Introduction to Simulation

April 5, 2017

Outline

Random Numbers

A fundamental step in a simulation study is the generation of random numbers, where a random number represents the value of a random variable uniformly distributed on $(0, 1)$.

Pseudo-random Number Generation

- Subroutines for generating random numbers are called random-number generators and are generally included in scientific packages or simulation programming languages. Most of these random-number generators use algorithms that produce a deterministic string of numbers that have all the appearances of being independent uniform $(0, 1)$ random variables.

- One of the most common approaches for generating pseudorandom numbers starts with an initial value x_0, called the seed, and then recursively computes successive values x_n, $n \geq 1$, by letting

$$x_n = (ax_{n-1} + c) \mathrm{mod} m \tag{1}$$

  where a, c and m are given positive integers. Each x_n is either $0, 1, \ldots, m-1$ and the quantity $\frac{x_n}{m}$,called a pseudorandom number is taken as an approximation to an uniform $(0, 1)$ random variable.
- This approach specified by equation 1 is called congruential method if $c \neq 0$. If $c = 0$, then it is called the multiplicative congruential method.
- For a 32-bit machine, $m = 2^{31} - 1$ and $a = 7^5$ result in desirable properties.

Example: Use the mixed congruential method to generate a sequence of pseudorandom numbers $R_1, R_2, \cdots$, with $x_0 = 27, a = 17, c = 43$, and $m = 100$.
Solution:

$$\begin{aligned}
x_0 &= 27 \\
x_1 &= [(17)(27) + 43]\mathrm{mod}100 = 502\mathrm{mod}100 = 2 \\
R_1 &= 2/100 = 0.02 \\
x_2 &= [(17)(2) + 43]\mathrm{mod}100 = 77\mathrm{mod}100 = 77 \\
R_2 &= 77/100 = 0.77 \\
\ldots
\end{aligned}$$

We shall not pursue further the question of how to construct "good" random number generators. Rather, we assume in conducting computer simulation of systems, that we have a "black box" that gives a random number on request.

Using Random Numbers to Evaluate Definite Integrals

One of the earliest applications of random numbers was in the computation of integrals. Let $f(x)$ be a function and suppose we wanted to compute the integral

$$I = \int_0^1 f(x)dx.$$

If U is uniformly distributed over $(0, 1)$ then

$$I = E(f(U)).$$

One of the applications of the random numbers is the computation of definite integrals.

To estimate $E(f(U))$, suppose $U_1, U_2, \cdots, U_n$ are independent random variables uniformly distributed in (0,1), then $f(U_1), f(U_2), \cdots, f(U_n)$ are independent and identically distributed random variables having mean I.
Therefore by **Strong Law of Large Numbers**, with probability one,

$$\lim_{n\to\infty} \sum_{i=1}^{n} \frac{f(U_i)}{n} = I$$

This means that when n is large, $I_n \approx \sum_{i=1}^{n} \frac{f(U_i)}{n}$ will be close to I.

Example: Find an approximation of the definite integral

$$I = \int_0^1 \sin(x)dx$$

by using simulation method discussed. The true value of I is 0.4597. The following table shows the results for different number of simulation n.

n	10	100	1000	10000
I_n	0.5170	0.4811	0.4564	0.4596

Table : The Approximation of I for Different n.

Let us extend the idea of simulation to evaluate

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

To apply the result, we perform the following substitution

$$y = \frac{(x-a)}{(b-a)}$$

Therefore

$$dy = dx/(b-a) \text{and} I = \int_0^1 f(a+(b-a)y)(b-a)dy.$$

How about the case when

$$I = \int_0^\infty f(x)dx?$$

We can apply the substitution

$$y = (1+x)^{-1}.$$

Therefore we have

$$dy = -dx/(x+1)^2 = -y^2 dx$$

and

$$I = \int_0^1 \frac{f(\frac{1}{y}-1)}{y^2} dy.$$

Generate Discrete Random Variables (The Inverse Transform Method)

