Results Based Financing for Health Impact Evaluation Workshop

Tunis, Tunisia

October 2010

Dealing with STATA and a Single Data Set

Basics of Using Stata

Session 1

Outline of Session

1) Beginning: opening a dataset
2) Getting help
3) Do files
4) Log files
5) Exploring the data
6) Summarizing variables
7) Changing data

A Quick Comment

- There are many different ways to do the same thing in STATA. Don't worry if you see something done a different way

- **You will find additional information and examples in your user guide "Getting started with STATA"**

Outline of Session

1) **Beginning: opening a dataset**
2) Getting help
3) Do files
4) Log files
5) Exploring the data
6) Summarizing variables
7) Changing data

Beginning:
Opening STATA

Option 1:

Start → All programs → Stata

Option 2:

Double-click on STATA Icon on your Desktop

Intercooled Stata 9.lnk

Beginning:
STATA Windows

- **Variables**
  - Lists the variables that are in your data set
- **Command**
  - Where you type commands into STATA in order to obtain an output
- **Review**
  - Lists all the commands that you have already used.
  - Allows us to easily repeat command by clicking on the right one
- **Results**
  - Where all the output from our commands will appear

Beginning:
Opening STATA Data

- Step 1: Set Memory
  - Command: **set mem #m**
  - **set mem 100m / 200m**
- Step 2: Remove data from memory before opening new data
  - Command: **clear**
- Step 3: Tell STATA which folder you want to look in for your data file in STATA format (here "session1.dta")
  - Command
    - *Option 1*: **File→Open→** Find folder and dataset you want to open
    - *Option 2* : **use "filelocation\filename.dta" →** Use full directory information
    - *Option 3* (2 steps)
      - **gl data = "filelocation" →** replace by appropriate file path
      - **use "$data\session1.dta"**

Outline of Session

1) Beginning: opening a dataset
2) **Getting help**
3) Do files
4) Log files
5) Exploring the data
6) Summarizing variables
7) Changing data

Getting help:
2 scenarios

- **Scenario 1**
  - You know the name of the command but don't know how to use it
    - Command: **help** *command*
- **Scenario 2**
  - You know what you want to do but don't know the name of the command to use
    - Help→ search → then type what you are looking for
    - Or you can use the command: **search** *word*

Outline of Session

1) Beginning: opening a dataset
2) Getting help
3) **Do files**
4) Log files
5) Exploring the data
6) Summarizing variables
7) Changing data

Do-files:
Using a Do-file

- **Keeps a record of all commands in a text document**
  - Can execute all commands in one go
  - Don't have to repeat work→ re-create dataset by running do-file
  - Easy to correct mistakes you find later

- **To create/ edit do-file**
  - Starting a new do-file: click on the 'new do-file editor' icon or enter the command **doedit**
  - Editing an existing do-file: **doedit DOFILENAME.do**
  - Saving do-file: *control* + s or, in do-file click File→Save

- **To run the do-file**
  - **do DOFILENAME.do**

Do-files:

Helpful Hints for Good Do-Files

- **Be Organized!**
  - Include headings for different sections of do-file
  - Include comments and notes to yourself

➔*For all text in do-files that are not commands, begin the line with " * "*

- **Include at the beginning of every do-file**
  - set mem #m
  - Headings
    - To describe do-file purpose, date, author, etc.
  - File location
    - If file location changes, only need to change it once at start of do-file
      - Command: **cd** *FILELOCATION*
      - **cd "C:\Documents and Settings\Desktop\STATA workshop"**

Do-files: *#delimit* Command

- **#delimit resets the character that marks the end of a command**
  - At beginning of do-file, type **#delimit;**
  - Mark the end of a command line with a semi-colon

- **Why?**
  - Allows commands to go on more than one line
  - Easier to read do-file
  - Can open do-file in other programs (Word, etc.) more easily

➔ *Don't forget to include a semi-colon after every command in do-file!*

Outline of Session

1) Beginning: opening a dataset
2) Getting help
3) Do files
4) **Log files**
5) Exploring the data
6) Summarizing variables
7) Changing data

