An Introduction to Stata for Economists -
Part I: Data Management

Steve Bond and Stefan Hubner*

* We thank Kerry Papps (Bath) and Daniel Gutknecht (Mannheim) for sharing these slides.

1. Overview

- Brief guide to the display windows and toolbar
- Interactive vs. batch mode
- Introduction to Stata commands
- Options for entering data
- “Log” files
- Formats
- Modifying the data
- Combining datasets
- Creating a dataset of means or medians etc.

2. Comment on notation used

- Consider the following syntax description:

  `list` [*varlist*] [`in` *range*]

  – Text in `typewriter-style font` should be typed exactly as it appears (although there are possibilities for abbreviation).

  – *Italicised* text should be replaced by desired variable names *etc*.

  – Square brackets (*i.e.* []) enclose optional Stata commands (do not actually type these).

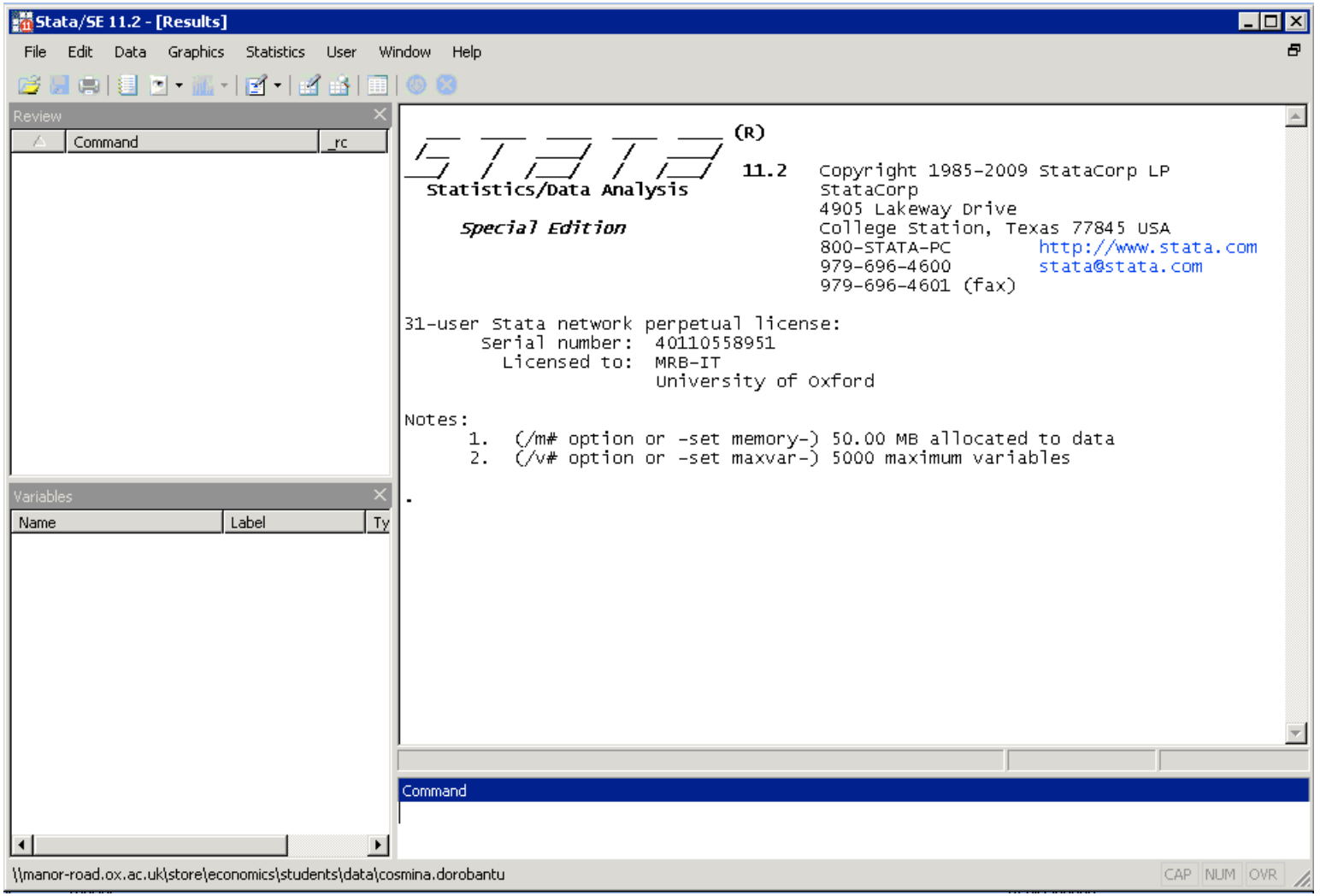
3. Comment on notation used (cont.)

