Day 3A
Simulation Basics and Monte Carlo Experiments

A. Colin Cameron
Univ. of Calif. - Davis
... for
Center of Labor Economics
Norwegian School of Economics
Advanced Microeconometrics

Aug 28 - Sept 1, 2017

1. Introduction

- Begin with
  - How to make draws of random variables.
  - How to compute integrals using random draws.
- Then Monte Carlo simulation from a known model
  - can be used to check validity of estimation and testing methods
  - and also can learn a lot
  - exceptionally useful and under-utilized.
- For estimation
  - Markov chain Monte Carlo simulation is basis for modern Bayesian methods
  - Monte Carlo integration is used for maximum simulated likelihood
    - ★ For low dimension integrals may instead use Gaussian quadrature.

Outline

2. Simulation: Pseudo random uniform draws

- Pseudo random uniform numbers draws
  - building block for draws from other distributions
  - want $X \sim Uniform(0, 1)$
  - use a deterministic rule that mimics independent random draws
  - get $X_j, j = 1, 2, ...$ where $X_j \sim Uniform(0, 1)$ and X_j independent of X_k, $k \neq j$.
- IMPORTANT: always set the seed
  - this is the initial value X_0 that starts the sequence
  - then can reproduce the same sequence later
  - otherwise the computer clock is used to form the seed.
- IMPORTANT: Stata 14 uses different random numbers.

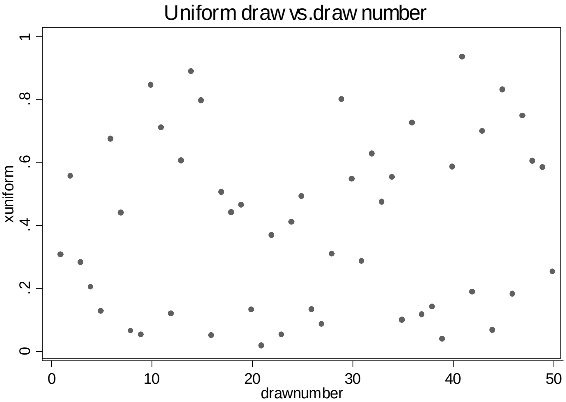
Pseudo random uniform draws

- Simple rule is of form

$$X_j = (kX_{j-1} + c) \bmod m$$

  - where a mod b = remainder when a is divided by b
  - for good choices of k, c and m
  - e.g. $X_j = (69069X_{j-1} + 1234567) \bmod 2^{32}$
- Stata 13 and earlier function `runiform()` uses the KISS generator
  - this combines four such generators
  - it is 32 bit so at most $2^{32} \simeq 4$ billion unique random numbers.
- Stata 14 on uses the Mersenne twister 14 bit generator
  - In Stata 14 to instead use KISS: `version 13` or `set rng kiss32`

- Example: Draw 50 uniforms and plot the sequence of 50 draws
  - set obs 50
  - set seed 10101
  - generate xuniform = runiform()
  - generate drawnumber = _n
  - scatter xuniform drawnumber



