Monte Carlo Methods

Lecture notes for MAP001169

Based on Script by Martin Sköld

adopted by Krzysztof Podgórski

Contents

Part I

Simulation and Monte-Carlo Integration

Chapter 1

Simulation and Monte-Carlo integration

1.1 Issues in simulation

Whatever the application, the role of simulation is to generate data which have (to all intents and purposes) the statistical properties of some specified model. This generates two questions:

1. How to do it; and

2. How to do it efficiently.

To some extent, just doing it is the priority, since many applications are sufficiently fast for even inefficient routines to be acceptably quick. On the other hand, efficient design of simulation can add insight into the statistical model itself, in addition to CPU savings. We'll illustrate the idea simply with a well–known example.

1.2 Buffon's Needle

We'll start with a simulation experiment which has intrinsically nothing to do with computers. Perhaps the most famous simulation experiment is Buffon's needle, designed to calculate (not very efficiently) an estimate of π. There's nothing very sophisticated about this experiment, but for me I really like the 'mystique' of being able to trick nature into giving us an estimate of π. There are also a number of ways the experiment can be improved on to give better estimates which will highlight the general principle of *designing* simulated experiments to achieve optimal accuracy in the sense of minimizing statistical variability.

Buffon's original experiment is as follows. Imagine a grid of parallel lines of spacing d, on which we randomly drop a needle of length l, with $l \leq d$. We repeat this experiment n times, and count R, the number of times the

needle intersects a line. Denoting $\rho = l/d$ and $\phi = 1/\pi$, an estimate of ϕ is

$$\hat{\phi}_0 = \frac{\hat{p}}{2\rho}$$

where $\hat{p} = R/n$.

Thus, $\hat{\pi}_0 = 1/\hat{\phi}_0 = 2\rho/\hat{p}$ estimates π.

The rationale behind this is that if we let x be the distance from the centre of the needle to the lower grid point, and θ be the angle with the horizontal, then under the assumption of random needle throwing, we'd have $x \sim U[0, d]$ and $\theta \sim U[0, \pi]$. Thus

$$\begin{aligned} p &= \Pr(\text{needle intersects grid}) \\ &= \frac{1}{\pi}\int_0^\pi \Pr(\text{needle intersects } |\theta = \phi) d\phi \\ &= \frac{1}{\pi}\int_0^\pi \left(\frac{2}{d} \times \frac{l}{2}\sin\phi\right) d\phi \\ &= \frac{2l}{\pi d} \end{aligned}$$

A natural question is how to optimise the relative sizes of l and d. To address this we need to consider the variability of the estimator $\hat{\phi}_0$. Now, $R \sim \text{Bin}(n, p)$, so $\text{Var}(\hat{p}) = p(1-p)/n$. Thus $\text{Var}(\hat{\phi}_0) = 2\rho\phi(1-2\rho\phi)/4\rho^2 n = \phi^2(1/2\rho\phi - 1)/n$ which is minimized (subject to $\rho \leq 1$) when $\rho = 1$. That is, we should set $l = d$ to optimize efficiency.

Then, $\hat{\phi}_0 = \frac{\hat{p}}{2}$, with $\text{Var}(\hat{\phi}_0) = (\phi/2 - \phi^2)/n$.

Figure 1.1 gives 2 realisations of Buffon's experiment, based on 5000 simulations each. The figures together with an estimate can be produced in R by

```
buf=function(n,d,l){
x=runif(n)*d/2
theta=runif(n)*pi/2
I=(l*cos(theta)/2>x)
R=cumsum(I)
phat=R/(1:n)
nn=1:n
  plot(nn[phat>0],2*l/d/phat[phat>0],xlab='proportion of hits',ylab='pi estimate',type='l')
}
```

Exercise 1.1. *Provide with the full details in the argument above which showed that the optimality is achieved for the estimator $\hat{\phi}$.*

Use the R-code given above and design a Monte Carlo study that confirms (or not) that optimality is also achieved for $\hat{\pi}$ when $\rho = 1$, i.e. $d = l$. First, explain why it is not obvious. When discussing this review the concepts of the bias, the variance and the mean-square error and relations between these three. Then explain or/and analyze numerically what is the bias, the variance and the mean-square error of $\hat{\phi}$ and $\hat{\pi}$. Hint: Note that the event that the needle does not crosses the line in any trial has a positive probability

