STATISTICAL SIMULATION AND NUMERICAL PROCEDURES

Ion Văduva
University of Bucharest, Romania

Keywords: Random numbers, Random variates, Random number generators, Linear congruential generators, Inverse method, Composition method, Acceptance-rejection method, Accepting probability, Ratio-of-uniform method, Bootstrap method, Monte Carlo method, Monte Carlo procedure, Primary estimator, Secondary estimator, Simulated annealing, Markov Chain Monte Carlo (MCMC) method, Metropolis-Hastings algorithm, Gibbs sampler.

Contents

Summary

The paper presents in short the main questions related to the use of simulation in studying statistical problems and solving some classes of numerical problems. First, methods for simulating various types of statistical distributions are presented. Then, some applications of simulation in statistics, including bootstrap techniques, are also discussed. A special attention is paid to Monte Carlo techniques and the Markov Chain Monte Carlo method. References contain only some of the representative or recent publications.

1. Introduction

The term *simulation* represents today science of a wide class of problems, solved or analyzed via computers. There are many ways to define and understand *simulation* but all of them assume the use of *random numbers* to perform a *computer experiment* for solving some mathematical or practical problem. In this article, the word simulation is also associated with terms like *Monte Carlo* techniques and *resampling* techniques, the last one involving statistical problems.

Random numbers are sampling values on the uniform distribution over $(0,1)$ which has the probability density function (*pdf*)

$$f(u)=\begin{cases}1, & \text{if } u\in(0,1)\\ 0, & \text{otherwise.}\end{cases}$$

(Note that it makes no difference if we consider the interval $(0,1)$ as being open or closed at any of its limits).

The following proposition (due to Khintchine) plays a great role in simulating non-uniform random variates.

Theorem 1: *If X is a random variable having the cumulative distribution function* (*cdf*) $F(x)$, $x\in R$, *and U denotes the random variable uniformly distributed over* $(0,1)$, *then the random variable $F^{-1}(U)$ (with F^{-1}-the inverse of F), has the cdf F .*

In other words, this theorem gives a general method for simulating a sampling value x of the random variable X when we have a sampling value u of U, namely, $x=F^{-1}(u)$. That is why the next chapter will be dedicated to simulating random numbers. The same theorem suggests that there could be various methods which transform sequences of random numbers into non-uniform variates. Thus, another chapter will discuss these methods.

Since one purpose of this paper is to discuss the use of simulation in solving statistical problems, one section is devoted to the *bootstrap* method and some applications. The Monte Carlo method for solving various numerical problems is introduced in another chapter. Some applications of the so-called *Markov Chain Monte Carlo* are also presented.

2. Random Number Generation

Random numbers, i.e. sampling values on the random variable U uniformly distributed over $(0,1)$ (denoted $U \sim (0,1)$), are very important for the problems to be treated in this paper.

The aim of this chapter is to present in short some methods for generating with the computer sampling values on the random variable U, which are independent and uniformly distributed over $[0,1)$. As Knuth and other authors have shown, the computer calculations necessary to produce good random numbers, require first to generate uniformly distributed integers over some interval $[0, m)$, and then to divide this by m in order to obtain the required random number. The calculations needed to produce uniformly distributed integers in $[0, m)$ must be simple. In other words, the generation algorithm must have a low complexity, both regarding computing time and memory complexities. Details on random number generation are found in many books (see for instance Devroye-1986, Ermakov-1971, Gentle-1998, Ripley-1986 and Ross-1997).

2.3. Linear Congruential Generators

A *linear congruential generator* is of the form

$$x_n = \left(\sum_{i=1}^{k} a_i x_{n-i} + c \right) (\text{mod } m), \; x_n \in \mathcal{N}, \tag{1}$$

where m is a large positive integer, $k \le n$, and $a_i, x_i, i = 1,\ldots,k, c$ are given constants, all chosen such that the produced numbers x_n, $n > k$ are integers uniformly distributed over the interval $[0, m-1)$ and $\mathcal{N}$ is the set of natural numbers. Then the uniform $U[0, 1)$ random numbers are obtained as

$$u_n = \frac{x_n}{m}. \tag{1a}$$

The usual linear (*mixed*) congruential generator is the one with $k = 1$, i.e. $x_{n+1} = \left(a^* x_n + c\right)(\text{mod}\, m)$. If a, c and m are properly chosen, then, in this case, u_n "look like" they are randomly and uniformly distributed between 0 and 1. Even if this linear congruential generator has a low complexity, the most used is the *multiplicative congruential generator*

$$x_{n+1} = (a x_n) \; (\text{mod} \quad m). \tag{2}$$

If $x_0 \neq 0$ is prime with m, and a is a *primitive root* $\bmod m$, close to $\sqrt{m}$, then the numbers u_n, produced by this generator have a large *period* λ, (defined as the minimum λ such as $x_n = x_{n+\lambda}$), they are approximately uniform $(0,1)$ distributed and have a small *serial correlation coefficient* $\rho = corr(u_n, u_{n+1}) \forall n$ (i.e. are *almost independent*). Of course, the *modulus m* must be very large (usually close to the computer word, i.e. close to 2^{31}).

