Monte Carlo Simulations using Stata

Lee C. Adkins

Oklahoma State University
Stillwater, OK 74078

lee.adkins@okstate.edu
www.learneconometrics.com

25 March 2012

General Motivation

- Monte Carlo simulations are a great way to learn about the sampling properties of estimators

General Motivation

- Monte Carlo simulations are a great way to learn about the sampling properties of estimators
- Using them in class presents some challenges, though. To do it, students have to generate data, program, and analyze results.

General Motivation

- Monte Carlo simulations are a great way to learn about the sampling properties of estimators
- Using them in class presents some challenges, though. To do it, students have to generate data, program, and analyze results.
- This paper guides a student through these steps. The examples can be used as templates.

General Motivation

- Monte Carlo simulations are a great way to learn about the sampling properties of estimators
- Using them in class presents some challenges, though. To do it, students have to generate data, program, and analyze results.
- This paper guides a student through these steps. The examples can be used as templates.
- MC is also very useful for prototyping and developing DGP for more sophisticated simulations. It can serve as a quick reference for researchers as well.

Desirable Characteristics of Teaching Software

- For teaching purposes, I consider two pieces of software: Stata 12 and gretl

Desirable Characteristics of Teaching Software

- For teaching purposes, I consider two pieces of software: Stata 12 and gretl
- Software should be self-contained. Add-ons can make it hard to use in labs.

Desirable Characteristics of Teaching Software

- For teaching purposes, I consider two pieces of software: Stata 12 and gretl
- Software should be self-contained. Add-ons can make it hard to use in labs.
- Language should be simple to use and the syntax straightforward. Many students have never programmed.

Desirable Characteristics of Teaching Software

- For teaching purposes, I consider two pieces of software: Stata 12 and gretl
- Software should be self-contained. Add-ons can make it hard to use in labs.
- Language should be simple to use and the syntax straightforward. Many students have never programmed.
- Results should be relatively easy to get and to interpret.

Desirable Characteristics of Teaching Software

- For teaching purposes, I consider two pieces of software: Stata 12 and gretl
- Software should be self-contained. Add-ons can make it hard to use in labs.
- Language should be simple to use and the syntax straightforward. Many students have never programmed.
- Results should be relatively easy to get and to interpret.
- Output streams should be easy to analyze further.

Desirable Characteristics of Teaching Software

- For teaching purposes, I consider two pieces of software: Stata 12 and gretl
- Software should be self-contained. Add-ons can make it hard to use in labs.
- Language should be simple to use and the syntax straightforward. Many students have never programmed.
- Results should be relatively easy to get and to interpret.
- Output streams should be easy to analyze further.
- Numerical accuracy and quality RNG are essential.

Gretl vs. Stata

Gretl advantages

Gretl and Stata both satisfy these. Gretl has several advantages, though.

- Easier to program – DGP syntax is particularly straightforward. Gretl's Gauss-like scripting language (HANSL) is much easier to use than Stata's MATA. Gretl's `--store` option is easier to use than Stata's `postfile`.

Gretl vs. Stata

Gretl advantages

Gretl and Stata both satisfy these. Gretl has several advantages, though.

- Easier to program – DGP syntax is particularly straightforward. Gretl's Gauss-like scripting language (HANSL) is much easier to use than Stata's MATA. Gretl's `--store` option is easier to use than Stata's `postfile`.
- More portable – Windows, Mac, Linux (all freely available). Gretl can be used from a thumb-drive.

Gretl vs. Stata

Gretl advantages

Gretl and Stata both satisfy these. Gretl has several advantages, though.

- Easier to program – DGP syntax is particularly straightforward. Gretl's Gauss-like scripting language (HANSL) is much easier to use than Stata's MATA. Gretl's `--store` option is easier to use than Stata's `postfile`.
- More portable – Windows, Mac, Linux (all freely available). Gretl can be used from a thumb-drive.
- Cheaper – No charge! Can't beat the price.

Gretl vs. Stata

Gretl advantages

Gretl and Stata both satisfy these. Gretl has several advantages, though.

- Easier to program – DGP syntax is particularly straightforward. Gretl's Gauss-like scripting language (HANSL) is much easier to use than Stata's MATA. Gretl's `--store` option is easier to use than Stata's `postfile`.
- More portable – Windows, Mac, Linux (all freely available). Gretl can be used from a thumb-drive.
- Cheaper – No charge! Can't beat the price.
- More Accurate (in my opinion) and faster.

Gretl vs. Stata

Stata advantages

- Stata has a large library of canned procedures that can easily be studied using the automatic **simulate** command. `simulate` executes code a given number of times and prints summary statistics when done. For non-programmers, or prototyping this is a plus.
- As demonstrated in this paper, it is also easy to loop Stata over several design parameters, though it requires (I believe) the more complicated `postfile` commands and the programming of loops.
- Stata's documentation is complete and clear; Stata ships with a complete pdf version of all manuals.
- Stata is well-supported by a large user base.