- For example, an actual Stata command might be:

```
list name occupation
```

- This notation is consistent with notation in Stata Help menu and manuals.

4. The Stata windows



5. Navigating around Stata

- **Results window:** The big window. Results of all Stata commands appear here (except graphs which are shown in their own windows).
- **Command window:** Below the results window. Commands are entered here.
- **Review window:** Records all Stata commands that have been entered. A previous command can be repeated by double-clicking the command in the Review window (or by using Page Up).

6. Navigating around Stata (cont.)

- **Variables window:** Shows a record of all variables in the dataset that is currently being used.
- **Toolbar:** Across the top of the screen. Note the  (break) button, which allows any Stata command taking a long time to be interrupted.
- **Spreadsheet:** Click the  (editor) button. All data (both imported and derived) are visible here. Note that no commands can be executed when the data editor is open for versions prior to Stata11.

EXERCISE 1
7. Getting to know Stata

- Open Stata.
- Identify the Results window, Command window, Review window, Variables window.
- Open the data editor () and experiment with entering some data (type values and press Enter).
- Exit the data editor and then clear the memory by typing `clear` in Command window.
- Look at Help Menu (Help → Contents).

8. Ways of running Stata

- There are two ways to operate Stata.
  - Interactive mode: Commands can be typed directly into the Command window and executed by pressing Enter.
  - Batch mode: Commands can be written in a separate file (called a do-file) and executed together in one step.
- We will use interactive mode for the exercises in this set of slides, and batch mode in the next set.

9. Ways of running Stata (cont.)

- Note: one can also execute many commands using the drop-down menus.

10. Introduction to Stata commands

- Stata syntax *is* case sensitive. All Stata command names must be in lower case.
- Many Stata commands can be abbreviated (look for underlined letters in “Help”).
- Up to Stata 12, it may be necessary to increase the memory limit in Stata from the default of 1 megabyte when working with large datasets (note that there must be no data in memory at this stage):

  `set memory #`

  (# represents a number of kilobytes (`k`), megabytes (`m`) or gigabytes (`g`).)

11. Introduction to Stata commands (cont.)

- For example:

  ```
  set memory 100m
  ```

- By default, Stata assumes all files are in `c:\data`.
- To change this working directory, type:

  `cd` *foldername*

- If the folder name contains blanks, it must be enclosed in quotation marks.

12. Using Stata datasets

- Stata datasets always have the extension .dta.
- Access existing Stata dataset *filename.dta* by selecting File → Open or by typing:

  `use` *filename* [`, clear`]
- If the file name contains blanks, the address must be enclosed in quotation marks.
- *filename* can also be a Stata file stored on the internet.

13. Using Stata datasets (cont.)

- If a dataset is already in memory (and is not required to be saved), empty memory with `clear` option.
- To save a dataset, click  or type:

  `save` *filename* `[, replace]`
- Use `replace` option when overwriting an existing Stata (.dta) dataset.

14. Creating Stata datasets

- There are various ways to enter data into Stata; the choice depends on the nature of the input data:
  - Manual entry by typing or pasting data into data editor
  - Inputting ASCII files using `infile`, `insheet` or `infix`
  - Use of other software to directly create a new Stata dataset from another format (*e.g.* SAS, SPSS)

15. Using ASCII datasets

- Must have data in ASCII (text) format.
- If using text editing package to assemble dataset, save as text (.txt) file, not default (*e.g.* .xlsx).
- Options:
  - Free format data (*i.e.* columns separated by space, tab or comma *etc.*): use `infile` or `insheet`.
  - Fixed format data (*i.e.* data in fixed columns): use `infix`.

16. Entering free format data

- Can use `insheet` when input data created in spreadsheet package, *e.g*. Excel:

  `insheet using` *filename*
- First row of data file assumed to contain the variable names.
- Can use `infile` for other types of free format data, but more complicated (need to list all variables).

EXERCISE 2

17. Entering free format data

- Create a folder for your Stata files (*e.g.* `c:\stataworkshop`) and change the working directory to that using `cd`.
- Use `insheet` to read in the dataset:

  `http://people.bath.ac.uk/klp33/stata_data.txt`
- Save file (in your working directory) as "`Economic data.dta`".