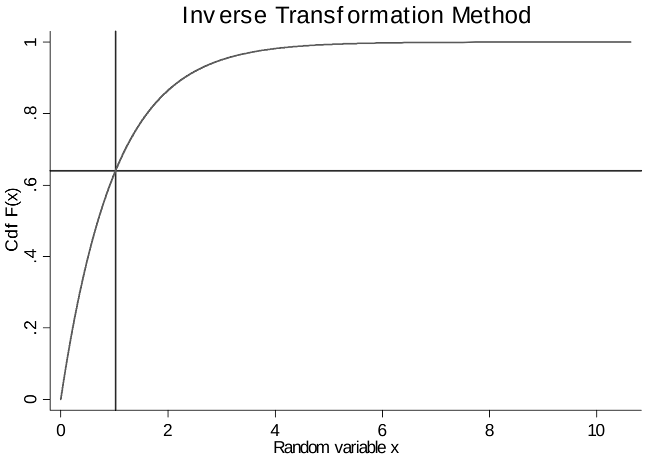
- Draws have flat histogram and kernel density plot
  - mean $\simeq 0.5$; standard deviation$\simeq \sqrt{1/12} = 0.288675$; and uncorrelated.

```
. summarize x

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
           x |     10,000    .4997462     .288546   .0000276   .9999758

. display "Theoretical mean = 0.5 and standard deviaion = " 1/sqrt(12)
Theoretical mean = 0.5 and standard deviaion = .28867513

. histogram x, start(0) width(0.1)
(bin=10, start=0, width=.1)

. 
. *** Autocorrelations for the uniform draws should be zero
. generate t = _n

. tsset t
        time variable:  t, 1 to 10000
                delta:  1 unit

. * line x t if t <= 100
. pwcorr x L.x L2.x L3.x, star(0.05)

             |        x      L.x     L2.x     L3.x
-------------+------------------------------------
           x |   1.0000 
         L.x |  -0.0006   1.0000 
        L2.x |   0.0054  -0.0005   1.0000 
        L3.x |  -0.0004   0.0054  -0.0005   1.0000 
```

Inverse transformation method

- Common method to draw nonuniform draws
  - but there can be computationally quicker methods.
- Choose x so that values of the c.d.f. $F(x) = \Pr[X \leq x]$ between 0 and 1 are equally likely
  - Then $F(x) = u \sim Uniform(0, 1)$
  - So $x = F^{-1}(u)$.
- Example: standard normal
  - draw $u = .975$
  - then $x = \Phi^{-1}(.975) = 1.955964$.
  - Stata: `generate x = invnorm(runiform())`
- For normals Box-Mueller method is instead preferred as quicker
  - Stata: `generate x = rnormal(0,1)`

- Example: Draw from unit exponential
  - $u = F(x) = 1 - \exp(-x)$
  - $x = F^{-1}(u) = -\ln(1 - u)$
  - e.g. $x = F^{-1}(0.64) = -\ln(1 - 0.64) = 1.0216.$



Other methods of pseudo random draws

- Transformation method
  - Transform r.v. to one whose distribution is easy to draw from
  - e.g. To draw $X \sim \chi^2(2)$ draw $X = Z_1^2 + Z_2^2$ where $Z_1, Z_2 \sim \mathcal{N}[0,1]$.
- Accept-reject method
  - Want X from density $f(x)$
  - Suppose $f(x) \leq kg(x)$ for all x for some $k > 1$
  - Then draw from $g(x)$ and accept draw if $r \leq \frac{f(x)}{kg(x)}$ where r is $Uniform(0,1)$ draw.
- Composition
  - e.g. Negative binomial is Poisson-gamma mixture
  - So draw gamma and then Poisson given this gamma
- Stata 10.1 added new suite of generators beginning with `r`
- Stata 14 added new 64 bit random number generator as default.

Multivariate normal draws (Cholesky decomposition)

- Way to make draws from (correlated) multivariate normal that requires only independent draws from independent univariate normals.
- Given $\mathbf{Z} \sim \mathcal{N}[\mathbf{0}, \mathbf{I}_k]$
  - then $\mathbf{X} = \boldsymbol{\mu} + \mathbf{LZ} \sim \mathcal{N}[\boldsymbol{\mu}, \mathbf{LL}']$
- So to draw $\mathbf{X} = \mathcal{N}[\boldsymbol{\mu}, \Sigma]$ where $\Sigma = \mathbf{LL}'$
  - use $\mathbf{X} = \boldsymbol{\mu} + \mathbf{LZ}$ where $\mathbf{Z} \sim \mathcal{N}[\mathbf{0}, \mathbf{I}_k]$ are draws from i.i.d. standard normal.
- More than one $\mathbf{L}$ satisfies $\mathbf{LL}'$
  - Cholesky decomposition sets $\mathbf{L}$ to be lower triangular

