Data Management Using Stata

Iowa Social Research Center (ISRC) Workshop

Desmond D. Wallace

Department of Political Science
The University of Iowa
Iowa City, IA

September 22, 2017

RECALL: Opening a Dataset

- The command for opening a dataset in Stata is `use`.
- If a dataset is already open, opening a new dataset requires including the option `clear` with the `use` command.
- Examples
  - Example: `use` *`filename`* works if there is no data in Stata's memory.
  - Example: `use` *`filename`*`, clear` works if data is already in memory.

Importing Data

- Can read non .dta files into memory using the `import` command.
  - Excel files: `import excel [using]` *`filename`*`, firstrow clear`
  - Delimited files: `import delimited [using]` *`filename`*`, clear`
- See `help import_excel` and `help import_delimited` for more information.

Saving and Exporting Data

- Use the `save` and `saveold` commands to save data in memory to a .dta file.
  - Stata 15 and 14: `save [`*filename*`], replace`
  - Stata 13, 12, and 11: `saveold [`*filename*`], version(#) replace`
  - **NOTE: Stata 11 through 13 files NOT COMPATIBLE with Stata 14 and 15!!!**
- Can export data in memory to Excel and delimited files.
  - Excel files: `export excel [using]` *filename*`, firstrow(variables) replace`
  - Delimited files: `export delimited [using]` *filename*`, replace`
- See `help export_excel` and `help export_delimited` for more information.

Sorting Data

- Use the `sort` and `gsort` commands to arrange data.
  - `sort` arranges data in ascending order only.
  - `gsort [+|-] varname [[+|-] varname ...]`
    - `+` – Sort in ascending order
    - `-` – Sort in descending order

Subsetting Data

- Use `drop` or `keep` in combination with an `if` or `in` statement to subset observations.
  - `drop [in` *`range`*`] if` *`exp`* eliminates observations from memory satisfying specified condition(s).
  - `keep [in` *`range`*`] if` *`exp`* keeps observations from memory satisfying specified condition(s).
- Use `drop` *`varlist`* to eliminate variables or `keep` *`varlist`* to keep variables
- **NOTE:** `drop` **and** `keep` **are NOT reversible.**

Generating Variables

- `generate` command creates a new variable.
  - `generate [type] =exp [if] [in]`
  - If *type* is not specified, variable type is determined by *exp*
- `replace` command replaces the contents of an existing variable.
  - `replace oldvar =exp [if] [in]`
- `egen` command used to create variables based on special functions.
  - `egen [type] newvar = fcn(arguments) [if] [in], by(varlist)`
  - `by` option is used to calculate values over specified variables.
  - Functions written for use with `egen` are ONLY for `egen`
- See `help generate` and `help egen` for more information.

Recoding Variables

- Use the `recode` command to change values of categorical variables.
  - `recode` *varlist* `(`*rule*`) [(`*rule*`) ...], generate(`*newvar*`)` *options*
  - `recode` *varlist* `(`*erule*`) [(`*erule*`) ...], generate(`*newvar*`)` *options*
- Use the `generate` option to save recoded variable to new variable.

Recoding Variable Rules

- *rule*
  1. `# = #`
  2. `# # = #`
  3. `#/# = #`
  4. `nonmissing = #`
  5. `missing = #`
- *erule*
  1. `# | #/# = el [``label'']`
  2. `nonmissing = el [``label'']`
  3. `missing = el [``label'']`
  4. `else | * = el [``label'']`

Recoding Variable Rules

- Keywords `missing`, `nonmissing`, and `else` must be the last rules specified.
- `else` cannot be combined with `missing` or `nonmissing`.
- Must use the `generate` option when recoding a variable, and specifying value labels.
- See `help recode` for more information.

Summarizing Data

- Recall, the `summarize` command is used to report summary statistics for variables.
- Can use the `collapse` command to create a dataset of summary statistics.
  - `collapse [(stat)] varlist [ [(stat)] ... ] [if] [in], by(varlist)`
  - `by` option is used to calculate summary statistics over specified variables.
  - If *stat* is not specified, default statistic calculated is the mean.
  - **NOTE: Only the variables specified in `collapse` command remain in memory.**
  - See `help collapse` for more information, including full list of statistics.

Append Datasets

- Appending datasets is when one wishes to combine two, or more, datasets vertically.
- Adding observations to existing datasets.
- `append using` *`filename`* `[`*`filename`* `...], [`*`options`*`]`
- See `help append` for more information.

Merge Datasets

- Merging datasets is when one wishes to combine two, or more, datasets horizontally.
- Adding variables to existing datasets.
- merge *join varlist* using *filename*, [*options*]
- Unlike append, can only use one dataset on file at a time.
- Specified variables must uniquely identify observations.

Example Merging Join Types

- One-to-one merge (1:1)
  - Merging datasets that have the same uniquely identified observations.
  - Each dataset holds one observation per unique case.
  - `merge 1:1` *`varlist`* `using` *`filename`*`, [`*`options`*`]`
  - Does not matter which dataset is master (file loaded into memory) and which dataset is using (file not loaded into memory).
- Many-to-one merge (m:1)/One-to-many merge (1:m)
  - Merging datasets where one dataset has multiple observations per unit and another dataset has single observation per unit.
  - `merge m:1` *`varlist`* `using` *`filename`*`, [`*`options`*`]`
  - `merge 1:m` *`varlist`* `using` *`filename`*`, [`*`options`*`]`
  - Matters which dataset is master and which dataset is using.

Any Questions?