Unversitat Pompeu Fabra
Short Guides in Microeconometrics

Kurt Schmidheiny
October 2008

Monte Carlo Experiments

1 Introduction

Doing econometrics means estimating parameters, such as the mean of a population, the coefficients in a linear regression or the autocorrelation of a time series, given a sample of real world data. Besides the point estimate itself, we would like to know how close our estimate is to the true value. In other words we would like to know its "accuracy" or "precision". An estimator is a (maybe complicated) function of random variables and therefore itself a random variable. The properties of an estimator are fully described by its probability distribution (the so-called sampling distribution). The sampling-distribution can then be used to perform tests against hypothesis. Often we are especially interested in some moments of the sampling distribution, such as the mean and the variance.

In some cases it is possible to calculate the sampling distribution from the econometric model. But sometimes, especially for finite (small) samples, this is either not possible or very difficult. In these cases Monte Carlo experiments are an intuitive way to obtain information about the sampling distribution and hence about the "quality" of the estimator.

2 The Method

The term "Monte Carlo" refers to procedures in which quantities of interest are approximated by generating many random realisations of a stochastic process and averaging them in some way. In statistics, the

quantities of interest are the distributions of estimators and test statistics, the size of a test statistic under the null hypothesis, or the power of a test statistic under some specified alternative hypothesis (see Davidson and MacKinnon 1993, 731). In economic theory Monte Carlo techniques are used to explore the quantitative properties of models with stochastic elements, for example the correlation between variables in real business cycle models.

How can we use Monte Carlo techniques to find the sampling distribution of an estimator? In the real world, we usually observe just one sample of a certain size N, that will give us just one estimate. The Monte Carlo experiment is a lab situation, where we replicate the real world study many (R) times. Every time, we draw a different sample of size N from the original population. Thus, we can calculate the estimate many times and any estimate will be a bit different. The empirical distribution of these many estimates approximates the true of the estimator.

A Monte Carlo experiment involves the following steps:

(1) Draw a (pseudo) random sample of size N for the stochastic elements of the stochastic model from their respective probability distribution functions

(2) Assume values for the exogenous parts of the model or draw them from their respective distribution function

(3) Calculate the endogenous parts of the statistical model

(4) Calculate the value (e.g. the estimate) you are interested in

(5) Replicate step 1 to 4 R times

(6) Examine the empirical distribution of the R values

Let's explain the above elements in an example: the bivariate ordinary least squares model

$$y_t = \beta_0 + \beta_1 x_t + \varepsilon_t$$

with $\varepsilon_t \sim N(0, \sigma^2)$. The stochastic element in the model is ε_t, the exogenous part is x_t is either fixed are also stochastic. Assuming values for the true parameters a and b and drawing values for the stochastic element, we can simulate the endogenous variable y_t. The values of interest are then the least squares estimates $\hat{\alpha}$ and $\hat{\beta}$ in the simulated data set.

In the core of Monte Carlo Experiments is the random number generator. A random number generator produces a sequence of numbers, that are draws from a specific identically and independently distributed random variable. In practice, this is a mathematical algorithm, that produces a sequence of so-called pseudo random numbers. These numbers are in fact not random as the algorithm describes the purely deterministic relationship between the numbers. However, with a good generator, they are indistinguishable from sequences of genuinely random numbers and pass usual statistical tests of independence. Judd (1998) provides a thorough treatment of different pseudo-random number generators.

There is an important limitation of Monte Carlo experiments: We must completely specify the Statistical Model (Data Generating Process DGP). This implies, that we must assume the deterministic parts of the model, the form and the exact parameters of the distribution of the stochastic (error) term and the distribution of exogenous variables. This is a great loss of generality as the results of the experiment are apply only to the assumptions made.

3 Implementation in Stata 10.0

Stata has a built-in random number generator:

```
uniform()
```

returns uniformly distributed pseudo-random numbers on the interval $[0, 1)$. Random numbers for other continuous distributions are calculated using the inverse of the desired distribution, for example

```
generate z = invnorm(uniform())*2+5
```

generates a new variable z with `_N` (the number of observations in the current dataset) independent draws from a normal distribution with variance 2^2 and mean 5. See `help drawnorm` on how to draw a random vector from the *multivariate* normal distribution. You can reset the random number generator with `set seed 0`.

