Trade - Assignment 2

Jeanne Sorin

October 26, 2020

Table 1: Table 1

	reg	xtreg	areg	reghdfe
	(1)	(2)	(3)	(4)
	log(flow)	log(flow)	log(flow)	log(flow)
Contiguous countries (dummy)	0.974***	0.977***	0.974***	0.974***
	(0.0396)	(0.0399)	(0.0396)	(0.0396)
Common language (dummy)	0.906***	0.906***	0.906***	0.906***
	(0.0184)	(0.0185)	(0.0184)	(0.0184)
Log(distance)	-1.658***	-1.656***	-1.658***	-1.658***
	(0.00875)	(0.00880)	(0.00875)	(0.00875)
N	156248	156248	156248	156178
R2	0.718	0.577	0.718	0.718
Time (sec)	290.524	2.765	38.002	3.02

Standard errors in parentheses

* $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

The table above shows the results for the regression of log (non-zero) trade flows on log bilateral distance, a contiguity indicator, a common-language indicator, exporter-year FE, importer-year FE for the years 2000-2006.

We can see that the point estimates and standard errors are numerically identical across the different estimators, as they should be. Indeed, we are estimating the same model, with the same data! We are simply trying to exploit different numerical "shortcuts". If one looks carefully, we see that the estimates and standard errors differ slightly across columns. I believe this is due to numerical approximation.

The number of observations are the same for the first 3 columns, but different for the 4th column. My understanding is the observations dropped in the last columns are *"singleton"* observations, i.e. corresponding to pairs of FE *exporter-year + importer-year* containing a single observation. Therefore, no variation remains to explain once we control jointly for the 2 pairs of FE.

THe R2 in column 2 is different than in column 2 because of the panel structure we impose by using *xtreg*: the R2 reported with panel data models is cross-sectional rather than also across time (as for the 3 other columns).

Finally, we notice that the relative computation times of these estimators depend on the "*net*" dimensionality and size of the data. To be more precise, we see that *reg*, which takes the *raw* data is obnoxiously slow, while *reghdfe*, which "nets" out both pairs of FE before performing the regression, is much faster, as the net dimensionality is exponentially smaller. The speed doesn't seem to depend on the number of observations.

Table 2: Table 2

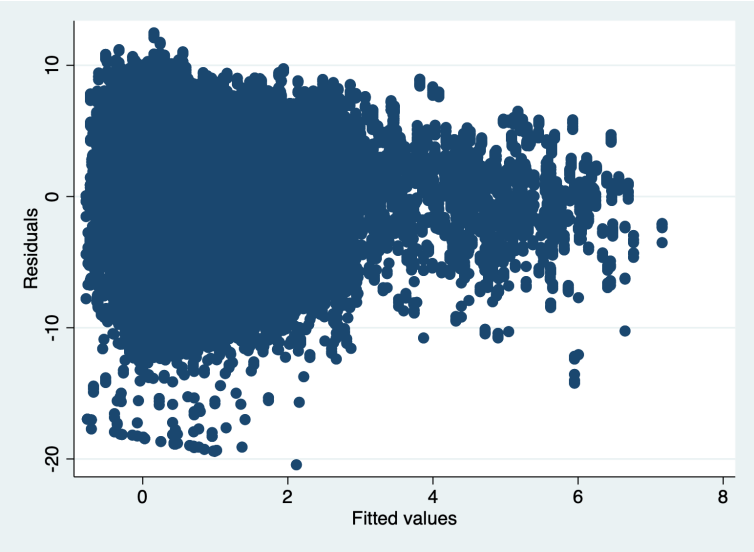
	(1) lflow	(2) log(flow+1)	(3) log(flow+1)	(4) flow	(5) flow	(6) flow	(7) flow	(8) flow
1 for contiguity	0.971***	1.017***	1.118***	0.462	0.462***	1.118***	0.462***	0.464***
	(0.0617)	(0.0343)	(0.0320)	(.)	(0.0377)	(0.0320)	(0.0377)	(0.0376)
Common 1st Language	0.975***	0.470***	0.433***	0.212	0.212***	0.433***	0.212***	0.211***
	(0.0286)	(0.0159)	(0.0132)	(.)	(0.0358)	(0.0132)	(0.0358)	(0.0357)
ldistw	-1.704***	-1.069***	-0.890***	-0.899	-0.899***	-0.890***	-0.899***	-0.899***
	(0.0136)	(0.00756)	(0.00652)	(.)	(0.0157)	(0.00652)	(0.0157)	(0.0157)
N	67350	67350	91652	91685	91652	91685	91652	67350
R2	0.718	0.758	0.742	0.907		0.907		
Command	reg	reg	reg	ppml	poi2hdfe	ppml_panel_sg	ppmlhdfe	ppmlhdfe
Flow > than 0 only	Yes	Yes	No	No	No	No	No	Yes
Time (sec)	.894	.848	1.061	2178.349	23.092	456.983	7.291	4.488

