Stochastic Simulation

Michaelmas Term 2002

Dr. Gesine Reinert

Organization of the class

This class will take place Wednesdays 12-1, from week 1 to week 8, in the Department of Statistics. In addition there will be a practical class on

Tuesday, November 19 (week 6), in the Computer lab in the department

Lecture notes and a problem sheet will be handed out in the beginning of every second week. For computer exercises, we will use S-PLUS. Note that lectures may differ slightly from these lecture notes.

Recommended reading

1. J.M. Hammersley and D.C. Handscomb (1964). *Monte Carlo Methods.* Methuen.

2. A.M. Law and W.D. Kelton (1999). *Simulation Modeling and Analysis.* Third edition, McGraw-Hill.

3. B.J.T. Morgan (1984). *Elements of Simulation.* Chapman and Hall.

4. B.D. Ripley (1987). *Stochastic Simulation.* Wiley.

5. S.M. Ross (1996). *Simulation.* Second edition. Academic Press.

Contents

1 Introduction

Over the last 50 years, simulation has become an important tool in applied mathematics, in computer science, and in engineering. Often a real-life problem is too complex for a closed-form solution; think of the aerodynamics of an airplane, or of weather systems. In principle, these complex deterministic systems could be described completely, for example by differential equations, but these are too large and/or too complicated to be solvable exactly. Hence one would resort to simulation to understand the underlying behaviour.

In addition, there are complex problems that involve randomness and hence are not solvable deterministically at all. Among these would be, for example, internet traffic, and the spread of disease in a population. In this class we will restrict our attention to this type of problems, namely those that have an element of randomness. Some uses of simulation would be

- exploratory modeling of the behaviour of complex stochastic systems, such as queuing systems, spatial spread of epidemics
- to gain insight in statistical behaviour, such as simulating from a binomial distribution to illustrate the central limit theorem
- intractable deterministic problems in analysis sometimes have a solution involving stochastic processes, and these could be studied by simulation
- examination of the properties of statistical procedures, such as estimating the power of tests, assessing robustness of procedures
- statistical inference: determine the null distribution of a statistic

Advantages of simulations are that we could try different scenarios easily, that we could analyze systems of almost arbitrary complexity, and that we could visualize processes. Disadvantages are that a simulation might be slow, and that optimization of the algorithms can be hard. Moreover, rare event simulation is extremely difficult. Large scale modeling is still very hard and time consuming.

2 Pseudo-random number generators

For any simulation of stochastic systems, we need to be able to simulate "randomness", that is, to simulate observations that look as if they came from a prespecified probability distribution. One way of achieving this would be for example to flip a fair coin repeatedly, to roll a fair die repeatedly, or to draw a card from a deck of cards repeatedly.

For example, if you wanted to determine the probability that with 4 cards drawn at random from a deck of 52 cards, exactly two of them are aces, you could simulate this event by shuffling repeatedly, dealing 4 cards, and recording whether or not they contained two aces.

In simulations often we need many random numbers, so this is not feasible. (Think of assessing insurance risks - these are typically rare events.) On the other hand, computer algorithms are deterministic in nature, so it is not obvious how to generate random numbers in a computer. Of course statistical software comes with pseudo-random number generators, but this class is partly designed to look behind that curtain. In particular we will see that these pseudo-random generators have its flaws, and these flaws are important to know when setting up a simulation.

2.1 Generating from the uniform distribution

The best distribution to start with is the uniform distribution on $[0, 1]$, denoted by $\mathcal{U}([0, 1])$. Ideally, a pseudo-random number generator (RNG) for $\mathcal{U}([0, 1])$ would

- provide a very good approximation to $\mathcal{U}([0, 1])$ marginally
- have very close to independent output in a moderate number of dimensions
- be repeatable from a specified starting point
- be fast
- not repeat itself too often
- be portable (for different operating systems)

- be analyzable
- in cryptography often it is desired that the RNG is unpredictable.

Virtually all random number generators are based on the following idea. We have a finite set E and a function $f : E \to E$. Given an initial value X_0, the generated sequence is

$$X_0, X_1 = f(X_0), X_2 = f(X_1) = f^2(X_0), \ldots.$$

The three most common types of pseudo-random number generators are the so-called congruential generators, the shift-register generators, and lagged-Fibonacci generators, with variants.

2.1.1 Congruential generators

The *(first-order) congruential generators* were introduced by Lehmer (1951). They are of the form

$$X_n = aX_{n-1} + b \pmod{M}$$

so that $X_n \in [0, M-1]$. If $b = 0$ this is called a *multiplicative* generator, if $b \neq 0$ it is called a *mixed* generator. We then put

$$U_i = \frac{X_i}{M}.$$

The starting point X_0 is called the *seed*. The seed is often chosen by the programmer, or using the internal clock of the computer. Ideally, the seed should be chosen at random from $\mathcal{U}([0,1])$.

A convenient choice for M would equal the computer's word length, since then division by M is quite efficient. However, to reduce periodicity, M should be large.

We also want the sequence not to repeat itself too often. A congruential generator yields a periodic sequence with period k, say. For mixed congruential generators, $k \leq M$. If $k = M$, the generator is said to have *full period*. For multiplicative congruential generators, $k \leq M - 1$, since 0 repeats itself indefinitely. If $k = M - 1$, the multiplicative generator is called *maximal*.

In Hammersley and Hanscomb (1964), p.28 (see also Ripley (1987)) the following can be found.

Proposition 1 *For a mixed congruential generator, the full period of M can always be achieved provided that*

1. *b and M have no common divisor*
2. *$a \equiv 1 \pmod{p}$ for every prime factor of M*
3. *$a \equiv 1 \pmod{4}$ if M is a multiple of 4.*

For multiplicative generators, we can use the following fact (Ripley (1987), p.21).

Proposition 2 *A multiplicative generator has period $M-1$ only if M is prime. Then the period divides $M-1$, and is $M-1$ if and only if a is a primitive root, that is, $a \neq 0$ and $a^{\frac{M-1}{p}} \not\equiv 1 (mod\ M)$ for each prime factor p of $M-1$.*

Popular examples would be $M = 2^{32}, a = 69,069, b = 1$, or the *Learmonth-Lewis generator* $M = 2^{31} - 1, a = 7^5 = 16,807, b = 0$. The S-PLUS function `congrval` uses $a = 69,069, b = 0, M = 2^{32}$. A rather unpopular example is $M = 2^{31}, a = 2^{16} + 3, c = 0$, see Ripley (1987), p.23-24.

Note that successive numbers generated by these methods would always be correlated. Depending on the choice of parameters the correlation could be quite strong. Successive values all lie on a lattice, see Marsaglia (1968), and the goal is to choose a such that the lattice is rather evenly spread, and that the points are not too far apart. Shuffling can also help.

2.1.2 Shift-register generators

For *shift-register* generators, the set E is the set of binary $k-$vectors $X = (X_1, X_2, \ldots, X_k)$, and f is a linear transformation $f(x) = xT$, with T a binary $k \times k$ matrix and all calculations being carried out *mod*2. This is convenient as it corresponds to the exclusive "or " (EOR). An example is

$$X_n = a_n X_{n-1} + a_2 X_{n-2} + \cdots + a_k X_{n-k} \quad d(\text{mod } 2),$$

where $a_1, \ldots, a_k \in \{0, 1\}$. Then put

$$U_n = 0.X_{n-1} \ldots X_{n-k}.$$

Very customary is to use

$$X_n = X_{n-p} + X_{n-q} \pmod 2;$$

here, $1 \leq q < p$. Choices of (p, q) that give the maximal period $2^p - 1$ are given in Ripley (1987), p.27.

The *Tausworthe* generator, suggested by Tausworthe (1965), uses L-bit binary fractions taken t apart:

$$U_n = \sum_{s=1}^{L} 2^{-s} X_{nt+s} = 0.X_{nt+1}X_{nt+2}\ldots X_{nt+L},$$

that is, L-bit binary fractions taken t apart. Here, $0.X_{nt+1}X_{nt+2}\ldots X_{nt+L}$ is a binary representation. The parameter $t \geq L$ is called the *decimation*, and if t and $2^p - 1$ are relatively prime, then the decimation is said to be *proper* and the sequence attains its full period.

Typical values would be $p = 33, q = 13, t = L = 32$, having period $2^{33} - 1$.

Recently the Mersenne Twister, developed by Matsumoto and Nishimura (1998), has become rather popular. It is based on a matrix version of a shift-register generator.

2.1.3 Lagged-Fibonacci generators

Here, E is the set of r-vectors with elements in some finite set. The function f is defined by

$$f(X_1, X_2, \ldots, X_r) = (X_2, X_3, X_4, \ldots, X_r, X_1 + X_{r+1-s}) (\text{mod } M).$$

An example is

$$X_n = X_{n-607} - X_{n-243} (\text{mod } 2^{32}).$$

The advantage of lagged-Fibonacci generator is that their period is considerably larger than the period of congruential generators. The above generator, for example, has period about 2^{607+32}.

L'Ecuyer (1999) showed that the combined linear multiple recursive generator

$$\begin{aligned} X_{1,n} &= (a_{1,1}X_{1,n-1} + \cdots + a_{1,k}X_{1,n-k} \pmod{m_1} \\ X_{2,n} &= (a_{2,1}X_{2,n-1} + \cdots + a_{2,k}X_{2,n-k} \pmod{m_2} \\ U_n &= \left(\frac{X_{1,n}}{m_1} - \frac{X_{2,n}}{m_2}\right) \pmod 1 \end{aligned}$$

has good properties for

$$\begin{aligned} k &= 3 \\ m_1 &= 2^{32} - 209 \\ m_2 &= 2^{32} - 22,853 \\ (a_{1,1}, a_{1,2}, a_{1,3}) &= (0; 1,403,480; -810,728) \\ (a_{2,1}, a_{2,2}, a_{2,3}) &= (527,612; 0; -1,370,589) \end{aligned}$$

It has two main cycles of length $\approx 2^{191}$; the lattice test is satisfactory for tuples of length at most 48.

Lagged-Fibonacci Generators have been refined by *subtract-with-borrow* generators, see Marsaglia and Zaman (1991). These are of the form

$$X_n = X_{n-s} - X_{n-r} - b (\text{mod } M).$$

Here, b is a "borrow" flag. The iterating function here is

$$\begin{aligned} &f(x_1, x_2, \ldots, x_r, b) \\ &= \begin{cases} (x_2, \ldots, x_r, x_{r+1-s} - x_1 - b, 0) & \text{if } x_{r+1-s} - x_1 - b \geq 0 \\ (x_2, \ldots, x_r, x_{r+1-s} - x_1 - b + M, 1) & \text{if } x_{r+1-s} - x_1 - b < 0. \end{cases} \end{aligned}$$

These have much longer periods, for example, with $M = 32, s = 607, r = 243$ the above generator has period about $2^{607 \times 32}$. Unfortunately, also this generator has the problem of falling mainly on the planes; indeed, Tezuka, L'Ecuyer and Couture (1993) showed that all triples (X_n, X_{n-s}, X_{n-r}) lie on only two planes in $[0,1]^3$.