In simulation we use sequences of random numbers $u_1, u_2, \ldots, u_n$ produced by a random number generator. These numbers must pass any test which assumes that they are uniformly distributed and stochastically independent. It is obvious that a random number generator cannot produce "pure" random numbers to pass the mentioned tests. Therefore we call them *pseudo-random numbers*. A "good" random number generator must produce sequences close to pure random numbers. A linear congruential generator cannot produce good random numbers. It can be used when there is no need to perform *very accurate calculations* or to obtain *exact* solutions to the problems.

One trouble with using pseudorandom numbers produced by a linear congruential generator is that pairs (u_i, u_{i+1}) or triplets (u_i, u_{i+1}, u_{i+2}) are lying on lines or planes (i.e. have a *lattice structure*). This means that these generators must be used with care in numerical calculations. In order to obtain "better" random numbers from a uniform pseudo-random number generator, the numbers produced by the generator must be transformed. If we consider the *binary representation* of the numbers u_i, then one way to obtain better numbers is to use the *bit stripping*, i.e. to obtain the new numbers by selecting some bits from the sequences of bits representing previous given numbers (e.g. odd bits or even bits, etc).

2.4. Other Sources of Uniform Random Numbers

Note that if in (2) we take a^k instead of a and start with x_s, then the sequence of pseudo-random numbers obtained is $x_s + x_{s+k}, x_{s+2k}, \ldots$ and therefore, for various values of s, the corresponding stream can be used by one processor in a parallel architecture.

Shuffling random numbers: A way of improving the quality of a uniform pseudo-random number generator is to define the *new* number *y* by mixing (or *shuffling*) two generators $G1$, $G2$. One mixing algorithm (due to MacLaren and Marsaglia) is:

Take an array (i.e. a table) $T[1\ldots k]$, $k =$ = fixed, and initialize (fill in) it using G_1; generate with G_2 a random index $j \in \{1, 2, \ldots k\}$; take $y := T[j]$; generate x with G_1, and put $T[k] = x$.

(The expression $a := b$ means that b is assigned to a). The *better* generated number is y. This mixed generator can have a larger period and can break up the lattice structure of the generated sequence $\{y_i\}$. If instead of two generators we use only one, $G = G_1 = G_2$, then the above algorithm (called Bays-Durham shuffling of random numbers) can be easily changed by generating only one x in the initial step and determining j by the "*bit stripping*" procedure mentioned before.

Lagged Fibonacci sequences: Apart from linear congruential generators, another way of generating random numbers is to use the *lagged Fibonacci generator*, defined as

$$x_i = (x_{-j} + x_{i-k}) \pmod m \qquad (3)$$

which, when m is prime and $k > j$, gives a period close to $m^k - 1$.

Inversive congruential generators: This method produces uniform integers over $[0, m-1]$ by the relation

$$x_i = \left(a x_{i-1}^{-1} + c\right) \pmod m \qquad (4)$$

where x^{-1} denotes the multiplicative *inverse* modulo m if it exists, or else is 0. Even if these inverting generators imply computational difficulties, they promise to give high quality random sequences.

Matrix congruential generators: Such a generator is of the form

$$\mathbf{x}_i = (\mathbf{A}\mathbf{x}_{i-1} + \mathbf{C}) \pmod m$$

where $\mathbf{x}_i$ are vectors of dimension d and $\mathbf{A}$ and $\mathbf{C}$ are $d \times d$ matrices. This kind of generators are important when parallel computers are used to produce correlated random vectors.

Feedback shift register generators: Such a generator takes into consideration the binary representation of integers in registers of the computer. If $a_i, i = 1, \dots, p$, denote the binary digits of the random number, and c_i are given (not all zero) binary digits, then the digits a_i of the *new* generated number are produced by

$$a_i = (c_p a_{i-p} + c_{p-1} a_{i-p+1} + \dots + c_1 a_{i-1}) \pmod 2. \qquad (5)$$