Log files:
Keeping a Log file

- **Do files vs. log files**
  - Do file = keeps a record of all the *commands*
  - Log file = makes *a full record of your Stata session*
    - record of all the outputs created as a result of the commands used
    - stores your entire statistical analysis

- Opening a new log:
  - **log using LOGNAME.txt**
- Writing over an existing log:
  - **log using LOGNAME.txt, replace**
- Adding to an existing log
  - **log using LOGNAME.txt, append**
- Closing log to save it:
  - **log close**

Outline of Session

1) Beginning: opening a dataset
2) Getting help
3) Do files
4) Log files
5) **Exploring the data**
6) Summarizing variables
7) Changing data

Exploring the Data:

Browsing and Editing the Data

- **browse** command
  - Opens a matrix with actual data
    - Column holds the **variable**
    - Row holds the **observation**
    - Cell of a particular variable for a particular observation = **value**
    - When no information is recorded on a particular observation for a particular variable = **missing value**
  - Look at only some variables: **browse a1_11a a1_12**
  - Look at only some observations: **browse if a1_12==2**
  - Look at only some observations of only some variables: **browse a1_11a if a1_11a<30**

- **edit** command
  - Opens a matrix with actual data
  - Can change any value by highlighting a cell

➔ *Close data browser or editor before continuing with STATA*

➔ ***IMPORTANT: If you are a simple user of the data, you should not edit it. In case you edit the data, keep a do-file/log-file of what you have done.***

Exploring the Data:
Getting to Know the Variables

- **describe** command
  - Overview of the dataset:
    - Number of observations in the dataset
    - Number of variables in the dataset
    - Amount of memory the dataset is using and how much memory you still have to work with
    - Basic information about the variables in our dataset
  - Overview of some variables only:
    - **describe a1_11a a1_12**

- **codebook** command
  - More detailed overview of the variables
    - Variable name, label and type
    - Some basic descriptive statistics for variable
  - More detailed overview of some variables only:
    - **codebook a1_11a a1_12**

Exploring the Data: *Variable Types*

- **Numeric**
  - Stata reads as number
  - Different types: byte, int, long, float, double (different numbers of decimal points stored)
  - Missing value is denoted by "."
- **String**
  - Stata reads as text
  - String types are str1, str2, str3, etc. (# after *str* indicates the maximum length of the string)
  - Missing value is denoted by " " (blank)

➔ For further details: **help data types**

Exploring the Data: *Variable Labels and Value Labels*

- Variable labels give a brief description of the variable
  - a1_11a is labeled "age/years"
  - a1_12 is labeled "Marital status"

- Value labels put word labels on numeric category variables
  - E.g. code for marital status (a1_12)
    - 1 Never Married
    - 2 Married/Civil Union (Monogamous)
    - 3 Married (Polygamous)
    - 4 Cohabitating
    - 5 Divorced/Separated
    - 6 Widowed
  - We read label names in output even though variable values are numeric (**tab a1_12**)

Outline of Session

1) Beginning: opening a dataset
2) Getting help
3) Do files
4) Log files
5) Exploring the data
6) **Summarizing variables**
7) Changing data

Summarizing variables:
One- and Two-Way Tables of Summary

- **One-way table: tabulate varname1**
  - E.g. **tab a1_12**
  - # of times each value appears in data for specified variable
  - % of observations that take on that value
  - Can include missing values: **tab a1_12, miss**
- **Two-way table: tabulate varname1 varname2**
  - E.g. **tab a1_12 b12_12**
  - Varname1 appears as row, varname2 appears as column
- **Can also look at:**
  - % of obs with each combination: **tab a1_12 b12_12, cell**
  - % of obs with each varname1 value for each varname2 column separately: **tab a1_12 b12_12, col**
  - % of obs with each varname2 value for each varname1 row separately: **tab a1_12 b12_12, row**
- **To eliminate variable count:**
  - Add *nofreq* at the end: **tab a1_12 b12_12, row nofreq**

Summarizing variables: *Summary Statistics Tables*

- **summarize (or sum)** command
  - **sum varname1 varname2 …** (all the variables you want)
    - E.g. **sum a1_11a b12_17**
    - Returns basic summary statistics
    - # non-missing obs, mean, sd, min & max of values
- **sum …,d** command
  - **sum varname1 varname2 …, d** (all the variables you want)
    - E.g. **sum a1_11a b12_17, d**
    - Returns additional statistics
    - Skewness, kurtosis, smallest and largest values, and various percentiles.