There is no uniformly best pseudo-random number generator available. The Mersenne Twister seems to display good properties. Combining generators is also a good choice. For example, one could add the binary output of two generators mod 2 and use that as a new random number. Or one could have two random number generators and a third one to switch between the two. Another option is shuffling the output. There are nonlinear generators available, displaying more favorable statistical properties, but unfortunately they are still rather slow, see L'Ecuyer (1998, 2002).

2.2 Testing randomness

A good RNG should produce an output which does not differ significantly from that of a (memoryless and fair) monkey hitting keys on a numeric keyboard. Given a pseudo-random number generator, statistical tests are advisable. Here are some possibilities.

2.2.1 Testing for independence

The *gap test* is based on the following observation. Choose two numbers, $\alpha \leq \beta \in [0,1]$, say, and record the lengths of the subsequence lying between occurrences of the same sequence within $[\alpha, \beta]$. If a sequence $X_1, X_2, \ldots X_n$ was chosen uniformly and independently, then the distribution of the gap length K should be geometric with probability of success

$$P(\alpha \leq U \leq \beta) = \beta - \alpha,$$

where U denotes a $\mathcal{U}([0,1])$ random variable. Under the independence assumption, successive gap lengths are independent, and a Chisquare test for independence can be used for testing whether the gap lengths are indeed independent. This assumes that $U \sim \mathcal{U}([0,1])$; indeed the gap test can be used for other distributions as well, with a modified probability of success.

The *run test* follows a related idea. Record the length of *runs*, these are monotonically increasing subsequences of $X_1, X_2, \ldots X_n$. For example, $(3,1,4,1,5,9,2,6,5,3,5) = 3|14|159|26|5|35$ has 2 runs of length 1, 3 runs of length 2, and one run of length 3. Under the null hypothesis that the observations are independent, the run lengths are independent, with expectation

$$\mathbf{E}(\text{ number of runs with length } k) = \frac{(n+1)k}{(k+1)!} - \frac{k-1}{k!}, \quad k = 1, \ldots, n.$$

A *permutation test* would divide $X_1, X_2, \ldots X_n$ into blocks of length t, say; $(X_1, \ldots, X_t), (X_2, \ldots, X_{t+1})$, and so on. Under the null hypothesis of independence, all $t!$ different orderings should be equally likely, and a Chisquare test can be used to test this.

There are also tests based on *return time analysis*, see (*Wegenkittl (1999)*

2.2.2 Testing for the distribution

For this we could easily use the Kolmogorov-Smirnov test; if

$$F_n(x) = \frac{1}{n}\sum_{i-1}^{n} \mathbf{1}(X_i \leq x)$$

denotes the empirical distribution function of the observations, and F is the cumulative distribution function of the true distribution (which is assumed to be continuous), then the Kolmogorov-Smirnov statistic

$$\sup_x |F_n(x) - F(x)|$$

should be small if the distribution is indeed uniform, and larger otherwise. For $\mathcal{U}([0,1])$, we have the special case $F(x) = x, \quad 0 \leq x \leq 1$. The distribution of this test statistic is not easy to determine; it can be shown (see Shorack (2000), p. 316) that

$$\lim_{n\to\infty} \mathbf{P}(\sqrt{n}\sup_x |F_n(x) - x| > \lambda) = 2\sum_{k=1}^{\infty}(-1)^{k+1}\exp(-2k^2\lambda^2).$$

Instead, one could also bin the observations and use a Chisquare test. Make a histogram of d equal size bins; then, under H_0, the probability for each bin is $\frac{1}{d}$. Use a Chisquare test with $d-1$ degrees of freedom.

Another alternative is the *Maximum-of-t Test.* Divide the sequence into n groups of t elements each, and calculate the maximum of each group. Under H_0 these maxima should follow the cumulative distribution function $F(x) = x^t, \quad 0 \leq x \leq 1$. Use a Kolmogorov-Smirnov test.

For testing uniformity of k-vectors, under the null hypothesis of uniformity, the k-vectors $(U_1, \ldots, U_k), (U_{k+1}, \ldots, U_{2k}) \ldots$ should be independent and identically uniformly distributed on the cube $[0,1]^k$. Thus divide $[0,1]^k$ into d subcubes of same size, count the number of observations in each cube, and use a Chisquare test.

For testing for independence when uniformity is assumed, there is the *Serial Correlation Test.* Let $U_0, U_1, \ldots, U_{n-1}$ and $V_0, V_1, \ldots, V_{n-1}$ be two series. Then a correlation coefficient between the two series can be defined by

$$C = \frac{n\sum_j U_j V_j - \sum_j U_j \sum_j V_j}{\sqrt{\left(n\sum_j U_j^2 - (\sum_j U_j)^2\right)\left(n\sum_j V_j^2 - (\sum_j V_j)^2\right)}}.$$

Let $U_0, U_1, \ldots, U_{n-1}$ be uniform, and put $V_j = U_{(j+1) mod n}, j = 0, \ldots, n-1$. Then

$$C = \frac{n(U_0U_1 + U_1U_2 + \cdots + U_{n-1}U_0) - (U_0 + U_1 + \cdots + U_{n-1})^2}{n(U_0^2 + U_1^2 + \cdots + U_{n-1}^2) - (U_0 + U_1 + \cdots + U_{n-1})^2}.$$

The exact distribution for C is not known; empirically "good" values of C lie between $\mu_n - 2\sigma_n$ and $\mu_n + 2\sigma_n$ 95% of the time. Here. $\mu_n = -\frac{1}{n-1}$, $\sigma_n = \frac{1}{n-1}\sqrt{\frac{n(n-3)}{n+1}}$, for $n > 2$.

Recently the *Collision test* has been advertised, (*Knuth (1997)*). Cut $[0,1)$ into k equal intervals, generate n points independently in $[0,1)$, and let $C =$ number of times a point falls in a box that already has a point in it. For large k, C is approximately Poisson $(\frac{n^2}{2k})$. Choose $k \approx$ equal to the period, and test for Poisson. This can be generalized to higher dimensions.

A variant of this is the *Birthday spacings test* (*L'Ecuyer and Simard (2001)*). Cut $[0,1)$ into k equal intervals, and number these boxes in natural order. Generate n points independently in $[0,1)$, and let $I_1 \leq I_2 \leq \ldots \leq I_n$ be the intervals where points fell. Consider the spacings

$$S_j = I_{j+1} - I_j, \quad j = 1, \ldots, n-1.$$

Let Y be the number of collisions between these spacings (that is, $S_{(j)} = S_{(j+1)}$). This corresponds to the birthday problem with n people and year with k days. Under H_0, Y is approximately Poisson(λ), if $\lambda = \frac{n^3}{4k}$ is small.

In (*L'Ecuyer, Simard and Wegenkittl (2002)*) these tests are studied in a more general framework. Partition $[0,1)$ into d equal segments. This generates a partition of $[0,1)^t$ into $k = d^t$ cubes of equal size. Generate $U_0, \ldots, U_{nt-1}$ random numbers, put

$$V_{ti} = (U_{ti}, \ldots, U_{ti+t-1}), \quad i = 0, \ldots, k-1,$$

and let X_j be the number of these points falling into cube $j, j = 0, \ldots, k-1$. Let

$$\lambda = \frac{n}{k}$$

denote the average number of points per cube under the null hypothesis that the data are i.i.d. $\mathcal{U}([0,1])$. Pearson's Chi-square statistic is

$$X^2 = \sum_{j=0}^{k-1} \frac{(X_j - \lambda)^2}{\lambda};$$

under the null hypothesis, X^2 is approximately $\chi^2_{(k-1)}$ distributed, when $\lambda \geq 5$, say. This can be generalized to test statistics

$$Y = \sum_{j=0}^{k-1} f_{n,k}(X_j),$$

where $f_{n,k}$ is a real valued function. For example, the function $f_{n,k}(x) = \frac{(x-\lambda)^2}{\lambda}$ gives Pearson's Chi-square statistics, the function $f_{n,k}(x) = \mathbf{1}[x = b]$ gives the number of cells with exactly b points, and $f_{n,k}(x) = (x-1)\mathbf{1}[x > 1]$ gives the number of collisions. It is calculated in (*L'Ecuyer, Simard and Wegenkittl (2002)*) that, under the null hypothesis,

$$\begin{aligned}
EY &= k\mu = \sum_{x=0}^{n} \binom{n}{x} \frac{(k-1)^{n-x}}{k^{n-1}} f(x) \\
VarY &= \sum_{x=0}^{n} \binom{n}{x} \frac{(k-1)^{n-x}}{k^{n-1}} (f(x)-\mu)^2 \\
&\quad + \sum_{x=0}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{x} \binom{n-x}{x} \frac{(k-1)(k-2)^{n-2x}}{k^{n-1}} (f(x)-\mu)^2 \\
&\quad + \sum_{x=0}^{n} \sum_{y=0}^{\min(n-x,x-1)} \binom{n}{x} \binom{n-x}{y} \frac{(k-1)(k-2)^{n-x-y}}{k^{n-1}} \\
&\quad \cdot (f(x)-\mu)(f(y)-\mu).
\end{aligned}$$

There are two asymptotic regimes to distinguish: the *sparse* case where λ is small, and the *dense* case where $\lambda > 1$. In the sparse case, the count statistic and the collision statistic will be approximately Poisson distributed, whereas in the dense case, they will be approximately normal distributed.

These tests can be extended to overlapping vectors.

Due to the periodic nature of many RNGs, for large sample sizes the null hypothesis is likely to be rejected. (*L'Ecuyer, Simard and Wegenkittl (2002)*) also give an empirical evaluation for RNGs. They conclude that all linear congruential generators and Lagged-Fibonacci shift register generators fail the above tests as soon as the sample size exceeds a few times the square root of their period length, regardless of the choice of their parameters. Thus they advice against using RNGs with small periods (*small* here meaning less than 2^{50}).

Another type of tests uses the generated numbers to approximate something that is known already, and see how well it does. An example for this is the *Monte Carlo Value for* π. Each successive sequence is used as X and Y co-ordinates within a square. If distance from zero of a point is smaller than the radius of a circle inscribed within the square, a hit is recorded. The percentage of hits approximates (very slowly) π.

Marsaglia compiled a battery of tests of randomness called DIEHARD, to be found at `http://stat.fsu.edu/pub/diehard/`. A newer version is at `http://www.helsbreth.org/random/diehard.html`.

Another battery called ENT has been developed by John Walker, see `http://www.fourmilab.ch/random/`.

Further reading

1. D.E. Knuth (1997). *The Art of Computer Programming, Volume 2: Seminumerical Algorithms.* 3*rd* ed., Addison-Wesley; Reading, Mass.

2. P. L'Ecuyer (2002). Random Numbers. In *the International Encyclopedia of the Social and Behavioral Sciences*, N. J. Smelser and Paul B. Baltes Eds., Pergamon, Oxford, 12735–12738. (A short and easy introduction to random number generation.)

3. P. L'Ecuyer (2001). Software for uniform random number generation: Distinguishing the good and the bad. *Proceedings of the 2001 Winter Simulation Conference*, IEEE Press, 95–105.

4. P. L'Ecuyer (1999). Good parameters and implementations for combined multiple recursive random number generators. *Operations Research*,**47** (1), 159–164.

5. P. L'Ecuyer (1998). Uniform random number generators. *Proceedings of the 1998 Winter Simulation Conference*, IEEE Press, 579–586.

