Intro to Data Management in Stata

Alicia Doyle Lynch

Harvard-MIT Data Center (HMDC)

Documents for Today

- Find class materials at: http://libraries.mit.edu/guides/subjects/data/training/workshops.html
  - Several formats of data
  - Presentation slides
  - Handouts
  - Exercises
- Let’s go over how to save these files together

Organization

- Please feel free to ask questions at any point if they are relevant to the current topic (or if you are lost!)
- There will be a Q&A at the end of class for more specific, personalized questions
- Collaboration with your neighbors is encouraged
- If you are using a laptop, you will need to adjust paths accordingly

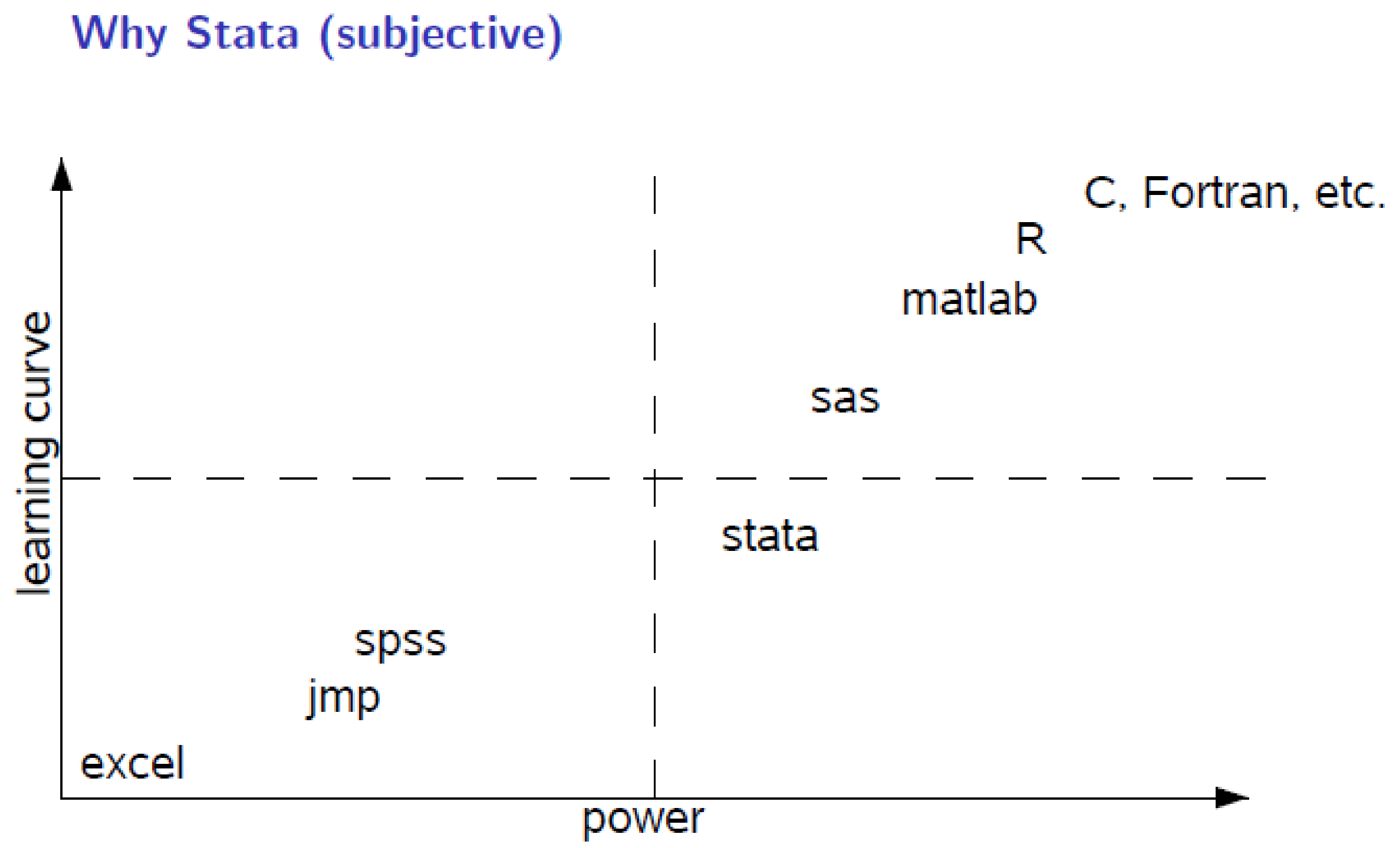
Organization

- Make comments in your Do-file rather than on hand-outs
  - Save on flash drive or email to yourself
- Stata commands will always appear in red
- “Var” simply refers to “variable” (e.g., var1, var2, var3, varname)
- Pathnames should be replaced with the path specific to your computer and folders

Assumptions and Disclaimers

- This is an **INTRODUCTION** to data management in Stata
- Assumes basic knowledge of Stata
- Not appropriate for people already well familiar with Stata
- If you are catching on before the rest of the class, experiment with command features described in help files

Why Stata?


Why Stata (subjective)
C, Fortran, etc.
R
matlab
sas
stata
spss
jmp
excel
learning curve
power

How do I get Stata?

- Your Department IT
- HMDC labs
- Athena terminals at MIT
- Buy it: educational or grad plan
- http://libraries.mit.edu/guides/subjects/data/software/index.html

Opening Stata

- In your Athena terminal (the large purple screen with blinking cursor) type

```
add stata
xstata
```

- Stata should come up on your screen
- Always open Stata FIRST and THEN open Do-Files (we'll talk about these in a minute), data files, etc.

Let’s get started

- Open up a new Do-File
- Before we do anything, we need to tell Stata how much memory to use

```
set mem 500m, perm
```

  – “Perm” makes this permanent (everytime you open Stata, it will allow 500m of memory)