The different steps of a Monte Carlo experiment in Stata are explained by an investigation into the properties of the OLS estimator in a bivariate regression model.

The first task in setting up the Monte Carlo experiment in Stata is to define a program that produces the result of a single experiment, i.e. that performs the steps (1) to (4).

```
program olssim, rclass
   version 10.0
   drop _all
   set obs 100
   generate e = invnorm(uniform())*2
   generate x = uniform()*10
   generate y = 1 + 0.5 * x + e
   regress y x
   return scalar b0 = _coef[_cons]
   return scalar b1 = _coef[x]
end
```

The above program clears the data in memory and sets the number of observations in each sample to $N = 100$. Step (1): the realized error terms e are drawn form the centered normal distribution with variance 2^2. Step (2): the independent variable x is drawn from a uniform distribution on [0,10). Step (3): the realizations of the dependent variable y are calculated according to the DGP as $y = \beta_0 + \beta_1 x + \varepsilon$, with true parameter values $\beta_0 = 1$ and $\beta_1 = 0.5$. Step (4): the estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ are estimated in the regression of y on x. The last two lines of the program specify the values that are investigated in this Monte Carlo experiment: $\hat{\beta}_0$ and $\hat{\beta}_1$ which will be returned under the names `aConst` and `aSlope`, respectively. The definition of the program can be directly typed into the command window or is part of a do-file.

Step (5) is the replication of the single experiment R times. There is a special Stata command `simulate` that performs this replication and produces a new dataset with the results.

```
simulate "olssim" b0 = r(b0) b1 = r(b1), reps(1000)
```

performs the single experiment $R = 1000$ times and produces a new dataset with 1000 observations of the two variables $b0$ and $b1$. Each row contains the estimated parameters of a single experiment.

In step (6), we examine the results of the Monte Carlo experiment. This is done by inspecting the new variables $b0$ and $b1$ using the usual command for descriptive statistics, such as `summarize` and `histogram`.

We will save the program together with the analysis of the results in a do-file.[1] This do-file should begin with the command

```
capture program drop olssim
```

in order to clear the program from the memory before it is re-defined.

[1]More sophisticated implementations of Monte-Carlo experiment would declare the program in a separate so-called ado-file. The program will generally take several arguments that describe the details of the experiments, such as the true parameter values.

4 Implementation in Matlab

Matlab has a good built-in random number generator. We will use the functions rand and randn, which are included in the basic Matlab installation. The rand function generates random numbers whose elements are uniformly distributed in the interval (0,1). `Y = rand(m,n)` returns an m-by-n matrix of random entries. rand, by itself, returns a scalar whose value changes each time it's referenced. We can reset the random number generator with `rand('state',0)`. The `randn` function generates random numbers whose elements are normally distributed with mean 0 and variance 1. `Y = randn(m,n)` returns an m-by-n matrix of random entries. `randn`, by itself, returns a scalar whose value changes each time it's referenced. Again, `randn('state',0)`. resets the generator. Matlab's statistics toolbox contains random number generators for a variety of distribution functions.

The Monte Carlo experiment is then performed by a simple `for` loop over the $r = 1...R$ replications. Each replication within the loops first creates the data vectors with the N random draws, calculates the statistic(s) of interest and stores it(them) in the rth row of a new $R \times 1$ vector(s).

References

Davidson, Russell and James G. MacKinnon (1993), Estimation and Inference in Econometrics, Oxford University Press, chapter 21.

Greene, William H. (2003), Econometric Analysis, Prentice Hall, appendix E.2.1 - E.2.5 and E.3.

Kennedy, Peter (2003), A Guide to Econometrics, 5th ed., Blackwell Publishing, section 2.10.

Judd, Kenneth L. (1998), Numerical Methods in Economics, MIT Press, chapter 8.