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

1.2 Raw ingredients

Chapter 2

Simulating from specified distributions

2.1 Transforming uniforms

2.2 Transformation methods

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 2.1. *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

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

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 2.1 (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

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

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.

The technique is illustrated in Figure 2.1.

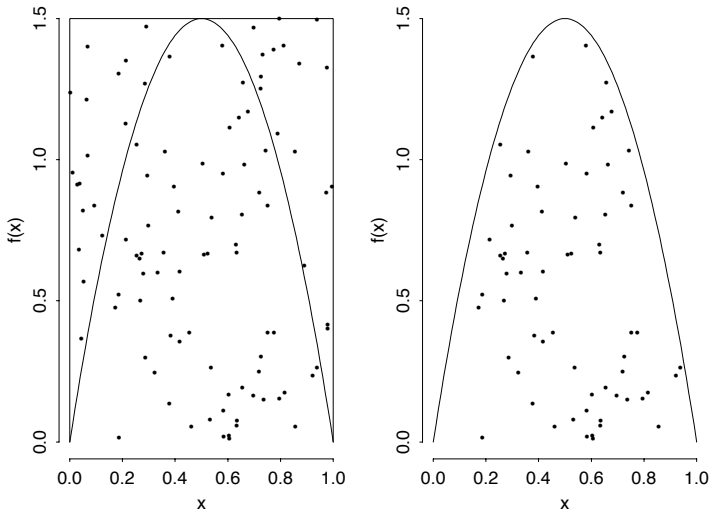

Figure 2.1: Simulation by rejection sampling from an $U[0,1]$ distribution (here $M = 1$ and $G = 1.5$); the x–coordinates of the points in the right panel constitute a sample with density f

As an example, consider the distribution with density

$$f(x) \propto x^2 e^{-x}; \quad 0 \leq x \leq 1, \tag{2.1}$$

a truncated gamma distribution. Then, since $f(x) \leq e^{-x}$ everywhere, we can set $g(x) = \exp(-x)$ and so simulate from an exponential distribution, rejecting according to the above algorithm. Figure 2.2 shows both $f(x)$ and $g(x)$. Clearly in this case the envelope is very poor so the routine is highly inefficient (though statistically correct). Applying this to generate a sample of 100 data by

```
RS=rejsim(100)
hist(RS$sample)
RS$count
```

using the following code

```
rejsim=function(n){
   x=vector("numeric",n)
   m=0
```

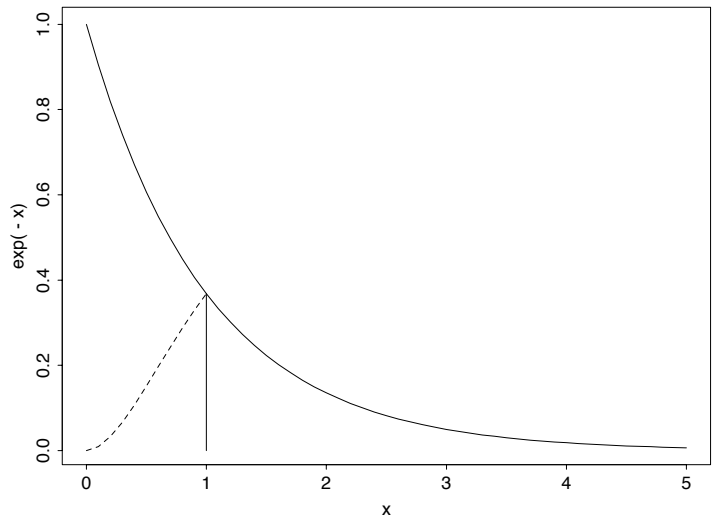

Figure 2.2: Scaled density and envelope

```
   for(i in 1:n)
   {
     acc=0
     while(acc<1){
       m=m+1
       z1=-log(runif(1))
       z2=runif(1)*exp(-z1)
       if(z2<z1^2*exp(-z1)*(z1<1)){
       acc=1
       x[i]=z1
       }
     }
   }
   rejsim=list(sample=x,count=m)
}
```

gave the histogram in Figure 2.3. The variable `m` contains the number of random variate pairs (Z_1, Z_2) needed to accept 100 variables from the correct distribution, in our simulation `m=618` suggesting that the algorithm is rather poor. What values of M and G did we use in this example?

