1

Random number generation

The topic of this book is the study of statistical models using computer simulations. Here we use the term 'statistical models' to mean any mathematical models which include a random component. Our interest in this chapter and the next is in simulation of the random component of these models. The basic building block of such simulations is the ability to generate random numbers on a computer, and this is the topic of the present chapter. Later, in Chapter 2, we will see how the methods from Chapter 1 can be combined to simulate more complicated models.

Generation of random numbers, or more general random objects, on a computer is complicated by the fact that computer programs are inherently deterministic: while the output of computer program may look random, it is obtained by executing the steps of some algorithm and thus is totally predictable. For example the output of a program computing the decimal digits of the number

$$\pi = 3.14159265358979323846264338327950288419716939937510\cdots$$

(the ratio between the perimeter and diameter of a circle) looks random at first sight, but of course π is not random at all! The output can only start with the string of digits given above and running the program twice will give the same output twice.

We will split the problem of generating random numbers into two distinct subproblems: first we will study the problem of generating any randomness at all, concentrating on the simple case of generating independent random numbers, uniformly distributed on the interval [0, 1]. This problem and related concerns will be discussed in Section 1.1. In the following sections, starting with Section 1.2, we will study the generation of random numbers from different distributions, using the independent, uniformly distributed random numbers obtained in the previous step as a basis.

An Introduction to Statistical Computing: A Simulation-based Approach, First Edition. Jochen Voss.
© 2014 John Wiley & Sons, Ltd. Published 2014 by John Wiley & Sons, Ltd.

1.1 Pseudo random number generators

There are two fundamentally different classes of methods to generate random numbers:

(a) True random numbers are generated using some physical phenomenon which is random. Generating such numbers requires specialised hardware and can be expensive and slow. Classical examples of this include tossing a coin or throwing dice. Modern methods utilise quantum effects, thermal noise in electric circuits, the timing of radioactive decay, etc.

(b) Pseudo random numbers are generated by computer programs. While these methods are normally fast and resource effective, a challenge with this approach is that computer programs are inherently deterministic and therefore cannot produce 'truly random' output.

In this text we will only consider pseudo random number generators.

Definition 1.1 A pseudo random number generator (PRNG) is an algorithm which outputs a sequence of numbers that can be used as a replacement for an independent and identically distributed (i.i.d.) sequence of 'true random numbers'.

1.1.1 The linear congruential generator

This section introduces the linear congruential generator (LCG), a simple example of a PRNG. While this random number generator is no longer of practical importance, it shares important characteristics with the more complicated generators used in practice today and we study it here as an accessible example. The LCG is given by the following algorithm.

Algorithm 1.2 (linear congruential generator)

```
input:
    $m > 1$ (the *modulus*)
    $a \in \{1, 2, \ldots, m-1\}$ (the *multiplier*)
    $c \in \{0, 1, \ldots, m-1\}$ (the *increment*)
    $X_0 \in \{0, 1, \ldots, m-1\}$ (the *seed*)
output:
    a sequence $X_1, X_2, X_3, \ldots$ of pseud random numbers
1: **for** $n = 1, 2, 3, \ldots$ **do**
2:     $X_n \leftarrow (aX_{n-1} + c) \bmod m$
3:     output $X_n$
4: **end for**
```

In the algorithm, 'mod' denotes the modulus for integer division, that is the value $n \bmod m$ is the remainder of the division of n by m, in the range $0, 1, \dots, m - 1$. Thus the sequence generated by algorithm 1.2 consists of integers X_n from the range $\{0, 1, 2, \dots, m-1\}$. The output depends on the parameters m, a, c and on the seed X_0. We will see that, if m, a and c are carefully chosen, the resulting sequence behaves 'similar' to a sequence of independent, uniformly distributed random variables. By choosing different values for the seed X_0, different sequences of pseudo random numbers can be obtained.

Example 1.3 For parameters $m = 8$, $a = 5$, $c = 1$ and seed $X_0 = 0$, algorithm 1.2 gives the following output:

n	$5X_{n-1} + 1$	X_n
1	1	1
2	6	6
3	31	7
4	36	4
5	21	5
6	26	2
7	11	3
8	16	0
9	1	1
10	6	6

The output $1, 6, 7, 4, 5, 2, 3, 0, 1, 6, \dots$ shows no obvious pattern and could be considered to be a sample of a random sequence.