6. P. L'Ecuyer, R. Simard, and S. Wegenkittl (2002). Sparse serial tests of uniformity for random number generators. *SIAM Journal of Scientific Computing.* To appear.

7. P. L'Ecuyer and R. Simard (2001). On the performance of birthday spacing tests with certain families of random number generators. *Mathematics and Computers in Simulation* **55**, 131–137.

8. D.H. Lehmer (1951). Mathematical methods in large-scale computing units. *Proceedings of the Second Symposium on Large-Scale Digital Calculating Machinery.* Harvard University Press, Cambridge, MA, 141–146.

9. G. Marsaglia (1968). Random numbers fall mainly in the planes. *Proc. Nat. Acad. Sci. USA*, **61**, 25–28.

10. G. Marsaglia and A. Zaman (1991). A new class of random number generators. *Ann. Appl. Probab.* **1**, 462–480.

11. M. Matsumoto and T. Nishimura (1998). Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom generator. *ACM Transactions on Modeling and Computer Simulations: Special Issue on Uniform Random Number Generation.*

12. B.D. Ripley (1990). Thoughts on pseudorandom number generators. *J. Comput. Appl. Math.* **31**, 153–163.

13. G.R. Shorack (2000). *Probability for Statisticians.* Springer.

14. R.C. Tausworthe (1965). Random numbers generated by linear recurrence modulo two. *Math. Comp.* **19**, 201–209.

15. S. Tezuka (1995). *Uniform Random Numbers: Theory and Practice.* Kluwer; Norwell, Mass.

16. S. Tezuka, P. L'Ecuyer, and R. Couture (1993). On the lattice structure of the add-with-carry and subtract-with-borrow random number generators. *ACM Transactions on Modeling and Computer Simulations* **3**, 315–331.

17. Wegenkittl, S. (1999). Monkeys, gambling, and return times: Assessing Pseudorandomness. *Proceedings of the 1999 Winter Simulation Conference*, IEEE Press, 625–631.

Some useful URLs

Pierre L'Ecuyer's home page
`http://www.iro.umontreal.ca/~lecuyer/`

Mersenne Twister home page
`http://www.math.keio.ac.jp/~matumoto/emt.html`

Marsaglia's random number CDRom at
`http://stat.fsu.edu/~geo`

P-Lab
`http://random.mat.sbg.ac.at/`

3 Generators for other distributions

Suppose we know how to generate a random variable U having $\mathcal{U}([0,1])$ distribution. How can we use this to produce a random variable X having a certain distribution function F?

3.1 The inverse transform method

Proposition 3 *Define* F^{-1} *by* $F^{-1}(u) = \min\{x : F(x) \geq u\}$. *Then, if* $U \sim \mathcal{U}([0,1])$, *the random variable* $X = F^{-1}(U)$ *has distribution function* F.

Proof. First note that $F(F^{-1}(u)) \geq u$, and $F^{-1}(F(x)) = \min\{y : F(y) \geq F(x)\} \leq x$. Thus we have

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

and

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

Example: Generating from a discrete distribution. Suppose that X has the probability mass function

$$\mathbf{P}(X = x_j) = p_j, \quad j = 0, 1, 2, \ldots; \quad \sum_j p_j = 1.$$

Let $U \sim \mathcal{U}([0,1])$. Construct X by

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

Then it is easy to check that $\mathbf{P}(X = x_j) = \mathbf{P}(\sum_{i=0}^{j-1} \le U < \sum_{i=0}^{j}) = p_j$ for $j = 0, 1, \ldots$.

Algorithm

1. Generate a random number U
2. If $U < p_0$ set $X = x_0$ and stop
3. If $U < p_o + p_1$ set $X = x_1$ and stop
   $\vdots$
4. If $U < \sum_{i=0}^{j} p_i$ set $X = x_j$ and stop
   $\vdots$

Example: exponential distribution. Suppose

$$F(x) = 1 - e^{-\lambda x}, \quad x \ge 0.$$

Then

$$F^{-1}(u) = -\frac{1}{\lambda} \ln(1-u)$$

and, if $U \sim \mathcal{U}([0,1])$,

$$F^{-1}(U) = -\frac{1}{\lambda} \ln(1-U) \stackrel{\mathcal{D}}{=} -\frac{1}{\lambda} \ln U.$$

If we want to generate from the Gamma $\Gamma(n, \lambda)$-distribution, we could sum up n independently generated exponential variables. In particular, suppose we want to generate from the χ^2_2-distribution. Note that $\chi^2_2 = \Gamma\left(1, \frac{1}{2}\right) = \exp\left(\frac{1}{2}\right)$. Thus we can generate $X \sim \chi^2_2$ by $X = -2 \ln U$.

Example: Box-Muller method for generating normal variates (in pairs). Suppose we want to generate a pair (Y_1, Y_2) of independent $\mathcal{N}(0,1)$ random variables. Let R and Θ denote the polar coordinates of (Y_1, Y_2), that is,

$$\begin{aligned} R^2 &= Y_1^2 + Y_2^2 \\ \tan\Theta &= \frac{Y_2}{Y_1}, \end{aligned}$$

so that

$$Y_1 = R\cos\Theta \text{ and } Y_2 = R\sin\Theta.$$

Note that $R^2 = Y_1^2 + Y_2^2$ has χ^2_2-distribution. Thus we can generate R^2 by $R^2 = -2\ln U_1$, where $U_1 \sim \mathcal{U}([0,1])$. Moreover, the joint density of R^2 and Θ is given by

$$f(r,\theta) = \frac{1}{2\pi} \times \frac{1}{2} e^{-\frac{r}{2}},$$

see Ross (1996), p. 73. Hence R^2 and Θ are independent, with Θ being uniformly distributed over $(0, 2\pi)$. Thus we can generate (Y_1, Y_2) by

$$\begin{aligned} Y_1 &= \sqrt{-2\ln U_1}\cos(2\pi U_2) \\ Y_2 &= \sqrt{-2\ln U_1}\sin(2\pi U_2), \end{aligned}$$

where $U_1, U_2 \sim \mathcal{U}([0,1])$. Unfortunately this is computationally not very efficient.

Warning: This algorithm, as well as all the other algorithms, assume that the underlying uniform variables are truly random. When they are not, strange effects can occur, see *Ripley* (1987), p.56-58.

Example: Poisson process. Suppose we want to generate the first n event times of a Poisson process with rate λ. Recall that the interarrival times E_i for such a process are independent and $exp(\lambda)$-distributed, hence we can use $E_i = -\frac{1}{\lambda}\ln U_i, i = 1, \ldots, n$, where $U_1, \ldots, U_n$ are i.i.d. $\mathcal{U}([0,1])$. Let $N(t)$ be the number of events by time t, and let $S_n = E_1 + \cdots + E_n$. Then

$$N(t) = n \iff S_n \le t < S_{n+1}.$$

So

$$\begin{aligned} N(t) &= \max\{n : S_n \le t\} \\ &= \max\left\{n : -\frac{1}{\lambda}\sum_{i=1}^{n} \ln U_i \le t\right\} \\ &= \max\left\{n : \sum_{i=1}^{n} \ln U_i \ge -\lambda t\right\} \\ &= \max\left\{n : \ln(U_1 \cdots U_n) \ge -\lambda t\right\} \\ &= \max\left\{n : U_1 \cdots U_n \ge e^{-\lambda t}\right\}. \end{aligned}$$

Thus we generate successively $\mathcal{U}([0,1])$ random numbers until their product falls below $e^{-\lambda t}$, and then N equal to 1 less than the number of random numbers required, $N(t) = \min\{n : U_1 \cdots U_n < e^{-\lambda t}\} - 1$.

Algorithm

1. Set $N = 0, P = 1$
2. Repeat. Generate U_i, $P = P \times U_i, N = N + 1$ until $P < e^{-\lambda t}$
3. $X = N - 1 \sim Poisson(\lambda t)$.

Example: Random permutation. Suppose we want to create a random permutation of $\{1, \ldots, n\}$, so that all orderings are equally likely. We could first choose one of $1, \ldots, n$ at random, and put it in position 1. Then choose one of the remaining $n - 1$ numbers, put it in position 2, and so on. Note, though, that we do not have to consider exactly which of the numbers remain to be positioned. Starting with an initial ordering $P_1, P_2, \ldots, P_n$, we pick one of the positions $1, \ldots, n$ at random and then interchange the number in that position with position n. Now we randomly choose one of the positions $1, \ldots, n - 1$ and interchange the number in this position with the one in position $n - 1$, etc.

Example. Suppose $n = 4$, and we start with the permutation $(4, 3, 1, 2)$. We pick 3, and interchange positions 3 and 4, yielding $(4, 3, 2, 1)$. In the next step, suppose we pick 2. Thus we interchange positions 2 and 3, yielding $(4, 2, 3, 1)$. Lastly, suppose we pick 2, nothing to exchange; we have obtained $(4, 2, 3, 1)$.

Algorithm

1. Let $(P_1, \ldots, P_n)$ be any permutation of $\{1, \ldots, n\}$
2. Set $k = n$
3. Generate $U \sim \mathcal{U}([0,1])$, let $I_k = Int[kU] + 1$
4. Interchange the values of P_{I_k} and P_k
5. Let $k = k - 1$ and if $k > 1$ go to Step 3
6. $(P_1, \ldots, P_n)$ is the desired random permutation.

This procedure can also be used to create random subsets, such as a simple random sample of size n from a population of N individuals.

Problem: Sometimes F^{-1}, and indeed F, are not explicitly available; an example is the Gamma $\Gamma(\alpha, \lambda)$ distribution with general α. Another example are multivariate distributions. See also: exact sampling, next term.

3.2 The acceptance-rejection method

Suppose we want to simulate from a distribution F with density f, where F^{-1} is difficult to calculate. The idea is to start from a random variable Y with a density $g(x)$ which is easily simulated and has the property $f(x) \leq Cg(x)$, where $C < \infty$ is a constant. Given $Y = x$, one accepts Y and let $X = Y$ with probability $\frac{f(x)}{Cg(x)}$. Otherwise a new Y is generated, and one continues until eventual acceptance. The function g is also called an *envelope function*.

Algorithm

1. Generate Y from the density $g(x)$
2. Generate $U \sim \mathcal{U}([0,1])$
3. If $U \leq \frac{f(Y)}{Cg(Y)}$ let $X = Y$; this is called *acceptance*. Otherwise (*rejection*) return to Step 1.

Note that then

$$\begin{aligned}
\mathbf{P}(X \in dx) &= \mathbf{P}(Y \in dx | \text{acceptance}) \\
&= \frac{\mathbf{P}(Y \in dx; \text{acceptance})}{\mathbf{P}(\text{acceptance})} \\
&= \frac{g(x) \cdot f(x)/(Cg(x))}{\int_{-\infty}^{\infty} g(y)f(y)/(Cg(y))dy} dx \\
&= \frac{f(x)}{\int_{-\infty}^{\infty} f(y)dy} dx \\
&= f(x)dx,
\end{aligned}$$

thus X has the desired density.

