Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

nwxtregress: Network regressions in Stata

UK Stata Conference 2023

Jan Ditzen[1], William Grieser[2], Morad Zekhnini[3]

[1]Free University of Bozen-Bolzano, Italy
jan.ditzen@unibz.it

[2]Texas Christian University, USA
w.grieser@tcu.edu

[3]Michigan State University, USA
zekhnini@msu.edu

September 8, 2022

Work in progress - latest version available at
https://github.com/JanDitzen/nwxtregress

Many applications of interactions are best represented using networks I

- Empirical analysis in social sciences (nearly) invariably relies on the assumption of cross-sectional independence.
- Most real-world applications involve interactions between units of observation.
  - E.g., companies buy and sell from one another, individuals share information with family and friends, etc.
- A key question remains: how do we analyze outcomes in a regression framework in the context of networks?
  - cross-sectional independence cannot be assumed!

Many applications of interactions are best represented using networks II

- Spatial econometrics provide an answer:
  - Models dependence across cross-sectional units
  - Initially used in regional science to model neighbouring regions
  - Empirical models and estimation techniques with a priori knowledge of relationship between units (LeSage and Pace, 2009; Kelejian and Piras, 2017)
- In Stata implemented in the `Sp` environment with some limitations.

A parsimonious model of interactions

- General panel model with N units:

$$y_{it} = \sum_{j \neq i} \rho_{ij} y_{jt} + X_{it}\beta + \epsilon_{it}$$

- Considering all interactions ($\approx N^2$) is impractical
- Ord (1975) proposed the Spatial Autoregressive (SAR) model:

$$y_{it} = \rho \sum_{j \neq i} w_{ij,t} y_{jt} + X_{it}\beta + \epsilon_{it}$$

- $w_{ij,t}$ represents a priori link between i and j
- In matrices with additional spatial lag of X (SDM):

$$y_t = \rho W_t y_t + X_t\beta + W_t X_t \theta + \epsilon_t$$

- Estimating the model "as is" poses various challenges (Manski, 1993; Angrist, 2014)

How to solve?

- Reduced form of SAR (omitting time indices):

$$y = (I - \rho W)^{-1}(X\beta + \epsilon)$$

- $(I - \rho W)^{-1}$ can be "difficult" to calculate and model is non linear in parameters.
- Given mathematical restrictions on ρ and W:

$$(I - \rho W)^{-1} = I + \rho W + \rho^2 W^2 + \dots \quad (1)$$

$$\Rightarrow y = \left(I + \rho W + \rho^2 W^2 + \dots\right)(X\beta + \epsilon) \quad (2)$$

- Interpretation can be split into
    - Own effect (I term)
    - Immediate peers' effect (W term)
    - Peers of peers effect (W^2 term)
    - Direct ($\partial y_i / \partial x_i$) and indirect effects ($\partial y_i / \partial x_j$) Details

A short primer on estimation

- Focusing on one cross-section (for notational convenience), the likelihood function of the model is:

$$\begin{aligned} f(Y,X;\rho,\beta,\sigma^2) &= |I_N - \rho W|(2\pi\sigma^2)^{-N/2}\exp(-\frac{e'e}{2\sigma^2}) \\ e &= (I-\rho W)Y - X\beta \end{aligned}$$

- If ρ is known (say ρ_0), then β (and σ^2) can be integrated out in a maximum likelihood estimation (MLE).
- The problem becomes an optimization w.r.t. ρ only.
- The estimation proceeds with an MCMC sampler using the above likelihood over a grid of different values for ρ.
- $|I-\rho W|$ is the determinant, usually calculated via LU decomposition and challenging to calculate for large matrices.

How to estimate the model then?

`nwxtregress`

- estimates SAR and SDM models with a mix of a MLE and MCMC sampling (LeSage and Pace, 2009)
- allows the estimation of spatial/network models with
  - unbalanced datasets
  - time varying spatial weights/network dependencies
  - several formats to define the spatial weights/network dependencies
- calculates direct, indirect and total effects.
- Speed improvements using Python

Types of Spatial Weight Matrices

Two challenges

1. Dimension
   - Spatial weight matrix W is $N \times N$, often sparse. Example
   - Requires unnecessary amount of memory
2. Time Varying W_t and unbalanced data
   - Most economic data comes as flows, i.e. origin and destination.
   - Flows can change over time.
   - Square spatial weight matrix "unusual" format.