- **When to use tab vs. sum? *General guidelines:***
  - **tab**: categorical variables (sex, marital status, province) & discrete values (# children)
  - **sum**: continuous variables (income, out-of-pocket payments) & discrete variables

➔ *A discrete variable can be either tabulated or summarized (e.g. age, household size)*

Summarizing variables:
Producing Tables of Statistics

- **Tabstat** command
  - Produces table of statistics you choose for as many variables as you want
  - Better tables for presentation than with **sum** command

- Specify variables and stats you want in table
  - **tabstat varname1 varname2 … , s(statistics…)**
  - E.g. **tabstat a1_11a b12_17, s(mean sd)**

➔*For the list of available statistics :* ***help tabstat***

- For stats by sub-group, add **by(*subgroupvariable*)** as option
  - E.g. to see mean & sd of age by marital status
    - **tabstat a1_11a , s(mean sd) by(a1_12)**

Summarizing variables:
Exporting Tables to Excel, etc.

- **Copy & Paste Method**
  - Paste into Word for informal tables
  - Use Courier New 9

- **Copy Table & Paste Method**
  - Paste into Excel for formal, formatted tables

➔ *Copy only the table and not other output to maintain formatting!*

Summarizing variables: *Identifying outliers*

- **Outliers**
  - extreme values of observed variables that can distort estimates
- **Detecting the problem**
  - **tab1** *varlist*: produces one-way tables for each variable
  - **histogram** *var1*
  - **scatter** *var1 var2*: produces twoway scatterplots
- **Dealing with outliers**
  - Use measures that are not sensitive to them, such as the median instead of the mean
  - Delete outliers from the data set (usually by setting them equal to a missing value)
  - Command: **mvdecode varlist, mv(numlist)** to change numeric values to missing values
- **Example**
  - Rule: let's consider any figure over 6 as outlier for the variable "total children given birth (male)".
  - Create a variable equal to b12_01a → **gen boyspw= b12_01a**
  - Send outliers to missing → **replace boyspw=. if b12_01a>6**
  - Or → **mvdecode boyspw, mv (7 8 9)**
  - Check results → **tab boyspw, miss**

Outline of Session

1) Beginning: opening a dataset
2) Getting help
3) Do files
4) Log files
5) Exploring the data
6) Summarizing variables
7) **Changing data**

Changing data: *Command Structure*

[*1 By*] : [2 Command] [3 Var] *[4 Specify] [5 If], [6 Options]*

- **1 "by"**
  - Qualifying clause (optional)
  - Repeats the command on subsets of the data
- **2 Command**
  - Primary instruction to STATA
- **3 Variables**
  - One or more variables
- **4 Specify more information for the command**
  - For some commands
- **5 "if"**
  - Qualifying clause (optional)
  - Means that the command is to use only the data specified.
- **6 Options**
  - Extra specifications
  - Always at the end and always after a comma

Changing data: *"by" clause*

[*1 By*] : [2 command] [3 Var] [4 Specify] [*5 If*], [*6 Options*]

- Performs commands by a sub-group (specified by a variable)
  - Sometimes at the beginning (before the command in [1])
    - **by varname, sort: ……**
    - **bysort varname: ……**
  - Sometimes at the end as an option (in [6])
    - ….., **by (varname)**

- Example:
  - Currently using contraceptive method by marital status:
    - by a1_12, sort: tab b12_12
    - bysort a1_12: tab b12_12

Changing data: *"if" clause*

[*1 By*] : [2 command] [3 Var] [4 Specify] [*5 If*], [*6 Options*]

- Tells STATA to only apply command to certain observations
- Comes after you have told STATA what you want to do
- Common "if" expressions:
  - >, >=, <, <=, ==, != or ~=
- For more than one restriction use & (and)
  - E.g. if age>20 & age<=30
- For multiple possibilities use | (or)
  - E.g. if year==2004 | year==2005