Standard errors in parentheses

* $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

Regarding whether the results are sensitive to the inclusion of 0, the answer is "*it depends*". Indeed the log-linear model gives dignificantly different results between columns 2 and 3. This is unsurprising: by excluding the zero trade flows observations, we actually estimate a different (conditional) model. Therefore, we see here that estimating a gravity equation through OLS and log is problematic because it de facto gets rid of observations with 0 trade flow.
However, when one estimates the constant-elasticity specification by running the *"ppmlhdfe"* command to implement the PPML estimator (columns 7 and 8), we see that estimates are really close to each other. ppml doesn't seem to suffer from the same flawn as the OLS approach to estimate gravity equations.
Regarding the impact of making the dependent variable $log(x+1)$, the answer is that it performs quite poorl for the log-linear regression (columns 1 vs 2), i.e. we get quite different results. If ever one found that the results displayed in table 2 were not that different (see for example the coefficient for *log distance* in column 3), one should try to run the same regression with $log(x+0.1)$ or any arbitrarily small number and would see that estimates are really off.
Results from the log-linear regression are quite heteroskedastic: running a Breusch-Pagan test in stata on the model corresponding to Table 2, Col 1, we find that $chi2 = 181.32$ and $Prob > chi2 = 0.0000$. Since the latter is < 0.05, we can reject the null hypothesis and we conclude that there is heteroskedasticity in the data.
Moreover, as one can see on the scatter plot below, the residuals' variance decreases with the fitted value, suggesting that the error does not have constant variance, i.e. that we are in the case of heteroskedasticity.
Finally, with respect to computation times, well, in a nutshell: the OLS approach can be really fast when

Figure 1: Illustrating Heteroskedasticity



using *reghdfe*, but is problematic because it doesn't deal with the zeros (no free lunch I guess). On the other hand, the ppml approach is slower, but is able to deal with zeros. While *ppml* is the turtle of the race, *ppmlhdfe* does quite well (and is not sensitive to including the zeros or not!). *poihdfe* is very decent.

Table 3: Table 3

	(1) log(flow)	(2) log(flow)	(3) log(flow)
Contiguous countries (dummy)	0.550***	0.5495***	0.550***
	(0.0160)	(0.0160)	(0.016)
Common language (dummy)	0.762***	0.7620***	0.762***
	(0.00774)	(0.0077)	(0.008)
Log(distance)	-1.325***	-1.325***	-1.325***
	(0.00379)	(0.0038)	(0.004)
N	709,248	709,573	709,248
R2	0.702	0.70258	0.702
Time (sec)	7.789	0.66837	2.43
Software	Stata	R	Julia

Robust Standard errors in parentheses

* $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

As one can see on the table above, the three packages return the same estimates. The standard errors are also equal, up to numerical approximation.
Well, R is faster, and by an order of magnitude at least!