While the output of the LCG looks random, from the way it is generated it is clear that the output has several properties which make it different from truly random sequences. For example, since each new value of X_n is computed from X_{n-1}, once the generated series reaches a value X_n which has been generated before, the output starts to repeat. In example 1.3 this happens for $X_8 = X_0$ and we get $X_9 = X_1$, $X_{10} = X_2$ and so on. Since X_n can take only m different values, the output of a LCG starts repeating itself after at most m steps; the generated sequence is eventually periodic.

Sometimes the periodicity of a sequence of pseudo random numbers can cause problems, but on the other hand, if the period length is longer than the amount of random numbers we use, periodicity cannot affect our result. For this reason, one needs to carefully choose the parameters m, a and c in order to achieve a long enough period. In particular m, since it is an upper bound for the period length, needs to be chosen large. In practice, typical values of m are on the order of $m = 2^{32} \approx 4 \cdot 10^9$ and a and c are then chosen such that the generator actually achieves the maximally possible period length of m. A criterion for the choice of m, a and c is given in the following theorem (Knuth, 1981, Section 3.2.1.2).

Theorem 1.4 The LCG has period m if and only if the following three conditions are satisfied:

(a) m and c are relatively prime;

(b) $a - 1$ is divisible by every prime factor of m;

(c) if m is a multiple of 4, then $a - 1$ is a multiple of 4.

In the situation of the theorem, the period length does not depend on the seed X_0 and usually this parameter is left to be chosen by the user of the PRNG.

Example 1.5 Let $m = 2^{32}$, $a = 1\,103\,515\,245$ and $c = 12\,345$. Since the only prime factor of m is 2 and c is odd, the values m and c are relatively prime and condition (a) of the theorem is satisfied. Similarly, condition (b) is satisfied, since $a - 1$ is even and thus divisible by 2. Finally, since m is a multiple of 4, we have to check condition (c) but, since $a - 1 = 1\,103\,515\,244 = 275\,878\,811 \cdot 4$, this condition also holds. Therefore the LCG with these parameters m, a and c has period 2^{32} for every seed X_0.

1.1.2 Quality of pseudo random number generators

PRNGs used in modern software packages such as R or Matlab are more sophisticated (and more complicated) than the LCG presented in Section 1.1.1, but they still share many characteristics of the LCG. We will see that no PRNG can produce a perfect result, but the random number generators used in practice, for example the Mersenne Twister algorithm (Matsumoto and Nishimura, 1998), are good enough for most purposes. In this section we will discuss criteria for the quality of the output of general PRNGs, and will illustrate these criteria using the LCG as an example.

1.1.2.1 Period length of the output

We have seen that the output of the LCG is eventually periodic, with a period length of at most m. This property that the output is eventually periodic is shared by all PRNGs implemented in software. Most PRNGs used in practice have a period length which is much larger than the amount of random numbers a computer program could ever use in a reasonable time. For this reason, periodicity of the output is not a big problem in practical applications of PRNGs. The period length is a measure for the quality of a PRNG.

1.1.2.2 Distribution of samples

The output of almost all PRNGs is constructed so that it can be used as a replacement for an i.i.d. sample of *uniformly distributed* random numbers. Since the output takes values in a finite set $S = \{0, 1, \ldots, m-1\}$, in the long run, for every set $A \subseteq S$ we should have

$$\frac{\#\left\{i \;\middle|\; 1 \leq i \leq N, X_i \in A\right\}}{N} \approx \frac{\#A}{\#S}, \tag{1.1}$$

where $\#A$ stands for the number of elements in a finite set A.

Uniformity of the output can be tested using statistical tests like the chi-squared test or the Kolmogorov–Smirnov test (see e.g. Lehmann and Romano, 2005, Chapter 14).

One peculiarity when applying statistical tests for the distribution of samples to the output of a PRNG is that the test may fail in two different ways: The output could either have the wrong distribution (i.e. not every value appears with the same probability), or the output could be too regular. For example, the sequence $X_n = n \bmod m$ hits every value equally often in the long run, but it shows none of the fluctuations which are typical for a sequence of real random numbers. For this reason, statistical tests should be performed as two-sided tests when the distribution of the output of a PRNG is being tested.

Example 1.6 Assume that we have a PRNG with $m = 1024$ possible output values and that we perform a chi-squared test for the hypothesis

$$P\left(X_i \in \{64j, 64j+1, \ldots, 64j+63\}\right) = 1/16$$

for $j = 0, 1, \ldots, 15$.