Opening Files in Stata

- Look at bottom left hand corner of Stata screen
  - This is the directory Stata is currently reading from
- We can also see this by typing pwd in our Do-File editor
- Use dir to see what is in the directory
  - If your datafile is not there, Stata will not open it!

Opening Files in Stata

- When I open Stata, it tells me it’s using the directory:
  - afs/athena.mit.edu/a/d/adlynch
- But, my files are located in:
  - afs/athena.mit.edu/a/d/adlynch/DataManagement
- I’m going to tell Stata where it should look for my files:

```
cd “~/DataManagement”
```

GSS

- General Social Survey
- Has tracked opinions of Americans since 1970s
- Monitors social change in U.S. and compares U.S. to other countries
- http://www.norc.org/GSS+Website/

Useful Data Manipulation Commands

- == equal to (status quo)
- = used in assigning values
- != not equal to
- > greater than
- >= greater than or equal to
- & and
- | or

Basic Data Manipulation Commands

- Basic commands you'll use for generating new variables or recoding existing variables:
  - egen, gen, replace, recode
- Many different means of accomplishing the same thing in Stata
  - Find what is comfortable (and easy) for you

Basic Data Manipulation Commands

- Generate & Replace
  - Pretty much the same command – Stata just gives you a separate "replace" command so you don't accidently write over an existing variable

```
gen income = .
replace income=1 if employed==1
replace income=1 if unemployed==1 & student==1
replace income=3 if retired==1 | disabled==1
```

Basic Data Manipulation

- Recode
  - Basically generate & replace combined
  - You can recode an existing variable OR use recode to create a new variable

```
recode unemployed (1=0)
recode unemployed (1=0), gen(unemployed2)
```

Basic Rules for Recode

Rule	**Example**	**Meaning**
#=#	3=1	3 recoded to 1
# # = #	2 . = 9	2 and . recoded to 9
#/# = #	1/5=4	1 through 5 recoded to 4
nonmissing=#	nonmiss=8	all nonmissing recoded to 8
missing = #	miss=9	all missing recoded to 9