18. Entering fixed format data

- Basic structure of `infix` command:

  `infix` *var1 startcol1-fincol1 var2 startcol2-fincol2* … `using` *filename*
- If a variable contains non-numeric data, precede the variable name by `str`.
- Example:

```
infix year 1-4 unemplrate 6-9 str
  country 11-30 using
  c:\unempldata.txt
```

19. Entering fixed format data (cont.)

- Also possible to begin reading data at a particular line in file or for each observation to spread over more than one line.

EXERCISE 3

20. Entering fixed format data

- Read in the following dataset using `infix`:
  `http://people.bath.ac.uk/klp33/stata_data_2.txt`
- This is fixed format data. Variables, types and positions are:
  - country string 1-14
  - capital string 17-26
  - area real 30-35
  - eu_admission real 41-44
- Save file as "`EU data.dta`".

21. Transferring other files into Stata format

- If data in another format (*e.g.* SAS, SPSS), Stat/Transfer can be used to create a Stata dataset directly.
- Can also handle Excel files.
- Able to optimise the size of the file (in terms of the memory required for each variable).

22. Labelling data

- A label is a description of a variable in up to 80 characters. Useful when producing graphs *etc*.
- To create/modify labels either double-click on appropriate column in spreadsheet or type:

  `label variable` *varname "label"*
- Value labels can also be defined.

23. Log files

- All Stata commands and their results (except graphs) are stored in a log file.
- At the start of each Stata session, it is good practice to open a log file, using the command:

  `log using` *filename*

  (where *filename* is chosen)
- To close the log, type:

  `log close`

24. Formats

- All variables are formatted as either numeric (real) or alphanumeric (string).
- You can instantly tell the format of a variable in the spreadsheet by its colour: black for numeric and red for alphanumeric.
- Alternatively, look at the “Type” column in the Variables window or type:

  `describe [`*varlist*`]`

25. Formats (cont.)

- The letter at the end of the “display format” column tells you what the format is: “`s`” for string and any other letter (*e.g.* “`g`”) for numeric.
- Missing values are denoted as dots (`.`) for numeric variables and blank cells for string variables.

26. Inspecting the data

- `codebook` is useful for checking for data errors. This gives information on each variable about data type, label, range, missing values, mean, standard deviation *etc*.
- Alternatively, `list` simply prints out the data for inspection. (Remember the break option.)
- Both `codebook` and `list` can be restricted to specific variables or observations.

27. Inspecting the data (cont.)

- `tabulate` generates one or two-way tables of frequencies (also useful for checking data):

  `tabulate` *rowvar* [*colvar*]

- For example, to obtain a cross-tabulation of `sex` and `educ` type:

  ```
  tab sex educ
  ```

28. Variable transformations

- New variables can be created using `generate`:

  `generate` *newvar* `=` *exp*
- *exp* can contain functions or combinations of existing variables, *e.g.*:

  `gen gdp=c+i+g`
- `replace` may be used to change the contents of an existing variable:

  `replace` *oldvar* `=` *exp1* [`if` *exp2*]
- Any functions that can be used with `generate` can be used with `replace`.

29. Variable transformations (cont.)

- `if` is used to restrict the command to a desired subset of observations, *e.g.*:

```
replace unemplrate=. if
  unemplrate==999
```

- Note that the double equal sign == is used to test for equality, while the single equal sign = is used for assignment.
- Logical operators can be used after `if`:
  - & denotes “and”
  - | denotes “or”
  - ~ or ! denote “not” (*e.g.* ~= is “not equal to”)

30. Variable transformations (cont.)

- For example, to create a dummy variable use:

```
gen highun=0
replace highun=1 if unemplrate>=8
  & unemplrate~=.
```

- Note that "." treated as an infinitely large number (be careful!).
- A shorter alternative to the above code is:

```
gen highun=(unemplrate>=8 &
  unemplrate~=.)
```

31. Variable transformations (cont.)

- `rename` may be used to rename variables, as follows:

  `rename` *oldvarname newvarname*
- To drop a variable or variables, type:

  `drop` *varlist*
- Alternatively, `keep` *varlist* eliminates everything but *varlist*.
- To drop certain observations, use:

  `drop if` *exp*
- For example, `drop if unemplrate==.`

EXERCISE 4
32.Variable transformations

- Open the dataset "`Economic data.dta`".
- Use `describe` to ascertain which variables are in string format and which are in real format.
- Rename `percentagewithsecondaryeduc` as `secondary`.
- Convert `lfpr` from a decimal into a percentage using `replace` (*i.e.* multiply it by 100).
- Keep only those observations between 1992 and 2006 (use either `drop` or `keep`).