If we consider a sample $X_1, X_2, \ldots, X_N$, the test statistic of the chi-squared test is computed from the observed numbers of samples in each block, given by

$$O_j = \#\left\{i \;\middle|\; 64j \leq X_i < 64(j+1)\right\}.$$

The expected count for block j, assuming that (1.1) holds, is

$$E_j = N \cdot 64/1024 = N/16$$

for $j = 0, 1, \ldots, 15$ and the test statistic of the corresponding chi-squared test is

$$Q = \sum_{j=0}^{15} \frac{(O_j - E_j)^2}{E_j}.$$

For large sample size N, and under the hypothesis (1.1), the value Q follows a χ^2-distribution with 15 degrees of freedom. Some quantiles of this distribution are:

q	6.262	7.261	$\cdots$	24.996	27.488
$P(Q \leq q)$	0.025	0.05	$\cdots$	0.95	0.975

Thus, for a one-sided test with significance level $1 - \alpha = 95\%$ we would reject the hypothesis if $Q > 24.996$. In contrast, for a two-sided test with significance level $1 - \alpha = 95\%$, we would reject the hypothesis if either $Q < 6.262$ or $Q > 27.488$.

We consider two different test cases: first, if $X_n = n \bmod 1024$ for $n = 1, 2, \ldots, N = 10^6$, we find $Q = 0.244368$. Since the series is very regular, the value of Q is very low. The one-sided test would accept this sequence as being uniformly distributed, whereas the two-sided test would reject the sequence.

Secondly, we consider $X_n = n \bmod 1020$ for $n = 1, 2, \ldots, N = 10^6$. Since this series never takes the values 1021 to 1023, the distribution is wrong and we expect a large value of Q. Indeed, for this case we get $Q = 232.5864$ and thus both versions of the test reject this sequence.

Random number generators used in practice, and even the LCG for large enough values of m, pass statistical tests for the distribution of the output samples without problems.

1.1.2.3 Independence of samples

Another aspect of the quality of PRNGs is the possibility of statistical dependence between consecutive samples. For example, in the LCG each output sample is a deterministic function of the previous sample and thus consecutive samples are clearly dependent. To some extent this problem is shared by all PRNGs.

An easy way to visualise the dependence between pairs of consecutive samples is a scatter plot of the points (X_i, X_{i+1}) for $i = 1, 2, \ldots, N - 1$. A selection of such plots is shown in Figure 1.1. Figure 1.1(a) illustrates what kind of plot one would expect if $X_i \sim \mathcal{U}[0, 1]$ was a true i.i.d. sequence. The remaining panels correspond to different variants of the LCG. Figure 1.1(b) (using $m = 81$) clearly illustrates that each X_i can only be followed by exactly one value X_{i+1}. While the same is true for Figure 1.1(c) and (d) (using $m = 1024$ and $m = 2^{32}$, respectively), the dependence is much convoluted there and in particular the structure of Figure 1.1(d) is visually indistinguishable from the structure of Figure 1.1(a).

One method for constructing PRNGs where X_{i+1} is not a function of X_i is to use a function $f(X_i)$ of the state, instead of the state X_i itself, as the output of the PRNG. Here, $f\colon \{0, 1, \ldots, m - 1\} \to \{0, 1, \ldots, \tilde{m} - 1\}$ is a map where $\tilde{m} < m$ and where the same number of pre-images is mapped to each output value. Then a uniform distribution of X_i will be mapped to a uniform distribution for $f(X_i)$ but the output $f(X_{i+1})$ is not a function of the previous output $f(X_i)$. This allows to construct random number generators with some degree of independence between consecutive values.