Multivariate normal draws (continued)

- Cholesky decomposition for $k = 3$

$$\mathbf{L} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix}$$

- Then given z_1, z_2, z_3 draws from $\mathcal{N}[0,1]$

$$\begin{aligned} x_1 &= \mu_1 + l_{11} z_1 \\ x_2 &= \mu_2 + l_{21} z_1 + l_{22} z_2 \\ x_3 &= \mu_3 + l_{31} z_1 + l_{32} z_2 + l_{33} z_3 \end{aligned}$$

- Stata command `drawnorm` does this.

Multivariate draws: Gibbs sampler

- Consider bivariate $\mathbf{Y} = (Y_1, Y_2)$ with
  - bivariate density $f(Y_1, Y_2)$ hard to draw from
  - known conditional densities $f(Y_1|Y_2)$ and $f(Y_1|Y_2)$ that are easy to draw from
  - then make alternating draws from $f(Y_1|Y_2)$ and $f(Y_2|Y_1)$.
- Even though

$$\begin{aligned} f(Y_1, Y_2) &= f(Y_1|Y_2) \times f(Y_2) \\ &\neq f(Y_1|Y_2) \times f(Y_2|Y_1) \end{aligned}$$

  - if we make many successive alternating draws we eventually get a draw from $f(Y_1, Y_2)$
  - an example of a Markov chain
- **Markov chain Monte Carlo (MCMC) methods are the basis for modern Bayesian analysis.**

Gibbs sampler example

- Suppose (Y_1, Y_2) multivariate normal means $(0, 0)$, variances $(1, 1)$ and correlation 0.9.
  - Then $Y_1|Y_2 \sim \mathcal{N}[0, 1-\rho^2]$ and $Y_2|Y_1 \sim \mathcal{N}[0, 1-\rho^2]$
    - ★ So given initial $Y_1^{(1)}$ draw $Y_2^{(1)}$ from $\mathcal{N}[Y_1^{(1)}, 0.19]$
    - ★ then $Y_1^{(2)}$ from $\mathcal{N}[Y_2^{(1)}, 0.19]$
    - ★ then $Y_2^{(2)}$ from $\mathcal{N}[Y_1^{(2)}, 0.19]$ ....
- The chain takes a while to converge
  - so "burn-in" where discard e.g. first 1,000 draws
- The draws are serially correlated
  - but they are draws from the joint density $f(Y_1, Y_2)$ as desired.
- Following Stata code
  - In Mata get the draws and then pass these back to Stata
  - In Stata analyze the draws including serial correlation of the draws.

```
. set seed 10101

. mata:
─────────────────────────────────────────────── mata (type end to exit) ───────
:   s0 = 10000               // Burn-in for the Gibbs sampler (to be discarded)

:   s1 = 1000                // Actual draws used from the Gibbs sampler

:   y1 = J(s0+s1,1,0)        // Initialize y1

:   y2 = J(s0+s1,1,0)        // Initialize y2

:   rho = 0.90               // Correlation parameter

:   for(i=2; i<=s0+s1; i++) {
>       y1[i,1] = ((1-rho^2)^0.5)*(rnormal(1, 1, 0, 1)) + rho*y2[i-1,1]
>       y2[i,1] = ((1-rho^2)^0.5)*(rnormal(1, 1, 0, 1)) + rho*y1[i,1]
>   }

:   y = y1,y2

:   y = y[|(s0+1),1 \ (s0+s1),.|]  // Drop the burn-ins

:   // Skip view in mata:  mean(y), variance(y), correlation(y)
:   stata("quietly set obs 1000")          // This requires s1 = 1000

:   st_addvar("float", ("y1", "y2"))
       1   2
    +---------+
  1 |  1   2  |
    +---------+

:   st_store(., ("y1", "y2"), y)

: end
```

```
. summarize

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
          y1 |      1,000    .0629699    .9346203  -2.516921   3.217661
          y2 |      1,000    .0577031    .9378138  -2.553584   2.931325

. correlate y1 y2
(obs=1,000)

             |       y1       y2
-------------+------------------
          y1 |   1.0000
          y2 |   0.8887   1.0000


. gen s = _n

. tsset s
        time variable:  s, 1 to 1000
                delta:  1 unit

. corrgram y1, lag(5)

                                          -1       0       1 -1       0       1
 LAG       AC       PAC      Q     Prob>Q  [Autocorrelation]  [Partial Autocor]
-------------------------------------------------------------------------------
1        0.7850   0.7850   618.09  0.0000
2        0.6050  -0.0308    985.6  0.0000
3        0.4699   0.0107   1207.5  0.0000
4        0.3585  -0.0188   1336.8  0.0000
5        0.3037   0.0791   1429.7  0.0000
```

