

# Introduction to the Stata Language

Mark Lunt

Arthritis Research UK Epidemiology Unit  
University of Manchester



27/09/2011

## Topics Covered

- Getting help
- Stata Windows
- Basic Concepts
- Manipulation of variables
- Manipulation of datasets

## Command-line vs. Point-and-Click

- Command-line requires more initial learning than point-and-click
- Only option for any serious work
  - 1 Reproducible
  - 2 Editable
  - 3 More efficient

# Getting Help

- Help
- Search
- Manuals
- Stata website
- Statalist
- Stata Journal
- Me

# Stata Windows

- 2 must exist:
  - Results
  - Command
- 2 others usually exist
  - Review
  - Variables
- Others can exist (data editor, graph, do-file editor, help/log viewer)

## Command Window: Syntax

`command [varlist] [,options]`

- Roman letters: entered exactly
- *Italic letters*: replaced by some text you enter
- Square brackets: that item is optional
- Example above means means:
  - Command is called “command”
  - Command name may be followed by a list of variables
  - Options may follow a comma

## Command Window

- Can navigate through previous commands with `PageUp` and `PageDown`.
- Case-sensitive: `height` and `HEIGHT` are different variables
- Syntax must be *exact* (although abbreviations are possible)
  - Only one comma, before all options
  - No space before opening bracket in an option (e.g. `level(5)`, not `level (5)`).

## Variables window

- List of all variables in current dataset
- Clicking adds variable name to command window
- May contain label if one has been defined

## Review Window

- List of commands entered this session
- Clicking on a command puts it in command window
- Double-clicking runs the command
- Can be saved as a do-file

## Results Window

- Limited size: use a log file to preserve results
- Blue = clickable link
- Scrolling controlled by Return, Space and q keys.
- `set more [on | off]`

# Basic Concepts

- Do-files
- Log files
- Interaction with Operating System
- Macros
- Variable and number lists

## Do-Files

- List of commands
- Can be run from stata with the command `do do-file.do`
- All data manipulation and analysis should be done using a do-file.
  - Perfectly reproducible
  - Can see exactly what was done
  - Easy to modify

## Profile.do

- Stata looks for a file called `profile.do` every time it starts.
- If it finds it, it runs it
- Useful for
  - Setting memory
  - User-defined menus
  - Logging commands
- See `help profilew` for details

## Log Files

- Results window of limited size: must log results
- Can use plain text or SMCL (stata markup and control language)
- Top of do file should be:  
capture log close  
log using *myfile.log*, [append] [replace]  
([text]||[smcl])

## Interaction with Operating System

`cd`          Change directory  
`pwd`        Display current directory  
`mkdir`      Create directory  
`dir`        List files in current directory  
`shell`     Run another program

- Can use either "/" or "\" in directory names.
- Better to use "/"
- Path names containing spaces must be surrounded by inverted commas.

# Macros

- Macro name is replaced by definition text when command is run.
- Very useful for making do-files portable
  - Directories used are defined first using macros
  - Change in location of data or do-files only means changing macro definitions

## Macro Example

- **Definition:** `global mymac C:/Project/Data`
- **Use:**
  - use `"$mymac/data"`
  - **Loads the file** `C:/Project/Data/data`

## Local vs. Global

- Global macro retains definition until end of session
- Local macro loses definition at end of do-file

	Definition	Use
Global	<code>global mymac defn</code>	<code>\$mymac</code>
Local	<code>local mymac defn</code>	<code>'mymac'</code>

*Local vs Global macros*

## Variable Lists

- Shorthand for defining a lot of variables
- `prefix*` means all variables beginning with `prefix`
- `firstvar-lastvar` means all variables in the dataset from `firstvar` to `lastvar` inclusive.
- Type `help varlist` for more details

# Number Lists

Symbol	Meaning	Example	Expansion
	list of numbers	1 2 3	1 2 3
$x/y$	whole numbers from $x$ to $y$ inclusive	1/5	1 2 3 4 5
$x\ y\ to\ z$	numbers from $x$ to $z$ , increasing by $y - x$	5 10 to 20	5 10 15 20
$x\ y : z$	same as $x\ y\ to\ z$	5 10:20	5 10 15 20
$x(y)z$	numbers from $x$ to $z$ , increasing by $y$	10(10)50	10 20 30 40 50
$x[y]z$	same as $x(y)z$	10[10]50	10 20 30 40 50

## *Number Lists*

# Manipulating Variables

- generate & replace
- egen
- Labelling
- Selecting variables

# generate

- Used to create a new variable
- **Syntax:** `generate [type] newvar = expression`
- *newvar* must not already exist
- *type*, if present, defines the type of the data
- *expression* defines the values: e.g.
  - `generate ltitre = log(titre)`
  - `generate str6 head = substr(name, 1, 6)`

# Variable Types

type	size (bytes)	min	max	precision	missing
byte	1	-127	126	integers	.
int	2	-32,767	32,766	integers	.
long	4	-2,147,483,647	2,147,483,646	integers	.
*float	4	$-10^{36}$	$10^{36}$	7 digits	.
double	8	$-10^{308}$	$10^{308}$	15 digits	.
strn	n				" "

## *Available data types*

\*float is the default type.

