Empirical Economics

Violations of Regression Assumptions and Methods for Longitudinal Data (Stata Seminar 3)

Teacher: Andrew Proctor
andrew.proctor@phdstudent.hhs.se

September 22, 2017

Outline

1. Further general techniques for Stata
   - Tips to improve your Stata code
   - Loops in Stata
2. Post-Estimation and Specification Testing
   - Post-Estimation Commands
   - Violations of error distribution assumptions
3. Panel Methods in Stata

Tips to improve your Stata code

Here are a few tips to keep your do-files looking readable and make your execution more convenient:

- Keep your variable names short (but long enough to be a reasonable abbreviation)
- If a line is very long (you can use the do-editor guideline as a way too judge), try splitting your commands over multiple lines using three forward slashes: ///

```
recode monthlyincpr (1 = 5) (2 = 17.5) (3 = 37.5) (4 = 62.5) (5 = 82.5) (6 = 95), gen(income_pctile)

recode monthlyincpr (1 = 5) (2 = 17.5) (3 = 37.5) (4 = 62.5) (5 = 82.5) ///
    (6 = 95), gen(income_pctile)
```

Tips to make your work more convenient and more readable cont'd

- To prevent Stata output from pausing and requiring you to click "more" in the results window when the results of a command are too large for the screen, use the command (in your preliminaries section): **set more off**
- If you are writing an 'if' statement that requires multiple inequalities, two good alternatives are the **inrange** and **inlist** commands.

```
reg income_pctile cogn_samp_pctile potent_exper i.female if ///
    ((age >= 30) & (age <= 65))
reg income_pctile cogn_samp_pctile potent_exper i.female if ///
    inrange(age, 30, 65)
```

Loops in Stata

Another tool that saves both time and writing is to use loops to perform the same operations on multiple sets of observations is 'loops.' These are somewhat advanced but incredibly useful.

- There are two main types of loops in Stata:
- **forval** loops, and
- **foreach** loops.

forval loops

- **forval** loops are used to iterate a command over a range of integers.
  - For example, what if you wanted to create a series of indicator variables for different years in a dataset. You could manually write out the **generate** and **replace** commands for each year in the dataset, or you could write these commands once and then 'loop' the command over the range of years in your dataset.
- The way to run a forloop for a range starting with [startnum] and ending with [endnum] is:
  forval i=[startnum]/[endnum] {
  [YOUR COMMANDS HERE]
  }

forval loops cont'd

- Wherever you would usually be writing the number that is being looped, instead replace it with the index letter (i here) in quotes using first the wide quote (`) and then the straight quote, ('), as such **`i'**.
- Here's an example of creating 65 year indicator variables using a forval loop:

```
**** Use 'forval' loop to create y1950-y2017 year indicator variables
forval i=1950/2015 {
        gen y`i' = .
        replace y`i' = 0 if ( (year!=`i') & !missing(year) )
        replace y`i' = 1 if ( (year==`i') & !missing(year) )
    }
```

foreach loops

- **foreach** loops are used instead to loop over a series of objects (typically variables) instead of values.
  - For instance, suppose you want to conduct a balance test by running a bunch of simple linear regressions of a (hypothetically) randomized variable on a number of observable characteristics.
- The way to run a foreach loop is:
  foreach j in [LIST OBJECTS] {
  [YOUR COMMANDS HERE]
  }

foreach loops cont'd

- The letter j, like 'i' in the forval example, is just a signifier to tell Stata that whenever you use this character in quotes during the loop, that's where the objects should be inserted.
  - In the foreach loop, instead of using a letter to signify what you're looping across, a descriptive word is often used instead.
- Here's an example of a **foreach** loop:

```
/* A 'foreach' loop to run 7 simple regressions to test whether a supposedly
    random variable is correlated with any observable characteristics:    */

foreach var in parentaleduc hhincome rural female black latino nonnative {
      reg randomized `var'
     }
```

Basics of Post-Estimation Commands

Another feature of Stata that we have not talked much about so far is post estimation commands.

