

# First steps in Stata

Zane Mokhiber  
Economic Policy Institute

September 30, 2021  
EARNCon 2021

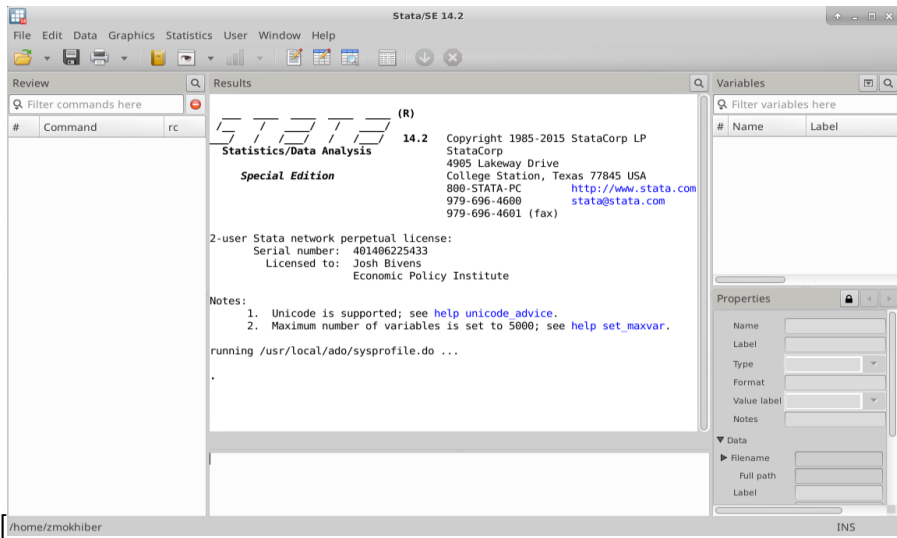
# Motivation

- ▶ What is Stata and why does EPI use it for data analysis?
  - ▶ Stata is statistical software
  - ▶ Easier to learn than other programs with similar functionality
  - ▶ Will enable EARN groups to conduct more sophisticated analysis than the EPI provided datasets allow (ie, swx, jobswatch, data library, etc)

# Overview

- ▶ This is a crash course for using Stata for data analysis
- ▶ Using individual level data from the Current Population Survey (CPS), we will learn about:
  1. Using Stata interactively
  2. Key data concepts
  3. What CPS microdata is
  4. Basic analysis using microdata

# Stata's graphical interface



## Getting familiar with the interface

- ▶ Type an operation into Stata's command pane and see the output in the results pane

```
display 2+2
```

# Open some data and look at it

- ▶ A simple example using boring data.
  - ▶ open the `auto.dta` example dataset

```
sysuse auto.dta
```

# What is a dataset anyways?

- ▶ Think of your data like a spreadsheet
- ▶ In Stata terminology,
  - ▶ columns are variables
  - ▶ rows are observations
- ▶ `browse` opens the data browser window
- ▶ `auto.dta` is not very much data so we can look at the entire thing with

```
list
```

# Getting more info about your data

- ▶ Here are some commands to describe what is in your data

```
describe  
summarize  
tabulate
```

- ▶ What is a Stata command?
  - ▶ type commands to perform operations in stata
  - ▶ all commands have a specific “syntax” that you have to follow

```
* typical stata syntax  
command {variable list}{expressions}, (options)
```



# Using commands to examine subsets of variables

- ▶ Instead of looking at all the variables, just investigate a few:

```
summarize price  
tabulate foreign
```

- ▶ Stata lets you abbreviate some commands

```
summarize price  
sum price  
  
tabulate foreign  
tab foreign  
  
describe  
d
```

- ▶ Help command

```
help describe
```

# Data types

- ▶ Stata stores variables with different data types.
  - ▶ there are many data types but the main distinction is character or “string” variables vs numeric variables
- ▶ String variables store plain text (ie. “sample text” “dataset\_1” “123”)
- ▶ Numeric variables store numbers (ie 123, 4,000, 1, 0, 26)
- ▶ you cannot have both numeric and string values in the same variable
- ▶ mathematical operations can only be done on numeric values
  - ▶ may seem obvious, but  $10 + 24$  is not the same as “10” + “24” in Stata.

# Value Labels

- ▶ When you look at a dataset with the `browse` function, many variables look like they have text, but Stata will say they are numeric.
  - ▶ these are “value labels”
  - ▶ they are a useful way to determine what a given number stands for in a categorical or indicator variable

```
label list foreign
```

\*can be confusing to assign value labels, but here is a good post on how to do it: <https://stats.idre.ucla.edu/stata/modules/labeling-data/>

Example research question: what are wages by race  
and gender in Ohio?

# CPS microdata

- ▶ Current Population Survey (CPS)
  - ▶ Survey conducted by Census and the Bureau of Labor Statistics
  - ▶ Basic monthly data are released on the [Census website](#)
  - ▶ Primary source of monthly labor force statistics
- ▶ Why can't I use Excel to analyze microdata?
  - ▶ Excel is limited to 1,048,576 rows
  - ▶ it's not good at processing even tens of thousands of rows
  - ▶ not (easily) programmable
  - ▶ mistakes are hard to catch

# EPI CPS extracts

- ▶ Use data from the [EPI microdata website](#)
  - ▶ variables recoded and harmonized across time by EPI
  - ▶ already in stata format
  - ▶ variables are consistent with EPI methodological choices
- ▶ Why CPS over ACS?
  - ▶ both surveys have their strengths
  - ▶ the CPS is typically better for answering questions about wages and unions