How many runs would we need to accept a value? Let Z be the number of attempts until we accept an X, and let $p = \mathbf{P}(\text{acceptance})$ at each step. Then $Z \sim Geometric(p)$. Note that

$$\begin{aligned}
p = \mathbf{P}(\text{acceptance}) &= \int_{-\infty}^{\infty} \frac{f(y)}{Cg(y)} g(y)dy \\
&= \frac{1}{C}.
\end{aligned}$$

Thus the expected number of attempts needed to get a new X is $\mathbf{E}Z = \frac{1}{p} = C$. Hence we want C to be small.

Example. Suppose we want to sample from a density f on $(0,1)$ that is bounded by f_{max}. Choose

$$g(x) = 1, \quad 0 < x < 1$$

, and generate $Y \sim \mathcal{U}([0,1])$. Generate $U \sim \mathcal{U}([0,1])$, and accept Y when $U_2 \leq \frac{f(Y)}{f_{max}}$; reject and restart otherwise.

Example. Suppose we want to sample from

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

Choose $g(x) = 1, 0 < x < 1$. Differentiating yields that

$$\frac{f(x)}{g(x)} \leq 20 \times \frac{1}{4} \times \left(\frac{3}{4}\right)^3 = \frac{135}{64} = C.$$

Thus generate $Y, U_2 \sim \mathcal{U}([0,1])$. If $U_2 \leq \frac{64}{135} \times 20Y(1-Y)^3$ then stop, set $X = Y$; otherwise sample again.

Example: Generating a normal variable. Suppose we want to generate $Z \sim \mathcal{N}(0,1)$, that is,

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}, \quad -\infty < x < \infty.$$

Then $|Z|$ has as density function

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

Use $g(x) = e^{-x}, \quad x \geq 0$. We have

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

Calculus shows that

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

Then

$$\frac{f(x)}{Cg(x)} = e^{x - \frac{x^2}{2} - \frac{1}{2}} = e^{-\frac{(x-1)^2}{2}}.$$

Algorithm.

1. Generate Y, an exponential random variable with rate 1
2. Generate $U \sim \mathcal{U}([0,1])$
3. If $U \leq e^{-\frac{(Y-1)^2}{2}}$ set $X = Y$, go to Step 4; otherwise return to Step 1
4. Generate $U \sim \mathcal{U}([0,1])$ and set

$$Z = \begin{cases} X & \text{if } U \leq \frac{1}{2} \\ -X & \text{if } U > \frac{1}{2} \end{cases}$$

This algorithm can be simplified, see Ross (1996), p.71, to yield

Algorithm.

1. Generate Y_1, an exponential random variable with rate 1
2. Generate Y_2, an exponential random variable with rate 1
3. If $Y_2 - \frac{(Y_1-1)^2}{2} > 0$ set $X = Y$, go to Step 4; otherwise return to Step 1
4. Generate $U \sim \mathcal{U}([0,1])$ and set

$$Z = \begin{cases} X & \text{if } U \leq \frac{1}{2} \\ -X & \text{if } U > \frac{1}{2} \end{cases}$$

For a $\mathcal{N}(\mu, \sigma^2)$-variable just take $\sigma Z + \mu$.

Example: Marsaglia's polar method for the normal distribution. To construct Y_1 and Y_2 as independent $\mathcal{N}(0,1)$-variables we employ an idea related to the Box-Muller method, where we used

$$\begin{aligned} Y_1 &= \sqrt{-2\ln U_1}\sin(2\pi U_2) \\ Y_2 &= \sqrt{-2\ln U_1}\cos(2\pi U_2), \end{aligned}$$

where $U_1, U_2 \sim \mathcal{U}([0,1])$. Instead of simulating all the angles, we contain the unit circle in a unit box and use the acceptance-rejection method. First generate independent $V_1, V_2 \sim \mathcal{U}(-1,1)$, (by setting $V = 2U - 1$) so that (V_1, V_2) is uniformly distributed over $[-1,1]^2$. Let S and Θ denote the polar coordinates of (V_1, V_2), that is,

$$\begin{aligned} S^2 &= V_1^2 + V_2^2 \\ \tan\Theta &= \frac{V_2}{V_1}. \end{aligned}$$

Let R have the conditional distribution of S^2 given that $S^2 \leq 1$. Then (see Ross (1996)), R^2 and Θ are independent, with $R^2 \sim \mathcal{U}([0,1])$ and $\Theta \sim$

$\mathcal{U}(0, 2\pi)$. Since Θ is a uniformly chosen angle, we can generate the sine and the cosine of Θ by setting

$$\begin{aligned} \sin\Theta &= \frac{V_2}{R} = \frac{V_2}{\sqrt{V_1^2 + V_2^2}} \\ \cos\Theta &= \frac{V_1}{R} = \frac{V_1}{\sqrt{V_1^2 + V_2^2}} \end{aligned}$$

Now set

$$\begin{aligned} Y_1 &= \sqrt{-2\ln U}\frac{V_2}{\sqrt{V_1^2 + V_2^2}} \\ Y_2 &= \sqrt{-2\ln U}\frac{V_1}{\sqrt{V_1^2 + V_2^2}}, \end{aligned}$$

where $U \sim \mathcal{U}([0,1])$. Indeed, $R^2 \sim \mathcal{U}([0,1])$, and we could use $S = R^2 = V_1^2 + V_2^2$. This gives the following algorithm for generating a pair (Y_1, Y_2) of independent $\mathcal{N}(0,1)$ random variables.

Algorithm

- Generate $U_1, U_2 \sim \mathcal{U}([0,1])$ independent
- Put $V_1 = 2U_1 - 1$, $V_2 = 2U_2 - 1, S = V_1^2 + V_2^2$
- If $S > 1$ return to Step 1
- Put

$$\begin{aligned} Y_1 &= \sqrt{\frac{-2\ln S}{S}}V_2 \\ Y_2 &= \sqrt{\frac{-2\ln S}{S}}V_1. \end{aligned}$$

Example: The Gamma distribution. Suppose we want to generate from the Gamma $\Gamma(\alpha, \lambda)$-distribution, $\alpha, \lambda > 0$. In general, this distribution has no simple closed form for which we could find an inverse; hence it still poses a problem. First note that given $X \sim \Gamma(\alpha, 1)$ we can obtain a $\Gamma(\alpha, \lambda)$ variable

by putting $Y = \lambda X$. So we only need to worry about $\Gamma(\alpha, 1)$, and indeed for $\alpha \neq 1$. For $\alpha < 1$ a good method seems to be the algorithm in Dagpunar (1988), p.109, which is based on an approach by Ahrens and Dieter, combining acceptance-rejection and inversion. For $\alpha > 1$, the algorithm by Best based on acceptance-rejection based on a student variate is recommendable, see Dagpunar (1988), p.111.

3.3 The composition method

Suppose we want to generate from a mixture distribution, with density

$$f = \pi_1 f_1 + \cdots + \pi_k f_k$$

where $\pi \geq 0, \sum_i \pi_i = 1$, and each f_i is a probability density. Then pick i with probability π_i, and generate from f_i.

Example. The double-exponential distribution. This distribution has density

$$f(x) \;=\; \frac{1}{2}e^x \mathbf{1}(x < 0) + \frac{1}{2}e^{-x}\mathbf{1}(x > 0),$$

Algorithm

- Generate $U_1, U_2 \sim \mathcal{U}([0,1])$ independent
- If $U_1 \leq \frac{1}{2}$ let $X = \ln U_2$
- If $U_1 > \frac{1}{2}$ let $X = -\ln U_2$.

3.4 Ratio of uniforms method

Suppose (U, V) is a uniformly distributed point within the unit disc. Then the ratio $\frac{U}{V}$ has the Cauchy distribution (**Exercise**). Thus a simple way of sampling from the Cauchy distribution is given by the following algorithm.

Algorithm

1. Generate $U_1, U_2 \sim \mathcal{U}([0,1])$, independent

2. Let $V = 2U_2 - 1$

3. If $U_1^2 + V^2 < 1$ let $X = \frac{V}{U_1}$, otherwise return to Step 1.

In general, the ratio of uniforms method is based on the acceptance-rejection method for a distribution generated from the ratio of two random numbers. It relies on the following result, see Dagpunar (1988), p.60.

Proposition 4 *Let $C = \{(u, v) : 0 \leq u \leq f^{1/2}(\frac{v}{u})\}$. Suppose points with coordinates (U, V) are uniformly distributed over C. Then the density function of $\frac{V}{U}$ is $f(x)$.*

Proof. Consider a transformation $(U, V) \to (U, Z)$ where $Z = \frac{V}{U}$. The Jacobian of the transformation is U. Thus the density of (U, Z) is

$$f_{U,Z}(u, z) = \frac{u}{\int\int_C du dv} \mathbf{1}(0 \leq u \leq f^{1/2}(z)).$$

Hence the marginal density of Z is

$$\begin{aligned} f_Z(z) &= \frac{\int_0^{f^{1/2}(z)} u du}{\int\int_C du dv} \\ &= \frac{1}{2} \frac{f(z)}{\int\int_C du dv}. \end{aligned}$$

Since f_Z and f are both probability densities, it follows that $f_Z = f$. This completes the proof.

Thus, for the ratio-of-uniforms method, we need to generate numbers within C. One way of doing this is to bound the C region by a rectangle $[0, a] \times [b, c]$. To determine a, b, c note

$$\begin{aligned} 0 &\leq U \leq \sup_x f^{1/2}(x) =: a \\ x &= \frac{V}{U}; \frac{V}{x} \leq f^{1/2}(x) \\ \text{for} \quad x &\leq 0 : V \geq x f^{1/2}(x); \text{ put } b := \inf_{x \leq 0} x f^{1/2}(x) \\ \text{for} \quad x &\geq 0 : V \leq x f^{1/2}(x); \text{ put } c := \sup_{x \geq 0} x f^{1/2}(x) \end{aligned}$$

Algorithm

1. Find bounding rectangle $[0, a] \times [b, c]$ for C
2. Generate $U_1, U_2 \sim \mathcal{U}([0, 1])$, independent
3. Set $U = aU_1, V = b + (c - b)U_2$
4. If $U \leq f^{1/2}\left(\frac{V}{U}\right)$ set $X = \frac{V}{U}$, otherwise return to Step 1.

3.5 Multivariate distributions

We have already seen how to generate pairs of independent normal variates; by a linear transformation, a sample of n i.i.d. $\mathcal{N}(0, 1)$ variates can be transformed into a sample from a multivariate normal distribution. The multinomial distribution is easily simulated using the methods for discrete variables: label the cells and sample the values of the label.

In general, the inverse transform does not work directly. The acceptance-rejection method however is straightforward and useful.

Example: Bounding by product densities. Assume that we want to generate a sample $\mathbf{X}$ from a multivariate density $f(x_1, x_2, \ldots, x_d)$ on $[0, \infty)^d$ which is ortho-monotone, that is, $f(x_1, x_2, \ldots, x_d)$ is nonincreasing in $x_1, x_2, \ldots, x_d$. Then we have

$$f(x_1, x_2, \ldots, x_d) \leq \min\{f(x_1, 0, \ldots, 0), \ldots, f(0, \ldots, 0, x_d)\}.$$

Set $f_i(u) = f(0, \ldots, 0, u, 0, \ldots, 0)$ gives

$$f(x_1, x_2, \ldots, x_d) \leq \min_i f_i(x_i) \leq \prod_{i=1}^{d} f_i^{\frac{1}{d}}(x_i).$$

Let $Q_i = \int_0^\infty f_i^{\frac{1}{d}}(x)dx$.

