AMM Tutorial 4

Panel Data Fun

Xinran Hu

August 20, 2024

1 Panel vs. Pooled Cross-section Data

Both panel and pooled cross-section data span multiple periods. But how are they different from each other?

Panel data: The definitive characteristic of panel data is that panel data have multiple observations of the SAME things over different time periods.

Pooled cross-Section data: Pooled cross-sectional data, on the other hand, does not require the observations in each period to be from the same pool of objects. The observations in each period could overlap with each other, but also could be completely unrelated.

You can think of pooled cross-section as a more scruffy version of panel data, where we give up the potential of ruling out specific unobserved factors by observing the same set of individuals over time. Pooled cross-section data can still be useful, though, especially when the factor we are interested in varies at a more aggregated level.

2 First Differencing

First differencing is a good way to remove any unobserved characteristics that are constant across time. (For simplicity, we'll be staying in a 2-period world.) By subtracting the relationship in period 2 from that in period 1, we eliminate any time-invariant components of the regression:

$$\begin{aligned} y_{it} &= \beta_0 + \delta_0 \mathbb{1}_t(t=2) + \beta_1 x_{1,it} + \beta_2 x_{2,it} + a_i + \varepsilon_{it} \\ y_{i1} &= \beta_0 + \delta_0 \times 0 + \beta_1 x_{1,i1} + \beta_2 x_{2,i1} + a_i + \varepsilon_{i1} \\ y_{i2} &= \beta_0 + \delta_0 \times 1 + \beta_1 x_{1,i2} + \beta_2 x_{2,i2} + a_i + \varepsilon_{i2} \\ \Rightarrow \Delta y_i &= \delta_0 + \beta_1 \Delta x_{1,i} + \beta_2 \Delta x_{2,i} + \Delta \varepsilon_i \qquad (1) \end{aligned}$$

The first thing to notice is that as everything we did are linear transformations, the slope coefficient for the term associated with Δx_i is still β_1. Next, the time-related difference across the two periods, $\delta_0 \mathbb{1}_t(t=2)$, conveniently provides us with the necessary intercept δ_0 for the differenced equation.

Keep in mind that first difference can only be applied to panel data - The purpose of taking first difference is to eliminate the term a_i. Hence we can't do the same thing with repeated cross-section data as there lacks clear-defined previous/next period counterparts to execute the subtraction.

The second pitfall for first differences is that you cannot include variables that experience the exact same magnitude of change for all individuals between the two periods. Why? Well, because imagine $x_{2,i2} = x_{2,i1} + 1$ for all i. Then $\Delta x_{2,i} = 1$ for all i and this means that Equation (1) becomes

$$\Delta y_i = \delta_0 + \beta_1 \Delta x_{1,i} + \beta_2 + \Delta \varepsilon_i,$$

where δ_0 and β_2 are perfectly collinear.

3 First Difference Regressions in Stata

How can we estimate a first difference model in Stata? Well, the only technical difficulty is to generate the first differences, which there are two approaches:

3.1 Manually

Remember what we did with `egen` last week? Turns out that we can do more than generating a value for all observations when doing cross-row manipulations.

Of course, whatever follows requires a sorted dataset. If you indicate more than one variable after the `sort` command, Stata will sort all observations in a nesting manner, i.e., start with all observations of the smallest value for variable 1 where observations are ranked according to values taken for variable 2, and the all observations of the second smallest value for variable 1 where observations are ranked according to values taken for variable 2, so on so forth.

With a sorted dataset, we can ask Stata to grab the previous or next observation for a specific variable! In Stata, `_n` gives you the index of the current observation, so `varable[_n]` pulls out the value of `variable` for each specific observation. Likewise, `varable[_n-1]` will give the value stored in the previous observation.

3.2 Automatically (kinda)

Stata allows a bunch of operators along the time dimensions if the dataset has been specified to have a time dimension. Given that we are working with panel data[1], we can inform Stata using the command `xtset`, followed by the variable identifying different individuals and then the variable identifying time periods. The only caveat is that the time period variable has to be consecutive.

[1]For time series data, use `tsset` command instead

After that, you can let Stata generate first differences by adding the `D.` operator in front of each variable!

4 After-the-tutorial Updates

4.1 Loops in Stata

Some of you might have noticed that in the Stata code at the end of week 3's lecture notes there is something that resembles a for loop. In Stata, there are two commands for executing for loops, `foreach` and `forvalues`.

The difference between the two is that `foreach` allows iteration over more general objects, while `forvalues` is for iterating over numbers only.

A `forvalues` iteration should look familiar if you have some programming background:

```
forvalues i = "some range" {
    Your command
}
```

where "some range" can take the form between several alternatives, but two of the most common options people use are `a(step)b` and `a/b` which allow you to execute a loop from number a to number b with an increment each time defined by *step* or 1 respectively.

On the other hand, `foreach` grants greater flexibility over the element of iterations. The most general version of a `foreach` loop is

```
foreach i in "a list"{
    Your command
}
```

The list specified in "a list" can be a list of variable names, strings, or numbers. There are more ways to specify the list for a `foreach` loop, which requires the use of `of` instead of `in` and specifying the type of list. `help forvalues` and `help foreach` will give you a more in-depth list of legitimate inputs.

Another thing to pay attention to is that when referring to the item of iteration in the "`Your command`" section, you'll need to surround `i` with single quotation marks as mentioned in week 2's appendix (Pressing the key to the left of 1, then i, then the good old single quote key). It should show up like `` `i' `` on your screen, and colored in some teal-ish green automatically.

4.2 Heteroskedasticity Adjustment

If you've read the answer sheet to this tutorial, you'll realize that I kinda made a dumb mistake - It's actually NOT correct to use heteroskedasticity robust standard errors for our regression!

Why? Well, it's not that assuming heteroskedasticity is wrong per se, but that the way `robust` standard errors are calculated has its own problem. Given that we are an applied econometrics course and not a theoretical one, I'll try to avoid going through the nerdy details.

Anyway, the reason why we would like to execute heteroskedasticity adjustments is because if the error terms are not identically distributed, then the OLS estimators are not efficient, i.e., they won't get you the correct variance estimates. The `robust` command tries to address this problem by estimating the standard errors using something called the White estimator instead. The White estimator relies on some very fancy statistical magic called asymptotics, which is the art of if the sample size is **large enough** then you can argue that the summary statistics have very neat distributions.

Our problem here is that the sample size is NOT large enough. If you recall from our regression, there are only 51 observations utilized in the first difference model. White estimators behave poorly under "small" sample as it involves taking one extra layer of averages and assuming that the sample average is as good as the true mean. "Behaving poorly" here means that it could be even more wrong than the vanilla standard errors, so you'll need to be extra cautious if, say, homoskedastic standard errors tell you to accept the null while robust standard errors under a small sample tells you to reject it.