1 Acceptance-Rejection Method

As we already know, finding an explicit formula for $F^{-1}(y)$ for the cdf of a rv X we wish to generate, $F(x) = P(X \le x)$, is not always possible. Moreover, even if it is, there may be alternative methods for generating a rv distributed as F that is more efficient than the inverse transform method or other methods we have come across. Here we present a very clever method known as the *acceptance-rejection method.*

We start by assuming that the F we wish to simulate from has a probability density function $f(x)$; that is, the continuous case. Later we will give a discrete version too, which is very similar.

The basic idea is to find an alternative probability distribution G, with density function $g(x)$, from which we already have an efficient algorithm for generating from (e.g., inverse transform method or whatever), but also such that the function $g(x)$ is "close" to $f(x)$. In particular, we assume that the ratio $f(x)/g(x)$ is bounded by a constant $c > 0$; $\sup_x\{f(x)/g(x)\} \le c$. (And in practice we would want c as close to 1 as possible.)

Here then is the algorithm for generating X distributed as F:

Acceptance-Rejection Algorithm for continuous random variables

1. Generate a rv Y distributed as G.
2. Generate U (independent from Y).
3. If
$$U \le \frac{f(Y)}{cg(Y)},$$
then set $X = Y$ ("accept") ; otherwise go back to 1 ("reject").

Before we prove this and give examples, several things are noteworthy:

- $f(Y)$ and $g(Y)$ are rvs, hence so is the ratio $\frac{f(Y)}{cg(Y)}$ and this ratio is independent of U in Step (2).
- The ratio is bounded between 0 and 1; $0 < \frac{f(Y)}{cg(Y)} \le 1$.
- The number of times N that steps 1 and 2 need to be called (e.g., the number of iterations needed to successfully generate X) is itself a rv and has a geometric distribution with "success" probability $p = P(U \le \frac{f(Y)}{cg(Y)})$; $P(N = n) = (1-p)^{n-1}p,\ n \ge 1$. Thus on average the number of iterations required is given by $E(N) = 1/p$.
- In the end we obtain our X as having the conditional distribution of a Y given that the event $\{U \le \frac{f(Y)}{cg(Y)}\}$ occurs.

A direct calculation yields that $p = 1/c$, by first conditioning on Y:
$P(U \le \frac{f(Y)}{cg(Y)} \mid Y = y) = \frac{f(y)}{cg(y)}$; thus, unconditioning and recalling that Y has density $g(y)$ yields

$$p = \int_{-\infty}^{\infty} \frac{f(y)}{cg(y)} \times g(y)dy$$

$$
\begin{aligned}
&= \frac{1}{c}\int_{-\infty}^{\infty} f(y)dy \\
&= \frac{1}{c},
\end{aligned}
$$

where the last equality follows since f is a density function (hence by definition integrates to 1). Thus $E(N) = c$, the bounding constant, and we can now indeed see that it is desirable to choose our alternative density g so as to minimize this constant $c = \sup_x\{f(x)/g(x)\}$. Of course the optimal function would be $g(x) = f(x)$ which is not what we have in mind since the whole point is to choose a different (easy to simulate) alternative from f. In short, it is a bit of an art to find an appropriate g. In any case, we summarize with

The expected number of iterations of the algorithm required until an X is successfully generated is exactly the bounding constant $c = \sup_x\{f(x)/g(x)\}$.

Proof that the algorithm works

Proof : We must show that the conditional distribution of Y given that $U \leq \frac{f(Y)}{cg(Y)}$, is indeed F; that is, that $P(Y \leq y \mid U \leq \frac{f(Y)}{cg(Y)}) = F(y)$. We will prove this by showing that the conditional density is indeed $f(y)$; that is, that the conditional density of $P(Y = y \mid U \leq \frac{f(Y)}{cg(Y)}) = f(y)$. Using $P(A \mid B) = P(AB)/P(B)$ yields, since $p = P(U \leq \frac{f(Y)}{cg(Y)}) = 1/c$, and U and Y are independent,

$$
\begin{aligned}
P(Y = y \mid U \leq \frac{f(Y)}{cg(Y)}) &= cP(Y = y\ ,\ U \leq \frac{f(Y)}{cg(Y)}) \\
&= cP(Y = y\ ,\ U \leq \frac{f(y)}{cg(y)}) \\
&= cg(y)P(U \leq \frac{f(y)}{cg(y)}) \\
&= cg(y)\frac{f(y)}{cg(y)} \\
&= f(y).
\end{aligned}
$$

■

2 Applications of the acceptance-rejection method

2.1 Normal distribution

If we desire an $X \sim N(\mu, \sigma^2)$, then we can express it as $X = \sigma Z + \mu$, where Z denotes a rv with the $N(0, 1)$ distribution. Thus it suffices to find an algorithm for generating $Z \sim N(0, 1)$. Moreover, if we can generate from the absolute value, $|Z|$, then by symmetry we can obtain our Z by independently generating a rv S (for sign) that is ± 1 with probability 1/2 and setting $Z = S|Z|$. In other words we generate a U and set $Z = |Z|$ if $U \leq 0.5$ and set $Z = -|Z|$ if $U > 0.5$.

$|Z|$ is non-negative and has density

$$f(x) = \frac{2}{\sqrt{2\pi}}e^{-x^2/2},\ x \geq 0.$$

For our alternative we will choose $g(x) = e^{-x}$, $x \geq 0$, the exponential density with rate 1, something we already know how to easily simulate from (inverse transform method, for example). Noting that $h(x) \overset{\text{def}}{=} f(x)/g(x) = e^{x-x^2/2}\sqrt{2/\pi}$, we simply use calculus to compute its maximum (solve $h'(x) = 0$); which must occur at that value of x which maximizes the exponent $x - x^2/2$; namely at value $x = 1$. Thus $c = \sqrt{2e/\pi} \approx 1.32$, and so $f(y)/cg(y) = e^{-(y-1)^2/2}$.