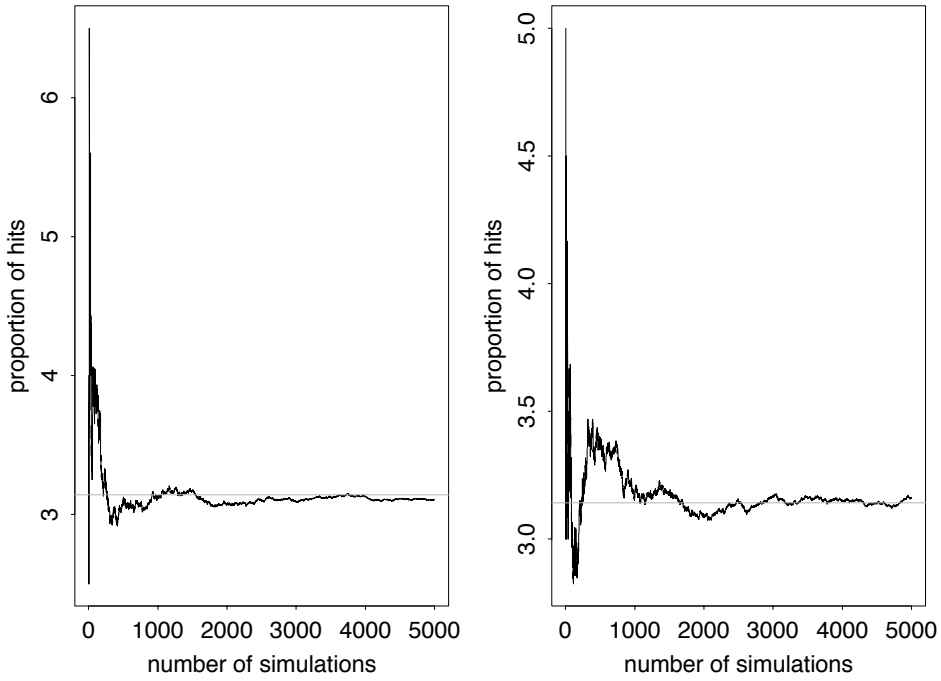

Figure 1.1: Two sequences of realisations of Buffon's experiment

and this affects existence of the mean and the variance of $\hat{\pi}$*. Modify the estimator to avoid the problem.*

The argument given above assumed that $l \leq d$*. Modify the algorithm to investigate also the case of* $d < l$*. Investigate the optimality in this case.*

Thus I've used the computer to simulate the physical simulations. You might like to check why this code works.

There are a catalogue of modifications which you can use which might (or might not) improve the efficiency of this experiment. These include:

1. Using a grid of rectangles or squares (which is best?) and basing estimate on the number of intersections with either or both horizontal or vertical lines.

2. Using a cross instead of a needle.

3. Using a needle of length longer than the grid separation.

So, just to re–iterate, the point is that simulation can be used to answer interesting problems, but that careful design may be needed to achieve even moderate efficiency.

1.3 Raw ingredients

The raw material for any simulation exercise is random digits: transformation or other types of manipulation can then be applied to build simulations of more complex distributions or systems. So, how can random digits be generated?

It should be recognised that any algorithmic attempt to mimic randomness is just that: a mimic. By definition, if the sequence generated is deterministic then it isn't random. Thus, the trick is to use algorithms which generate sequences of numbers which would pass all the tests of randomness (from the required distribution or process) despite their deterministic derivation. The most common technique is to use a *congruential generator*. This generates a sequence of integers via the algorithm

$$x_i = ax_{i-1}(\text{mod } M) \tag{1.1}$$

for suitable choices of a and M. Dividing this sequence by M gives a sequence u_i which are regarded as realisations from the Uniform $U[0,1]$ distribution. Problems can arise by using inappropriate choices of a and M. We won't worry about this issue here, as any decent statistics package should have had its random number generator checked pretty thoroughly. The point worth remembering though is that computer generated random numbers aren't random at all, but that (hopefully) they look random enough for that not to matter.

