NCES Datasets

LPO 9951 | Fall 2015

PURPOSE Today we will work on downloading data files from the NCES and OECD databases. While these data files represent only a fraction of publicly available data (which themselves are only a fraction of all potentially available data), they are some of the most widely used in education research.

EDAT

NCES has created a very useful data tool called EDAT, which I assume stands for something clever. The intent behind this web-based application is to allow you to select the variables that you would like to work with, then to generate syntax (in our case, do files) that will allow you to access this data. Today, we'll go through how to generate a very basic analysis dataset from all four of the surveys that you will be working with as a group. The process for each is broadly the same:

- Select variables
- Download syntax and data (only need to download the data the first time)
- Adjust syntax as appropriate (rewrite do-file to our specifications)
- Generate analysis dataset

This process, like all other work, should be tracked carefully. Some simple steps at the beginning can save you lots of time at the end.

I will also show you how to download the full datasets for each survey today. While EDAT may entirely serve your data-gathering needs (at least for these surveys), you may in the future easier to simply gather all of available data on your computer with a few lines of code, avoiding the point-and-click aspects of EDAT altogether and increasing the reproducibility of your project.

Directory structure

First things first, let's add to our directory structure. As you remember from the first lecture, data files and Stata do files have been stored in their own subdirectories. While it's possible to simply dump everything in one big directory, you may find that over time, as the folder grows, it becomes very difficult to find what you need and almost impossible to share your work with others. Yes, your computer can search really well. An organized directory structure is for you, the human. Get into the habit now, and you'll be thankful later.

Today we're adding an `./aux` subdirectory. Your directory structure should now look like this:

```
.
|-- /aux
|   |
|   |-- <auxiliary files>
|
|-- /data
|   |
|   |-- <data files>
|
|-- /do
|   |
|   |-- <Stata do files>
```

```
|
|-- /plots
|   |
|   |-- <plot files>
```

Global variables (macros)

Now that your project/course directory is structured this way, it is very easy to find files across subdirectories. But rather than retype something like `../data/` in front of every data file name, it is useful to store the relative link name in a type of variable that we can then call as we want. One type of variable that Stata allows us to use for this procedure (among others) is called a global variable or macro. See the top part of the do file for an example of how to store a relative path in a global variable:

```
. global workdir `c(pwd)'

. global datadir "../data/"

. global auxldir "../aux/"

. global ghuburl "https://raw.githubusercontent.com/btskinner/lpo9951/master/au
> x/"
```

Note that globals follow the pattern: `global <name> <value>`. To call a global macro in Stata, you place a `$` in front of the name you gave it. Stata will replace that with the value. Here are some examples with `display`:

```
. di "$workdir"
/Users/benski/Github/lpo9951/markdown

. di "$datadir"
../data/

. di "$auxldir"
../aux/

. di "$ghuburl"
https://raw.githubusercontent.com/btskinner/lpo9951/master/aux/
```

NOTE: Most calls of the Stata global macro do not require quotation marks. That is a quirk of Stata's `display` command.

Educational Longitudinal Study, 2002 (ELS)

Unzip data

Prior to running this file, we will have to have downloaded the entirety of the ELS student file. While this may seem like overkill, note that even if you use EDAT and subset the variables you actually want, you will end up downloading all of ELS anyway (just the way it works). First, let's set new globals:

```
. global els_zip "ELS_2002-12_PETS_v1_0_Student_Stata_Datasets.zip"
```

```
. global els_dta "els_02_12_byf3pststu_v1_0.dta"

. global elssave "els_reduced.dta"
```

As you probably noticed, the downloaded ELS file is zipped. This means that we need to unzip it with `unzipfile`. Stata will only unzip a file into the current directory, so in order to have it go into our data directory like we want, we need to use the `cd` command to change directory into our data directory and return to the working directory after we've unzipped the file. This is why we saved the `workdir` at the top of the file.

```
. cd $datadir
/Users/benski/Github/lpo9951/data

. unzipfile $datadir$els_zip, replace
    inflating: els_02_12_byf3pststu_v1_0.dta
successfully unzipped ../data/ELS_2002-12_PETS_v1_0_Student_Stata_Datasets.zip
> to current directory

. cd $workdir
/Users/benski/Github/lpo9951/markdown
```