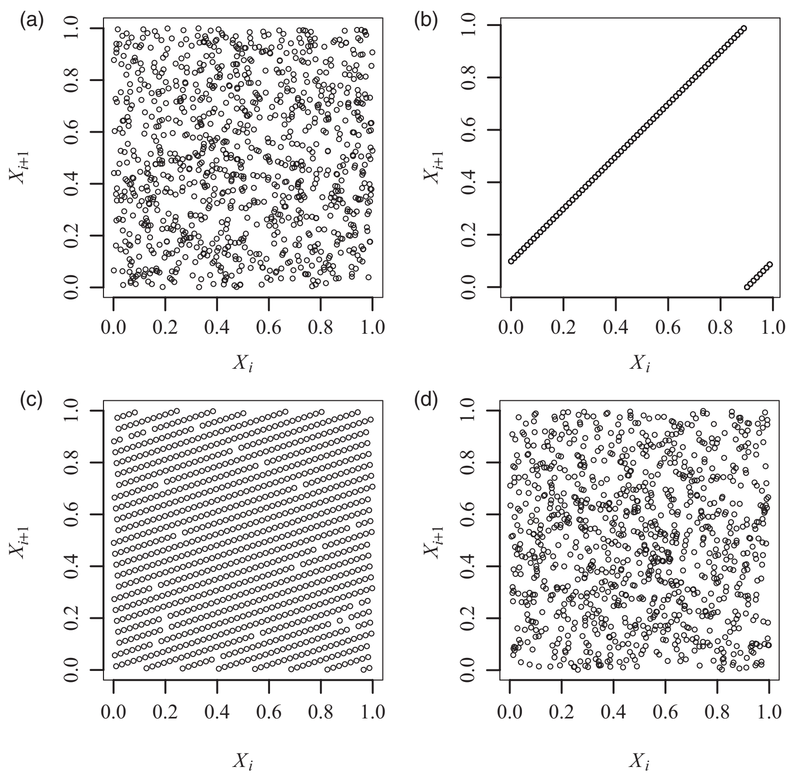


Figure 1.1 Scatter plots to illustrate the correlation between consecutive outputs X_i and X_{i+1} of different pseudo random number generators. The random number generators used are the `runif` *function in R (a), the LCG with $m = 81$, $a = 1$ and $c = 8$ (b), the LCG with $m = 1024$, $a = 401$, $c = 101$ (c) and finally the LCG with parameters $m = 2^{32}$, $a = 1\,664\,525$, $c = 1\,013\,904\,223$ (d). Clearly the output in the second and third example does not behave like a sequence of independent random variables.*

One way to quantify the independence of the output samples of a PRNG is the following criterion.

Definition 1.7 A periodic sequence $(X_n)_{n\in\mathbb{N}}$ with values in a finite set S and period length P is k-dimensionally equidistributed, if every possible subsequence $x = (x_1, \ldots, x_k) \in S^k$ of length k occurs equally often in the sequence X, that is if

$$N_x = \#\left\{ i \mid 0 \le i < P, X_{i+1} = x_i, \ldots, X_{i+k} = x_k \right\}$$

does not depend on x.

A random number generator is good, if the output is k-dimensionally equidistributed for large values of k.

JWST355-c01 JWST355-Voss Printer: Yet to Come July 27, 2013 13:27 Trim: 229mm × 152mm

1.1.3 Pseudo random number generators in practice

This section contains advice on using PRNGs in practice.

First, it is normally a bad idea to implement your own PRNG: finding a good algorithm for pseudo random number generation is a difficult problem, and even when an algorithm is available, given the nature of the generated output, it can be a challenge to spot and remove all mistakes in the implementation. Therefore, it is advisable to use a well-established method for random number generation, typically the random number generator built into a well-known software package or provided by a well-established library.

A second consideration concerns the rôle of the seed. While different PRNGs differ greatly in implementation details, they all use a seed (like the value X_0 in algorithm 1.2) to initialise the state of the random number generator. Often, when non-predictability is required, it is useful to set the seed to some volatile quantity (like the current time) to get a different sequence of random numbers for different runs of the program. At other times it can be more useful to get reproducible results, for example to aid debugging or to ensure repeatability of published results. In these cases, the seed should be set to a known, fixed value.

Finally, PRNGs like the LCG described above often generate a sequence which behaves like a sequence of independent random numbers, uniformly distributed on a finite set $\{0, 1, \ldots, m-1\}$ for a big value of m. In contrast, most applications require a sequence of independent, $\mathcal{U}[0, 1]$-distributed random variables, that is a sequence of i.i.d. values which are uniformly distributed on the real interval $[0, 1]$. We can obtain a sequence $(U_n)_{n\in\mathbb{N}}$ of pseudo random numbers to replace an i.i.d. sequence of $\mathcal{U}[0, 1]$ random variables by setting

$$U_n = \frac{X_n + 1}{m + 1},$$

where $(X_n)_{n\in\mathbb{N}}$ is the output of the PRNG. The output U_n can only take the m different values

$$\frac{1}{m+1}, \frac{2}{m+1}, \ldots, \frac{m}{m+1}$$

and thus U_n is not exactly uniformly distributed on the continuous interval $[0, 1]$. But, since the possible values are evenly spaced inside the interval $[0, 1]$ and since each of these values has the same probability, the distribution of U_n is a reasonable approximation to a uniform distribution on $[0, 1]$. This is particularly true since computers can only represent finitely many real numbers exactly.