3. Monte Carlo integration

- Suppose X is distributed with density $g(x)$ on (a, b)
- Then
$$\mathrm{E}[h(X)] = \int_a^b h(x)g(x)dx.$$
- If not tractable we could approximate by making draws $x^1, ..., x^s$ from $g(x)$, and average the corresponding values $h(x^1), ..., h(x^s)$, so
$$\widehat{\mathrm{E}}[h(X)] = \frac{1}{S}\sum_{s=1}^{S} h(x^s).$$
- Problems:
  - may require many draws
  - "works" even if $\mathrm{E}[h(X)]$ does not exist!
- **Monte Carlo integration is the basis for maximum simulated likelihood.**

Importance sampling

- Importance sampling re-expresses the integral as follows

$$\begin{aligned} \mathrm{E}[h(X)] = \int h(x)g(x)dx &= \int \left(\frac{h(x)g(x)}{p(x)}\right) p(x)dx \\ &= \int w(x)p(x)dx \end{aligned}$$

  - where density $p(x)$ is easy to draw from and has same support as original domain of integration
  - and weights $w\,(x) = h(x)g(x)/p(x)$ are easy to evaluate, bounded and finite variance.

- Then

$$\widehat{\mathrm{E}}[h(X)] = \frac{1}{S}\sum_{s=1}^{S} w(x^s),$$

  - x^s, $s = 1, ..., S$, are draws from $p(x)$ rather than $g(x)$
  - weights $w\,(x)$ determines weight or "importance" of draw
  - optimal is $w(x) = \mathrm{E}_g[h(x)]$ as this minimizes $\mathrm{Var}[\widehat{\mathrm{E}}[h(X)]$.
  - essentially best if $p(x)$ is chosen so that $w(x)$ is fairly flat.

4. Numerical Integrations

- Numerical method for computing integral

$$I = \int_a^b f(x)\, dx$$

- Mid-point rule calculates the Riemann sum at n midpoints

$$\widehat{I}_M = \sum_{j=1}^{n} \frac{b-a}{n} f(\bar{x}_j)$$

- Better variants are trapezoidal rule and Simpson's rule.
- But big problem if range of integration is unbounded
  - $a = -\infty$ or $b = \infty$!
  - so use Gaussian quadrature.
- **Gaussian quadrature is the basis for mixed model estimation in Stata.**

Gaussian quadrature (continued)