- `nwxtregress` uses internally sparse spatial weight matrices.
- Allows for time varying spatial weight matrices, unbalanced data and is memory/speed efficient.

Speed

- For estimation $(I - \rho W)$ needs to be inverted. Usually calculated via LU decomposition.
- Two caveats:
  - No sparse matrix support in Stata/mata. If W is sparse, then matrices are converted back to square matrices.
  - Calculations challenging for large matrices.
- Solution: use Python's sparse matrix environment.
- Implemented in `nwxtregress` with option `python`.
- Up to 10x faster.

Example: BEA I/O Tabels I

Data

- We collect USE/MAKE table data from the BEA's website
- These data represent the goods that were used (USE) and made (MAKE) by each industry in the US
- To construct links between industries, we convert into flows between industries
- Loaded data as *Sp* matrix using `spmatrix fromdata W = sam* , replace`, but only for year 1998. W is $N \times N$.
- We also collect key variables about each industry: capital consumption, compensation, and net surplus.

Example: BEA I/O Tabels II

Data

- We are estimating using `nwxtregress` Syntax:
  - SAR:

$$\begin{aligned} cap\_cons =& \beta_0 + \rho W_1 cap\_cons \\ & + \beta_1 compensation + \beta_3 net\_surplus + \epsilon \end{aligned}$$

  - SDM:

$$\begin{aligned} cap\_cons =& \beta_0 + \rho W_1 cap\_cons + \gamma_1 W_2 compensation \\ & + \beta_1 compensation + \beta_3 net\_surplus + \epsilon \end{aligned}$$

SAR

Time constant spatial weights

```
. nwxtregress cap_cons compensation net_surplus , ///
> dvarlag(W) seed(1234)

Order Spatial Weights (22)
────┼─── 10 ───┼─── 20 ───┼─── 30 ───┼─── 40 ───┼─── 50   %
..................................................     50
..................................................    100
Initialise Grid (100)
────┼─── 10 ───┼─── 20 ───┼─── 30 ───┼─── 40 ───┼─── 50   %
..................................................     50
..................................................    100
Griddy Gibbs (2000)
────┼─── 10 ───┼─── 20 ───┼─── 30 ───┼─── 40 ───┼─── 50   %
..................................................     50
..................................................    100
                                                 Number of obs     =       1358
Panel Variable (i): ID                           Number of groups  =         62
Time Variable (t): Year                          Obs. of group:              22
                                                              min =          19
                                                              avg =          22
                                                              max =          22
                                                 R-squared         =       0.77
                                                 Adj. R-squared    =       0.77
```

cap_cons	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
compensation	-.7195389	.0123628	-58.20	0.000	-.767359	-.6791014
net_surplus	-.7595604	.014257	-53.28	0.000	-.8046761	-.708425
_cons	.7214415	.0117927	61.18	0.000	.6828125	.7663866
W						
cap_cons	.120138	.0247156	4.86	0.000	.038	.2
/sigma_u	.0029666	.0001126			.0026178	.0033713

Example

Time varying spatial weight

- Network data in *timesparse* format as mata matrix `W`.[1]
- Especially for large datasets gains in speed and memory are considerable. Example
- The first column identifies the year, second and third the IDs and the last one the value of the weight.
- Non standardized timesparse W:

```
. mata W[1..4,.]
```

	1	2	3	4
1	1997	1	1	120.445105
2	1997	1	2	2646.806067
3	1997	1	3	0
4	1997	1	4	1594.653373

[1]Often called Coordinate (COO) list format.