Algorithm

1. For $1 \leq i \leq d$ generate X_i from the density $f_i^{\frac{1}{d}}(x_i)/Q_i$
2. Generate $U \sim \mathcal{U}([0, 1])$

3. If $U \prod_{i=1}^{d} f_i^{\frac{1}{d}}(X_i) \leq f(X_1, \ldots, X_d)$ let $\mathbf{X} = (X_1, \ldots, X_n)$. Otherwise return to Step 1.

Further reading

1. L. Barabesi (1993). *Random Variate Generation by Using the Ratio-of-Uniforms Method.* Università degli Studi di Siena Dipartimento di Metodi Quantitativi Collana di Pubblicazioni, Siena, 1993.

2. J. Dagpunar (1988). *Principles of Random Variate Generation.* Oxford University Press.

3. L. Devroye (1986). *Non-Uniform Random Variate Generation.* Springer, New York.

4. L. Devroye (1996). Random variate generation in one line of code. *1996 Winter Simulation Conference Proceedings*, J.M. Charnes, D.J. Morrice, D.T. Brunner, and J.J. Swain, eds., ACM, 265–272. Springer, New York.

5. L. Devroye (1997). Random variate generation for multivariate unimodal densities. *ACM Transactions on Modeling and Computer Simulation* **7**, 447–477.

6. M.E. Johnson (1987). *Multivariate Statistical Simulation.* Wiley, New York.

Luc DeVroye's web page
`http://cgm.cs.mcgill.ca/ luc/rng.html`

Wolfgang Hörmann's web page
`http://statistik.wu-wien.ac.at/staff/hoermann/publications.html`

4 Simulation design and analysis

Often in simulations one is interested in determining

$$\theta = \mathbf{E}\phi(X) = \int \psi(x)dx.$$

Here, X has density f, and $\psi(x) = \phi(x)f(x)$. We think of θ as a parameter connected with some stochastic model. To estimate θ, the model is simulated to obtain the output $X_1, \ldots, X_n$ which are such that $\theta = \mathbf{E}\phi(X_i), i = 1, \ldots, n$. Thus we can estimate θ by the so-called *raw estimate* or *crude Monte Carlo estimate*

$$\hat{\theta}_0 = \frac{1}{n}\sum_{i=1}^{n} \phi(X_i).$$

From the law of large numbers, this is an unbiased estimate of θ, and

$$Var\hat{\theta}_0 = \frac{1}{n} Var\phi(X).$$

Here we will analyze estimates $\hat{\theta}$ of θ with respect to their variance. In particular, the aim of variance reduction is to produce an alternative estimator of θ having hopefully a much smaller variance than $\hat{\theta}_0$. Note that the order of magnitude cannot be improved in general.

Example 1 *Suppose we want to estimate*

$$\theta = \int_0^1 \sqrt{1 - x^2} dx.$$

(We know that $\theta = \frac{\pi}{4}$). Then $nVar(\hat{\theta}_0)$ can be calculated to be $\frac{2}{3} - \frac{\pi^2}{16} = .0498$. This example is from Morgan (1984) and will recur.

Remark: Hit-or-miss Monte Carlo. If ψ is zero outside a finite interval (a, b) and $0 \leq \psi(x) \leq c$ for some constant c, and for all x, one could think of estimating $\theta = \int_a^b \psi(x)dx$ by simulating $(X_i, Y_i), i = 1, \ldots, n$ uniformly from the box $[a, b] \times [0, c]$ and count the number of observations that fall under the curve ψ, that is,

$$\hat{\theta}_1 = \frac{c(b-a)}{n}\sum_{i=1}^{n} \mathbf{1}(Y_i \leq \psi(X_i)).$$

This yields again an unbiased estimate of θ. It can easily be seen, see, for example, Ripley (1987), p. 121, that this *hit-or-miss Monte Carlo* method is

less efficient than $\hat{\theta}_0$. In particular,

$$\begin{aligned}
\mathbf{P}(Y_i \leq \psi(X_i)) &= \frac{1}{(b-a)c}\int_a^b \int_0^c \mathbf{1}(y_i \leq \psi(x_i))dy_i dx_i \\
&= \frac{1}{(b-a)c}\int_a^b \psi(x_i)dx_i \\
&= \frac{\theta}{(b-a)c},
\end{aligned}$$

so that

$$\begin{aligned}
Var(\hat{\theta}_1) &= \frac{\theta(c(b-a)-\theta)}{n} \\
&= \frac{c}{n}\int_a^b \psi(x)(b-a)dx - \frac{\theta^2}{n} \\
&\geq \frac{1}{n}\int_a^b \psi^2(x)(b-a)dx - \frac{\theta^2}{n} \\
&= Var\hat{\theta}_0,
\end{aligned}$$

where we took $f(x)$ to be the uniform density on $[a, b]$. Note that equality holds only if $\psi \equiv c$, in which case both variances vanish. *Hit-or-miss is always worse than crude Monte-Carlo.*

In Example 1, it can be shown that $nVar(\hat{\theta}_1) = \frac{\pi(4-\pi)}{16} \approx .1685$.

Note that hit-or-miss and crude Monte Carlo differ in replacing the indicator $\mathbf{1}(Y_i \leq \psi(X_i))$ by its conditional expectation given X_i, namely $\psi(X_i)/c$. This illustrates a general principle for variance reduction: If, at any point of a Monte Carlo simulation, we can replace an estimate by an exact value, we shall reduce the sampling error in the final result.

4.1 Stratified sampling

If ψ was piecewise constant, then we could easily estimate θ by sampling one observation each from of the intervals where ψ is constant. The idea of stratified sampling for

$$\theta = \mathbf{E}\phi(X) = \int_a^b \psi(x)dx$$

on a finite interval (a, b) is to break the interval (a, b) into pieces where ψ is approximately constant. This idea is related to stratified sampling from populations. Say, we partition

$$a = \alpha_0 < \alpha_1 < \cdots < \alpha_k = b$$

and sample n_j observations from $(\alpha_{j-1}, \alpha_j), j = 1, \ldots, k$. Let $X_{1j}, X_{2j}, \ldots, X_{n_j j}$ be i.i.d. uniform on (α_{j-1}, α_j). Then we use the crude Monte Carlo estimate

$$\hat{\theta}_j = \frac{1}{n_j} \sum_{i=1}^{n_j} \psi(X_{ij})$$

on each of the intervals, and we combine them to give

$$\hat{\theta}_s = \sum_{j=1}^{k} (\alpha_{j-1} - \alpha_j) \hat{\theta}_j.$$

Due to the independence of the components, we obtain

$$\begin{aligned} Var(\hat{\theta}_s) &= \sum_{j=1}^{k} \frac{1}{n_j} \left((\alpha_j - \alpha_{j-1}) \int_{\alpha_{j-1}}^{\alpha_j} \psi^2(x) dx - \left\{ \int_{\alpha_{j-1}}^{\alpha_j} \psi(x) dx \right\}^2 \right) \\ &= \sum_{j=1}^{k} \frac{a_j}{n_j}, \end{aligned}$$

say. If $A = \sum_{j=1}^{k} \sqrt{a_j}$, then a Lagrange multiplier argument shows that the optimal allocation is to have $n_j = \frac{n}{A} \sqrt{a_j}, j = 1, \ldots, n$. Unfortunately the a_j's are typically not available.

In Example 1, when splitting the range of integration at the point $x = \frac{1}{\sqrt{2}}$, the minimum variance obtainable is $\frac{.01169}{n}$, and this is achieved with the sample sizes in the ratio $1 : 1.249$.

4.2 Importance sampling

The idea here pushes the stratified sampling approach further: sample more frequently from those parts of the curve that display more variability. Now, for

$$\theta = \int \psi(x) dx = \int \phi(x) f(x) dx = \mathbf{E}\phi(X),$$

we would choose f non-uniform. Ideally, we would like to choose $f(x) = \frac{\psi(x)}{\theta}$, but of course θ is not available. In general, if g is another density, we can write

$$\int \psi(x)dx \;\; = \;\; \int \frac{\psi(x)}{g(x)} g(x)dx = \mathbf{E}\rho(Y),$$

where Y has density g, and $\rho(x) = \frac{\psi(x)}{g(x)}$. Suppose that Y_n are i.i.d. with density g, then put

$$\theta_g = \frac{1}{n} \sum_{j=1}^{n} \frac{\psi(Y_i)}{g(Y_i)}.$$

If a density function g can be chosen so that the random variable $\frac{\psi(X)}{g(X)}$ has a small variance, then this approach can result in a more efficient estimator of θ. Thus a good choice of g would be one that mimics the shape of ψ.

In Example 1, $g(x) = 2(1-x)$ gives $nVar(\hat{\theta}) = .1331$, whereas $g(x) = \frac{2}{3}(2-x)$ gives $nVar(\hat{\theta}) = .01339$.

Example: Exponential tilting. (See Ross (1996), p.170 ff.) If θ is very small, then exponential tilting might be useful. Let $M(t) = \int e^{tx} f(x)dx$ be the moment-generating function corresponding to the density f. Then the density

$$f_t(x) = \frac{e^{tx}}{M(t)} f(x)$$

is called a *tilted* density of f, $-\infty < t < \infty$. Similarly a tilted probability mass function can be defined. For a Bernoulli(p)-variable, for example, we have $f(x) = p^x(1-p)^{1-x}$, $M(t) = pe^t + 1 - p$ and

$$f_t(x) = \frac{1}{M(t)}(pe^t)^x(1-p)^{1-x} = \left(\frac{pe^t}{pe^t + 1 - p}\right)^x \left(\frac{1-p}{pe^t + 1 - p}\right)^{1-x}.$$

This is the probability mass function of Bernoulli($pe^t/(pe^t + 1 - p)$). Note that

$$\frac{f(x)}{f_t(x)} \;\; = \;\; M(t) \left(\frac{p}{pe^t}\right)^x \left(\frac{1-p}{1-p}\right)^{1-x} = M(t)e^{-tx}.$$

In certain situations, the quantity of interest might be the sum of independent random variables $X_1, \ldots, X_n$ with density f each. In this case, the joint

density f is the product of the one-dimensional densities. In this situation it may be useful to generate the X_i's according to their tilted densities. For instance, suppose we are interested in estimating the probability that a sum S_n of n independent Bernoulli-random variables, $X_i \sim Be(p_i), i = 1, \ldots, n$, exceed a large value a. Then

$$\theta = \mathbf{E}\mathbf{1}(S_n \geq a).$$

Thus $\phi(x_1, \ldots, x_n) = \mathbf{1}(\sum_{i=1}^n x_i \geq a)$. Now simulate Y_i according to the t-tilted Bernoulli distribution with parameter

$$p_{t,i} = \frac{p_i e^t}{1 - p_i + p_i e^t}; \quad i = 1, \ldots, n.$$

Then the importance sampling estimator of θ is

$$\begin{aligned}
\hat{\theta} &= \mathbf{1}(\sum_{i=1}^n Y_i \geq a) \prod_{i=1}^n \frac{f_i(Y_i)}{f_{i,t}(Y_i)} \\
&= \mathbf{1}(\sum_{i=1}^n Y_i \geq a) \prod_{i=1}^n M_i(t) e^{-tY_i} \\
&= \mathbf{1}(\sum_{i=1}^n Y_i \geq a) M(t) e^{-t \sum_{i=1}^n Y_i},
\end{aligned}$$

where $M(t) = \prod_{i=1}^n M_i(t)$. Since $t > 0$ it follows that $0 \leq \hat{\theta} \leq M(t) e^{-t \sum_{i=1}^n Y_i}$.
To make the bound as small as possible, choose t to minimize $M(t)e^{-at}$. It can be shown that this minimizing t^* satisfies $\frac{p_i e^t}{1 - p_i + p_i e^t} = a$.
The optimal choice of t can be approximated numerically.
For example, if $n = 20, p_i = .4, a = 16$, then $\mathbf{E}(\sum_{i=1}^n Y_i) = 20 \frac{.4e^t}{.4e^t + .6}$; this equals 16 is $t = \ln 6$. the importance sampling estimator is

$$\hat{\theta} = \mathbf{1}(\sum_{i=1}^n Y_i \geq 16) 6^{-\sum_{i=1}^n Y_i} 3^{20}.$$

It can be shown that $Var\hat{\theta} \leq 2.9131 \times 10^{-7}$, whereas $Var\hat{\theta}_0 = 3.160 \times 10^{-4}$. NOte that $\theta \approx 3.17 \times 10^{-4}$.