This concludes our discussion of how a replacement for an i.i.d. sequence of $\mathcal{U}[0, 1]$-distributed random numbers can be generated on a computer.

1.2 Discrete distributions

Building on the methods from Section 1.1, in this and the following sections we will study methods to transform an i.i.d. sequence of $\mathcal{U}[0, 1]$-distributed random variables into an i.i.d. sequence from a prescribed target distribution. The methods from the previous section were inexact, since the output of a PRNG is not 'truly random'. In contrast, the transformations described in this and the following sections can be carried out with complete mathematical rigour. We will discuss different methods for generating samples from a given distribution, applicable to different classes of target distributions. In this section we concentrate on the simplest case where the target distribution only takes finitely or countably infinitely many values.

As a first example, we consider the uniform distribution on the set $A = \{0, 1, \ldots, n-1\}$, denoted by $\mathcal{U}\{0, 1, \ldots, n-1\}$. Since the set A has n elements, a random variable X with $X \sim \mathcal{U}\{0, 1, \ldots, n-1\}$ satisfies

$$P(X = k) = \frac{1}{n}$$

for all $k \in A$. To generate samples from such a random variable X, at first it may seem like a good idea to just use a PRNG with state space A, for example the LCG with modulus $m = n$. But considering the fact that the maximal period length of a PRNG is restricted to the size of the state space, it becomes clear that this is not a good idea. Instead we will follow the approach to first generate a continuous sample $U \sim \mathcal{U}[0, 1]$ and then to transform this sample into the required discrete uniform distribution. A method to implement this idea is described in the following lemma.

Lemma 1.8 Let $U \sim \mathcal{U}[0, 1]$ and $n \in \mathbb{N}$. Define a random variable X by $X = \lfloor nU \rfloor$, where $\lfloor \cdot \rfloor$ denotes rounding down. Then $X \sim \mathcal{U}\{0, 1, \ldots, n-1\}$.

Proof By the definition of X we have

$$P(X = k) = P(\lfloor nU \rfloor = k) = P(nU \in [k, k+1)) = P\left(U \in \left[\frac{k}{n}, \frac{k+1}{n}\right)\right)$$

for all $k = 0, 1, \ldots, n-1$.

The uniform distribution $\mathcal{U}[0, 1]$ is characterised by the fact that $U \sim \mathcal{U}[0, 1]$ satisfies

$$P(U \in [a, b]) = b - a$$

for all $0 \leq a \leq b \leq 1$. Also, since U is a continuous distribution, we have $P(U = x) = 0$ for all $x \in [0, 1]$ and thus the boundary points of the interval $[a, b]$ can be included or excluded without changing the probability. Using these results, we find

$$P(X = k) = P\left(U \in \left[\frac{k}{n}, \frac{k+1}{n}\right)\right) = \frac{k+1}{n} - \frac{k}{n} = \frac{1}{n}$$

for all $k = 0, 1, \ldots, n-1$. This completes the proof. □

Another common problem related to discrete distributions is the problem of constructing random events which occur with a given probability p. Such events will, for example, be needed in the rejection algorithms considered in Section 1.4. There are many fascinating aspects to this problem, but here we will restrict ourselves to the simplest case where the probability p is known explicitly and where we have access to $\mathcal{U}[0, 1]$-distributed random variables. This case is considered in the following lemma.

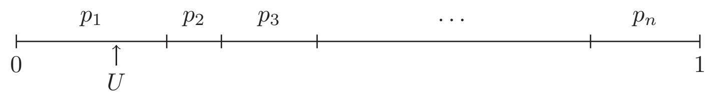
Lemma 1.9 Let $p \in [0, 1]$ and $U \sim \mathcal{U}[0, 1]$ and define the event E as $E = \{U \leq p\}$. Then $P(E) = p$.

