Monte Carlo Analysis in Dynamic Panel Data Models

Giovanni S.F. Bruno
Università Bocconi, Milano, Italy
giovanni.bruno@uni-bocconi.it

Presentation for the 11th SUG UK 2005

Motivations and background

Estimators in the dynamic panel data literature (Arellano and Bond (1991); Kiviet (1995); Blundell and Bond (1998); Judson and Owen (1999); Bun and Kiviet (2003); Bruno (2005)) are typically evaluated carrying out Monte Carlo experiments based on the following simple model

$$y_{it} = \gamma y_{i,t-1} + \beta x_{it} + \eta_i + \varepsilon_{it};$$
$$x_{it} = \rho x_{i,t-1} + \xi_{it}$$

$$\varepsilon_{\mathrm{it}} \sim N\left(0, \sigma_\varepsilon^2\right)$$
$$\xi_{\mathrm{it}} \sim N\left(0, \sigma_\xi^2\right)$$

Traditionally, start-up values are obtained by determining a time length $T = T_0 + T_1$. Then, after fixing an *arbitrary* initial value for the process, the first T_0 time observations are used to estimate the start-up value, so that eventually the resulting time dimension in the simulated sample consists of only T_1 observations.

A more efficient procedure (McLeod and Hipel (1978), Kiviet (1995)) obtains start-up values according to the data generation process, so it avoids wasting random numbers in the estimation of start-up values and also small sample non-stationarity problems.

This presentation shows my progresses towards a Stata implementation of the McLeod and Hipel (1978) procedure

Let L denote the lag operator and decompose the process for y into the sum of AR(2) and AR(1) processes

$$y_{it} = \frac{\beta \xi_{it}}{(1-\gamma L)(1-\rho L)} + \frac{\eta_i + \varepsilon_{it}}{(1-\gamma L)} = \beta \varphi_{it} + \psi_{it} + \frac{\eta_i}{(1-\gamma)}$$

The start-up values are the following

$$x_{i,0} = \xi_{i,0} \left(1-\rho^2\right)^{-1/2}$$

$$\psi_{i,0} = \varepsilon_{i,0} \left(1-\gamma^2\right)^{-1/2}$$

$$\varphi_{i,0} = \xi_{i,0} \operatorname{var}(\varphi_{it})^{1/2}$$

$$\varphi_{i,1} = \varphi_{i,0} \, cor\left(\varphi_{it}, \varphi_{i,t-1}\right) +$$

$$+ \xi_{i,1} \operatorname{var}(\varphi_{it})^{1/2} \left\{1 - cor^2\left(\varphi_{it}, \varphi_{i,t-1}\right)\right\}^{1/2}$$

A nice feature…

In Stata we can generate data according to autoregressive models effortlessly, by simply exploiting the ability of **`replace`** to work sequentially (see the messages by N. J. Cox and D. Kantor to Statalist on May 25, 2004 in response to a question by D. V. Masterov).

A nice feature…

More specifically, for the AR(1) process we just need two lines of coding:

1) **`generate`** the **`x`** variable using its start-up expression

```
gen x=eps_x*(1-rho^2)^(-1/2)
```

2) starting from the second observation **`replace`** the values of **`x`** according to the AR(1) recursion

```
replace x=rho*L.x + eps_x if obs>1
```

For the AR(2) process three lines are required:

1) As in the AR(1) process **`generate`** the dependent variable using its *first* start-up expression.

2) Starting from the second observation, **`replace`** the values in the resulting dependent variable with the *second* start-up expression

3) Starting from the third observation **`replace`** the values in the resulting dependent variable according to the AR(2) recursion:

```
replace ph=(gamma+rho)*L.ph-
(gamma*rho)*L2.ph + eps_x if obs>2
```

A Stata code

My Stata code **`xtarsim`** simulates the model following the McLeod and Hipel procedure. The basic syntax is as follows

`xtarsim` ***`newdepvar newindepvar newindeffect,`*** **`nid(`***`#`***`)`** **`time(`***`#`***`)`** **`gamma(`***`real`***`)`** **`beta(`***`real`***`)`** **`rho(`***`real`***`)`** **`snratio(`***`real`***`)`** **`[sigma(`***`real`***`)`** **`seed(`***`#`***`)]`**

where **`nid(`***`#`***`)`** specifies the number of individuals, **`time(`***`#`***`)`** the number of time periods, **`gamma(`***`real`***`)`**, **`beta(`***`real`***`)`**, **`rho(`***`real`***`)`** the parameter values and **`snratio(`***`real`***`)`** the signal to noise ratio.

`xtarsim` can be for use to generate a pseudo-random sample from a compatible panel data model inside a wrapper program computing the statistics of interest (estimators, estimation errors, tests, etc.)

Then, a Monte Carlo experiment is carried out by computing the statistics of interest for a possibly large number of repetitions, each giving rise to a different pseudo-random sample through the **`xtarsim`** call in the wrapper program.

Finally, depending on the aim of the analysis, Monte Carlo means and standard deviations can be calculated for the collected results set of simulated statistics (see **`[R] simulate`**). For example, to estimate the finite-sample bias of a given estimator, one can collect estimation errors from Monte Carlo simulations and then take the sample mean of the estimation errors. For more, **`help xtarsim`**