Suppose we want to generate the value of a discrete random variable X having probability mass function

$$P\{X = x_j\} = p_j,\ j = 0, 1,\ \dots,\ \sum_j p_j = 1.$$

We generate a random number U and set

$$X = \begin{cases} x_0 & \text{if } U < p_0 \\ x_1 & \text{if } p_0 \leq U < p_0 + p_1 \\ \vdots & \vdots \\ x_j & \text{if } \sum_{i=0}^{j-1} p_i \leq U < \sum_{i=0}^{j} p_i \\ \vdots & \vdots \end{cases} .$$

since, for

$$0 < a < b < 1, \; P\{a \leq U < b\} = b - a,$$

we have

$$P\{X = x_j\} = P\{\sum_{i=1}^{j-1} p_i \leq U < \sum_{i=1}^{j} p_i\} = p_j$$

and so X has the desired distribution.
This method can be written algorithmically as

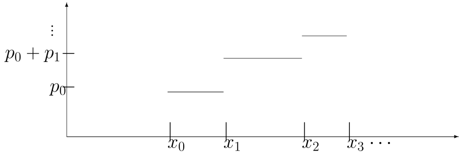
- Generate a random number U.
- If $U < p_0$ set $X = x_0$ and stop.
- If $U < p_0 + p_1$ set $X = x_1$ and stop.
- If $U < p_0 + p_1 + p_2$ set $X = x_2$ and stop.
- $\cdots$

If the $x_i, i \geq 0$ are ordered so that $x_0 < x_1 < \cdots$ and if we let F denote the distribution function of x, then

$$F(x_j) = \sum_{i=0}^{j} p_i$$

and so

$$X = x_j \text{ if } F(x_{j-1}) \leq U < F(x_j).$$



- In other words, after generating a random number U we determine the value of X by finding the interval $(F(x_{j-1}),\ F(x_j))$ in which U lies (or, equivalently finding the inverse of $F(U)$).
- Thus this method is called the discrete inverse transform method for generating X.
- In general, after generating U, one has to go through a table-lookup procedure in order to find X. However, for some cases, one can determine X by an algebraic formula.

Example: Suppose we want to generate the value of X such that

$$P\{X = j\} = \frac{1}{n},\ j = 1, 2,\ \ldots,\ n.$$

Using the inverse transform method,

$$X = j \text{ if } \frac{j-1}{n} \leq U < \frac{j}{n}, j = 1, 2, \ldots, n.$$

or

$$X = \lfloor nU \rfloor + 1$$

where $\lfloor x \rfloor$ (called the floor of x) is the greatest integer less than or equal to x.
In this example, we have an algebraic formula for determining X in terms of U.

Example: (Calculating Averages) Suppose we want to approximate

$$\bar{a} = \sum_{i=1}^{n} \frac{a(i)}{n}$$

where n is large and the values $a(i), i = 1, 2, \ldots, n$ are complicated and not easily calculated.
Let X be a discrete uniform random variable such that

$$P\{X = j\} = \frac{1}{n}, j = 1, 2, \ldots, n.$$

Then

$$E[a(X)] = \sum_{i=1}^{n} a(i)P\{X = i\} = \sum_{i=1}^{n} \frac{a(i)}{n} = \bar{a}.$$

Hence if we generate k random numbers U_i and setting $X_i = \lfloor nU_i \rfloor + 1$, then each of the k random variables $a(X_i)$ will have mean $\bar{a}$ and so by the strong law of large numbers it follows that when k is large,

$$\sum_{i=1}^{k} \frac{a(X_i)}{k} \to \bar{a} \text{ with probability 1.}$$

Example: Consider the geometric random variable X where

$$P\{X = i\} = pq^{i-1}, i \geq 1 \text{ and } q = 1 - p.$$

Since

$$\sum_{i=1}^{j-1} P\{X = i\} = 1 - q^{j-1},\ j \geq 1,$$

we can generate the value of X by generating a random number U and setting X equal to that value j for which

$$1 - q^{j-1} \leq U < 1 - q^{j}$$

or, equivalently, for which

$$X = \min_j (q^j < 1-U) = \min_j (j \log q < \log(1-U)) = \min_j (j > \frac{\log(1-U)}{\log q}).$$

We note that $\log q$ and $\log(1-U)$ are both negative. Hence

$$X = \lfloor \frac{\log(1-U)}{\log q} \rfloor + 1. \tag{2}$$

Since $1-U$ is also uniformly distributed on $(0,1)$, (2) is equivalent to

$$X = \lfloor \frac{\log U}{\log q} \rfloor + 1$$

Thus we again have an algebraic formula for determining X.