In general, variance reduction may or may not be obtained, depending on the choice of g.

4.3 Control variates

The idea behind control variates is formally related to regression. Suppose we have Y (perhaps $Y = \phi(X)$) and we want to estimate its mean

$$\theta = \mathbf{E}Y.$$

If Z is a related random variable with known mean μ, then put

$$W = Y - c(Z - \mu)$$

for some constant c. Then $\mathbf{E}W = \theta$ for any c. Thus, if $W_1, \ldots, W_n$ are i.i.d. with same distribution as W, then

$$\hat{\theta} = \bar{W} = \frac{1}{n}\sum_{i=1}^{n} W_i$$

is unbiased for θ. Moreover,

$$\begin{aligned} nVar\hat{\theta} &= VarW \\ &= VarY + c^2 VarZ - 2cCov(Y,Z) \\ &< VarY \quad \text{if and only if } Cov(Y,Z) > \frac{c}{2}VarZ. \end{aligned}$$

The above variance is minimized for

$$c^* = \frac{Cov(Y,Z)}{VarZ}.$$

Then we obtain

$$VarW = VarY - \frac{Cov^2(Y,Z)}{VarZ}.$$

Thus, variance reduction is always achievable by suitable choice of c whenever $Cov(Y,Z) \neq 0$.

Example 2 *(See Ross (1996), p.144-145.) Suppose we want to use simulation to compute*

$$\theta = \mathbf{E}e^U$$

for $U \sim \mathcal{U}([0,1])$. *Note that* $nVar\hat{\theta}_0 = .2402$. *A natural choice for a control variate is* U. *We then have*

$$\begin{aligned} Cov(e^U, U) &= \mathbf{E}(Ue^U) - \mathbf{E}(U)\mathbf{E}(e^U) \\ &= \int_0^1 xe^x dx - \frac{e-1}{2} = .14086. \end{aligned}$$

Moreover, $VarU = \frac{1}{12}$. *Hence, with* $c^* = 12 \times .14086$ *we have*

$$\begin{aligned} nVar\hat{\theta} &= Var(e^U) - 12 \times (.14086)^2 \\ &\approx .2402 - .2380 = .0039. \end{aligned}$$

In Example 1, with $Z = 1 - X$ as control variate, $c^* = \frac{3}{2}\pi - 4$, and $nVar\theta \approx .00752$.

Of course this method can be generalized to

$$W = Y - \sum_{i=1}^{k} c_k(Z_k - \mu_k)$$

when such $Z_1, \ldots, Z_n$ are available.

In general, c^* will not be available. One could estimate c from the experiment, but then $\bar{W} = \frac{1}{n}\sum_{i=1}^{n} W_i$ will in general not be an unbiased estimator of θ. Instead, it is better to use a pilot simulation to estimate c^*, and then use this estimated c^* for the larger simulation.

It is appealing that even when this method is not very successful, the resulting variance is never increased.

Due to the relation to standard regression analysis, often also the term *regression-adjusted control variates* is used. The similarity is formal, though: regression analysis via least squares is based upon the assumption of linear dependence (and preferably normal errors) whereas nothing like this is needed for regression-adjusted control variates.

4.4 Antithetic variates

Here the idea is to generate two (or $2n$) correlated unbiased estimators Y_1, Y_2 of θ with the same marginal distribution, described by Y, say. We then put

$$\hat{\theta} = \frac{1}{2}(Y_1 + Y_2).$$

Thus

$$Var\hat{\theta} = \frac{1}{2} VarY(1 + corr(Y_1, Y_2)).$$

If

$$corr(Y_1, Y_2) < 0$$

then we obtain a smaller variance than with independent estimators.

A standard way of generating such correlated unbiased estimators from a distribution function F is to put

$$Y_1 = F^{-1}(U), \quad Y_2 = F^{-1}(1 - U),$$

where $U \sim \mathcal{U}([0,1])$. Then the correlation is negative, following from (see Ripley (1987), p.129)

Proposition 5 *Suppose g is a monotonic function on $(0,1)$. Then*

$$corr(g(U), g(1-U)) < 0.$$

In Example 1, letting $Y_1 = \sqrt{1 - U^2}$ and $Y_2 = \sqrt{1 - (1-U)^2}$ gives $Var\hat{\theta} = .0052$. In Example 2, with $Y_1 = e^U$ and $Y_2 = e^{1-U}$, we obtain $Var\hat{\theta} = \frac{1}{2}.2420(1 - .9677) = .0039$.

4.5 Conditional Monte Carlo

In general, we have for any random variables Y, W that

$$\begin{aligned} \mathbf{E}Y &- \mathbf{E}(\mathbf{E}(Y|W)) \\ VarY &- Var(\mathbf{E}(Y|W)) + \mathbf{E}(Var(Y|W)). \end{aligned}$$

Hence $Var(\mathbf{E}(Y|W)) \leq VarY$. If we can evaluate $\mathbf{E}(Y|W)$ analytically as a function h of W, we can estimate $\theta = \mathbf{E}Y$ by

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} h(W_i),$$

where $W_1, \ldots, W_n$ are i.i.d. copies of W. If the distribution of Y is built up by mixing over values of some W, this becomes an obvious target for the technique.

Example: Suppose $W \sim Poisson(\lambda)$, and, given $W = w$, we have that $Y \sim Beta(w, w^2 + 1)$. Then

$$\mathbf{E}(Y|W = w) = \frac{w}{w^2 + w + 1}.$$

Thus, to simulate $\theta = \mathbf{E}Y$, we simulate $W_1, \ldots, W_n \sim Poisson(\lambda)$ i.i.d. and put

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} \frac{W_i}{W_i^2 + W_i + 1}.$$

Conditional Monte Carlo always provides variance reduction. The difficulty is to find W such that the conditional expectation is computable.

4.6 Isolating known components

In many cases, some parts of the expectation θ of $\phi(X)$ can be evaluated analytically. One may then attempt to organize the output analysis to that these known parts need not be simulated.

Example. Let $T_1, T_2, \ldots$ be i.i.d. and nonnegative with mean μ, and let $Z = \sup\{n : S_n \leq t\}$ be the number of renewals up to time t, where $S_n = T_1 + T_2 + \cdots + T_n$. Let $\theta = \mathbf{E}Z$. Letting $\tau = \inf\{n : S_n > t\}$, we then have $Z = \tau - 1$. By Wald's identity,

$$\mathbf{E}S_\tau = \mu \mathbf{E}\tau = \mu(\theta + 1).$$

This suggests the estimator

$$\hat{\theta}_1 = \frac{S_\tau}{\mu} - 1.$$

But we can write $S_\tau = t + \xi$, where $\xi = S_\tau - t$ is the overshoot. This yields

$$\theta = \frac{t + \mathbf{E}\xi}{\mu} - 1$$

and an alternative estimator is

$$\hat{\theta}_2 = \frac{t + \xi}{\mu} - 1.$$

For example, if the T_i are standard exponential and $t = 50$, then $Z \sim Poisson(50)$ so that $Var\hat{\theta}_1 = 50$. In contrast, since ξ is again standard exponential, $Var(\hat{\theta}_2) = 1$.

4.7 Experimental design

Many simulation experiments are designed to compare the effect of choosing different parameter values in the model. In such cases, ideas from the design of experiments can be used, see also Box *et al.* (1978). The analogue of a randomized block is a set of random numbers which can be re-used (*common random numbers*). Suppose we are simulating

$$\mathbf{E}\phi(X;\alpha),$$

where α is a parameter value, to be varied. To assess the variation between two different parameter values α_1 and α_2, we are interested in

$$\theta = \mathbf{E}\phi(X,\alpha_1) - \mathbf{E}\phi(X,\alpha_2).$$

If we use the same random numbers X for both parts, it is likely that $Cov(\phi(X,\alpha_1),\phi(X,\alpha_2) \neq 0$, and so we would obtain a variance reduction compared to choosing independent X's.

4.8 Discussion

Variance reduction techniques are typically most readily available for well structured problems. Typically, they involve a fair amount of both theoretical study of the problem and of additional programming effort. For this reason, variance reduction is most often only recommendable for large experiments.

Different variance reduction techniques used in combination may produce diminishing returns and may even conflict with each other to give adverse results.

It will generally be advantageous to break down a problem into components and to push the analytic treatment of the problem through as far as possible.

Further reading

1. S. Asmussen (1999). *Stochastic Simulation with a view towards stochastic processes.* MaPhySto, University of Aarhus. On the web at `http://www.maphysto.dk`

2. G.E.P. Box, W.G. Hunter, and J.S. Hunter (1978). *Statistics for Experimenters.* Wiley.

5 Simulation of stochastic processes

Many stochastic processes can be described as Markov chains (in discrete time) or Markov processes (in continuous time). A discrete-time Markov chain $\{X_n\}_{n\in\mathbf{N}}$ with state space S and transition matrix P can be simulated as follows: *If $X_n = s$ then select X_{n+1} from the conditional distribution of X_{n+1}, given $X_n = s$.*

Example: A simple time series model. Consider the discrete-time model

$$Z_t = \alpha Z_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2),$$

where $|\alpha| < 1$, and assume that $\{\epsilon_t\}_{t\in\mathbf{Z}}$ are independent. Then the conditional distribution of Z_t given $Z_{t-1} = z_{t-1}$ is $\mathcal{N}(\alpha z_{t-1}, \sigma^2)$, and $\{Z_t\}_{t\in\mathbf{Z}}$ is a Markov chain. (This model is called an *autoregressive process of order 1*). Thus, for any given starting point X_0, we can simulate the process by simulating normal random variables.

An alternative approach is to simulate the time until the next jump, which has a geometric distribution in the discrete-time case, and an exponential distribution in the continuous-time case.

5.1 Construction of algorithms

The design of an algorithm may involve very varied methods, and much is left to the ingenuity of the programmer. However, below is a selection of simple points to observe.

