2 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?

2.1 The inverse transform method

Proposition 1 *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

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

Check that $\mathbf{P}(X = x_j) = \mathbf{P}(\sum_{i=0}^{j-1} p_i \leq U < \sum_{i=0}^{j} p_i) = p_j$, $j = 0, 1, \ldots$.

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

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

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

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

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. $(P_1, \dots, P_n)$ 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.

2.2 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 $Z \sim \mathcal{N}(0,1)$. 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^2 have the conditional distribution of S^2 conditioned on $S^2 \leq 1$. 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.

2.3 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 $U_1 \le \frac{1}{2}$ let $X = \ln U_2$
- If $U_1 > \frac{1}{2}$ let $X = -\ln U_2$.

2.4 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 2 *Let $C = \{(u,v) : 0 \le u \le f^{1/2}(\frac{v}{u})\}$. 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

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

Hence the marginal density of Z is

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

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.

2.5 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.

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`