The Stata module `felsdvreg` to fit a linear model with two high-dimensional fixed effects

Thomas Cornelissen
Leibniz Universität Hannover, Germany

German Stata Users Group meeting
WZB Berlin
June 27, 2008

Motivation

- Fixed-effects models allow to take into account time-constant unobserved heterogeneity that may be correlated with observables
- Datasets often allow to include more than one fixed effect into the analysis.
- Examples: linked employer-employee data, linked student-teacher data, individual effects and region (county) effects
- With high number of panel units ("high-dimensional fixed effects") $\Rightarrow$ Computer restrictions
- I propose the Stata module `felsdvreg` for a memory saving way to compute such a model

Linear two-way fixed-effects model

- Notation for one observation:

$$y_{it} = x'_{it}\beta + \theta_i + \psi_{J(it)} + \epsilon_{it}, \tag{1}$$

- Matrix notation:

$$y = X\beta + D\theta + F\psi + \epsilon, \tag{2}$$

- X - observed time-varying characteristics (may include further fixed effects: e.g. time effects, school effects, etc.)
- D - person effects, F - firm effects
  (Dummy variable matrices)
- coefficient vectors β, θ and ψ
- assumption: error term is orthogonal to all regressors, including the individual and firm effects

How estimate the model?

Include the firm effects as dummy variables and sweep-out the person effects by the within-transformation ("time-demeaning"). Andrews et al. (2006) call this "$FE_i LSDV_j$ method".

$$\text{Transformed model: } \tilde{y} = \tilde{X}\beta + \tilde{F}\psi + \tilde{\epsilon}. \tag{3}$$

Imagine a big linked employer-employee dataset:
$J = 10,000$ firms, $N^* = 20$ million person-years
$K = 50$ time-varying regressors, 4 bytes per data cell (float)

How estimate the model?

Include the firm effects as dummy variables and sweep-out the person effects by the within-transformation ("time-demeaning"). Andrews et al. (2006) call this "$FE_i LSDV_j$ method".

$$\text{Transformed model: } \tilde{y} = \tilde{X}\beta + \tilde{F}\psi + \tilde{\epsilon}. \tag{3}$$

Imagine a big linked employer-employee dataset:
$J = 10,000$ firms, $N^* = 20$ million person-years
$K = 50$ time-varying regressors, 4 bytes per data cell (float)

Matrix	Dimension	Storage requirement
$(\tilde{X}, \tilde{F})$	$N^* \times (K + J)$	800 GB.
$(\tilde{X}, \tilde{F})'(\tilde{X}, \tilde{F})$	$(K + J) \times (K + J)$	0.4 GB

How to reduce storage requirements ?

Note the cross-product matrices $(\tilde{X}, \tilde{F})'(\tilde{X}, \tilde{F})$ and $(\tilde{X}, \tilde{F})'\tilde{y}$ are much smaller than $(\tilde{X}, \tilde{F})$.

The OLS normal equations $(\tilde{X}, \tilde{F})'(\tilde{X}, \tilde{F}) \begin{pmatrix} \hat{\beta} \\ \hat{\psi} \end{pmatrix} = (\tilde{X}, \tilde{F})'\tilde{y}$ involve these cross-product matrices.

Main idea of the paper: Create cross-product matrices without fully creating the F and $\tilde{F}$ matrix.

F is a sparse matrix and it is a very inefficient way to store the information in which firm worker i works at time t. This information is much more efficiently stored in the firm identifier variable.

The cross product matrix

Let's look in more detail at $(\tilde{X}, \tilde{F})'(\tilde{X}, \tilde{F})$. This can be written as:

$$(\tilde{X}, \tilde{F})'(\tilde{X}, \tilde{F}) = \begin{pmatrix} \tilde{X}'\tilde{X} & \tilde{X}'\tilde{F} \\ \tilde{F}'\tilde{X} & \tilde{F}'\tilde{F} \end{pmatrix}$$

Due to the possibility of the row-wise decomposition this can be written as:

$$\sum_i \begin{pmatrix} \tilde{X}_i'\tilde{X}_i & \tilde{X}_i'\tilde{F}_i \\ \tilde{F}_i'\tilde{X}_i & \tilde{F}_i'\tilde{F}_i \end{pmatrix}$$

Are there any regularities about $\tilde{F}_i$?