In subsequent sections then, we'll take as axiomatic the fact that we can generate a sequence of numbers $u_1, u_2, \ldots, u_n$ which may be regarded as n independent realisations from the $U[0,1]$ distribution.

Chapter 2

Simulating from specified distributions

In this chapter we look at ways of simulating data from a specified distribution function F, on the basis of a simulated sample $u_1, u_2, \ldots, u_n$ from the distribution $U[0,1]$.

2.1 Transforming uniforms

We start with the case of constructing a draw x from a random variable $X \in \mathbf{R}$ with a continuous distribution F on the basis of a single u from $U[0,1]$. It is natural to try a simple transformation $x = h(u)$, but how should we choose h? Let's assume h is increasing with inverse $h^{-1} : \mathbf{R} \mapsto [0,1]$. The requirement is now that

$$\begin{aligned} F(v) &= P(X \leq v) = P(h(U) \leq v,) \\ &= P(h^{-1}(h(U)) \leq h^{-1}(v)) = P(U \leq h^{-1}(v)) \\ &= h^{-1}(v), \end{aligned}$$

for all $v \in \mathbf{R}$ and where in the last step we used that the distribution function of the $U[0,1]$ distribution equals $P(U \leq u) = u, u \in [0,1]$. The conclusion is clear, we should choose $h = F^{-1}$. If F is not one-to-one, as is the case for discrete random variables, the above argument remains valid if we define the inverse as

$$F^{-1}(u) = \inf\{x; F(x) \geq u\}. \tag{2.1}$$

The resulting algorithm for drawing from F is *the Inversion Method*:

Algorithm 2.1 (The Inversion Method)**.**

1. Draw u from $U[0,1]$.
2. $x = F^{-1}(u)$ can now be regarded a draw from F.

Figure 2.1 illustrates how this works. For example, to simulate from the

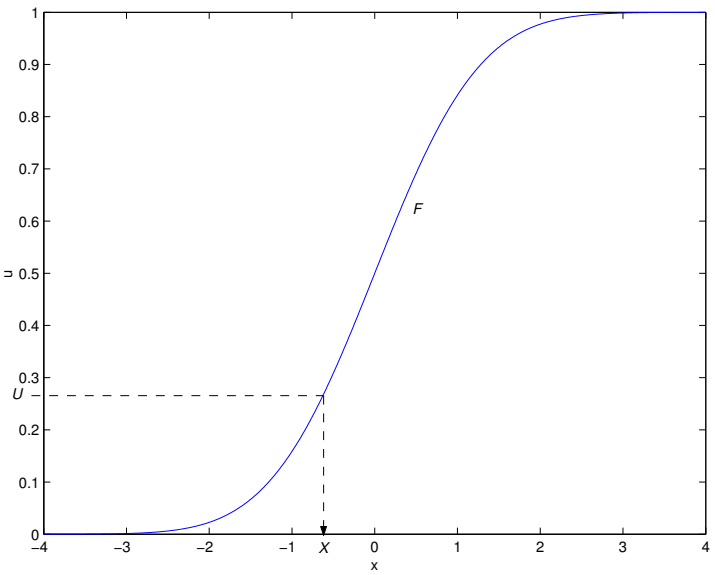

Figure 2.1: Simulation by inversion; the random variable $X = F^{-1}(U)$ will have distribution F if U is uniformly distributed on $[0, 1]$.

exponential distribution we have $F(x) = 1 - \exp(-\lambda x)$, so

$$F^{-1}(u) = -\lambda^{-1} \log(1 - u).$$

Thus with

```
u=runif(1,n);
x=-(log(1-u))/lambda;
```

we can simulate `n` independent values from the exponential distribution with parameter `lambda`. Figure 2.2 shows a histogram of 1000 standard ($\lambda = 1$) exponential variates simulated with this routine.