Subset data

ELS data files are very large, so they take up a ton of memory if you try to load them in their entirety. For any given project, you don't want or need all of the variables anyway. To subset the full dataset to only those variables we want requires the `use using` setup, as follows (note how we temporariliy change the delimiter to make our lives a little easier):

```
. #delimit ;
delimiter now ;
. // keep only selected variables in ELS
> use
>    F1RGPP2
>    F2B01
>    STU_ID
>    SCH_ID
>    STRAT_ID
>    PSU
>    F1SCH_ID
>    F1UNIV1
>    F1UNIV2A
>    F1UNIV2B
>    F2UNIV_P
>    BYSTUWT
>    F1QWT
>    F1PNLWT
>    F1TRSCWT
>    F2QTSCWT
>    F2QWT
>    F2F1WT
>    F2BYWT
> using $datadir$els_dta;
```

```
. // change delimiter back to carriage return
> #delimit cr
delimiter now cr
. // lower all variable names using wildcard
. renvars *, lower

. // save reduced ELS dataset
. save $datadir$elssave, replace
file ../data/els_reduced.dta saved
```

Because nobody (nobody) likes variables with capital letters—except in very specific sitations—we use the `renvars` command to set all the variable names to lowercase. Once that is finished, we save the new working dataset with `save` command. By prepending the name of the saved dataset with the relative link to the data folder (via the global), the save dataset goes into the correct subdirectory. The option after the comma, `replace`, simply tells Stata to overwrite any files with same name.

QUICK EXERCISE

> Using EDAT, generate an ELS dataset that includes student demographics and whether or not the student attended college.

Early Childhood Longitudinal Study - Kindergarten 98-99 (ECLS-K)

Unzip data

Unzipping ECLS-K follows the same method that we used for ELS. Note that the unzipped files is *really* big, so sure you have enough room for it (no 2 GB thumbdrives!).

```
. // set globals for ECLS-K files
. global ecl_zip "ECLSK_1998-99_v1_0_Stata_Datasets.zip"

. global ecl_dat "ECLSK_98_99_K8_CHILD_v1_0.dat"

. global ecl_dct "nces_datasets_ecls.dct"

. global eclsave "eclsk_reduced.dta"

.
. // unzip ECLS file
. cd $datadir
/Users/benski/Github/lpo9951/data

. unzipfile $datadir$ecl_zip, replace
    inflating: eclsk_98_99_k8_tch_v1_0.dta
    inflating: eclsk_98_99_k8_child_v1_0.dat
    inflating: eclsk_98_99_k8_sch_v1_0.dta
successfully unzipped ../data/ECLSK_1998-99_v1_0_Stata_Datasets.zip to current
> directory

. cd $workdir
/Users/benski/Github/lpo9951/markdown
```

Subset data

Also like the ELS files, the ECLS-K data files are too large to load all at once. Unlike ELS, however, they require a separate dictionary file to properly read in the data. When downloading from EDAT, you will also get a `.dct` file that you will use to properly parse the `.dat` format of ECLS-K. It will look something like this:

```
dictionary {

_column(1)     str8 CHILDID   %8s       "CHILD IDENTIFICATION NUMBER"
_column(1418)  double C3R4RSCL%6.2f     "C3 RC4 READING IRT SCALE SCORE"
_column(1510)  double C3R4MSCL%6.2f     "C3 RC4 MATH IRT SCALE SCORE"

}
```

I've already created this file and hosted it in the course repository. Since it is a small file, we can use the `copy` command, which takes the format `copy <from> <to>` and can handle URL paths, to download the file to the auxiliary directory. Note again the use of global macros:

```
. copy $ghuburl$ecl_dct $auxldir$ecl_dct, replace
```

To read the ECLS-K `.dat` file, we use the `infile` command. Note the weird double `using` subcommands. The first tells Stata which dictionary file to use; the second, after the comma, tells which dataset to use. There are a couple of other ways to use this command, but this is the most transparent. As before, we lower variable names and save the working dataset.

```
. infile using $auxldir$ecl_dct, using($datadir$ecl_dat) clear

dictionary {

_column(1)     str8 CHILDID   %8s       "CHILD IDENTIFICATION NUMBER"
_column(1418)  double C3R4RSCL%6.2f     "C3 RC4 READING IRT SCALE SCORE"
_column(1510)  double C3R4MSCL%6.2f     "C3 RC4 MATH IRT SCALE SCORE"

}

(21,409 observations read)

. // lower all variable names using wildcard
. renvars *, lower

. // save reduced ECLS-K dataset
. save $datadir$eclsave, replace
file ../data/eclsk_reduced.dta saved
```

QUICK EXERCISE

> Using EDAT, generate an ECLS-K dataset that includes student demographics and student IRT math scores from the first three waves.