## Missing Values

- Numerical variables can have several different missing values:
  - `., .a, .b`, etc
  - May be useful if you know why a variable is missing
  - `if variable != .` may not catch all missing values
- All missing values are *greater* than any number representable by that datatype.
  - Can exclude all missing values with `if variable < .`
  - `gen old = age > 65 if age < .`

# replace

- Similar to generate
- Cannot change type
- *newvar* must already exist

- Extended GENerate
- Has more functions available
- No `erename`: must drop the existing variable and create a new one
- Examples of its use in the practical
- See `help egen` for details

# Labelling

- Need to label variables themselves
  - show exactly what the variable measures
- Need to label values of a variable
  - Only for categorical variables
  - First define a label
  - Then assign it to a variable

## Labelling a variable

Syntax: `label variable varname "Description"`

Example: `label variable height "Height in m."`

## Labelling values

**Syntax:** `label define labelname 1 "string1" ...`  
`label values varname labelname`

**Example:** `label define yesno 0 "No" 1 "Yes"`  
`label values question1 yesno`  
`label values question2 yesno`

# Selecting variables

- `drop varlist`
- `keep varlist`

# Manipulating Datasets

- use & save
- append
- merge
- browse and edit
- preserve and restore

## use

- `use "filename"` reads a file into stata
- If there is already a file in stata, need `use "filename", clear`
- Always use inverted commas
- Easier to use the menu or button-bar

## save

- `save "filename"` saves the current dataset as `"filename"`
- If `"filename"` already exists, need `save "filename", replace`
- Option `saveold` allows saving in format of a previous version of stata

# Combining Datasets

- append
  - more subjects, same variables
  - append using *filename*
- merge
  - same subjects, more variables
  - merge *identifier* using *filename*

## Appending Data: Example

ID	common_1	common_2	file1_1	file1_2
1	$a_1$	$b_1$	$c_1$	$d_1$
2	$a_2$	$b_2$	$c_2$	$d_2$
3	$a_3$	$b_3$	$c_3$	$d_3$

*Appending Data: File 1*

ID	common_1	common_2	file2_1	file2_2
4	$a_4$	$b_4$	$e_4$	$f_4$
5	$a_5$	$b_5$	$e_5$	$f_5$
6	$a_6$	$b_6$	$e_6$	$f_6$

*Appending Data: File 2*

## Appending Data: Example

ID	common_1	common_2	file1_1	file1_2	file2_1	file2_2
1	$a_1$	$b_1$	$c_1$	$d_1$	.	.
2	$a_2$	$b_2$	$c_2$	$d_2$	.	.
3	$a_3$	$b_3$	$c_3$	$d_3$	.	.
4	$a_4$	$b_4$	.	.	$e_4$	$f_4$
5	$a_5$	$b_5$	.	.	$e_5$	$f_5$
6	$a_6$	$b_6$	.	.	$e_6$	$f_6$

*Appending Data: Combined Files*

## Merging Data

- Need an identifier (one or more variables on which to match observations)
- Both files must be sorted by this identifier
- All observations from both files are used
- Variable `_merge` says whether observation was in first file, second file or both.

## Merging Files: example

idno	var1	var2
1	$a_1$	$b_1$
2	$a_2$	$b_2$
3	$a_3$	$b_3$

*Merging Data: File 1*

idno	var3	var4
1	$c_1$	$d_1$
3	$c_3$	$d_3$
4	$c_4$	$d_4$

*Merging Data: File 2*

## Merging Files: example

idno	var1	var2	var3	var4	_merge
1	$a_1$	$b_1$	$c_1$	$d_1$	3
2	$a_2$	$b_2$	.	.	1
3	$a_3$	$b_3$	$c_3$	$d_3$	3
4	.	.	$c_4$	$d_4$	2

*Merging Data: Combined Files*

## Ensuring Uniqueness

- Usually, should only be one observation per unique identifier
- May not be the case (e.g. adding family-level data to individual-level data)
- Can ensure uniqueness with the option `unique`
- Other uniqueness options: see `help merge`

## browse & edit

- Can open a data editor window with `browse`
- Can choose variables to browse with `browse varlist`
- Cannot modify data while browsing
- `edit` allows data to be changed: don't use it

## preserve & restore

- You may wish to change your data temporarily
- E.g. collapse to means by group
- Type `preserve` before changing data, `restore` after