For discrete distributions, the procedure then simply amounts to searching through a table of the distribution function. For example, the distribution function of the Poisson(2) distribution is

```
x      F(x)
------------
0   0.1353353
1   0.4060058
2   0.6766764
3   0.8571235
4   0.9473470
5   0.9834364
6   0.9954662
```

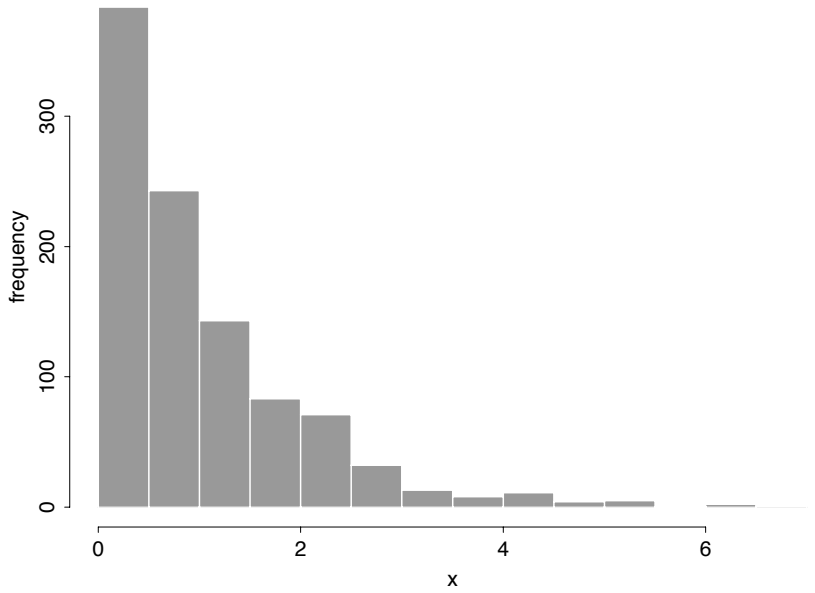

Figure 2.2: Histogram of 1000 simulated unit exponential variates

```
 7  0.9989033
 8  0.9997626
 9  0.9999535
10  0.9999917
```

so, we generate a sequence of standard uniforms $u_1, u_2, \ldots, u_n$ and for each u_i obtain a Poisson(2) variate x_i where $F(x_i - 1) < u_i \leq F(x_i)$. So, for example, if $u_1 = 0.7352$ then $x_1 = 3$.

The limitation on the efficiency of this procedure is due to the necessity of searching through the table, and there are various schemes to optimize this aspect.

Returning to the continuous case, it may seem that the inversion method is sufficiently universal to be the only method required. In fact, there are many situations in which the inversion method is either (or both) complicated to program or excessively inefficient to run. The inversion method is only really useful if the inverse distribution function is easy to program and compute. This is not the case, for example, with the Normal distribution function for which the inverse distribution function, Φ^{-1}, is not available analytically and slow to evaluate numerically. An even more serious limitation is that the method only applies for generating draws from univariate random variables. To deal with such cases, we turn to a variety of alternative schemes.

2.2 Transformation methods

The inversion method is a special case of more general transformation methods. The following theorem can be used to derive the distribution of $Z = h(X)$ for a more general class of real-valued random variables X.

Theorem 2.1 (Transformation theorem)**.** *Let $X \in \mathcal{X} \subseteq \mathbf{R}$ have a continuous density f and h a function with differentiable inverse $g = h^{-1}$. Then the random variable $Z = h(X) \in \mathbf{R}$ has density*

$$f(g(z))|g'(z)|, \tag{2.2}$$

for $z \in h(\mathcal{X})$ and zero elsewhere.

Proof. The proof is a direct application of the change-of-variable theorem for integrals. Note that two random variables X and Z have the same distribution iff $P(X \in A) = P(Z \in A)$ for all sets A. □

This device is used extensively in simulation, for example when we want to generate a $N(\mu, \sigma^2)$ variate y, it is common to first draw x from $N(0, 1)$ and then set $y = \sigma x + \mu$. Use Theorem 2.1 to show that this works. Sums of random variables can also be useful in creating new variables. Recall that

Theorem 2.2. *Let $X \in \mathbf{R}$ and $Y \in \mathbf{R}$ be independent with densities f and g respectively, then the density of $Z = X + Y$ equals $f * g(z) = \int f(t - z)g(t)\, dt$.*