- Gaussian quadrature re-expresses the integral as

$$I = \int_a^b f(x)\,dx = \int_c^d w(x)r(x)dx,$$

  - where $w(x)$ is one of the following functions depending on range of x (unbounded from above and below; or unbounded on one side only; or bounded on both sides)
    - ★ $(a,b) = (-\infty,\infty)$: Gauss-Hermite: $w(x) = e^{-x^2}$ & $(c,d) = (-\infty,\infty)$.
    - ★ $[a,b) = [a,\infty)$: Gauss-Laguerre: $w(x) = e^{-x}$ and $(c,d) = (0,\infty)$.
    - ★ $[a,b] = [a,b]$: Gauss-Legendre: $w(x) = 1$ and $(c,d) = [-1,1]$.
  - In simplest case $r(x) = f(x)/w(x)$, but may need transformation of x.
- Gaussian quadrature approximates the integral by the weighted sum

$$\widehat{I}_G = \sum_{j=1}^m w_j r(x_j),$$

  - the researcher chooses m with often $m = 20$ enough
  - given m, the m points of evaluation x_j and associated weights w_j are given in e.g. computer code of Press et al. (1993).

5. Monte Carlo experiments: Properties of OLS

- D.g.p.: $y_i = \beta_1 + \beta_2 x_i + u_i$ where $x_i \sim \chi^2(1)$ and $\beta_1 = 1$, $\beta_2 = 2$.
  Error: $u_i \sim \chi^2(1) - 1$ is skewed with mean 0 and variance 2.

```
. * Small sample - estimates will differ from d.g.p. values
. clear

. set seed 101

. quietly set obs 30

. generate double x = rchi2(1)

. generate y = 1 + 2*x + rchi2(1)-1

. regress y x, noheader
```

y	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
x	2.11194	.1380605	15.30	0.000	1.829136	2.394744
_cons	.73183	.2574067	2.84	0.008	.2045563	1.259104

- For $N = 30$: $\widehat{\beta}_1 = 0.732$ differs appreciably from $\beta_1 = 1.000$.
  - This is due to sampling error as $\text{se}[\widehat{\beta}_1] = 0.257$.

Consistency

- How to verify consistency: set N very large.

```
. * Consistency - sample size is set large at e.g. 100000
. clear

. set seed 10101

. quietly set obs 100000

. generate double x = rchi2(1)

. generate y = 1 + 2*x + rchi2(1)-1

. regress y x, noheader
```

y	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
x	2.001847	.0031711	631.27	0.000	1.995632	2.008063
_cons	.9955892	.0054482	182.74	0.000	.9849108	1.006268

- For $N = 100,000$: $\widehat{\beta}_2 = 0.9956$, $\widehat{\beta}_2 = 2.0018$ are very close to $(1, 2)$.

Monte Carlo experiment program

- How to check asymptotic results: compute $\widehat{\beta}$ many times.
  - Here $S = 1000$. Also sample size $N = 150$
  - This code uses `postfile`. Could instead use `simulate`.

```
. set seed 54321

. postfile simalternative b se t r p using simresults, replace

. forvalues i = 1/$numsims {
          drop _all
          quietly set obs $numobs
          quietly generate double x = rchi2(1)
          quietly generate y = 1 + 2*x + rchi2(1)-1     // demeaned ch
          quietly regress y x
          scalar b2 =_b[x]
          scalar se2 = _se[x]
          scalar t2 = (_b[x]-2)/_se[x]
          scalar r2 = abs(t2)>invttail($numobs-2,.025)
          scalar p2 = 2*ttail($numobs-2,abs(t2))
          post simalternative (b2) (se2) (t2) (r2) (p2)
  }
```

- Then look at the distribution of these $\widehat{\beta}'s$ and test statistics.

Monte Carlo experiment results

```
. * Analyze the simulation results
. use simresults, clear

. summarize

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
           b |      1,000    1.997102    .0884986    1.70497   2.626617
          se |      1,000    .0837861    .0166873   .0419276   .1469177
           t |      1,000    -.028814      1.0165  -2.930943   4.571772
           r |      1,000        .059    .2357426          0          1
           p |      1,000    .5172642    .2902078   .0000101   .9995021

. mean b se t r p

Mean estimation                   Number of obs     =      1,000

--------------------------------------------------------------
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
           b |   1.997102   .0027986       1.99161   2.002594
          se |   .0837861   .0005277      .0827506   .0848217
           t |   -.028814   .0321445     -.0918926   .0342645
           r |       .059   .0074548      .0443711   .0736289
           p |   .5172642   .0091772      .4992555    .535273
--------------------------------------------------------------
```