SDM

```
. nwxtregress cap_cons compensation net_surplus , python ///
> dvarlag(W,mata timesparse) ///
> ivarlag(W: compensation,mata timesparse )  seed(1234)

Order Spatial Weights (22)
----+--- 10 ---+--- 20 ---+--- 30 ---+--- 40 ---+--- 50   %
..................................................    50
..................................................   100
Initialise Grid using Python (22)
----+--- 10 ---+--- 20 ---+--- 30 ---+--- 40 ---+--- 50   %
..................................................    50
..................................................   100
Griddy Gibbs (2000)
----+--- 10 ---+--- 20 ---+--- 30 ---+--- 40 ---+--- 50   %
..................................................    50
..................................................   100
                                                 Number of obs     =      1358
Panel Variable (i): ID                           Number of groups  =        62
Time Variable (t): Year                          Obs. of group:             22
                                                              min =        19
                                                              avg =        22
                                                              max =        22
                                                 R-squared         =      0.77
                                                 Adj. R-squared    =      0.77

------------------------------------------------------------------------------
    cap_cons |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
compensation |  -.7015208   .0123799  -56.67   0.000    -.7482582   -.6556848
 net_surplus |  -.7472091   .0141113  -52.95   0.000    -.7920688   -.6965197
       _cons |   .7730435   .0143202   53.98   0.000     .7288873    .8187064
-------------+----------------------------------------------------------------
W            |
    cap_cons |    .031389   .0283132    1.11   0.268        -.054        .119
compensation |  -.0895239   .0181129   -4.94   0.000    -.1565037   -.0227881
-------------+----------------------------------------------------------------
    /sigma_u |   .0029277   .0001141                      .0025583    .0033638
------------------------------------------------------------------------------
```

SDM

Direct Indirect Effects

```
. estat impact
────┼─── 10 ───┼─── 20 ───┼─── 30 ───┼─── 40 ───┼─── 50   %
..................................................   50
..................................................  100
Average Impacts                                 Number of obs =      1358
```

cap_cons	dy/dx	Std. Err.	z	P>|z|	[95% Conf.	Interval]
direct						
compensation	-.7002594	.0124331	-56.32	0.000	-.7490398	-.6526457
net_surplus	-.7472824	.0141128	-52.95	0.000	-.7922827	-.6965584
indirect						
compensation	.7741688	.014668	52.78	0.000	.7288423	.8191533
net_surplus	-.0248066	.0225161	-1.10	0.271	-.1018671	.0382383
total						
compensation	.0739094	.0151583	4.88	0.000	.0152978	.1373628
net_surplus	-.772089	.027054	-28.54	0.000	-.8611195	-.6896129

Conclusion

- `nwxtregress` extends `spxtregress`:
  - Allows for unbalanced datasets and time varying spatial weight matrices
  - Spatial weights can be directly loaded from datasets, frames, mata matrices or `spmatrix` objects.
- Available on GitHub (https://janditzen.github.io/nwxtregress/) or directly in Stata:

  net install nwxtregress ,
  from(https://janditzen.github.io/nwxtregress/)

- Please, help us by providing feedback

References I

Kelejian, H., and G. Piras. 2017. Spatial Econometrics. Academic Press.

LeSage, J. P., and R. K. Pace. 2009. Introduction to Spatial Econometrics. Florida CRC Press.

nwxtregress[2]

Syntax

Spatial Autocorrelation Model (SAR)

`nwxtregress` *depvar indepvars* [*if*] , *dvarlag(W1[,*`options1`*)])*
[`mcmc_options options2`]

Spatial Durbin Model (SDM)

`nwxtregress` *depvar indepvars* [*if*] , *dvarlag(W1[,*`options1`*)])*
ivarlag(W2[,`options1`*)* [`mcmc_options options2`]

- `W1` and `W2` define spatial weight matrices, default is Sp object.