Exercise 2.1. *Suggest and implement a more efficient method of rejection sampling for the above truncated distribution. Compare numerical efficiency of both the methods through a Monte Carlo study.*

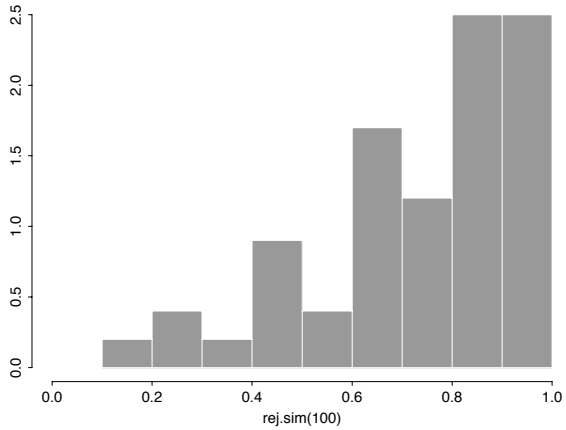

Figure 2.3: Histogram of simulated data

2.4 Conditional methods

The inversion method is strictly univariate, since the inverse F^{-1} is not well-defined for functions $F : \mathbf{R}^d \mapsto [0, 1]$ when $d > 1$. The rejection method is not limited to $d = 1$, but for large d it becomes increasingly difficult to find a bounding function $Kg(x)$ that preserves a reasonably high acceptance rate. A general technique to simulate from a multivariate distribution, using steps of univariate draws, is suggested by a factorization argument. Any d-variate density function f can be factorised recursively as

$$f(x_1, \ldots, x_d) = f_1(x_1) f_2(x_2|x_1) f_3(x_3|x_2, x_1) \cdots f_d(x_d|x_{d-1}, \ldots, x_1). \quad (2.2)$$

Given the above factorisation, a draw from f can now be produced recursively by

Algorithm 2.2.

1. Draw x_1 from the distribution with density $f_1(\cdot)$.
2. Draw x_2 from the distribution with density $f_2(\cdot|x_1)$.
3. Draw x_3 from the distribution with density $f_3(\cdot|x_1, x_2)$.

   $\vdots$

d. Draw x_d from the distribution with density $f_d(\cdot|x_1, x_2, \ldots, x_{d-1})$.

$(x_1, \ldots, x_d)$, is now a draw from $f(x_1, \ldots, x_d)$ in (2.2).

In the above algorithm, each step could be performed with an univariate method. The problem is that, commonly, the factorisation in (2.2) is not explicitly available. For example, deriving f_1 involves the integration

$$f_1(x_1) = \int f(x_1, \ldots, x_d)\, dx_2 \cdots dx_d,$$

which we might not be able to perform analytically.

Example 2.1 (Simulating a Markov Chain)**.** Recall that a Markov Chain is a stochastic process $(X_0, X_1, \ldots, X_n)$ such that, conditionally on $X_{i-1} = x_{i-1}$, X_i is independent of the past $(X_0, \ldots, X_{i-2})$. Assuming x_0 is fixed, the factorisation (2.2) simplifies to

$$f(x_1, \ldots, x_n) = f_1(x_1|x_0) f(x_2|x_1) \cdots f(x_n|x_{n-1}),$$

for a common transition density $f(x_i|x_{i-1})$ of $X_i|X_{i-1} = x_{i-1}$. Thus, to simulate a chain starting from x_0, we proceed recursively as follows

1. Draw x_1 from the distribution with density $f(\cdot|x_0)$.
2. Draw x_2 from the distribution with density $f(\cdot|x_1)$.

$\vdots$

n. Draw x_n from the distribution with density $f(\cdot|x_{n-1})$.

Exercise 2.2. *Consider that on a day one there is an individual that on any day and independently of anything else can give a birth to another individual with probability $p \in (0, 1)$. Each individual on every next day after its birthday can give a birth to another individual under the same conditions as the first individual. Let $(X_1, \ldots, X_{50})$ be the number of individuals on the first ten days. Implement a method of simulating from this vector. Perform MC investigation of behavior of such vector starting from small values of p.*

Exercise 2.3. *Extend the above model by assuming that at any given day each alive individual can die with probability $q(1 - 1/X_{n-1})$, $q \in (0, 1]$. This mimics the fact that with bigger population there is less resources to survive. Propose implementation of a generator from the vector $(X_1, \ldots, X_n)$ and perform study considering various values of q.*

Example 2.2 (Bivariate Normals)**.** Another application of the factorisation argument is useful when generating draws from the bivariate Normal distribution. If

$$(X_1, X_2) \sim N_2\left((\mu_1, \mu_2), \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}\right),$$

we obviously have that $X_1 \sim N(\mu_1, \sigma_1^2)$ and it is a straightforward exercise to show that $X_2|X_1 = x_1 \sim N(\mu_2 + \rho\sigma_2(x_1 - \mu_1)/\sigma_1, \sigma_2^2(1 - \rho^2))$.

Exercise 2.4. *Propose and implement the conditional method of simulation for a normal vector $(X_1, \ldots, X_2) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Perform numerical comparison of the efficiency of your method with the one based on Cholesky's decompostion.*