The Plan

- I'll review the two methods used for simulation in Stata.
- I'll demonstrate with some examples from the paper.
- I'll compare speed of `simulate`, `postfile`, and gretl.

Two Ways to Experiment

There are basically two ways to run simulations in Stata:

- `postfile` commands
- `simulate` command

I'll demonstrate each with some examples from the paper.

postfile

- Create a place to store temporary results (`tempname`)
- Initiate the postfile. Tell it what to name results and where to put them
- Make things quiet! `quietly {`
- create loops: `foreach`, `forvalues`, or `while`
- `post` desired results to the file identified by `tempname`
- close `loops`, `quietly`, and the `postfile`

postfile Example

```
set obs 100
gen b = .

tempname sim
postfile `sim' mean using results, replace
  quietly {
     forvalues i = 1/1000 {
     replace b = rnormal(0,1)
     summarize
     scalar mean = r(mean)
     post `sim' (mean)
  }
}
postclose `sim'
```

simulate

- Create a `program` (rclass) to compute and return the desired computations
- use `simulate`, specifying number of replications, where to save results, and the name of the program to simulate

simulate

```
program means, rclass
     replace b = rnormal(0,1)
     qui summarize
     return scalar mean = r(mean)
end

clear
set obs 100
gen b = .

simulate b=r(mean), reps(1000) ///
         saving(results, replace): means
```

Using the Results

```
use results, clear          /* Open the dataset */
summarize mean              /* Summary Stats    */

     /* Density plot with normal overlay */

kdensity mean, normal normopts(lwidth(medium) ///
    lpattern(dash))
```

Paper Examples

- Classical Normal linear regression – Coverage rates of confidence intervals
- Antithetic variates
- Lagged dependent variable model with autocorrelation
- Performance of HAC standard errors
- Heteroskedastic model – variance a function of regressors
- Instrumental variables
- Binary choice
- Censored regression
- Nonlinear least squares
- Looping over several designs

Example: Autocorrelated LDV Model

Consider the model

$$y_t = \beta x_t + \delta y_{t-1} + u_t \qquad t = 1, 2, \ldots, N \tag{1}$$

$$u_t = \rho u_{t-1} + e_t \qquad x_t = \theta x_{t-1} + v_t \tag{2}$$

where $|\rho| < 1$ and $|\theta| < 1$ are parameters, $e_t \sim N(0, \sigma_e^2)$ and $v_t \sim N(0, 1)$. Various values of ρ, θ, and δ present possibilities.

- If $\delta = 0$ is the usual AR(1) model.

Example: Autocorrelated LDV Model

Consider the model

$$y_t = \beta x_t + \delta y_{t-1} + u_t \qquad t = 1, 2, \ldots, N \tag{1}$$

$$u_t = \rho u_{t-1} + e_t \qquad x_t = \theta x_{t-1} + v_t \tag{2}$$

where $|\rho| < 1$ and $|\theta| < 1$ are parameters, $e_t \sim N(0, \sigma_e^2)$ and $v_t \sim N(0, 1)$. Various values of ρ, θ, and δ present possibilities.

- If $\delta = 0$ is the usual AR(1) model.
- If $\rho = 0$ is a lagged dependent variable model.

Example: Autocorrelated LDV Model

Consider the model

$$y_t = \beta x_t + \delta y_{t-1} + u_t \qquad t = 1, 2, \ldots, N \tag{1}$$

$$u_t = \rho u_{t-1} + e_t \qquad x_t = \theta x_{t-1} + v_t \tag{2}$$

where $|\rho| < 1$ and $|\theta| < 1$ are parameters, $e_t \sim N(0, \sigma_e^2)$ and $v_t \sim N(0, 1)$. Various values of ρ, θ, and δ present possibilities.

- If $\delta = 0$ is the usual AR(1) model.
- If $\rho = 0$ is a lagged dependent variable model.
- If both $\delta = 0$ and $\rho = 0$ the model reduces to equation CNLRM.

Example: Autocorrelated LDV Model

Consider the model

$$y_t = \beta x_t + \delta y_{t-1} + u_t \qquad t = 1, 2, \ldots, N \tag{1}$$

$$u_t = \rho u_{t-1} + e_t \qquad x_t = \theta x_{t-1} + v_t \tag{2}$$

where $|\rho| < 1$ and $|\theta| < 1$ are parameters, $e_t \sim N(0, \sigma_e^2)$ and $v_t \sim N(0, 1)$. Various values of ρ, θ, and δ present possibilities.

- If $\delta = 0$ is the usual AR(1) model.
- If $\rho = 0$ is a lagged dependent variable model.
- If both $\delta = 0$ and $\rho = 0$ the model reduces to equation CNLRM.
- $\delta = 1$ implies change $(y_t - y_{t-1})$ and $|\delta| < 1$ implies partial adjustment.

