basic organization and documentation

adam okulicz-kozaryn

adam.okulicz.kozaryn@gmail.com

this version: Saturday 29th September, 2018 16:49

outline

misc

directory (folder) structure

code structure

naming, labeling

outline

misc

directory (folder) structure

code structure

naming, labeling

datasets of the day

- climate! (easy access!)
  - https://wonder.cdc.gov/EnvironmentalClimateData.html
- religion!
  - http://www.thearda.com/Archive/Files/Descriptions/RCMSCY10.asp
  - http://www.thearda.com/Archive/Files/Descriptions/RCMSCY.asp
  - http://www.thearda.com/Archive/Files/Descriptions/CMS90CNT.asp
  - http://www.thearda.com/Archive/Files/Descriptions/CMS52CNT.asp
  - more: http://www.thearda.com/Archive/Browse_s.asp?pg=Browse_s.asp&sr=0&m=31&t=Search%20Data%20Archive&searchterms=county&p=B&c=N
- state level policy

  http://www.statepolicyindex.com/the-research/

outline

misc

directory (folder) structure

code structure

naming, labeling

replication again

- have a dofile that produces final results from raw data
- always keep raw data intact
- then manipulate it and save again, even several times
- at the end of your project you may end up
  - with several datasets at different levels of advancement
- then you may begin your stata session at any level

many ways to do it

- I am just giving an example of how I do it; but see:
- Scott Long "The Workflow of Data Analysis Using Stata"
  - I do not like his way!
  - no one's way is the best way
  - whatever floats your boat
- but always have it, be consistent
  - and give it some thought!

always have it!

- directory structure probably seems to you unnecessary
- but trust me, it is useful, just get in habit of having it
- you will see it's useful, once you start doing merging and outputing tables and graphs
- without directory structure, it'll get messy
- the more complex the project, the more important the directory structure
- in this class, try to make the project as complicated as possible

it's automatic! automate and standarize rules

- as discussed earlier, Stata can create directories and move files around
- so just have a generic dofile with a preamble
  - clear, version, set more off, etc
  - and a bunch of cap mkdir to create dir structure
- if I start a new project, I just start with my template
- also, standardization is good!
- it makes you move faster, you're on "autopilot"
  - it frees your mind to do more interesting things
- and it is easier to spot things that are out of normal
- so standardize and automate as much as possible (more about this later in theory.pdf)

files in general **singularity rule**

- organize dofiles and datafiles in folders
- **always one version of a dofile or datafile in one place**
  (see 'singularity' principle in theory.pdf)
- if you have 2 versions of the same file
  - sooner or later there will be problems!
  - you will update/change one, but forget the other one, etc
- exception is backup; but you never edit the backup!
- and you may and should keep historical record of your files
- mark it clearly, and always have only one current (working) file
- again, all that is best done and automatic in GIT

code in general **singularity rule**

- just like with files, so with code:
- **have the same chunk of code only in one place**
- if you have code that does the same thing multiple times (in same or many dofiles)
- then it is time to build some hierarchy and have
- some parent and some child dofiles
- typically, a parent will do something basic and generic
- and then different children will pick up the data from parent and each will be doing something differently

these rules are necessary!

- standardization helps: just doing things in the same way
  - it's faster and easier to spot mistakes
- and singularity helps–just do it one time!
  - say you work with GSS
  - then just manipulate it into the shape you need once and for all
  - then use it for all the other projects in your lifetime
  - well, of course you'll make some updates
  - but they're small and just in one file

hierarchy of dofiles

- an example when having many dofiles is useful is when you use the same data for many projects
- this happens more often than it doesn't
- it makes sense to have one dofile that makes data ready
  - it processes raw data and saves it in usable format
  - and then always start from there
- again, you always want to start from the very raw data
  - so just include at the beginning of each project

    ```
    do datMan.do
    ```

  - and then do your project specific analysis

hierarchy of dofiles

- always extract common chunks into one file
- typically there will be one (parent) file
  - doing general data management for each dataset
- say you use GSS for multiple projects,
  - typically for each project, you have to first do same things to get data usable
  - recode, label, calculate new vars, etc
- then just have a "root" directory for that dataset
  - and then each project will start with data from that root directory and do project specific-things
- otherwise, if you have multiple files doing the same things
- it will get mixed up!

datafiles

- never overwrite the original datafile, and a good idea to keep datafiles at different stage of advancement
  - especially if data are complex:
  - rawFile− >file1− >file2 –and those are produced by: dofile0− >dofile1− >dofile2 (or subsequent sections in one dofile!)
- and again dofile0 will be common for all projects
- but there may be for project A abd B: dofile1A and dofile1B
- in other words one parent dofile0 will have 2 children: dofile1A and dofile1B
- likewise, rawFile will have 2 different children file1A and

the one dofile to rule them all

- if you have a complicated project you may want to have many dofiles
- still you want to have a master dofile that runs them
- "the one dofile to rule them all"

branching

- not only dofies and datafiles have parents
- whole projects do!
- usually a project spins off other projects
- then you may want to clearly mark who is a parent and who is a child (for record keeping)
- and start a new project folder and directory structure for each new one

backup

- **backup all files at least once a week**–computers break regularly; flash drives break really often
- have automatic system for backups (i use cron)
  - otherwise you'll forget
- backup to remote places!
  - if your backup hd is in the same physical place
  - then in case of fire, flooding, burglary, etc
  - the backup is gone!

outline

misc

directory (folder) structure

code structure

naming, labeling

sections, subsections

- especially for beginners, one dofile would do
- (again, later, when you have multiple projects from same data, extract common code to one parent)
- dofile should have a multi-layerd structure
  - eg chapters, sections, subsections etc (like a paper or book)
- it is useful to mark large chunks of code, eg "datMan"
  - i do it in my code
- for different levels, use different kinds of comments: box, block, one line, horizontal line, etc

type them in dofiles and scroll down to already existing

  - now i just use '***', '**', '*', '//'

outline

misc

directory (folder) structure

code structure

naming, labeling

general

- naming and labeling looks like waste of time
- but at the end saves time
- labels are like “postit” notes
- importantly, it prevents mistakes/misinterpretations
  - especially, if a project is big and/or you share it with others
  - or if it takes long time

variable names, labels, notes, and value labels

- variable name is...a variable name, eg educ
- var lab describes var, eg "highest degree completed"
- note is like label, except it can be>80 chars
  - eg put there full svy question: "how would you describe highest level of your education?"
- value label describes values that a variable takes on
  - (output of codebook, or tab and tab,nola), eg:
  - "primary school" 1
  - "high school" 2
  - "college or university"3
- dofile

labels tips

- give variables short names, eg inc
- labels, on the other hand should be descriptive, eg "2004 hh income"
- labels prevent confusion later and for others
- they automatically appear on graphs, regressions, etc.
- use lookfor, especially if you have many variables
- be lazy (remember it's our core value)
  - only label what is necessary
  - indeed, only keep data and variables that are necessary
  - you have the code, so you can always add back in later

more tips on var names

- i dont like '_' anymore
- i just use Caps to denote words, eg
- hhInc as opposed to hh_inc; i guess it's cleaner
- and typicaly i have 3 letter var namees 'swb'
- or 6 letter that combine 2 words: say menHea for mental health
- but do whatever is natural to you!
- and is simple clean and consistent