# Using the data

- ▶ what data should we choose?
  - ▶ lets start with one year of data
  - ▶ 2019 data is provided for you already
  - ▶ make sure the data is in your working directory
- ▶ The working directory is where stata looks for data files and also where it saves/outputs files if not otherwise specified

```
pwd /* this prints the working directory to  
the output window */  
cd /* this allows you to change the working directory */
```

# Using the data

- ▶ The use command loads data into stata
  - ▶ stata data files have the .dta extension

```
use epi_cpsorg_2020.dta, clear
```

- ▶ The clear option allows you to use the data even if there is already data in memory
- ▶ If the file is already in the working directory, you don't need to specify a file path



## Creating new variables: indicator variables

- ▶ Use `tabulate` to look at the state variable, then `generate` to create an indicator variable

```
generate oh = 0  
replace oh = 1 if statefip == 39
```

- ▶ What's the going on with the mixture of `=` and `==`?
- ▶ `=` is the assignment operator
- ▶ `==` is the equivalence operator
- ▶ Other ways to create a OH indicator

```
generate oh = 1 if statefip == 39  
replace oh = 0 if statefip != 39
```

- ▶ If you use this syntax, `stata` knows you want to make an indicator variable

```
generate oh = statefip == 39
```

# Restricting the sample

- ▶ It is often useful to select just those rows of your data based on a condition
  - ▶ for example select only rows where individuals are older than 16
- ▶ The following operators allow you to do this:
  - `==` equal to
  - `!=` not equal to
  - `>` greater than
  - `<` less than
  - `>=` greater than or equal to
  - `<=` less than or equal to
  - `&` and
  - `|` or (this is the “pipe” character, `ctrl + \` on your keyboard )

Use `=` when you are assigning values, like `generate` or `replace`

Use `==` when you are testing a true/false condition ex: `if state == 39`

# Restricting the sample

- ▶ We want to look at wages for people with positive wages age 16+ in Ohio in 2019
  - ▶ use the keep and drop commands to restrict the sample
  - ▶ will keep or drop observations based on a condition

```
* age restriction
```

```
keep if age >= 16
```

```
* Ohio only
```

```
keep if oh == 1
```

# Note on dealing with missing values

- ▶ Sometimes, an observation for a given variable will be “missing”.
  - ▶ Not everyone answers all questions in a survey
  - ▶ Some questions don't apply to certain individuals
- ▶ Stata stores missing values as ‘.’

```
sum wage  
count if wage == .
```

## Note on dealing with missing values (cont.)

- ▶ Stata treats missing values as the largest number.
- ▶ This keeps missing values

```
keep if wage > 0  
tab wage if age < 16, m
```

- ▶ This drops missing values

```
keep if wage > 0 & wage ~= .  
tab wage if age < 16, m
```

# Race and ethnicity variables in the CPS

- ▶ The race variable in the EPI CPS extracts has several categories
  - ▶ check out the [race variable methodology](#) documentation for more information

```
tab wbho
```

- ▶ In our example we want to look at wages for various demographic cuts
  - ▶ many analyses by race have Hispanic as a category so we include it in the same variable
  - ▶ Census defines Hispanic ethnicity in a separate variable (`hispanic` also exists by itself the EPI extracts)
  - ▶ in `wbho`, race/ethnicity categories are mutually exclusive

# Calculating some earnings statistics

- ▶ Use summarize to look at mean earnings
- ▶ Always use weights!
  - ▶ surveys are rarely true random samples
  - ▶ weights are added to make the sample look like the overall population

```
sum wage [w=orgwgt]
sum wage [w=orgwgt] if wbho == 1
sum wage [w=orgwgt] if wbho == 2
sum wage [w=orgwgt] if wbho == 1 & female == 1
```

## Exercise: (wages by race and by sex in Ohio)

- ▶ Many stata commands allow the by and bysort commands

```
bysort wbho female: sum wage [w=orgwgt]
```

- ▶ but what if we want to do something more useful like make a graph or compare to other states?
  - ▶ you could copy the output into excel...



# Transforming data: Collapse

- ▶ Instead of using the `bysort: sum` command, we can use the `collapse` command to transform the data
  - ▶ `collapse` replaces the data in memory with the new collapsed data
  - ▶ Extremely helpful when you want to calculate aggregate statistics from individual-level data
  - ▶ Allows you to continue your analysis with `stata`

```
collapse (mean) wage [aw=orgwgt], by(wbho female)
```

- ▶ **WARNING:** this will replace the data in memory, so ensure you don't save over the original individual level data
  - ▶ `preserve` and `restore` are useful when using `collapse` in `stata` interactively

## Exporting the analysis

The collapsed data is easily exported to excel using the export command

```
export excel using ohio_wages.xlsx, ///  
replace firstrow(variables)
```

# Resources/contact info

- ▶ All files associated with this presentation can be accessed at [https://economic.github.io/data\\_bootcamp/](https://economic.github.io/data_bootcamp/)
- ▶ Additional stata resources
  - ▶ Princeton intro to stata: <https://data.princeton.edu/stata>
  - ▶ UCLA learning modules  
<https://stats.idre.ucla.edu/other/mult-pkg/seminars/#Stata> and here  
<https://stats.idre.ucla.edu/stata/modules/>
  - ▶ Stata also has a large library of video tutorials:  
<https://www.stata.com/links/video-tutorials/> and webinars:  
<https://www.stata.com/training/webinar/>
  - ▶ Stata cheat sheets:  
<https://www.stata.com/bookstore/statacheatsheets.pdf>
- ▶ My contact info:
  - ▶ email: [zmokhiber@epi.org](mailto:zmokhiber@epi.org)
  - ▶ twitter: @zanemokhiber