- Unbiasedness of $\widehat{\beta}_2$
  - For $S = 1,000$ simulations each with sample size $N = 150$.
    - $\widehat{\beta}_2^{(1)}, \widehat{\beta}_2^{(2)}, \ldots, \widehat{\beta}_2^{(1000)}$ has distribution with mean 1.997
    - This is close to $\beta_2 = 2.000$ (within 95% sim. interval (1.992,2.003)).

Correct standard errors?

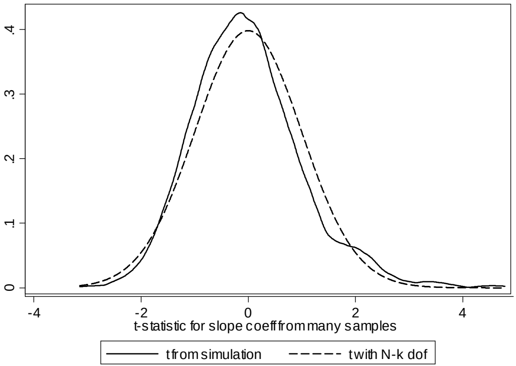
- How to verify that standard errors are correctly estimated.
  - The average of the computed standard errors of $\widehat{\beta}_2$ is 0.0838 (see `mean` of `se`)
  - This is close to the simulation estimate of se$[\widehat{\beta}_2]$ of 0.0885 (see `Std.Dev.` of `b`)
- Aside: Actually for this d.g.p. expect $\sqrt{1/150} \simeq 0.082$ using $\mathsf{V}[\widehat{\beta}_2] \simeq (\sigma_u^2/\mathsf{V}[x_i])/N = (2/2)/150 = 1/150)$

Correct test size?

- How to verify that test has correct size.
  - The Wald test of $H_0 : \beta_2 = 2$ at level 0.05 has actual size 0.059 (see `mean` of `r`)
- This is close enough as a 95% simulation interval when $S = 1000$ is
  - $0.05 \pm 1.96 \times \sqrt{0.05 \times 0.95/1000} = 0.05 \pm 1.96 \times 0.007 = (0.036, 0.064)$.
- More precisely

```
. * Simulation interval for test size when tet at 0.05 and 1000 simulations
. cii proportions 1000 0.05

                                                         -- Binomial Exact --
    Variable |        Obs  Proportion    Std. Err.       [95% Conf. Interval]
-------------+---------------------------------------------------------------
             |      1,000         .05     .006892        .0373354    .0653905
```

- Test $\beta_2 = 2$ using $z = (\widehat{\beta}_2 - \beta_2)/\mathrm{se}[\widehat{\beta}_2] = (\widehat{\beta}_2 - 2.0)/\mathrm{se}[\widehat{\beta}_2]$ to test $H_0 : \beta_2 = 2$.
  Histogram and kernel density estimate for z_1, z_2, ... , z_{1000}.



- Not quite $T(148)$ or $N[0,1]$:
  - $N = 150$ too small for CLT or the $T(148)$ approximation.

Test Power

- Test $H_{a:} : \beta_2 = 2.0$ against $H_a : \beta_2 = 2.1$
- Change the d.g.p. value to 2.1
  - so in the simulation program change to
  - `generate y = 1 + 2.1*x + rchi2(1) - 1`
- Rerun the program and now the rejection rate gives the power
  - Here the power is 0207.

6. Some References

- The material is covered in
  - CT(2005) MMA chapter 12
  - CT(2009) MUS chapter 4
- For Monte Carlo experiments see
  - Davidson, R. and J. MacKinnon (1993), *Estimation and Inference in Econometrics*, chapter 21, Oxford University Press.