ARDL(2,1) representation

If both $\delta \neq 0$ and $\rho \neq 0$ is an ARDL(2,1). The others are nested within this one.

$$y_t = \beta x_t + (\delta + \rho) y_{t-1} - (\rho\beta) x_{t-1} - (\rho\delta) y_{t-2} + e_t \tag{3}$$

- Recall that $x_t = \theta x_{t-1} + v_t$. So, when $\theta = 0$ it is possible to estimate β consistently using the simple regression $y_t = \beta x_t + \varepsilon_t$.
- ε_t includes y_{t-1}, y_{t-2}, and x_{t-1}, but $E[\varepsilon_t | x_t] = 0$.

DGP code

```
replace x = theta*L.x + rnormal() in 2/$nobs
replace u = rho*L.u + rnormal(0,sigma) in 2/$nobs
replace y = beta*x+delta*L.y + u in 2/$nobs
```

Estimation code

```
reg y x                /* b1 */
reg L(0/1).y x         /* b2 */
prais L(0/1).y x       /* b3 */
reg L(0/2).y L(0/1).x /* b4 */
```

Output LDV

Variable	Obs	Mean	Std. Dev.
b1	1000	15.01713	9.167224
b2	1000	5.307263	1.72269
b3	1000	8.235005	1.635338
b4	1000	9.96528	1.445586

Weak Instruments–Loop over different designs

```
program regIV, rclass
tempname sim
   postfile ‘sim’ gam r2 b biv using results, replace
   quietly {
     foreach gam of numlist 0.025 0.0375 0.05 0.1 0.15 {
       forvalues i = 1/$nmc {
       replace u = rnormal()
       replace x = ‘gam’*z+rho*u+rnormal(0,sige)
       replace y = slope*x + u
       ....
       }
     }
  }
  postclose ‘sim’
end
```

Rule-of-Thumb: 1 endogenous variable

Stock et al. propose the rule-of-thumb: instruments are weak if $F < 10$. It is based on:

$$E[\hat{\beta}_{IV}] - \beta \approx [plim(\hat{\beta}_{OLS}) - \beta]/(E(F) - 1) \tag{4}$$

where we take F to be the average for a given design, $\bar{F}$. Hence,

$$\frac{E[\hat{\beta}_{IV}] - \beta}{[plim(\hat{\beta}_{OLS}) - \beta]} \approx \frac{1}{(E(F) - 1)} \tag{5}$$

Weak Instruments–egen to get group means

```
regIV
use results, clear
by gam, sort: summarize r2 F biv b
by gam, sort: summarize p_ls p_iv

by gam: egen Fbar = mean(F)       /* Avg F by gamma   */
by gam: egen tslsbar = mean(biv) /* Avg biv by gamma */
by gam: egen olsbar = mean(b)    /* Avg b by gamma   */
by gam: egen r2bar = mean(r2)    /* Avg r2 by gamma  */
```

Weak Instruments–bias and a regression

```
gen tsls_bias = tslsbar-1         /* IV bias           */
gen ols_bias = olsbar-1           /* OLS bias          */
gen relb = tsls_bias/ols_bias     /* relative bias     */
gen rot = 1/(Fbar-1)              /* rule of thumb     */

gen t = _n                     /* observation numbers */
keep if mod(t,1000) == 0       /* keep 1 obs per design */
reg relb c.rot##c.rot, noconst /* rel. bias onto rot */
reg relb rot, noconst
test (rot=-1)                 /* directly proportional? */
```

Weak Instruments–bias and a regression

```
                                  F
    Variable |    2.03     3.64    5.19      18.21
    ----------------------------------------------
rule-of-thumb|  .97362   .37787  .23832    .05809
relative bias| -.76564  -.67957  .05188   -.07878

           F |   39.52    70.02   432.9
    ------------------------------------
rule-of-thumb|  .02595   .01449  .00232
relative bias| -.03569  -.01917 -.00224
```

Weak Instruments–bias and a regression

```
. reg relb rot, noconst
                              Number of obs =       7
                              R-squared     =  0.8131
-----------------------------------------------------
relative bias:   Coef.    Std. Err.      t    P>|t|
----------+------------------------------------------
rule-o-tmb|   -.864443   .1692116    -5.11   0.002
-----------------------------------------------------
```

Speed Kills

Elapsed time, in seconds

Program	gretl	Stata (`postfile`)	Stata (`simulate`)
Confidence Interval	0.578	2.019	3.059
Nonlinear Least Squares	2.558	47.77	-

No contest!

gretl code example

```
# Set the sample size and save it in n
   nulldata 100
   scalar n = $nobs
   set seed 3213799
# Set the values of the parameters
   scalar slope = 10
   scalar sigma = 20
   scalar delta = .7
   scalar rho = .9
# initialize variables
   series u = normal()
   series y = normal()
   series x = uniform()
```

gretl code example

```
loop 400 --progressive --quiet
   series e = normal(0,sigma)
   series u=rho*u(-1)+e
   series y = slope*x + delta*y(-1) + u
   ols y const x y(-1)
   ols y const x
   ar 1; y const x y(-1)
   ols y const x y(-1) x(-1) y(-2)
endloop
```