EXERCISE 4 (cont.)
33.Variable transformations

- Create a GDP per capita variable called `gdpcap` using `generate`.
- Create an employment/population rate using:

```
gen emplrate = (100-unemplrate)*
  lfpr/100
```

- Label `gdp` as `"GDP at market prices (2000 US$)"`.
- Save the modified dataset. (Remember to use `replace` option.)

34. Appending datasets

- To add another Stata dataset below the end of the dataset in memory, type:

  `append using` *filename*
- Dataset in memory is called “master dataset”.
- Dataset *filename* is called “using dataset”.
- Variables (*i.e.* with same name) in both datasets will be combined.
- Variables in only one dataset will have missing values for observations from the other dataset.

35. Merging datasets

- To join corresponding observations from a Stata dataset with those in the dataset in memory, type:

  `merge 1:1` *varlist* `using` *filename*
- Stata will join observations with common values of *varlist*, which must be present in both datasets.
- If more than one observation has the same value of *varlist* in the master dataset, use:

  `merge m:1` *varlist* `using` *filename*
- If more than one observation has the same value of *varlist* in the using dataset, use:

  `merge 1:m` *varlist* `using` *filename*

36. Merging datasets (cont.)

- The variable `_merge` is automatically added to the dataset, containing:

`_merge==1`	Observation from master data
`_merge==2`	Observation from using data
`_merge==3`	Observation from both master and using data

- Stata reports the number of observations with each value of `_merge`.

EXERCISE 5
37. Merging

- Open "`Economic data.dta`" (the master dataset) and merge with "`EU data.dta`" (the using dataset) using `country` as the match variable.
- Should you use `merge 1:1`, `merge m:1` or `merge 1:m`?
- Look at the values that `_merge` takes: what does this indicate?

EXERCISE 5 (cont.)
38. Merging

- Remove those observations that do not contain data from both files:

```
drop if _merge==1
```

- Create a dummy variable called `eu` for whether a country was a member of the EU in a given year.
- Save the modified dataset as `"Combined EU data.dta"`.

39. “By group” processing

- To execute a Stata command separately for each group of observations for which the values of the variables in *varlist* are the same, type:

  `by` *varlist*`:` *command*

- Most commands allow the `by` prefix.
- Requires that data be sorted by *varlist* (precede command with `sort` *varlist* or use `bysort`).

40. Collapsing datasets

- To create a dataset of means, sums *etc*., type:

  `collapse (`*stat*`)` *varlist1* `(`*stat*`)` …
  `[[`*weight*`]], by(`*varlist2*`)`
- *stat* can be `mean`, `sd`, `sum`, `median` or other statistics.
- `by(`*varlist2*`)` specifies the groups over which the means *etc*. are to be calculated.

41. Collapsing datasets (cont.)

- Be sure to save data before attempting collapse as there is no “undo” facility.
- Example:

```
collapse (mean) age educ (median)
  income, by(country)
```

42. Collapsing datasets (cont.)

- Four types of weight can be used in Stata:
  - `fweight` (frequency weights): weights indicate the number of duplicated observations.
  - `pweight` (sampling weights): weights denote the inverse of the probability that an observation is included in the sample.

43. Collapsing datasets (cont.)

- `aweight` (analytic weights): weights are inversely proportional to the variance of an observation to correct for heteroskedasticity. Often, observations represent averages and weights are number of elements that gave rise to the average.
- `iweight` (importance weights): weights have no other interpretation.

44. Collapsing datasets (cont.)

- Example:

```
collapse (mean) unemplrate
  [aweight=labforce], by(country)
```

- Weights may be used in many other Stata commands, *e.g.* `correlate`, `regress`.
- Note that the square brackets around the weight must be typed.

EXERCISE 6
45. Collapsing

- Collapse the dataset "`Combined EU data.dta`" by `year` to produce a dataset containing the sums of `pop` and `area` and the means of `gdpcap`, `lfpr`, `unemplrate` and `secondary` across the entire EU.
- Use `aweight=pop` so that the variables take into account the changing populations of the countries.
- In what years did the EU have the highest unemployment rate and the highest GDP per capita?

EXERCISE 6 (cont.)
46. Collapsing

- Looking at the data editor, can you spot a problem with the collapsed data?
- A more appropriate `collapse` step would use `rawsum` rather than `sum`, which computes the unweighted sum.