Example: Generating a Poisson random variable. Let X be a Poisson random variable with probability mass function

$$p_i = P\{X = i\} = e^{-\lambda}\frac{\lambda^i}{i!}, i = 0, 1, \ldots$$

By noting that

$$p_{i+1} = \frac{\lambda}{i+1}p_i,\ i \geq 0,$$

we can write the inverse transform method for generating X as follows:(where $F = F(i) = P\{X \leq i\}$)

- Step 1: Generate a random number U.
- Step 2: $i = 0, p = e^{-\lambda}, F = p$.
- Step 3: If $U < F$, set $X = i$ and stop.
- Step 4: $p = \lambda p/(i+1), F = F + p, i = i + 1$.
- Step 5: Go to Step 3.

The algorithm successively checks if the Poisson value is 0, then if it is 1, then 2, and so on.

Thus the number of comparisons needed will be 1 greater than the generated value of the Poisson. Hence on average, the above algorithm needs to make $(1 + \lambda)$ searches.

Example: Generating Binomial random variables.
Suppose X is a binomial random variable with probability mass function

$$P\{X = i\} = \frac{n!}{i!(n-i)!} p^i (1-p)^{n-i},\ i = 0, 1,\ \ldots,\ n.$$

Again we apply the inverse transform method by making use of the recursive relation

$$P\{X = i+1\} = \frac{n-i}{i+1} \frac{p}{1-p} P\{X = i\}.$$

- Step 1: Generate a random number U.
- Step 2: $c = p/(1-p), i = 0, r = (1-p)^n, F = r$.
- Step 3: If $U < F$, set $X = i$ and stop.
- Step 4: $r = [c(n-i)/(i+1)]r, F = F + r, i = i + 1$.
- Step 5: Go to Step 3.

In the algorithm with i denoting the value currently under consideration,

$$r = P\{X = i\} \text{ and } F = F(i) = P\{X \leq i\}.$$

The number of searches required is the value of X plus 1.
Since $E[X] = np$, on average, it will take $1 + np$ searches to generate X.

Generate Discrete Random Variables (The Acceptance-Rejection Technique)

Suppose we have an efficient method for simulating a random variable Y having probability mass function $\{q_j,\ j \geq 0\}$.
We can use this as the basis for simulating another random variable X with probability mass function $\{p_j,\ j \geq 0\}$.
Let c be a constant such that $\frac{p_j}{q_j} \leq c$ for all j such that $p_j > 0$.
Rejection Method (acceptance-rejection method)

- Step 1: Simulate the value of Y, having probability mass function q_j.
- Step 2: Generate a random number U.
- Step 3: If $U < p_Y / cq_Y$, set $X = Y$ and stop. Otherwise, return to Step 1.

Proposition 1

The acceptance-rejection method generates a random variable X such that

$$P\{X = j\} = p_j,\ j = 0, 1,\ \ldots\ .$$

In addition, the number of iterations of the algorithm needed to obtain X is a geometric random variable with mean c.

Proof.

Let us first determine the probability that a single iteration produces the accepted value j.

$$\begin{aligned} P\{Y=j, \text{it is accepted}\} &= P\{Y=j\}P\{\text{accepted}|Y=j\} \\ &= q_j \frac{p_j}{cq_j} \\ &= \frac{p_j}{c} \end{aligned}$$

Summing over j,

$$P\{\text{accepted}\} = \sum_j \frac{p_j}{c} = \frac{1}{c}.$$

As each iteration independently results in an accepted value with probability $\frac{1}{c}$, we see that the number of iterations needed is geometric with mean c. □

Proof.

