in short: **bys:**
- ◇ **bys marital: egen avgmlnc=mean(inc)**
- **bys:** and **egen** often work well together!
- ◇ don't forget to check if stata did what you think it did
- <http://stataproject.blogspot.com/2007/12/step-4-thank-god-for-egen-command.html>
- ◇ **dofile**

tostring/destring is about storage type

- ◇ after running `d` in “storage type” column **str** denotes a string(word), everything else is a number
- ◇ run `edit` and note colors: red is string, black is number, blue is number with label
- ◇ number can be stored as a string
- ◇ string cannot be stored as a number
- ◇ from number to string
`tostring marital, gen(m_s)`
- ◇ from string to number
`destring m_s, gen(m_n)`
- ◇ `dofile`

'destring, ignore' is dangerous!

- i tried to clean up `http://taxfoundation.org/article/state-individual-income-tax-rates`
- a bunch of footnotes with (a),(b),(1),(2), etc
- in general do not use options
- "ignore" "force"
- unless you know 100% what you are doing!
- 'destring, ignore' is dangerous!
- it works on individual characters not full strings;
- `destring, ignore("(1)")` drops '(', ')', and '1' too !!!!
- <http://www.stata.com/statalist/archive/2011-11/msg01050.html>

encode/decode is about values

- ◇ convert string into numeric

encode region, gen(regN)

- ◇ decode will replace values with labels

- ◇ **encode/decode is about values**

- ◇ **tostring/destring is about storage type**

- ◇ dofile

missing values

- ◇ stata understands missing as a very big number
- ◇ for instance, if income is coded from 1 to 26 and we generate high income, this is **wrong**:

```
gen hi_inc=0
```

```
replace hi_inc=1 if inc>15 (1 for >15 and ".")
```

- ◇ it should be:

```
gen hi_inc=.
```

```
replace hi_inc=1 if inc>15 & hi_inc<26
```

```
replace hi_inc=0 if inc>0 & hi_inc<16
```

- ◇ dofile

missing values

- you can and should assign specific missing values
- that are '.' and a lowercase letter
 - that depends on reason for missingness, say:
 - .i=missing because refused
 - .k=missing because inapplicable
 - .z=missing because nonsense reported
- typically, do not drop missing obs!
 - because that it is missing on one var,
does not mean it is missing on others!

tips

- ◇ use `tab, mi` to see if there are any missings
- ◇ be careful about strings
- ◇ remember that number can be stored as a string
- ◇ you cannot do math with strings
- ◇ use operators—you can do anything with your data using them
- ◇ manipulation of variables is easy, but can easily go wrong
- remember to double check what you did
- `tab <oldVar> <newVar> , mi`
- (typically use `,mi!` and can add `,nola`)

exercise 1

- ◇ load gss.dta
- ◇ generate age^2 from age.
- ◇ generate a divorced/separated dummy variable that will take on value 1 if a person is either divorced or separated and 0 otherwise
- ◇ generate a variable that is a deviation from income's mean ($x - \bar{x}$)
- ◇ generate a variable showing average income for each region
- ◇ change storage type of income variable into string and name it inc_str and then change it back into number and name it inc_num
- ◇ generate numeric codes for regions

keep/drop

- ◇ keep first 10 obs
`keep in 1/10`
- ◇ keep obs on condition
`keep if marital==1`
- ◇ instead of `keep` you may use `drop`
`drop if marital>1 & marital <.`
- ◇ `keep` and `drop` also work for variables:
`drop marital`
- ◇ `dofile`

sort, order

- ◇ sort on marital's values

```
sort marital
```

- ◇ sort on marital's and then income's values

```
sort marital inc
```

- ◇ make marital 1st var

```
order marital
```

- ◇ put vars in alphabetic order

```
aorder
```

- ◇ dofile

`_n` `_N`

- ◇ To make operations based on row order it is useful to use `_n` and `_N`
- ◇ `gen id=_n`
- ◇ `gen total=_N`
- ◇ `edit`
- ◇ `gen previous_id=id[_n-1]`
- ◇ `dofile`

collapse

- ◇ we already learned `bys:` and `egen:`
`bys marital: gen count_marital_group=_n`
`bys marital: egen count_id=count(id)`
- ◇ a similar, but more radical, is `collapse`
`collapse inc educ, by(region)` (mean is default)
`collapse (count) id, by(marital)`
- ◇ `dofile`

tips

- ◇ both `collapse` and `bys: egen` can be used to calculate group statistics
- ◇ `collapse` produces new dataset with n equal number of groups
- ◇ `bys: egen` adds a new variable with group statistic that is constant within a group
- ◇ `_n+/-<number>` is useful with panel/time series data

exercise 2

- ◇ load gss.dta
- ◇ Create a new dataset using 'collapse' by region that has mean income, mean happiness, mean education, number of people who are married and number of females.
Hint: to get number of married and females first generate respective dummy variables and then use 'sum' option with 'collapse'.