This generator was introduced by Tausworthe. In practice it has the form

$$a_i = (a_{i-p} + a_{i-p+q}) \pmod 2 \qquad (5a)$$

or, if we denote ⊕, the *binary exclusive-or operation*, as addition of 0'*s* and 1'*s* modulo 2, equation (5a) becomes

$$a_i = a_{i-p} \oplus a_{i-p+q}. \qquad (5b)$$

Note that this recurrence of bits a_i*'s* is the same as the recurrence of random numbers, (interpreted as l–tuples of bits), namely,

$$x_i = x_{i-p} \oplus x_{i-p+q}. \qquad (6)$$

If the random number has l binary digits $(l \le p)$, and l is relatively prime to $2^p - 1$, then the period of the l–tuples (i.e. of the sequence of generated numbers) is $2^p - 1$. A variation of the Tausworthe generator, called *generalized feedback shift register* (*GFSR*), is obtained if we use a bit-generator in the form (5a) to obtain an l–bit binary number and next bit-positions are obtained from the same bit-positions but with delay (by *shifting* usually to the left). A particular GFSR is $x_i = x_{i-3p} \oplus x_{i-3q}$, $p = 521, q = 32$ which gives a period $2^{521} - 1$. Another generator of this kind is the so-called *twisted GSFR generator*, which recurrently defines the random integers x_i as

$$x_i = x_{i-p} \oplus Ax_{i-p+q} \qquad (6a)$$

where A is a properly chosen $p \times p$ matrix.

A practical remark: Apart from *shuffling* random numbers as mentioned above, some other simple combinations could be used to produce "good" random numbers. Thus, if we use the following three generators

$$x_i = 171x_{i-1} (\text{mod} \quad 30269), y_i = 172y_{i-1} (\text{mod} \quad 30307), z_i = 170z_{i-1} (\text{mod} \quad 30323)$$

with positive initializations (*seeds*) (x_0, y_0, z_0) and take uniform $(0,1)$ numbers such as

$$u_i = \left(\frac{x_i}{30269} + \frac{y_i}{30307} + \frac{z_i}{30323} \right) \quad (\text{mod} \quad 1)$$

it can be shown that the sequence of u_i's has a period of order 10^{12}.

3. Non Uniform Random Variate Generation

In this chapter we assume that a uniform $(0,1)$ random number generator called **rnd** is given. The aim of this chapter is to present methods and algorithms which transform sequences of random numbers $u_i \cdot u_2, \ldots, u_n, n \ge 1$ into a sampling value of a given random variable X which has a cdf $F(x)$. (For further information see Devroye-1986, Gentle-1998 and Ross-1997).

3.3. General Methods

3.3.1. The Inverse Method

Theorem 1 leads to the following algorithm (the *inverse method*):

generate u with **rnd**; take $x := F^{-1}(u)$.

The following list gives some examples of the inverse method:

Distribution	*cdf*	*Inverse*
$\exp(\lambda)$	$F(x) = 1 - e^{-\lambda x}, x > 0, \lambda > 0$	$x := -\ln(u)$
$Weib(0,1,\nu)$	$F(x) = 1 - e^{-x^{\nu}}, \nu > 0$	$x := (-\ln(u))^{1/\nu}$
Cauch	$F(x) = \frac{1}{\pi}(\arctan \quad x + \frac{\pi}{2}), x \in R$	$x = \tan \pi(u - \frac{1}{2})$
Pears XI	$F(x) = 1 - \frac{1}{(1+\alpha x)^{\nu}}, x > 0, \nu > 0$	$x = \frac{1}{u^{1/\nu}}$

(The abbreviations are: exp for exponential; Weib for Weibull; Cauch for Cauchy; Pears XI for Pearson type XI).

In the multivariate case, there is a generalization of Theorem 1 which gives a similar algorithm for simulating a sampling value $\mathbf{x} = (x_1, x_2, \ldots, x_k)'$ of the k-dimensional random vector $\mathbf{X}$ which has the cdf $F(\mathbf{x})$. Let us denote

$$F_1(x_1) = P(X_1 < x_1), F_j(x_j) = P(X_j < x_j \mid X_{j-1} = x_{j-1}, \ldots, X_1 = x_1), 1 < j \leq k.$$

The algorithm is (the *multivariate inverse method*):

generate u with **rnd**;
take $x_1 = F_1^{-1}(u)$
for $i := 2$ **to** k **do**
begin
Generate u with **rnd**; take
$x_i = F_i^{-1}(u)$;
end.

An inverse algorithm for simulating a finite discrete random variate having probability distribution

$$X : \begin{pmatrix} a_1, & a_2, & \ldots, & a_n \\ p_1, & p_2, & \ldots, & p_n \end{pmatrix}$$

is:

calculate $F_i = \sum_{\alpha=1}^{i} p_\alpha$, $1 \le i \le n$; take $i := 0$;
generate u with **rnd**;
repeat
$i := i + 1$;
until $u < F_i$;
take $x := a_i$.

The loop in the algorithm searches for the value of index i; this can be better done by using the *binary search* technique.