(Continued) Also,

$$P\{X = j\} = \sum_n P\{j \text{accepted on iteration n}\}$$
$$= \sum_n (1 - \frac{1}{c})^{n-1} \frac{p_j}{c} = p_j.$$

$\square$

Remarks: (i) Since $\sum_j p_j \leq c \sum_j q_j$ and $\sum_j p_j = \sum_j q_j = 1$, we have $c \geq 1$. (ii) The number of iterations is smaller if c is smaller.

Example: Suppose we wanted to generate a random variable X that takes one of the values 1, 2, . . . , 10 with respective probabilities $0.11, 0.12, 0.09, 0.08, 0.10, 0.08, 0.10, 0.09, 0.11, 0.12$. One way is to use the inverse transform method.
However, here let us use the rejection method with Y being the random variable with the uniform discrete density

$$q_j = P\{Y = j\} = \frac{1}{10}\ ,\ j = 1, 2, \ldots, 10.$$

Let $p_j = P\{X = j\}$. Then

$$c = \max \frac{p_j}{q_j} = 1.2$$

and the algorithm is given as follows:
Step 1: Generate a random number U_1 and set $Y = \lfloor 10U_1 \rfloor + 1$.
Step 2: Generate a second number U_2.
Step 3: If $U_2 \leq \frac{p_Y}{0.12}$, set $X = Y$ and Stop. Otherwise return to Step 1.

The Composition Approach

Suppose we have efficient methods for generating random variables X_1 and X_2 with probability mass functions $\{p_j^{(1)}, j \geq 0\}$ and $\{p_j^{(2)}, j \geq 0\}$ respectively, and we want to generate a random variable X having mass function

$$P\{X = j\} = \alpha p_j^{(1)} + (1 - \alpha) p_j^{(2)}, \ j \geq 0$$

where $0 < \alpha < 1$.
The composition algorithm is as follows:
Step 1: Generate X_1 with mass function $\{p_j^{(1)}, j \geq 0\}$.
Step 2: Generate X_2 with mass function $\{p_j^{(2)}, j \geq 0\}$.
Step 3: Generate a random number U.
Step 4: If $U < \alpha$, set $X = X_1$. Otherwise, set $X = X_2$.

We can prove that the algorithm gives the correct value by observing

$$\begin{aligned} P\{X=j\} &= P\{U<\alpha \text{ and } X_1=j\} + P\{U\geq\alpha \text{ and } X_2=j\} \\ &= P\{U<\alpha\}P\{X_1=j\} + P\{U\geq\alpha\}P\{X_2=j\} \\ &= \alpha p_j^{(1)} + (1-\alpha)p_j^{(2)} \end{aligned}$$

Generating Continuous Random Variables (Inverse Transform Algorithm)

The inverse transform algorithm for generating continuous random variable is based on the following proposition:

Proposition 2

Let U be a uniform $(0, 1)$ random variable. For any continuous distribution function F the random variable X defined by

$$X = F^{-1}(U)$$

has distribution F.

Proof.

Let F_X denote the distribution function of $X = F^{-1}(U)$. Then

$$F_X(x) = P\{X \leq x\}$$

$$= P\{F^{-1}(U) \leq x\}.$$

Since F is a monotonic increasing function of x and so $F^{-1}(U) \leq x$ is equivalent to $F(F^{-1}(U)) \leq F(x)$, or

$$F_X(x) = P\{F(F^{-1}(U)) \leq F(x)\} = P\{U \leq F(x)\} = F(x) \ .$$

□

Example: Let X be an exponential random variable with mean $\frac{1}{\lambda}$ and distribution function

$$F(x) = 1 - e^{-\lambda x}$$

If we let

$$x = F^{-1}(u) \ ,$$

then

$$u = F(x) = 1 - e^{-\lambda x}$$

or

$$x = -\frac{1}{\lambda} \log(1 - u) \ .$$

Hence we can generate the exponential random variable by generating a random number U and then setting

$$X = F^{-1}(U) = -\frac{1}{\lambda} \log(1 - U) \ .$$

Since $U \sim 1 - U$ (i.e. they have the same distribution), equivalently we can write

$$X = -\frac{1}{\lambda} \log(U) \ .$$

Generating Continuous Random Variables (The Rejection Method)