The algorithm for generating Z is then

1. Generate Y with an exponential distribution at rate 1; that is, generate U and set $Y = -\ln(U)$.
2. Generate U
3. If $U \leq e^{-(Y-1)^2/2}$, set $|Z| = Y$; otherwise go back to 1.
4. Generate U. Set $Z = |Z|$ if $U \leq 0.5$, set $Z = -|Z|$ if $U > 0.5$.

Note that in 3, $U \leq e^{-(Y-1)^2/2}$ if and only if $-\ln(U) \geq (Y-1)^2/2$ and since $-\ln(U)$ is exponential at rate 1, we can simplify the algorithm to

Algorithm for generating $Z \sim N(0,1)$

1. Generate two independent exponentials at rate 1; $Y_1 = -\ln(U_1)$ and $Y_2 = -\ln(U_2)$.
2. If $Y_2 \geq (Y_1 - 1)^2/2$, set $|Z| = Y_1$; otherwise go back to 1.
3. Generate U. Set $Z = |Z|$ if $U \leq 0.5$, set $Z = -|Z|$ if $U > 0.5$.

As a nice afterthought, note that by the memoryless property of the exponential distribution, the amount by which Y_2 exceeds $(Y-1)^2/2$ in Step 2 of the algorithm when Y_1 is accepted, that is, the amount $Y \overset{\text{def}}{=} Y_2 - (Y_1 - 1)^2/2$, has itself the exponential distribution with rate 1 and is independent of Y_1. Thus, for free, we get back an independent exponential which could then be used as one of the two needed in Step 1, if we were to want to start generating yet another independent $N(0,1)$ rv. Thus, after repeated use of this algorithm, the expected number of uniforms required to generate one Z is $(2c+1) - 1 = 2c = 2.64$.

One might ask if we could improve our algorithm for Z by changing the rate of the exponential; that is, by using an exponential density $g(x) = \lambda e^{-\lambda x}$ for some $\lambda \neq 1$. The answer is no: $\lambda = 1$ minimizes the value of c obtained (as can be proved using elementary calculus).

2.2 Beta distribution

In general, a beta distribution on the unit interval, $x \in (0,1)$, has a density of the form $f(x) = bx^n(1-x)^m$ with n and m non-negative (integers or not). The constant b is the normalizing constant,

$$b = \Big[\int_0^1 x^n(1-x)^m dx\Big]^{-1}.$$

(Such distributions generalize the uniform distribution and are useful in modeling random proportions.)

Let us consider a special case of this: $f(x) = bx^n(1-x)^n = b(x(1-x))^n$. Like the uniform distribution on $(0,1)$, this has mean $1/2$, but its mass is more concentrated near $1/2$ than near 0 or 1; it has a smaller variance than the uniform. If we use $g(x) = 1$, $x \in (0,1)$ (the uniform density), then $f(x)/g(x) = f(x)$ and as is easily verified, $c = \sup_x f(x)$ is achieved at $x = 1/2$; $c = b(1/4)^n$.

Thus we can generate from $f(x) = b(x(1-x))^n$ as follows:

1. Generate U_1 and U_2.
2. If $U_2 \leq 4^n(U_1(1-U_1))^n$, then set $X = U_1$; otherwise go back to 1.

It is interesting to note that in this example, and in fact in any beta example using $g(x) = 1$, we do not need to know (or compute in advance) the value of b; it cancels out in the ratio $f(x)/cg(x)$. Of course if we wish to know the value of c, we would need to know the value of b. In the case when n and m are integers, b is given by $b = (n+m+1)!/n!m!$, which in our example yields $b = \binom{2n+1}{n}$, and so $c = \binom{2n+1}{n}/4^n$. For large values of n, this is not so efficient, as should be clear since in that case the two densities f and g are not very similar.

There are other ways to generate the beta distribution. For example one can use the basic fact (can you prove this?) that if X_1 is a gamma rv with shape parameter $n+1$, and independently X_2 is a gamma rv with shape parameter $m+1$ (and both have the same scale parameter), then $X = X_1/(X_1+X_2)$ is beta with density $f(x) = bx^n(1-x)^m$. So it suffices to have an efficient algorithm for generating the gamma distribution. In general, when n and m are integers, the gammas become Erlang (represented by sums of iid exponentials); for example, if X_1 and X_2 are iid exponentials, then $X = X_1/(X_1+X_2)$ is uniform on $(0,1)$.

The gamma distribution can always be simulated using acceptance-rejection by using the exponential density $g(x) = \lambda e^{-\lambda x}$ in which $1/\lambda$ is chosen as the mean of the gamma (it can be shown that this value of λ is optimal).

3 Discrete case

The discrete case is analogous to the continuous case. Suppose we want to generate an X that is a discrete rv with probability mass function (pmf) $p(k) = P(X = k)$. Further suppose that we can already easily generate a discrete rv Y with pmf $q(k) = P(Y = k)$ such that $\sup_k\{p(k)/q(k)\} \leq c < \infty$. The following algorithm yields our X:

Acceptance-Rejection Algorithm for discrete random variables

1. Generate a rv Y distributed as $q(k)$.
2. Generate U (independent from Y).
3. If $U \leq \frac{p(Y)}{cq(Y)}$, then set $X = Y$; otherwise go back to 1.