-
-
-

Bibliography

Davison, A.C. and Hinkley, D.V. (1997). *Bootstrap Methods and their Applications*, Cambridge University Press, Cambridge. [This is an up-to-date monograph on bootstrap resampling methods and their applications in statistics]

Devroye, L. (1986). *Non-Uniform Random Variate Generation*, Springer Verlag, New York. [The book, after fourteen years of being published, is still a complete monograph which contains all methods invented in the field of random variate simulation, including algorithms for simulating some stochastic processes and particular types of random vectors also]

Efron, B. and Tibshirani, R.J. (1993). *An Introduction to the Bootstrap*, Chapman & Hall, New York. [This is a pioneering book in the field of bootstrap methodology]

Ermakov, E.S. (1971). *Monte Carlo Method and Related Questions*, (Russian), "Nauka" Publishing House, Moscow. [The book dominated the literature on Monte Carlo methods in Eastern Europe during the 70's. It is written in an accurate mathematical form and discusses some particular simulation models]

Fishman, G.S. (1996). *Monte Carlo: Concepts, Algorithms and Applications*, Springer Verlag, New York. [A modern and consistent monograph studying a wide area of theoretical questions regarding computer simulation and Monte Carlo methods as well as various applications]

Garzia, R.F. and Garzia, M.R. (1990). *Network Modeling, Simulation, and Analysis*, Marcel Dekker, New York. [The book does not fit within the framework of this contribution, but it is important for the algorithmic way of analysing stochastic networks which may help in solving particular types of optimization problems]

Gentle, J.E. (1998). *Random Number Generation and Monte-Carlo Methods*, (Statistics and Computing Series), Springer Verlag, New York. [The book has inspired me to discuss some algorithms for simulating uniformly distributed random numbers; it contains almost all ideas in this respect, underlines the way of analysing the quality of random numbers and also presents in a modern way the problems related to Monte Carlo methods]

Knuth, D.E. (1981). *The Art of Computer Programming, Volume 2, Seminumerical Algorithms*, second edition, Addison-Wesley Publishing Company, Reading, Massachusetts. [This is the first monograph of the end of the 60's which describes and performs a deep analysis of the arithmetic problems of computer generation of random numbers and their testing]

Pham, D.T. and Karaboga, D. (2000). *Intelligent Optimisation Techniques*, Springer Verlag, Berlin. [The book is an introductory text to optimization methods based on simulated annealing and genetic algorithms]

Ripley, B. (1986). *Stochastic Simulation, Wiley*, New York. [This is a concise text underlining in an accurate way the main problems of Monte Carlo simulation. Various applications are also explained]

Robert, C.P. and Casella, G. (1999). *Monte Carlo Statistical Methods*, Springer Verlag, New York. [This is a complete monograph on statistical simulation which performs a good analysis of Markov Chain Monte Carlo methods and their applications. Monte Carlo integration and Monte Carlo optimization are also discussed]

Ross, S.M. (1997). *Simulation*, second edition, (Statistical Modeling and Decision Science series), Academic Press, London. [This is a good textbook discussing many issues related to statistical applications of simulation and to the Markov Chain Monte Carlo method]

Văduva, I. (1994). "Fast algorithms for computer generation of random vectors used in reliability and applications". *Preprint Nr. 1603, January 1994, TH-Darmstadt*. [Here are found methods for simulating various multivariate distributions, including those based on transforming uniformly distributed vectors into other types of random vectors. Algorithms for the simulation of univariate and multivariate distributions used in reliability are particularly discussed]

Biographical Sketch

Ion Vaduva is a professor of Faculty of Mathematics at the University of Bucharest, Romaina,. He was born on November 25, 1936 in Romania and graduated in Mathematics from the University of Bucharest in 1960. He received Ph.D. in Mathematical Statistics in 1968, from the Center of Mathematical Statistics of the Romanian Academy and M.Sc. in Automatic Computation in 1969, from the University of Manchester, Institute of Science and Technology (UMIST).

His research interests and activities are in the areas of Computer Simulation and Modeling, Mathematical Statistics and Data Analysis, Operation Research, Computation. He teaching experience spans the following subject: Computer Simulation and Monte Carlo Methods (for under-graduates), Software Reliability (for undergraduates), Computerized Stochastic Models (for M.Sc. students), Multivariate Statistical Analysis (for postgraduates and undergraduates), Background of Computer Science (for undergraduates). Since 1971 he has been supervising research work of scholars for Ph.D. He published more than 90 papers in various Romanian or International journals and 18 books or textbooks (all in Romanian). He received the "Simion Stoilow" Prize of the Romanian Academy in 1977. He is Associate Chief Editor of "Analele Universitatii Bucuresti Matematica-Informatica". He was a Visiting Researcher at the following places GMD-Bonn (1974,1976), Sheffield Hallam University (1972), TH-Darmstadt (1993), Alborg University (1998).

Professor Vaduva is a Member of Biometric Society, International Association of Statistical Computing (IASC) and American Mathematical Society (AMS).