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.