- These are commands that you enter immediately after a regression to tell it compute something else using the data stored from the regression.
- One of the most common post estimation commands is **margins**, which computes the marginal effects of a variable. For OLS, this is especially helpful when you have quadratic or higher order terms in your regression.

```
reg income_pctile cogn_samp_pctile c.potent_exper##c.potent_exper ///
    i.female  if inrange(age, 30, 65)
    *** Marginal Effects
    margins, dydx(potent_exper) at(potent_exper=(5 10 15 25 35)) // OR
    margins, dydx(potent_exper) atmeans
```

Basics of Post-Estimation Commands ctd

Perhaps more commonly, post-estimation commands are used to run tests.

- One common specification test is the Ramsey RESET test, which uses squared and higher powers of the fitted values to check whether the regression has been mispecified, omitting higher order terms or interactions of the explanatory variables.
- To perform the Ramsey RESET after a regression, the command is **ovtest**

```
regress incwage educ potexp female black asian native otherormulti immig
    *Run Ramsey RESET Test
    ovtest
```

Violations of error distribution assumptions

Two key types of tests that can be run after a regression involve testing Guass-Markov assumptions about the distribution of the error term.

- From Wooldridge, Assumptions MLR 4-6 can be summarized as:

$$u|x_1, ..., x_n \overset{iid}{\sim} N(0, \sigma^2)$$

- The two major violations of this condition are:

Heteroskedasticity: $\sigma_i^2 \neq \sigma^2 \; \forall i$, and

Autocorrelation: $Cov(u_i, u_j) \neq 0$for some i, j

Heteroskedasticity

The most pervasive violation of the two is heteroskedasticity.

- Rarely, in practice, do we have any reason to believe that the variance of unobserved factors affecting outcomes is constant among individuals.
- There are multiple ways to test if errors are heteroskedastic in the dataset. One good approach that we've previously mentioned in this course is the Breusch-Pagan (BP) test.
  - You can run the (generalized) Breusch-Pagan test in Stata via the command **estat hettest, iid** following the regression.

Autocorrelation and Violation of Error Independence

A problem that is somewhat less pervasive but often more problematic is when errors are not independently distributed.

- Errors in a dataset can be correlated for a number of reasons. Often, omitted factors captured in the residual may affects groups of observations, so that the 'true' error structure may be thought of as

$$e_{ig} = v_g + u_{ig},$$

where v_g is the group term and u_{ig} is is the individual term (which may be homoskedastic).

- For panel data (which has multiple observations for a person/group over time), it's instead likely that individuals' errors may be correlated over time (autocorrelation), so that:

$$u_{it} = \rho u_{i,t-1} + \eta_{it}$$

Autocorrelation and Violation of Error Independence ctd

There are multiple ways to test for autocorrelation in the error terms:

- If your dataset consists of a repeated observations for a single panel(i.e. individual/group):
  - If the error is a function only of a new shock and the immediately previous (i.e. lagged) residual, there is the Durbin-Watson test: **estat durbinalt**
  - If there is a persistent effect of multiple lags (more than the just the last residual), then the Breusch—Godfrey test is likely better: **estat bgodfrey**
- If you instead want to to test for autocorrelation in a regression with multiple panels:
  - You can instead perform tests such as the Wooldridge (2002) test, implemented by the **xtserial** command.
  - Or the newer Inoue and Solon (2006) test, which is implemented by the **xtistest** command.

Dealing with error distribution violations

OLS is generally still unbiased & consistent in the presence of heteroskedasticity and autocorrelation, but if you don't account for these violations, standard errors (and thus inference) will be wrong.

- If heteroskedasticity is the main issue suspected, use the option **robust** after your **regress** commands.
- If instead you suspect significant departure from independence of error terms (e.g. autocorrelation or within group correlation), then instead use the option **cluster([CLUSTERVAR])**, where [CLUSTERVAR] is the grouping variable in which errors are thought to be correlated.
- Generally speaking, you should always at least use the **robust** option. There is almost never reason to believe in homoskedasticity.