This can be used to generate Gamma random variables. A random variable X has a Gamma$(a, 1)$ distribution if its density is proportional to $x^{a-1} \exp(-x)$, $x > 0$. Using Theorem 2.2 we can show that if X and Y are independent Gamma$(a, 1)$ and Gamma$(a', 1)$ respectively, then $Z = X + Y$ has a Gamma$(a + a', 1)$ distribution. Since Gamma$(1, 1)$ (i.e. Exponential(1)) variables are easily generated by inversion, a Gamma$(k, 1)$ variable Z, for integer values k, is generated by

$$z = \sum_{i=1}^{k} -\log(u_k) \tag{2.3}$$

using independent draws of uniforms $u_1, \ldots, u_n$. As an alternative we can use a combination of Theorems 2.1 and 2.2 to show that

$$z = \sum_{i=1}^{2k} x_i^2/2 \tag{2.4}$$

is a draw from the same distribution if $x_1, \ldots, x_{2k}$ are independent standard Normal draws.

Example 2.1 (The Box-Muller transformation)**.** This is a special trick to simulate from the Normal distribution. In fact it produces two independent variates in one go. Let u_1, u_2 be two independently sampled U$[0, 1]$ variables, then it can be shown that

$$x_1 = \sqrt{-2\log(u_2)}\cos(2\pi u_1) \text{ and } x_2 = \sqrt{-2\log(u_2)}\sin(2\pi u_1)$$

are two independent N$(0, 1)$ variables.

Below we give the multivariate version of Theorem 2.1.

Theorem 2.3 (Multivariate transformation theorem)**.** *Let $X \in \mathcal{X} \subseteq \mathbf{R}^d$ have a continuous density f and $h : \mathcal{X} \mapsto \mathbf{R}^d$ a function with differentiable inverse $g = h^{-1}$. Further write $J(z)$ for the determinant of the Jacobian matrix of $g = (g_1, \ldots, g_d)$,*

$$J(x) = \begin{vmatrix} dg_1(z)/dz_1 & \ldots & dg_1(z)/dz_d \\ \vdots & \ddots & \vdots \\ dg_d(z)/dz_1 & \ldots & dg_d(z)/dz_d \end{vmatrix}. \tag{2.5}$$

Then the random variable $Z = h(X) \in \mathbf{R}^d$ has density

$$f(g(z))|J(z)|, \tag{2.6}$$

for $z \in h(\mathcal{X})$ and zero elsewhere.

Example 2.2 (Choleski method for multivariate Normals)**.** The Choleski method is a convenient way to draw a vector z from the multivariate Normal distribution $N_n(0, \Sigma)$ based on a vector of n independent $N(0, 1)$ draws $(x_1, x_2, \ldots, x_n)$. Choleski decomposition is a method for computing a matrix C such that $CC^T = \Sigma$, in R the command is `chol`. We will show that $z = Cx$ has the desired distribution. The density of X is $f(x) = (2\pi)^{-d/2} \exp(-x^T x/2)$ and the Jacobian of the inverse transformation, $x = C^{-1}z$, equals $J(z) = |C^{-1}| = |C|^{-1} = |\Sigma|^{-1/2}$. Hence, according to Theorem 2.3, the density of Z equals

$$\begin{aligned} f(C^{-1}z)|\Sigma|^{-1/2} &= (2\pi)^{-d/2} \exp(-(C^{-1}z)^T (C^{-1}z)/2)|\Sigma|^{-1/2} \\ &= (2\pi)^{-d/2} \exp(-z^T \Sigma^{-1} z/2)|\Sigma|^{-1/2}, \end{aligned}$$

which we recognise as the density of a $N_n(0, \Sigma)$ distribution. Of course, $z + \mu$, $\mu \in \mathbf{R}^d$ is a draw from $N_n(\mu, \Sigma)$.

2.3 Rejection sampling

The idea in rejection sampling is to simulate from one distribution which is easy to simulate from, but then to only accept that simulated value with some probability p. By choosing p correctly, we can ensure that the sequence of accepted simulated values are from the desired distribution.