Simulation via discrete events. Often stochastic processes can be described by discrete events - the arrival of a customer, birth or death, jumps. Typically a stochastic process would involve a time variable (the amount of simulated time that has elapsed), counter variables (the number of times that certain events have occurred by time t), and system state variables. Whenever an event occurs, these variables are updated. We hence keep an "event list", which lists the nearest future events and when they are scheduled to occur. Whenever an event occurs, we reset the time and all state and counter variables and collect the relevant data.

For example, in an epidemic process with N individuals, assume that individual i gets infected at time A_i, and it recovers at time $A_i + R_i$; assume that the A_i's and the R_i's are independent. Then we would first generate $(A_1, R_1), \ldots, (A_N, R_N)$ and then run the process, which is deterministic conditioned on $(A_1, R_1), \ldots, (A_N, R_N)$. Note that no Markovian structure is required.

Example: A single-server queueing system. (See Ross (1996), p. 88-89.) Customers arrive at a queue according Poisson process with parameter λ. There is a single server; when a customer arrives, if the server is busy he/she joins the queue, and otherwise he/she enters service ("first-in first-out queue"). The amount of time it takes to serve a customer is a random variable, independent of all the other variates in the system, with distribution G.

To simulate this system, we use as variables t, the time; $N_A(t)$ the number of arrivals by time t, $N_D(t)$ the number of departures by time t, and $n(t)$ the number of customers in the system at time t. There are two types of events: arrivals and departures. The event list contains the time t_A of the next arrival and the time t_D of the departure of the customer presently in service. If no customer is presently being served, put $t_D = \infty$. The output variables are $A(i) =$ the arrival time of customer i, and $D(i) =$ the departure time of customer i. We restrict the simulation to a fixed time interval $[0, T]$; after time T, no new customers are allowed in the system, although those present will still be served. Let T_p be the time past T that the last customer departs.

Algorithm.

1. Initialize $t = N_A = N_D = n = 0$. Generate T_0 from the $exp(\lambda)$-distribution, set $t_A = T_0, t_D = \infty$

2. Let $Y \sim G$. We distinguish 4 cases:

   (a) **Case 1.** $t_A \leq t_D, t_A \leq T$. Reset $t = t_A, N_A = N_A + 1, n = n + 1$. Generate $T_t \sim exp(\lambda)$, reset $t_A = T_t$. If $n = 1$, generate $Y \sim G$ and reset $t_D = t + Y$. Collect output data $A(N_A) = t$.

   (b) **Case 2.** $t_D \leq t_A, t_D \leq T$. Reset $t = t_D, N_D = N_D + 1, n = n - 1$. If $n = 0$, reset $t_D = \infty$; otherwise, generate $Y \sim G$ and reset $t_D = t + Y$. Collect the output data $D(N_D) = t$.

(c) **Case 3.** $\min(t_A, t_D) > T, n > 0$. Reset $t = t_D, N_D = N_D + 1, n = n - 1$. If $n > 0$, generate $Y \sim G$ and reset $t_D = t + Y$. Collect the output data $D(N_D) = t$.

(d) **Case 4.** $\min(t_A, t_D) > T, n = 0$. Collect the output data $T_p = \max(t - T, 0)$.

Similarly, tandem queues or two-server queues can be included, see Ross (1996).

Arrival times. If we wish to relax the restrictive assumption that an individual has an exponentially distributed arrival time, while maintaining a Markovian structure, we can make him/her go through k stages in series before arrival, leading to a Γ_k distribution. More generally we can achieve an arrival time having any distribution whose moment-generating function is a rational function by choosing a suitable network of stages.

Non-homogeneous Poisson processes. Suppose we have a realisation of a non-homogeneous Poisson process of intensity $\lambda(t)$ on $[0, T]$ and that the total number of points sampled in a given realisation is n. Then their distribution on the interval is identical to that of a random sample of size n from the density function $f(t) = \lambda(t)/\Lambda$, where $\Lambda = \int_0^T \lambda(s)ds$. Conversely, an obvious way to generate a realisation from the process is to sample from Poisson(Λ) and choose this many points from $f(t)$.

Exponential spacings. Let $E_1, E_2, \ldots, E_{n+1}$ be independent $exp(1)$-variates. Put $S_j = \sum_{i=1}^{j} E_i$. Then $\{S_j/S_{n+1}, j = 1, 2, \ldots, n\}$ have the same distribution as an ordered sample of n $\mathcal{U}([0, 1])$- variates. This provides a neat way to generate a sample of order statistics form $\mathcal{U}([0, 1])$ and, by transformation, from any univariate distribution with F^{-1} available. This would be more efficient than taking a random sample and sorting it for sufficiently large samples since sorting takes time $O(n \log n)$.

Simulating a two-dimensional Poisson process. (See Ross (1996).) A 2-dimensional Poisson (point) process having rate $\lambda > 0$ is defined by two properties:

1. The number of points in any region of area A is $Poisson(\lambda A)$

2. The number of points occurring in disjoint regions are independent.

Suppose we want to generate a Poisson point process on the circle $C(r)$ with radius r around the origin. Then, for any a, the number of points in the circle $C(a)$ is $Poisson(\lambda\pi a^2)$. For $i \geq 1$, let R_i denote the distance from the origin to its ith nearest point. Then

$$\begin{aligned}\mathbf{P}(\pi R_1^2 > x) &= \mathbf{P}\left(R_1 > \sqrt{\frac{x}{\pi}}\right) \\ &= \mathbf{P}\left(\text{ no point in } C\left(\sqrt{\frac{x}{\pi}}\right)\right) \\ &= e^{-\lambda x}\end{aligned}$$

and, similarly,

$$\begin{aligned}&\mathbf{P}(\pi R_2^2 - \pi R_1^2 > x | R_1 = a) \\ &= \mathbf{P}\left(R_2 > \sqrt{\frac{x+\pi a^2}{\pi}} | R_1 = a\right) \\ &= \mathbf{P}\left(\text{ no point in } C\left(\sqrt{\frac{x+\pi a^2}{\pi}}\right) \setminus C(a)\right) \\ &= e^{-\lambda x}.\end{aligned}$$

Similarly one can show that, with $R_0 = 0$, the spacings $\pi R_i^2 - \pi R_{i-1}^2 \sim exp(\lambda)$, for $i \geq 1$. Due to symmetry, the angles of the Poisson points are $\mathcal{U}(0, 2\pi)$. Hence we may use the following algorithm.

Algorithm.

1. Generate $X_1, X_2, \ldots \sim exp(\lambda)$ independently, until

$$N = min(n : X_1 + \cdots + X_n > \pi r^2).$$

2. If $N = 1$: stop, there are no points in $C(r)$. If $N > 1$: put

$$R_i = \sqrt{\frac{X_1 + \cdots + X_i}{\pi}}, \quad i = 1, \ldots, N$$

Note that then $\pi R_i^2 = X_1 + \cdots + X_i$.

3. Simulate independent $U_1, \ldots, U_{N-1} \sim \mathcal{U}([0,1])$

4. the polar coordinates of the $N-1$ Poisson points are

$$(R_i, 2\pi U_i), \quad i = 1, \ldots, N$$

The above algorithm can be generalized to simulating on any smooth region.

Verification of the simulation model. To check the simulation,

- Try to debug in small subroutines
- Write the simulation as general as possible, to that a special case that has been well studied, or that can be treated analytically, can be used for a comparison
- In the testing stage of a program, make the program give as output all the random quantities it generates
- When searching for errors, use a *trace*, so that the state variable, the event list, and all the counter variables are printed out after each event occurs.

Simulation languages. In simulation models there tend to exist similar features. This has lead to the development of special purpose languages,

- GPSS: "General Purpose Simulation System"; relatively easy to learn; no facilities for program structuring
- SIMSCRIPT: Originally based on FORTRAN; comprehensive language, incorporating all that GPSS achieves, but with many other features, resulting in it being more difficult to learn
- SIMULA: ALGOL related language; designed to be extendable by classes, providing programming tools for specific needs, for example: simulation, discrete-event systems, and combined simulations
- G.A.S.P.: "Genometric Analysis Simulation Program"; FORTRAN related; specifically geared to genetic analysis

- Simview: Graphical language for PC
- SLAM II: based on FORTRAN, geared to networks, discrete event simulations, and continuous simulations
- And many more
- Both SPLUS and MATLAB have convenient routines for simulation

5.2 Markov Chain Monte Carlo methods: The Gibbs sampler

In general it is very difficult to simulate the value of a random vector $\mathbf{X}$ whose component random variables are dependent. A powerful approach is given by Markov chain Monte Carlo methods.

Let $\{X_n\}_{n\in\mathbf{N}}$ be a discrete-time Markov chain with finite state space S and transition matrix P. Recall that a distribution π is a stationary distribution of the Markov chain if

$$\pi = \pi P.$$

The Markov chain is called *reversible* if the *detailed balance equations* hold:

$$\pi(x)P(x,y) = \pi(y)P(y,x) \quad ,x,y \in S. \tag{1}$$

If a Markov chain is reversible, then it follows that it has stationary distribution π. Under suitable conditions, the distribution of X_n will tend to the stationary distribution, no matter which starting point X_0 has been chosen for the chain.

Example: The simple time series model. Consider again the discrete-time model

$$Z_t = \alpha Z_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0,\sigma^2),$$

where $|\alpha| < 1$, and $\{\epsilon_t\}_{t\in\mathbf{Z}}$ are independent. Then it can be shown that the process has stationary distribution $\mathcal{N}(0,\sigma^2/(1-\alpha^2))$, and that (1) holds.

Markov chain Monte Carlo methods build on the converse problem - given a distribution π, can we find a transition matrix (kernel) such that (1) holds? If so, then we could simulate from the Markov chain corresponding to the

transition kernel, and use the convergence to the stationary distribution in order to obtain (approximate) samples from π.

The Gibbs Sampler. A popular Markov chain Monte Carlo method is the *Gibbs sampler.* Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a random vector with probability mass function $p(\mathbf{x})$, which may only be specified up to a multiplicative constant, and suppose that we want to generate a random vector whose distribution is that of the conditional distribution of $\mathbf{X}$ given that $\mathbf{X} \in A$ for some set A.

Algorithm.

1. Let $\mathbf{x} = (x_1, \ldots, x_n)$ be a vector in A for which $p(\mathbf{x}) > 0$.
2. Let I be uniformly chosen from $\{1, 2, \ldots, m\}$.
3. If $I = i$, generate the value of a random variable X such that

$$\mathbf{P}(X = x) = \mathbf{P}(X_i = x | X_j = x_j, j \neq i)$$

4. If $X = x$ and $(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n) \in A$ then reset $x_i = x$.
5. Go to 2.

At each step, one of the variables X_i is randomly chosen, and a random variable having the conditional distribution of X_i given that $X_j = x_j, j \neq i$ is generated. If the new vector, with this value replacing x_i, is in A, then that is the next state of the underlying Markov chain; if this vector is not in A, the state remains unchanged.

Example. Let $X_1, \ldots, X_n$ be independent exponential variates, so that $X_i \sim exp(\lambda_i), i = 1, \ldots, n$, and put $S_n = \sum_{i=1}^n X_i$. Suppose we want to generate $\mathbf{X} = (X_i, \ldots, X_n)$ conditional on the event $A = \{S_n > c\}$, for some large c. That is, $\mathbf{X}$ has density function

$$f(x_1, \ldots, x_n) = \frac{1}{P(S_n > c)} \prod_{i=1}^n \lambda_1 e^{-\lambda_i x_x}, \quad \sum_i x_i > c.$$

Algorithm.