Basics of panel data

- Panel data, where data is recorded for individuals (or aggregate groups) over time, offers some major advantages for empirical research, while also commonly implicating the assumption violations just mentioned.
    - The major advantage of panel data is that repeated observations of a person or group (e.g. region) allows for 'fixed effect' indicator variables that control for all time-invariant features in that group, the same way indicator variables control for common features of other groups (e.g. females).
    - However repeated observations of the same individuals over time allows for temporal correlation that is autocorrelation.

Using the 'reshape' command to work with panel data

- Often panel data is reported in 'wide' format, where each year (or other time increment) is reported as a new column (with rows corresponding to different variables).
- For Stata, we always want the columns to represent different variables and the rows to represent observations (the 'long' format).

Using the 'reshape' command ctd

Data before reshape:

	countrycode	indicatorcode	year47	year48	year49	year50	year51
1	ARB	domcredtopriv	35.992713	34.922893	34.007707	34.909079	35.053313
2	ARB	pcgdp	4720.1943	4865.9433	5204.4409	5378.3792	5594.8985
3	ARB	gdppcgrowth	-1.5141602	3.0877756	6.9564651	3.3421124	4.0257361
4	ARB	sanitation	73.415417	74.047607	74.716338	75.410072	76.080392
5	ARB	waterq	82.692883	82.766386	82.868783	82.948543	83.041854
6	ARB	tuburc	.	.	.	.	.
7	ARB	secondeduc	60.583778	60.689373	61.470268	62.970181	62.11697
8	ARB	cellph	25275663	35121269	52132873	84854521	1.260e+08
9	ARB	femmortal	143.5371	141.5055	139.47715	137.44558	135.30105
10	ARB	workingagepop	58.796763	59.344157	59.875579	60.376616	60.866008
11	ARB	totalpop	2.960e+08	3.024e+08	3.092e+08	3.163e+08	3.238e+08
12	ARB	anemia	36.160225	36.565608	36.927819	37.130876	36.983113
13	ARB	womenlegislseats	6.1323519	6.0439175	6.7026706	7.0417823	9.3578192
14	ARB	secondenroll	62.234928	64.037773	65.853859	65.931007	65.847931
15	ARB	tertenroll	19.569469	19.55899	20.69919	21.73148	22.191481
16	ARB	journals	9074.1756	10149.937	11516.188	12553.15	14389.76
17	ARB	tariff	.	.	.	.	.
18	ARB	tax	.	.	.	.	.
19	ARB	contractenftime	.	634.84615	642.6875	667.44444	667.44444
20	ARB	natresource	18.492381	21.746106	27.250342	34.337253	34.981006

Variables

Filter variables here

Name	Label
year34	1989
year35	1990
year36	1991
year37	1992
year38	1993
year39	1994
year40	1995
year41	1996
year42	1997
year43	1998
year44	1999
year45	2000
year46	2001
year47	2002
year48	2003
year49	2004
year50	2005
year51	2006
year52	2007
year53	2008

Reshaping data from 'wide' to 'long'

To change data from 'wide' to 'long', we can use the **reshape** command.

The syntax to reshape 'long' is **reshape long [PREFIX], i(IDENTIFIER VARIABLES) j(WIDEVAR)**, where:

WIDEVAR is the wide variable (that is the variable where each value is a different column),

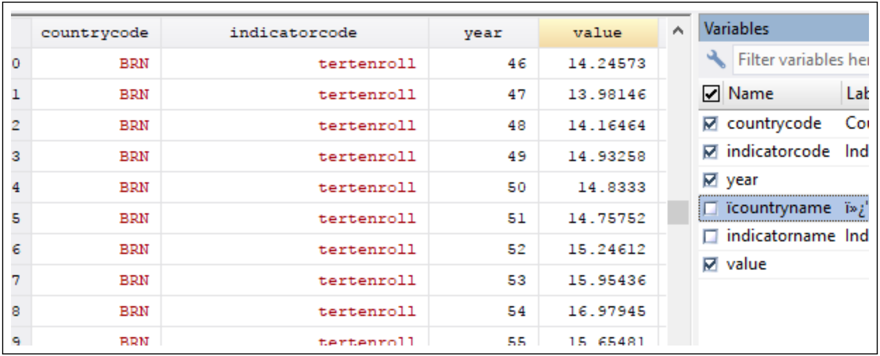
PREFIX is the common element of the wide variable column names (e.g. year), and