Proof We have

$$P(E) = P(U \leq p) = P\left(U \in [0, p]\right) = p - 0 = p.$$

This completes the proof. □

The idea underlying lemmas 1.8 and 1.9 can be generalised to sample from arbitrary distributions on a finite set A. Let $A = \{a_1, \ldots, a_n\}$ where $a_i \neq a_j$ for $i \neq j$ and let $p_1, \ldots, p_n \geq 0$ be given with $\sum_{i=1}^n p_i = 1$. Assume that we want to generate random values $X \in A$ with $P(X = a_i) = p_i$ for $i = 1, 2, \ldots, n$. Since the p_i sum up to 1, we can split the unit interval [0, 1] into disjoint sub-intervals lengths $p_1, \ldots, p_n$.



With this arrangement, if we choose $U \in [0, 1]$ uniformly, the value of U lies in the ith subinterval with probability p_i. Thus, we can choose X to be the a_i corresponding to the subinterval which contains U. This idea is formalised in the following lemma.

Lemma 1.10 Assume $A = \{a_i \mid i \in I\}$ where either $I = \{1, 2, \ldots, n\}$ for some $n \in \mathbb{N}$ or $I = \mathbb{N}$, and where $a_i \neq a_j$ whenever $i \neq j$. Let $p_i \geq 0$ be given for $i \in I$ with $\sum_{i \in I} p_i = 1$. Finally let $U \sim \mathcal{U}[0, 1]$ and define

$$K = \min \left\{ k \in I \,\middle|\, \sum_{i=1}^{k} p_i \geq U \right\}. \tag{1.2}$$

Then $X = a_K \in A$ satisfies $P(X = a_k) = p_k$ for all $k \in I$.

Proof We have

$$\begin{aligned} P(X = a_k) = P(K = k) &= P\left(\sum_{i=1}^{k-1} p_i < U, \sum_{i=1}^{k} p_i \geq U\right) \\ &= P\left(U \in \left(\sum_{i=1}^{k-1} p_i, \sum_{i=1}^{k} p_i\right]\right) = \sum_{i=1}^{k} p_i - \sum_{i=1}^{k-1} p_i = p_k \end{aligned}$$

for all $k \in I$, where we interpret the sum $\sum_{i=1}^{0} p_i$ for $k = 1$ as 0. This completes the proof. □

The numerical method described by lemma 1.10 requires that we find the index K of the subinterval which contains U. The most efficient way to do this is to find a function φ which maps the boundaries of the subintervals to consecutive integers and then to consider the rounded value $\lfloor \varphi(I) \rfloor$. This approach is taken in lemma 1.8 and also in the following example.

Example 1.11 The geometric distribution, describing the number X of individual trials with probability p until the first success, has probability weights $P(X = i) = p^{i-1}(1 - p) = p_i$ for $i \in \mathbb{N}$. We can use lemma 1.10 with $a_i = i$ for all $i \in \mathbb{N}$ to generate samples from this distribution.

For the weights p_i, the value sum in equation (1.2) can be determined explicitly: using the formula for geometric sums we find

$$\sum_{i=1}^{k} p_i = (1 - p) \sum_{i=1}^{k} p^{i-1} = (1 - p) \frac{1 - p^k}{1 - p} = 1 - p^k.$$

Thus, we can rewrite the event $\sum_{i=1}^{k} p_i \geq U$ as follows:

$$\begin{aligned}\left\{ U \leq \sum_{i=1}^{k} p_i \right\} &= \left\{ U \leq 1 - p^k \right\} \\ &= \left\{ p^k \leq 1 - U \right\} \\ &= \{ k \log(p) \leq \log(1 - U) \} \\ &= \left\{ k \geq \frac{\log(1 - U)}{\log(p)} \right\}.\end{aligned}$$

In the last expression, we had to change the $\leq$ sign into a $\geq$ sign, since we divided by the negative number $\log(p)$. By definition, the K from equation (1.2) is the smallest integer such that $\sum_{i=1}^{k} p_i \geq U$ is satisfied and thus the smallest integer greater than or equal to $\log(1 - U)/\log(p)$. Thus, the value

$$X = a_K = K = \left\lceil \frac{\log(1 - U)}{\log(p)} \right\rceil,$$

where $\lceil \cdot \rceil$ denotes the operation of rounding up a number to the nearest integer, is geometrically distributed with parameter p.