Suppose we have a method for generating a random variable Y having density function $g(x)$. We can use this as basis for generating a random variable X having density function $f(x)$.
Let c be a constant such that

$$\frac{f(y)}{g(y)} \leq c \text{ for all } y$$

The Rejection Method
Step 1: Generate Y having density g.
Step 2: Generate a random number U.
Step 3: If $U \leq \frac{f(Y)}{cg(Y)}$, set $X = Y$. Otherwise, return to Step 1.
The method is similar to the rejection method for discrete random variables except that density function replaces mass function. As in the discrete case, we can prove the following result.

Proposition 3

The random variable generated by the rejection method has density f. and the number of iterations of the algorithm that are needed is a geometric random variable with mean c.

Proof.

$$
\begin{aligned}
P[X \le x] &= P[Y \le x | U \le \frac{f(Y)}{cg(Y)}] \\
&= \frac{P[Y \le x, U \le \frac{f(Y)}{cg(Y)}]}{P[U \le \frac{f(Y)}{cg(Y)}]} \\
&= \frac{\int_{-\infty}^{x} [\int_0^{f(Y)/cg(Y)} du] g(y) dy}{\int_{-\infty}^{\infty} [\int_0^{f(Y)/cg(Y)} du] g(y) dy} \\
&= \frac{\int_{-\infty}^{x} \frac{f(y)}{cg(y)} g(y) dy}{\int_{-\infty}^{\infty} \frac{f(y)}{cg(y)} g(y) dy} \\
&= \int_{-\infty}^{x} f(y) dy
\end{aligned}
$$

$\square$

Example Let us use the rejection method to generate a random variable having density function

$$f(x) = 20x(1-x)^3,\ 0 < x < 1.$$

Consider $g(x) = 1, 0 < x < 1$.
To determine c such that $f(x)/g(x) \leq c$, we use calculus to determine the maximum value of

$$\frac{f(x)}{g(x)} = 20x(1-x)^3$$

$$\frac{d}{dx}(\frac{f(x)}{g(x)}) = 20[(1-x)^3 - 3x(1-x)^2].$$

The derivative is zero when $x = 1/4$, positive when $x < \frac{1}{4}$ and negative when $x > \frac{1}{4}$.

Hence max $\frac{f(x)}{g(x)}$ occurs at $x = 1/4$ and thus

$$\begin{aligned}\frac{f(x)}{g(x)} \leq \frac{f(\frac{1}{4})}{g(\frac{1}{4})} &= 20(\frac{1}{4})(\frac{3}{4})^3 \\ &= \frac{135}{64} \\ &\equiv c.\end{aligned}$$

Hence

$$\frac{f(x)}{cg(x)} = \frac{256}{27}x(1-x)^3$$

Hence the rejection method is as follows:
Step 1: Generate random numbers U_1 and U_2
Step 2: If $U_2 \leq \frac{256}{27}U_1(1-U_1)^3$, stop and set $X = U_1$. otherwise return to Step 1.
The average number of iterations that will be performed is $c = \frac{135}{64}$

Example: Suppose we wanted to generate a random variable having the density

$$f(x) = Kx^{1/2}e^{-x}\ ,\ x > 0$$

where $K = 2/\sqrt{\pi}$. Let us try the rejection method with an exponential random variable with mean λ^{-1}
Hence let $g(x) = \lambda e^{-\lambda x}, x > 0$. Now

$$\frac{f(x)}{g(x)} = \frac{K}{\lambda}x^{1/2}e^{-(1-\lambda)x},$$

and

$$\frac{d}{dx}(\frac{f(x)}{g(x)}) = \frac{K}{\lambda}[\frac{1}{2}x^{-\frac{1}{2}}e^{-(1-\lambda)x} - (1-\lambda)x^{\frac{1}{2}}e^{-(1-\lambda)x}]$$

We have

$$\frac{d}{dx}(\frac{f(x)}{g(x)}) = 0 \text{ when } \frac{1}{2}x^{-\frac{1}{2}} = (1-\lambda)x^{\frac{1}{2}}$$

or when $x = \frac{1}{2(1-\lambda)}$, provided that $\lambda < 1$.