[2] This command is work in progress. Options, functions and results might change.

nwxtregress

Spatial Weight Options

`nwxtregress` *depvar indepvars* [*if*] , *dvarlag(W1[,options1)])*
[`ivarlag(W2[,options1) mcmc_options options2`]

- `options1` controls the spatial weight matrices:
  - `mata` declares weight matrix is `mata` matrix. Details
  - `sparse` if weight matrix is sparse. Details
  - `timesparse` weight matrix is sparse and varying over time. Details
  - `frame(name)` use spatial weight from frame.
  - `id(string)` vector of IDs if W is a non sparse mata matrix.
  - `zero(real)` how to treat zeros in spatial weight matrix when using (time) sparse matrices.

nwxtregress

Further Options

`nwxtregress` *depvar indepvars* [*if*] , *dvarlag(W1[,options1)])*

[`ivarlag(W2[,options1) mcmc_options options2`]

- `options2` are:
  - `nosparse` do not convert weight matrix internally to a sparse matrix.
  - `noconstant` suppress constant.
  - `fe` add fixed effects.
  - `absorb()` absorb fixed effects using `reghdfe`
- `mcmc_options` control the Markov Chain Monte Carlo Details
  - `python` use Python to calculate $|I - \rho W|$
  - `usebp` use BarryPace trick instead of LUD for $|I - \rho W|$.

Weight Matrices back

Square

Square matrix format

- The spatial weights are a matrix with dimension $N_g \times N_g$. It is time constant. An Example for a 5×5 matrix is:

```
        1      2      3      4
    +---------------------+
  1 |   0     .1     .2     0   |
  2 |   0      0     .1    .2   |
  3 |  .3     .1      0     0   |
  4 |  .2      0     .2     0   |
    +---------------------+
```

Weight Matrices back

Sparse format

- The sparse matrix format is a $v \times 3$ matrix, where v is the number of non-zero elements in the spatial weight matrix.
- The weight matrix is time constant. The first column indicates the destination, the second the origin of the flow. A sparse matrix of the matrix from above is:

Destination	Origin	Flow
1	2	0.1
1	3	0.2
2	3	0.1
2	4	0.2
3	1	0.3
3	2	0.1
4	1	0.2
4	3	0.2

Weight Matrices back

Time-Sparse format

- The time sparse format can handle time varying spatial weights.
- The first column indicates the time period, the remaining are the same as for the sparse matrix. For example, if there are two time periods and we have the matrix from above for the first and the square for the second period:

Time	Destination	Origin	Flow
1	1	2	0.1
1	1	3	0.2
1	2	3	0.1
1	2	4	0.2
1	3	1	0.3
1	3	2	0.1
1	4	1	0.2
1	4	3	0.2
	(next time period)		
2	1	2	0.1
2	1	3	0.4
2	2	3	0.1
2	2	4	0.4
2	3	1	0.9
2	3	2	0.1
2	4	1	0.4
2	4	3	0.4

Example Sparse Matrix

back (COO) back (Intro)

- Contiguity matrix for 100 units.
- Total number of elements in W: 10,000 (100×100).
- Non-zero elements: 200, 9800 elements are zero.
- Square matrix will consume 80,000bytes (in mata).
- COO matrix with 3x200 elements will consume 2,400bytes.
- Calculation ρW on square matrix implies 10,000 mathematical operations, but only 200 lead to a non-zero result!
- Using COO matrix will limit the mathematical operations to 200!

mcmc_options back

Control the Markov Chain Monte Carlo:

- `python` use Python to calculate $|I - \rho W|$
- `draws(integer 2000)` number of griddy gibs draws.
- `gridlength(integer 1000)` grid length
- `nomit(integer 500)` number of omitted draws
- `barrypace(numlist)` settings for BarryPace Trick, iterations, maxorder default: 50 100
- `usebp` use BarryPace trick instead of LUD for $|I - \rho W|$.
- `seed(#)` sets the seed.

Partial derivatives are no longer βs

back

- In traditional model:

$$\frac{\partial y_i}{\partial x_i} = \beta, \text{ and } \frac{\partial y_i}{\partial x_j} = 0, \; i \neq j$$

- In the model with interactions:

$$\frac{\partial y_i}{\partial x_j} = (I - \rho W)^{-1}_{ij} \beta, \; \forall i, j$$

- Listing all partial derivatives is impractical.
- LeSage and Pace (2009) propose summarizing partial derivative estimates into direct and indirect effect averages:
  - Direct: $\frac{1}{N} \sum_i \frac{\partial y_i}{\partial x_i}$
  - Indirect: $\frac{1}{N} \sum_i \sum_{j \neq i} \frac{\partial y_i}{\partial x_j}$