Changing data:
Some Basic Commands

- **generate (gen) & extended generate (egen)**
  - Both generate new variables
- **replace**
  - Replaces values for existing variables
- **rename**
  - Changes the name of a variable
- **label var**
  - Gives label to explain variable content
- **label val**
  - Gives labels to values that a variable takes on
- **drop**
  - Drops the variables or observations specified
- **keep**
  - Keeps only variables or observations specified (drops all others)

Changing data:
Using "gen" (generate)

- Generates new variables

- Examples of **gen** command:
  - **gen ones = 1** (column of ones)
  - **gen age=a1_11a**
  - **gen age20=1 if a1_11a==20**
  - **gen over30=1 if a1_11a >30 & a1_11a!=.**
    - STATA reads missing values as infinity, so be careful when using > and >=
  - **gen varname = varname1 / varname2**
    - Divides varname1 by varname2
  - Functions that work with gen are basic ones:+, -, *, /, etc.

➔ *Note:Variable names can never start with a number*

Changing data: *Using "replace"*

- Used just like **gen** but for existing variables

- Example of **replace** command:
  - **replace over30=0 if age<=30 &age!=.**
    - **over30** becomes a dummy variable

➔ The single equal, =, is used as a set equal operator. It is used in the **generate** and **replace** commands

➔ The double equals, ==, is used to test for equality. It is part of a logical test that returns either a one (true) or a zero (false)

Changing data: *Using "egen"*

- Generates variables but often uses more sophisticated functions
  - statistical functions like mean, sd, etc.

- **egen** examples:
  - egen mean_varname=mean(varname)
    - **egen mean_age=mean(age)**
  - For only one sub-group of sample
    - **egen mean_age_over30=mean(age) if over30==1**
  - Separately for each sub-group
    - e.g. mean for those 30 and under & mean for those over 30
    - **by over30, sort: egen agegrpmean=mean(age)**
    - **egen agegrpmean=mean(age), by(over30)**

Changing data:

Renaming and Labeling Variables

- Renaming Variables
  - rename command changes the variable name
    - **rename [current variable name] [new variable name]**
    - E.g.: **rename over30 thirty_plus**

- Labeling Variables
  - Variable labels describe the variable you created
    - Good idea to do this so that you remember later and so others understand your dataset!
    - **label var varname "[short description of the variable]"**
    - E.g.: **label var over30 "=1 if woman is older than 30"**

Changing data: *Labeling Values*

- Value labels put word labels on category variables
  - E.g.: no=0 and yes=1 in dataset

- Step One: Define the label
  - **label def [lbl name] [value1] "[lbl for value1]" value2 "[lbl for value2]"**
  - E.g.: **label def ny 0 "No" 1 "Yes"**
  - *Value always comes first & labels go in quotes*

- Step Two: Apply the value label to that variable
  - **label val [variable you are labeling] [label you want to apply]**
  - E.g.: **label val over30 ny**

➔*Note: The same label can be used again for other variables!*

Changing data:

Dropping & Keeping Variables and Observations

- **Dropping and keeping variables**
  - **drop** deletes variables you tell STATA to drop
    - **drop *varname***
  - **keep** drops everything EXCEPT the variables you tell STATA to keep
    - **keep *varname1 varname2 …***

- **Dropping and keeping observations**
  - Specify the observations you want to delete/keep using "if" clause
    - **drop if over30==0**
      - Drops all observations for which variable over30 is equal to zero
    - **keep if over30==1**
      - Keeps all observations for which variable over30 is equal to one

Changing data
Repeated commands

- Command: **foreach**
- Loop over items
  - loops are used to do repetitive tasks
- Syntax

  foreach *item* **in** a-list-of-things **{**
  body of loop using `item' ...
  }
- Example

```
label def ny 0 "No" 1 "Yes"
foreach x in   b12_06a b12_06b b12_06c b12_06d b12_06e b12_06f
    b12_06g b12_06h b12_06i b12_06j b12_06k b12_06l {
replace `x'=0 if `x'==2
label value `x' ny
tab `x‘
}
```

Changing data:
Saving Changes

- Always save altered dataset ***with a new name***
  - **save NEWDATASET** to save a new dataset
  - **save NEWDATASET, replace** to save over old
    - E.g.: **save session1_changed.dta, replace**

➔ **Never** save over the original dataset!!!

Thank you