1. Let $\mathbf{x} = (x_1, \ldots, x_n)$ be such that $x_i > 0, i = 1, \ldots, n$, and $\sum_{i=1}^{n} x_i > c$.
2. Let $U \sim \mathcal{U}([0,1])$, put $I = Int[nU + 1]$.
3. If $I = i$, generate $X_i \sim exp(\lambda_i)$ conditioned on the event $X + \sum_{j \neq i} x_j > c$, i.e. $X > c - \sum_{j \neq i} x_j$. To do this, use the fact that the conditional distribution of X, given $X > c$, is the same as the distribution of $X + c$, by the memoryless property of the exponential distribution. So
   (a) Generate $Y \sim exp(\lambda_i)$
   (b) Set
   $$X = Y + (c - \sum_{j \neq i} x_j)^+$$
4. If $X = x$ then reset $x_i = x$.
5. Go to Step 2.

As the underlying Markov chain will converge to its stationary distribution, the Gibbs sampler will converge to the correct distribution. An interesting question with no clear answer is how many iterations would be needed to be sufficiently close to the target distribution. This is sometimes also discussed as *burn-in problem.*

The Gibbs sampler is a special case of

The Metropolis-Hastings algorithm. Let $b(j), j = 1, \ldots, m$ be positive numbers, and let $B = \sum_{j=1}^{m} b(j)$. Suppose that B is difficult to calculate, and that we want to simulate a random variate X with

$$\mathbf{P}(X = j) = \pi(j) = \frac{b(j)}{B}, \quad j = 1, \ldots, m$$

The idea is to simulate a Markov chain whose limiting probabilities are the $\pi(j)$.

One starts with an acceptance-rejection idea. Suppose you have simulated X_n. If $X_n = i$, then choose X such that $\mathbf{P}(X = j) = q(i,j)$, and put

$$\begin{aligned} X_{n+1} &= j \quad \text{with probability } \alpha(i,j) \\ X_{n+1} &= 1 \quad \text{with probability } 1 - \alpha(i,j). \end{aligned}$$

with some $\alpha(i,j)$ to be determined. Then

$$\begin{aligned} P_{i,j} &= q(i,j)\alpha(i,j), \quad j \neq i \\ P_{i,i} &= q(i,i) + \sum_{k \neq i} q(i,k)(1 - \alpha(i,k)) \end{aligned}$$

is a Markovian transition matrix. This Markov chain will have stationary distribution π if

$$\pi(i)P_{i,j} = \pi(j)P_{j,i}, \quad j \neq i,$$

i.e.

$$\pi(i)q(i,j)\alpha(i,j) = \pi(j)q(j,i)\alpha(j,i), \quad j \neq i, \tag{2}$$

Taking

$$\begin{aligned} \alpha(i,j) &= \min\left(\frac{\pi(j)q(j,i)}{\pi(i)q(i,j)}, 1\right) \\ &= \min\left(\frac{b(j)q(j,i)}{b(i)q(i,j)}, 1\right) \end{aligned}$$

satisfies (2). To see this, note that if $\alpha(i,j) = \frac{b(j)q(j,i)}{b(i)q(i,j)}$ then $\alpha_j, i) = 1$ and vice versa.

Algorithm.

1. Choose an irreducible Markov transition matrix $Q = (q(i,j))_{i,j=1,\ldots,m}$. Also choose $k \in \{1, 2, \ldots, m\}$.

2. Let $n = 0$ and $X_0 = k$.

3. Generate a random variable X such that $\mathbf{P}(X = j) = q(X_n, j)$ and generate $U \sim \mathcal{U}([0,1])$

4. If
$$U < \frac{b(X)q(X, X_n)}{b(X_n)q(X_n, X)}$$
then put $S = X$; else $S = X_n$

5. $n = n + 1, X_n = S$.

6. Go to 3.

Further reading

1. S. Coles. Computer-intensive statistics. Lecture notes at `http://www.statistics.bristol.ac.uk/ masgc/ast/notes.ps`

2. P. Diaconis and S. Holmes (1995). Three examples of Monte-Carlo Markov Chains: At the Interface between Statistical Computing, Computer Science, and Statistical Mechanics. In *Discrete Probability and Algorithms* (D. Aldous, P. Diaconis, J. Spencer, and J.M. Steele, eds). Springer-Verlag, 43–56.

3. S. Geman and D. Geman (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**, 721–724.

4. P.J. Green (2000). A primer on Markov chain Monte Carlo. In *Complex Stochastic Systems,* O.E. Barndorff-Nielsen, D.R. Cox and Claudia Klüppelbert, eds. Chapman and Hall: Boca Raton etc. pp. 1–62.

5. N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller (1953). Equations of state calculations by fast computing machines. *J. Chem. Phys.* **21**, 1087–1092.

6. E.A. Thompson (2000). Monte Carlo methods on genetic structures. In *Complex Stochastic Systems,* O.E. Barndorff-Nielsen, D.R. Cox and Claudia Klüppelbert, eds. Chapman and Hall: Boca Raton etc., pp/ 175–218.

6 Statistical Inference

6.1 Monte Carlo tests

The Monte Carlo test, attributed to Dwass (1957) and Barnard (1963), is an exact procedure of virtually universal application and correspondingly widely used. We only need to be able to simulate a random sample $T_{01}, T_{02}, \ldots$ from the distribution F_0 determined by the null hypothesis. We assume that F_0 is continuous, and, without loss of generality, that we reject H_0 for large values of T_0. Then, provided that $\alpha = \frac{m}{n+1}$ is rational, we can proceed as follows.

1. Observe the actual value t^* calculated from the data
2. Simulate a random sample of size n from F_0
3. Order set $\{t^*, t_{01}, \ldots, t_{0n}\}$
4. Reject H_0 if the *rank* of t^* in this set (in decreasing order) is $\geq m$.

The basis of this test is that, under H_0, the random variable T^* has the same distribution as the remainder of the set and so, by symmetry,

$$\mathbf{P}(t^* \text{ is among the largest } m \text{ values }) = \frac{m}{n+1}.$$

The procedure is exact however small n might be. However, increasing n increases the power of the test. The question of how large n should be is discussed by Marriott (1979), see also Hall and Titterington (1989). A reasonable rule is to choose n such that $m \geq 5$. Note that we will need more simulations to test at smaller values of α.

An alternative view of the procedure is to count the number M of simulated values $> t^*$. It is easy to see that $\hat{P} = \frac{M}{n}$ is an unbiased estimator of the true significance level P achieved by the data, i.e.

$$P = \mathbf{P}(T_0 > t^* | H_0).$$

In discrete data, we will typically observe ties. We can break ties randomly, then the above procedure will still be valid.

Unfortunately this test does not lead directly to confidence intervals.

6.2 Confidence intervals

The relationship between significance tests and confidence intervals suggests that we can construct the latter by simulation, but the problem is more difficult, as we need to estimate critical values for T with sufficient precision. For an exact interval we would need infinite precision at the critical values.

For a confidence interval for a population mean we would need an estimate for the population variance also, for example. One method is to choose first an acceptable value d for the standard deviation of the estimator, and continue to generate random variates until this number k is such that $S/\sqrt{k} < d$, where S is the estimated standard deviation.

More precisely, suppose that $\hat{\theta}$ is an unbiased estimator of θ, with distribution function F_θ. Let θ^* be a sample from $F_{\hat{\theta}}$. We want to make inference about the variation of $\hat{\theta}$ around θ by studying the variation of θ^* around $\hat{\theta}$.

Example. (See Ripley (1986), p.176.) Suppose we sample from a location-family model, such that

$$\hat{\theta} - \theta \sim F_0; \tag{3}$$

then

$$\theta^* - \hat{\theta} \sim F_0.$$

A $(1-\alpha)$ two-sided confidence interval for θ could then be constructed as follows. Put

$$\begin{aligned} L &= F_\theta^{-1}\left(\frac{\alpha}{2}\right) = \hat{\theta} + F_0^{-1}\left(\frac{\alpha}{2}\right) \\ R &= F_\theta^{-1}\left(1-\frac{\alpha}{2}\right) = \hat{\theta} + F_0^{-1}\left(1-\frac{\alpha}{2}\right) \end{aligned}$$

Then an exact $(1-\alpha)$ two-sided confidence interval for θ is

$$\begin{aligned} \theta &\in \left\{\hat{\theta} - F_0^{-1}\left(1-\frac{\alpha}{2}\right), \hat{\theta} - F_0^{-1}\left(\frac{\alpha}{2}\right)\right\} \\ &= (2\hat{\theta} - R, 2\hat{\theta} - L). \end{aligned}$$

This is only exact if (3) holds. Otherwise it could serve as an approximation. For more discussion, see Ripley (1986).

If we estimate a parameter using simulations from a Markov chain, then this dependence has to be taken into account.

Example: The simple time series model. Consider again the discrete-time model

$$Z_t = \alpha Z_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2),$$

where $|\alpha| < 1$, and $\{\epsilon_t\}_{t \in \mathbf{Z}}$ are independent. Suppose we want to estimate $\theta = \mathbf{E}Z_t$. We could do this by simulating $Z_1, \ldots, Z_n$ from the Markov chain, starting with the stationary distribution. and use $\hat{\theta} = \bar{X}$. Then

$$Var(X_i) = \frac{\sigma^2}{1 - \alpha^2} =: v^2$$

and it can be calculated that

$$Cov(X_i, X_{i+k}) = v^2 \alpha^k \neq 0,$$

so that

$$Var(\bar{X}) = \frac{v^2}{n^2} \left(n + \sum_{i=1}^{n-1} 2(n-i)\alpha^i \right) \neq \frac{v^2}{n}.$$

6.3 Output analysis

Output analysis is concerned with the analysis of the results of simulation experiments. This may involve special difficulties either because the experiments are large and it is difficult to infer general conclusions from the results, or, more particularly, because the results are correlated. The latter will happen, for example, in the simulation of stochastic processes.

To some extent the methods of time series analysis will be appropriate for the output from one-dimensional stochastic processes. The problems fall into two groups according to whether we are considering the transients of a process or the equilibrium state. For the former - for example, the first passage time from a particular state - we will usually obtain a single observation from each of a number of independent runs of a process. The more challenging problems are associated with processes in or near to equilibrium. One major problem is concerned with then we can consider a process to have reached equilibrium. To treat this problem, one could, for example,

- estimate and use the correlation structure (time series methods)

- take the means of successive blocks of the observations, exploiting the fact that these will be more nearly uncorrelated.

Further reading

1. G. Barnard (1963). Contribution to the discussion of Bartlett's paper. *J. Roy. Statist. Soc.***B**, 294.

2. M. Dwass (1957). Modified randomization tests for nonparametric hypotheses. *Ann. Math. Stat.* **28**, 181–187.

3. F. Marriott (1979). Barnard's Monte Carlo tests: how many simulations? *Appl. Statist.* **28**, 75–77.

4. P. Hall and D.M. Titterington (1989). The effect of simulation order on level accuracy and power of Monte Carlo tests. *J. Roy. Statist. Soc.* **B**, 459.