Basic Data Manipulation Commands

- Egen
  - Just means “extension” to generate
  - Contains a variety of more sophisticated functions
  - Type “help egen” in Stata to get a complete list of functions
- Let’s create a new variable that counts the number of “yes” responses on computer, email and internet use

```
egen compuser= anycount(usecomp usemail usenet),
   values(1)
```

Basic Data Manipulation Commands

- Let’s say we want to assess how much missing data each participant has:

```
egen countmiss = rowmiss(age-wifeft)
codebook countmiss
```

- What if we have multiple variables that we want to compare values on?

```
egen ftdiff=diff(wkft*)
codebook ftdiff
```

The “By” Command

- Sometimes, you’d like to create variables based on different categories of a single variable
  - For example, say you want to look at happiness based on whether an individual is male or female
- The “by” command does just this

```
bysort sex: tab happy
hist happy, by(sex)
```

The “By” Command

- Allows us to generate variables based on group statistics

```
bysort state: egen stateincome = mean(income)
bysort degree: egen degreeincome = mean(income)
bysort marital: egen marincomesd = sd(income)
```

Missing Values

- Always need to consider how missing values are coded when recoding variables
- Stata's symbol for a missing value is "."
- Stata interprets "." as a **large value**
  - What are implications of this?
  - An aside – SAS interprets "." as a small value

Missing Values

- If income is coded from 1-26 and we want to generate a new variable that identifies high income individuals, we might use the command:

```
gen hi_inc=0
replace hi_inc=1 if income>15
```

- What happens to our missing values when we use this command?

```
tab income, nola
```

Missing Values

- Instead, we might try:

```
gen hi_inc = 0 if income != .
replace hi_inc=1 if income >15 & income !=.
```

- Also be careful if your missing values equal “999” or another numeric value
- Add the “mi” command to tab to view your missing data values

```
tab income, mi
```

Missing Values

- What if you used a numeric value originally to code missing data (e.g., “999”)?
- The mvdecode command will convert all these values to missing

  mvdecode _all, mv(999)

    - The “_all” command tells Stata to do this to all variables
- Use this command carefully!
  - If you have any variables where “999” is a legitimate value, Stata is going to recode it to missing
  - As an alternative, you could list var names separately rather than using “_all” command

Variable Types

- Stata uses two main types of variables: String and Numeric
- String variables are typically used for text variables
- To be able to perform any mathematical operations, your variables need to be in a numeric format

Variable Types

- Stata’s numeric variable types:

```
Storage                                                 0 without
type                   Minimum                Maximum    being 0     bytes
----------------------------------------------------------------------------
byte                      -127                    100    +/-1           1
int                    -32,767                 32,740    +/-1           2
long            -2,147,483,647          2,147,483,620    +/-1           4
float   -1.70141173319*10^38  1.70141173319*10^38    +/-10^-38      4
double  -8.9884656743*10^307  8.9884656743*10^307    +/-10^-323     8
----------------------------------------------------------------------------
Precision for float  is 3.795x10^-8.
Precision for double is 1.414x10^-16.
```

Variable Types

- How can I deal with those annoying string variables?
- Sometimes you need to change a string variable into a numeric variable:

```
destring var1, gen(newvar)
```

- Other times, you want to convert a numeric variable to a string:

```
tostring var1, gen(newvar)
```

Variable Types: Date and Time

- Stata offers several options for date and time variables
- Generally, Stata will read date/time variables as strings
- You'll need to convert string variables in order to perform any mathematical operations
- Once data is in date/time form, Stata uses several symbols to identify these variables
  - %tc, %td, %tw, etc.