IDENTIFIER VARIABLES are the variables that, together with the [WIDEVAR], characterize what the value represents.

Reshape ctd

Data after **reshape long**:

```
*** Get year as a single column
reshape long yr, i(countrycode indicatorcode ) j(year)
rename yr value
```



	countrycode	indicatorcode	year	value
0	BRN	tertenroll	46	14.24573
1	BRN	tertenroll	47	13.98146
2	BRN	tertenroll	48	14.16464
3	BRN	tertenroll	49	14.93258
4	BRN	tertenroll	50	14.8333
5	BRN	tertenroll	51	14.75752
6	BRN	tertenroll	52	15.24612
7	BRN	tertenroll	53	15.95436
8	BRN	tertenroll	54	16.97945
9	BRN	tertenroll	55	15.65481

Reshaping data from 'long' to 'wide'

Sometimes you instead have the problem that what you want to use as variables (and therefore have formatted as columns in Stata) are instead expressed as different values of a single column/variable.

$\longrightarrow$ You then need to reshape the data so that these different rows (which are *long*) are reshaped to be different columns (*wide*).

Reshaping data from 'long' to 'wide' ctd

The syntax of **reshape wide** largely mirrors the syntax of **reshape long**.

WIDEVAR — now is simply used to specify which column contains the values you want to express as different variables.

PREFIX — is somewhat different for **reshape wide**. In [PREFIX], you should now specify which column in the dataset (before reshaping) contains the values for the new wide variables. This column name will then be assigned as a prefix in each of the new (wide) variable names.

IDENTIFIER VARIABLES — are again the variables that, together with the [WIDEVAR] variable, characterize what each value in the dataset represents.

Collapse

- If you want to run regressions on aggregate instead of individual data, you can create aggregated data using the **collapse** command.
- **collapse** will transform your dataset from the microdata to aggregate data only, using the commands you specify.
- The collapse syntax is: **collapse ([STAT1]) [VAR1] [VAR2] ([STAT2..]) [VAR1] [VAR2]..., by(BYVARS)**,
  - Where the STAT* inputs are the statistic you want for aggregates, such as mean, median, sd (standard deviation), etc, and the BYVAR are the variables that you want to aggregate from, usually your panel variable (eg region) and time variable.

```
collapse (mean) wageinc educ potexper female black hisp asian other (median) wageinc educ
```

Fixed Effects Regression

Fixed effects regression is the workhorse of linear regression for time series data.

- Allows for convenient inclusion of fixed effects (while suppressing output for the fixed effect indicators.
- More natively (and efficiently) takes the potentially correlated error structure of the data into account (when using **robust**).
- The first thing necessary to do in order to run a fixed effects regression is to declare the structure of the data with **xtset**.
    - The syntax of **xtset** is **xtset [PANELVAR] [TIMEVAR]**.

Fixed Effects Regression

- To then run fixed effects regression, the command is **xtreg**.
  - The syntax is almost the same as **regress**, except you can specify inclusion of fixed effects by option **fe** and random effects by **re**.
  - You can include lags (/ leads) of the dependent or independent variable by using the prefix **L.** (/ **F.**) in front of the variable in the regression.
    - More than one lag or lead can be accomodated by putting the # desired in parentheses. For instead **L(0/4).Income** would include current income (lagged 0) and the first four lags of income (i.e. T-1,T-2,T-3,T-4).
    - Lag and lead notation can also be used outset of the regression command once panel data is **xtset** (e.g. to generate new variables).