What are F and $\tilde{F}$ like?

$$\overbrace{\left(\begin{array}{ccc} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ \hline 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ \hline 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{array}\right)}^{F} = \left(\begin{array}{c} F_1 \\ \\ \\ F_2 \\ \\ \\ F_3 \\ \\ \end{array}\right)$$

What are F and $\tilde{F}$ like?

$$\overbrace{\left(\begin{array}{ccc} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ \hline 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ \hline 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{array}\right)}^{F} = \left(\begin{array}{c} F_1 \\ \\ \\ F_2 \\ \\ \\ F_3 \\ \\ \end{array}\right) \qquad \overbrace{\left(\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline \frac{1}{3} & -\frac{1}{3} & 0 \\ \frac{1}{3} & -\frac{1}{3} & 0 \\ -\frac{2}{3} & \frac{2}{3} & 0 \end{array}\right)}^{\tilde{F}} = \left(\begin{array}{c} \tilde{F}_1 \\ \\ \\ \tilde{F}_2 \\ \\ \\ \tilde{F}_3 \\ \\ \end{array}\right)$$

Reduction of the problem

1. Stayers contribute nothing to $\tilde{F}'\tilde{F}$ and $\tilde{F}'\tilde{X}$.

$$(\tilde{X},\tilde{F})'(\tilde{X},\tilde{F}) = \begin{pmatrix} \tilde{X}'\tilde{X} & 0 \\ 0 & 0 \end{pmatrix} + \sum_{i \in \text{ Movers}} \begin{pmatrix} 0 & \tilde{X}_i'\tilde{F}_i \\ \tilde{F}_i'\tilde{X}_i & \tilde{F}_i'\tilde{F}_i \end{pmatrix}$$

2. Each mover contributes only to specific cells of $\tilde{F}'\tilde{F}$ and specific rows of $\tilde{F}'\tilde{X}$. Which cells these are and which values a mover contributes can be computed from the **firm identifier** variable.

For the product $(\tilde{X},\tilde{F})'\tilde{y}$ the argument is similar.

$\Rightarrow$ This idea is implemented in a Stata program `felsdvreg`.

The Stata module -felsdvreg-

`net search felsdvreg`, The Stata Journal 8(2), pp. 170-189.

1. Identifies stayers and movers.
2. Identifies groups of workers and firms that are connected by worker mobility and drops one firm effect per group.
3. Estimates the model by least squares implementing the memory-saving creation of the cross-product matrices and returns:
   a) coefficient estimates and standard errors
   b) the predicted person and firm effects,
   c) a mover indicator variable
   d) a grouping indicator (identifying the different groups connected by mobility)
   e) a variable containing the number of movers per firm
   f) a variable containing the number of observations per person

The Stata module -felsdvreg-

```
. felsdvreg y x1 x2, ivar(i) jvar(j) feff(feffhat) peff(peffhat) xb(xb) res(res)
> mover(mover) group(group) mnum(mnum) pobs(pobs)
Memory requirement for moment matrices in GB:
  2.17600e-06

Computing generalized inverse, dimension: 11
  Start: 6 Mar 2008 18:06:02
  End:   6 Mar 2008 18:06:02
N=100
```

	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
x1	1.029258	.2151235	4.78	0.000	.6000987	1.458418
x2	-.7094819	.2094198	-3.39	0.001	-1.127263	-.2917009

```
F-test that person and firm effects are equal to zero: F(28,69)=9.81 Prob > F = 0
F-test that person effects are equal to zero:          F(19,69)=8.64 Prob > F = 0
F-test that firm effects are equal to zero:            F(9,69)=9.97 Prob > F = 0
```

Time constraint for solving the LGS ?

- `felsdvreg` uses the fact that $(\tilde{X}, \tilde{F})$ is a sparse matrix in order to create its cross-product in a memory-saving way
- The program then solves the system of normal equations in a direct way (Cholesky decomposition or generalized inverse)
- This way of solving the LGS is of mathematical complexity $O(N^3)$, i.e. multiplying the number of firm effects by 10 increases computing time by a factor of 1000.
- For big problems different methods might be worthwhile $\Rightarrow$ **preconditioned conjugate gradient method** (Abowd, Creecy and Kramarz 2002)
- Could take advantage of the fact that $(\tilde{X}, \tilde{F})'(\tilde{X}, \tilde{F})$ is sparse, too.

Alternatives:

- `a2reg` by Ouazad (2008) (`net search a2reg`)
  - Implements preconditioned conjugate gradient algorithm
  - Solves for coefficients, not for standard errors ($\Rightarrow$ Booststrap)
  - The algorithm does not automatically detect regressors collinear to fixed effects (the generalized inverse used in `felsdvreg` drops these regressors automatically)
- Andrews, Schank, Upward (2006): Spell fixed effects
  - Call the unique combinations of the two fixed effects 'spells'
  - Estimate a one-way fixed-effects model using the spells as panel units
    This controls effectively for the unobserved heterogeneity but does not allow to recover the two fixed effects.

Conclusion

- Creating the fixed-effects dummies in a two-way fixed effects estimation is inefficient (memory-wise)
- A memory-saving way of computing the cross-product matrices for the system of normal equations has been presented
- The method is implemented in the Stata module `felsdvreg`
- The program takes care of identification issues, drops collinear regressors and provides summary statistics on the mobility structure among panel units
- A further restriction one should be aware of is the computing time to solve the system of normal equations
- Algorithms for solving/inverting sparse systems can tackle this restriction, but will only be fully satisfactory if they also cope with regressors collinear to the fixed effects