Variable Types: Date and Time

```
Format  String-to-numeric conversion function
        -------+----------------------------------------
        %tc     clock(string, mask)
        %tC     Clock(string, mask)

        %td     date(string, mask)

        %tw     weekly(string, mask)
        %tm     monthly(string, mask)
        %tq     quarterly(string, mask)
        %th     halfyearly(string, mask)
        %ty     yearly(string, mask)

        %tg     no function necessary; read as numeric
        ------------------------------------------------
```

Variable Types: Date and Time

		----- Numerical value & interpretation ------		
Format	Meaning	Value = -1	Value = 0	Value = 1
%tc	clock	31dec1959 23:59:59.999	01jan1960 00:00:00.000	01jan1960 00:00:00.001
%td	days	31dec1959	01jan1960	02jan1960
%tw	weeks	1959w52	1960w1	1960w2
%tm	months	1959m12	1960m1	1960m2
%tq	quarters	1959q4	1960q1	1960q2
%th	half-years	1959h2	1960h1	1960h2
%tg	generic	-1	0	1

Variable Types: Date and Time

- To convert a string variable into date/time format, first select the date/time format you'll be using (e.g., %tc, %td, %tw, etc.)
- Let's say we create a string variable, today's date (today) that we want to format in a new

```
gen today = “Feb 18, 2011”
gen date1 = date(today, “MDY”)
```

Variable Types

- What if your variable is time admitted (time) formatted as DDMMYYYYhhmmss

  ```
  generate double time2 = clock(time, “DMYHMS”)
  ```

  - “double” command necessary for all clock formats – basically tells Stata to allow a long string of characters

Exercise 1: Generate, Replace, Recode & Egen

Merging Datasets

- 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
- If you want to add OBSERVATIONS, you’d use “append” (we’ll go over that next)

Merging Datasets

- Several different ways that you might be interested in merging data
  - Two datasets with same participant pool, one row per participant (1:1)
  - A dataset with one participant per row with a dataset with multiple rows per participant (1:many or many:1)

Merging Datasets

- Stata will create a new variable ("_merge") that describes the source of the data
  - Use option, "nogenerate" if you don't want _merge created
  - Use option, "generate(varname)" to give _merge your own variable name
- Need to add IDs to your dataset?

```
generate id _n
```

Merging Datasets

- Before you begin:
  - Identify the “ID” that you will use to merge your two datasets
  - Determine which variables you’d like to merge
  - In Stata 11, data does NOT have to be sorted
  - Make sure your “using dataset” is in the same directory as your master dataset (otherwise, you’ll have to write the path name)
  - Variable types must match across datasets
    - Can use “force” option to get around this, but not recommended

Merging Datasets

- Let’s say I want to perform a 1:1 merge using the dataset “data2” and the ID, “participant”

```
merge 1:1 participant using data2.dta
```

- Now, let’s say that I have one dataset with individual students (master) and another dataset with information about the students’ schools called “school”

```
merge m:1 schoolID using school.dta
```

  - The “m” stands for “many”

Merging Datasets

- What if my school dataset was the master and my student dataset was the merging dataset?

```
merge 1:m schoolID using student.dta
```

- It is also possible to do a many:many merge
  - This method is not recommended – very few circumstances under which this would be useful
  - Data needs to be sorted in both the master and using datasets

Merging Datasets

- 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

Merging Datasets

- Update and replace options:
  - What if you want data from your using dataset to overwrite that in your master?
    - Specify "replace update" – it will replace master data with using data UNLESS the value is missing in the using dataset

Appending Datasets

- Sometimes, you'll have observations in two different datasets, or you'd like to add observations to an existing dataset
- Append will simply add observations to the end of the observations in the master

```
append using dataset2
```

Appending Datasets

- Some options with Append:
  - generate(newvar) will create variable that identifies source of observation
    - append using dataset1, generate(observesource)
  - "force" will allow for data type mismatches (again, this is not recommended)