The method is based on the following theorem:

Theorem 2.4. *Let f be the density function of a random variable on $\mathbf{R}^d$ and let $Z \in \mathbf{R}^{d+1}$ be a random variable that is uniformly distributed on the set $A = \{z; 0 \leq z_{d+1} \leq Mf(z_1, \ldots, z_d)\}$ for an arbitrary constant $M > 0$. Then the vector $(Z_1, \ldots, Z_d)$ has density f.*

Proof. First note that

$$\begin{aligned}\int_A dz &= \int_{\mathbf{R}^d} \left(\int_0^{Mf(z_1,\ldots,z_d)} dz_{d+1} \right) dz_1 \cdots dz_d \\ &= M \int f(z_1, \ldots, z_d)\, dz_1 \cdots dz_d = M.\end{aligned}$$

Hence, Z has density $1/M$ on A. Similarily, with $B \subseteq \mathbf{R}^d$, we have

$$\begin{aligned}P((Z_1, \ldots, Z_d) \in B) &= \int_{\{z; z\in A\}\cap\{z;(z_1,\ldots,z_d)\in B\}} M^{-1}\, dz \\ &= M^{-1} \int_B Mf(z_1, \ldots, z_d)\, dz_1 \cdots dz_d \\ &= \int_B f(z_1, \ldots, z_d)\, dz_1 \cdots dz_d,\end{aligned}$$

and this is exactly what we needed to show. □

The conclusion of the above theorem is that we can construct a draw from f by drawing uniformly on an appropriate set and then drop the last coordinate of the drawn vector. Note that the converse of the above theorem is also true, i.e. if we draw $(z_1, \ldots, z_d)$ from f and then z_{d+1} from $U(0, Mf(z_1, \ldots, z_d))$, $(z_1, \ldots, z_{d+1})$ will be a draw from the uniform distribution on $A = \{z; 0 \leq z_{d+1} \leq Mf(z_1, \ldots, z_d)\}$. The question is how to draw uniformly on A without having to draw from f (since this was our problem in the first place); the rejection method solves this by drawing uniformly on a larger set $B \supset A$ and rejecting the draws that end up in $B\backslash A$. A natural choice of B is $B = \{z; 0 \leq z_{d+1} \leq Kg(z_1, \ldots, z_d)\}$, where g is another density, the *proposal density*, that is easy to draw from and satisfies $Mf \leq Kg$.

Algorithm 2.2 (The Rejection Method)**.**

1. Draw $(z_1, \ldots, z_d)$ from a density g that satisfies $Mf \leq Kg$.
2. Draw z_{d+1} from $U(0, Kg(z_1, \ldots, z_d))$.
3. Repeat steps 1-2 until $z_{d+1} \leq Mf(z_1, \ldots, z_d)$.
4. $x = (z_1, \ldots, z_d)$ can now be regarded as a draw from f.

It might seem superflous to have two constants M and G in the algorithm. Indeed, the rejection method is usually presented with $M = 1$. We include M here to illustrate the fact that you only need to know the density up to a constant of proportionality (i.e. you know Mf but not M or f). This situation is very common, especially in applications to Bayesian statistics.

The efficiency of the rejection method depends on how many points are rejected, which in turn depends on how close Kg is to Mf. The probability of accepting a particular draw $(z_1, \ldots, z_d)$ from g equals

$$\begin{aligned} &P(Z_{d+1} \leq Mf(Z_1, \ldots, Z_d)) \\ &\qquad = \int \left(\int_0^{Mf(z_1,\ldots,z_d)} (Kg(z_1, \ldots, z_d))^{-1} \, dz_{d+1} \right) g(z_1, \ldots, z_d \, dz_1 \cdots dz_d \\ &\qquad = \frac{M}{K} \int f(z_1, \ldots, z_d) \, dz_1 \cdots dz_d = \frac{M}{K}. \end{aligned}$$

For large d it becomes increasingly difficult to find g and K such that M/K is large enough for the algorithm to be useful. Hence, while the rejection method is not strictly univariate as the inversion method, it tends to be practically useful only for small d.