Moreover

$$\frac{d^2}{dx^2}\left(\frac{f(x)}{g(x)}\right)\Big|_{x=\frac{1}{2(1-\lambda)}} = \frac{-Ke^{-1/2}}{2\lambda(2(1-\lambda))^{1/2}} < 0.$$

Hence

$$c = \mathrm{Max}\left(\frac{f(x)}{g(x)}\right) = \frac{K}{\lambda}\frac{1}{[2(1-\lambda)]^{1/2}}e^{-\frac{1}{2}}$$

Algorithm:
Step 1: Generate a random number U_1 and set $Y = -\frac{1}{\lambda}\log U_1$.
Step 2: Generate a random number U_2.
Step 3: If $U_2 < \frac{f(Y)}{cg(Y)}$, set $X = Y$. Otherwise, return to Step 1.

The average number of iterations that will be needed is *c*. Since *c* depends on λ, we can try to minimize *c* by choosing suitable A. We differentiate *c* with respect λ,

$$c'(\lambda) = Ke^{-\frac{1}{2}}\left[-\frac{1}{\lambda^2}\frac{1}{2^{1/2}(1-\lambda)^{1/2}} + \frac{1}{\lambda}\frac{1}{2^{1/2}(1-\lambda)^{3/2}}\left(\frac{1}{2}\right)\right].$$

$c'(\lambda) = 0$ when

$$1 - \lambda = \frac{\lambda}{2} \text{ or } \lambda = \frac{2}{3}.$$

Hence

$$c = \frac{3K}{2}\left(\frac{3}{2}\right)^{1/2} e^{-1/2} = \frac{3^{3/2}}{(2\pi e)^{1/2}}.$$

Since *c* is minimized, the number of iterations is minimized also.

Example: Generating a Normal Random Variable. Let Z be the standard normal random variable, and let $X = |Z|$. The density function of X is

$$f(X) = \frac{2}{\sqrt{2\pi}} e^{-x^2/2},\ 0 < x < \infty$$

We first generate X by using the rejection method with g being the exponential density function with mean 1, that is, $g(x) = e^{-x}, 0 < x < \infty$. Now

$$\frac{f(x)}{g(x)} = \sqrt{2/\pi} e^{x - x^2/2}$$

and so the maximum value of $f(x)/g(x)$ occurs at the value of x that maximizes $x - x^2/2$. This occurs at $x = 1$ by simple calculus. So we can take

$$c = \max \frac{f(x)}{g(x)} = \frac{f(1)}{g(1)} = \sqrt{\frac{2e}{\pi}}$$

Hence

$$\frac{f(x)}{cg(x)} = \exp\{-\frac{(x-1)^2}{2}\}.$$

Algorithm for generating the absolute value of a standard normal random variable:
Step 1: Generate Y, an exponential random variable with mean 1.
Step 2: Generate a random number U.
Step 3: If $U \leq \exp\{-(Y-1)^2/2\}$, set $X = Y$. Otherwise, return to Step 1.
Once we have simulated the random variable X, we can generate the standard normal random variable Z by letting Z be equally likely to be either X or X.
This can be written algorithmically as follows:
Step 1: Generate X.
Step 2: Generate a random number U.
Step 3: If $U \leq \frac{1}{2}$, set $Z = X$. Otherwise set $Z = -X$.

A Summary on Simulation

- Generation of random numbers/uniformly distributed r.v. by using the congruential method.
- Approximation of definite integral by random numbers.
- Inverse transformation method for discrete and continuous r.v..
- Acceptance and rejection method for discrete and continuous r.v..
- Generation of Poisson and Exponential r.v.