Joinby

- Merge will add new observations from using that do not appear in master
- Sometimes, you need to add variables from using but want to be sure the list of participants in your master does not change

```
joinby participant using dataset1
```

- Any observations in using that are NOT in master will be omitted

Collapse

- Collapse will take master data and create a new dataset of summary statistics
- Useful in hierarchal linear modeling if you’d like to create aggregate, summary statistics
- Can generate group summary data for many descriptive stats
  - Mean, media, sd, sum, min, max, percentiles, standard errors
- Can also attach weights

Collapse

- Before you collapse
  - Save your master dataset and then save it again under a new name
    - This will prevent collapse from writing over your original data
  - Consider issues of missing data. Do you want Stata to use all possible observations? If not:
    - cw (casewise) option will make casewise deletions

Collapse

- Let’s say you have a dataset with patient information from multiple hospitals
- You want to generate mean levels of patient satisfaction for EACH hospital

```
save originaldata
save hospitalcollapse
collapse (mean) ptsatisfaction, by(hospital)
```

Collapse

- You could also generate different statistics for multiple variables

```
collapse (mean) ptsatisfaction length_stay (median)
   ptincome (sd) ptsatisfaction, by(hosptial)
```

- What if you want to rename your new variables in this process?

```
collapse (mean) ptsatmean=ptsatisfaction
lsmean=length_stay (median) ptincmed=ptincome
(sd) sdptsat=ptsatisfaction, by(hospital)
```

Exercise 2: Merge, Append, & Joinby

Comparing Two Datasets

- Stata allows you to compare the variable list of master to variable list of using dataset
- Important – Stata does not compare variable labels, value labels, etc. – Stata just compares variable values

Comparing Two Datasets

- Let’s say you’re interested in just seeing whether values on all variables in the master match values on all variables in using

```
cf _all using dataset1
```

  - The “_all” tells Stata to compare all variables
  - “cf” is short for “compare files”
- Stata only reports mismatches
  - If all values match, Stata will produce NO output

Comparing Two Datasets

- Let’s say Stata tells you that there are mismatching values on some of your variables. Now you’ll need to identify these values:

  cf _all using dataset1, verbose

  – “verbose” command tells Stata to print output that identifies where mismatches are

```
education:  1 mismatches
obs    2. 18 in master; 15 in using
hours:  1 mismatches
obs   20. 50 in master; -1 in using
marital:  1 mismatches
obs  995. . in master; 5 in using
income:  1 mismatches
obs 1353. 5 in master; 0 in using
age:  1 mismatches
obs 1413. 12 in master; . in using
```

Other Services Available

- MIT’s membership in HMDC provided by schools and departments at MIT
- Institute for Quantitative Social Science
  - www.iq.harvard.edu
- Research Computing
  - www.iq.harvard.edu/research_computing
- Computer labs
  - www.iq.harvard.edu/facilities
- Training
  - www.iq.harvard.edu/training
- Data repository
  - http://libraries.mit.edu/get/hmdc

Thank you!

Institute for Quantitative Social Science (IQSS) offers statistical workshops in Stata, SAS and R throughout the semester.

The R Series

- Introduction to R
- R and Statistics
- R Programming

Stata and SAS Courses

- Introduction to Stata
- Data Management in Stata
- Regression in Stata
- Graphics in Stata
- Introduction to SAS

For more information, visit:
http://support.hmdc.harvard.edu/kb-20/statistical_support
Sign up anytime by emailing:
dataclass@help.hmdc.harvard.edu

Thank you!

All of these courses will be offered during MIT's IAP and again at Harvard during the Spring 2011 semester.

- Introduction to Stata
- Data Management in Stata
- Regression in Stata
- Graphics in Stata
- Introduction to R
- Introduction to SAS

Sign up for MIT workshops at:
http://libraries.mit.edu/guides/subjects/data/training/workshops.html

Sign up for Harvard workshops by emailing:
dataclass@help.hmdc.harvard.edu