High School Longitudinal Study, 2009 (HSLS)

Unzip and subset data

The process for unzipping and subsetting data from HSLS is the same as for ELS. In interest of completeness, the code is reproduced in full below:

```
. // set globals for HSLS files
. global hsls_zip "HSLS_2009_v2_0_Stata_Datasets.zip"

. global hsls_dta "hsls_09_student_v2_0"

. global hslssave "hsls_reduced.dta"

. // unzip HSLS file
. cd $datadir
/Users/benski/Github/lpo9951/data

. unzipfile $datadir$hsls_zip, replace
    inflating: HSLS_09_SCHOOL_v1_0.dta
    inflating: hsls_09_student_v2_0.dta
successfully unzipped ../data/HSLS_2009_v2_0_Stata_Datasets.zip to current dire
> ctory

. cd $workdir
/Users/benski/Github/lpo9951/markdown

. // change delimiter to a semi-colon -2-
. #delimit ;
delimiter now ;
. // keep only selected variables in HSLS
> use
>    stu_id
>    sch_id
>    x1ncesid
>    w1student
>    w1parent
>    w1mathtch
>    w1scitch
>    s1avid
> using $datadir$hsls_dta;

. // change delimiter back to carriage return
> #delimit cr
delimiter now cr
. // lower all variable names using wildcard
. renvars *, lower
no renames necessary

. // save reduced HSLS dataset
. save $datadir$hslssave, replace
file ../data/hsls_reduced.dta saved
```

Programme for International Student Assessment (PISA)

Unzip data

Not all datasets, of course, come from the NCES. One major dataset, PISA, is conducted and hosted by the Organization for Economic Co-operation and Development or OECD. To get these data, we'll once again use the same process for unzipping files.

```
. // set globals for PISA files (note new baseurl)
. global pisa_zip "INT_COG12_DEC03.zip"

. global pisa_txt "INT_COG12_DEC03.txt"

. global pisasave "pisa_reduced.dta"

. // unzip PISA file
. cd $datadir
/Users/benski/Github/lpo9951/data

. unzipfile $datadir$pisa_zip, replace
    inflating: INT_COG12_DEC03.txt
successfully unzipped ../data/INT_COG12_DEC03.zip to current directory

. cd $workdir
/Users/benski/Github/lpo9951/markdown
```

Subset data

PISA uses a fixed-width format for storing data that is common but not particularly user friendly. The first 3 lines of text look like this (they are long so they wrap in this view):

```
ALB0080000ALB00060000800000000100001 777777777790100177237002777114393137900772077777777771077777777777
ALB0080000ALB00060000800000000100002 970777777770777377277102707779777737911777777199777777777777777777
ALB0080000ALB00060000800000000100003 397777777771777177477104707773777737019197777777772337742117777777
```

To read the data into Stata, we have to use the `infix` command, which requires that we know the column of each variable that we need. It also helps if we know the data format (e.g., string vs. integer vs. float, etc.). Having a codebook like this is absolutely necessary. Knowing what is needed, we can finally read in the data and save a reduced dataset. Because the string is long, I've stored the entire thing in a local variable using the comand `local <name> ...`. Locals work just like globals except they only last for as long as the script is running. To call a local, surround the local name by left and right ticks. Note, that a left tick is different from a right tick. On most keyboards, the left tick is above the tab key on the lefthand side of the keyboard.

```
. local vars str cnt 1-3 subnatio 4-10 stuid 32-36 ps527q03r 382-384

. // read in variables
. infix `vars' using $datadir$pisa_txt, clear
(485,490 observations read)

. // save reduced PISA dataset
. save $datadir$pisasave, replace
file ../data/pisa_reduced.dta saved
```

QUICK EXERCISE

Following the link above, figure out the column for the variable about the number of computers in the household, add it to the list, and read in the new expanded data.

Init: 03 August 2015; Updated: 16 August 2015