1.3 The inverse transform method

The inverse transform method is a method which can be applied when the target distribution is one-dimensional, that is to generate samples from a prescribed target

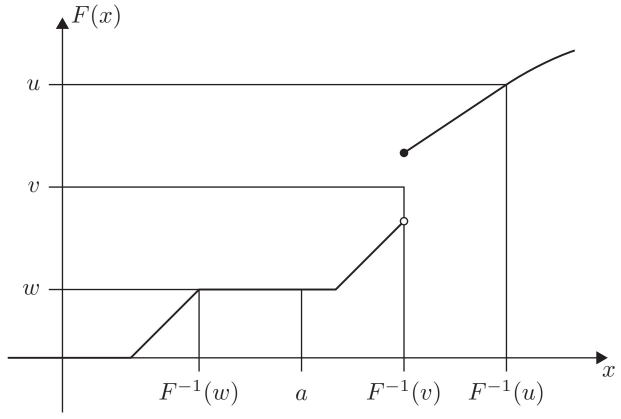


Figure 1.2 Illustration of the inverse F^{-1} of a CDF F. At level u the function F is continuous and injective; here F^{-1} coincides with the usual inverse of a function. The value v falls in the middle of a jump of F and thus has no preimage; $F^{-1}(v)$ is the preimage of the right-hand limit of F and $F(F^{-1}(v)) \neq v$. At level w the function F is not injective, several points map to w; the preimage $F^{-1}(w)$ is the left-most of these points and we have, for example, $F^{-1}(F(a)) \neq a$.

distribution on the real numbers $\mathbb{R}$. The method uses the cumulative distribution function (CDF) (see Section A.1) to specify the target distribution and can be applied for distributions which have no density.

Definition 1.12 Let F be a distribution function. Then the *inverse* of F is defined by

$$F^{-1}(u) = \inf\left\{x \in \mathbb{R} \;\middle|\; F(x) \geq u\right\}$$

for all $u \in (0, 1)$.

The definition of the inverse of a distribution function is illustrated in Figure 1.2. In the case where F is bijective, that is when F is strictly monotonically increasing and has no jumps, F^{-1} is just the usual inverse of a function. In this case we can find $F^{-1}(u)$ by solving the equation $F(x) = u$ for x. The following algorithm can be used to generate samples from a given distribution, whenever the inverse F^{-1} of the distribution function can be determined.

Algorithm 1.13 (inverse transform method)
```
input:
    the inverse $F^{-1}$ of a CDF $F$
randomness used:
    $U \sim \mathcal{U}[0, 1]$
```

```
output:
  $X \sim F$
1: generate $U \sim \mathcal{U}[0, 1]$
2: return $X = F^{-1}(U)$
```

This algorithm is very simple and it directly transforms $\mathcal{U}[0, 1]$-distributed samples into samples with distribution function F. The following proposition shows that the samples X generated by algorithm 1.13 have the correct distribution.

Proposition 1.14 Let $F : \mathbb{R} \to [0, 1]$ be a distribution function and $U \sim \mathcal{U}[0, 1]$. Define $X = F^{-1}(U)$. Then X has distribution function F.

Proof Using the definitions of X and F^{-1} we find

$$P(X \leq a) = P\left(F^{-1}(U) \leq a\right) = P\left(\inf\{\, x \mid F(x) \geq U \,\} \leq a\right).$$

Since $\inf\{\, x \mid F(x) \geq U \,\} \leq a$ holds if and only if $F(a) \geq U$, we can conclude

$$P(X \leq a) = P\left(F(a) \geq U\right) = F(a)$$

where the final equality comes from the definition of the uniform distribution on the interval [0, 1]. □

Example 1.15 The exponential distribution $\text{Exp}(\lambda)$ has density

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0 \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Using integration, we find the corresponding CDF as

$$F(a) = \int_{-\infty}^{a} f(x)\, dx = \int_{0}^{a} \lambda e^{-\lambda x}\, dx = -e^{-\lambda x}\Big|_{x=0}^{a} = 1 - e^{-\lambda a}$$

for all $a \geq 0$. Since this function is strictly monotonically increasing and continuous, F^{-1} is the usual inverse of F. We have

$$1 - e^{-\lambda x} = u \quad \iff \quad -\lambda x = \log(1 - u) \quad \iff \quad x = -\frac{\log(1 - u)}{\lambda}$$

and thus $F^{-1}(u) = -\log(1 - u)/\lambda$ for all $u \in (0, 1)$. Now assume $U \sim \mathcal{U}[0, 1]$. Then proposition 1.14 gives that $X = -\log(1 - U)/\lambda$ is $\text{Exp}(\lambda)$-distributed. Thus we have found a method to transform $\mathcal{U}[0, 1]$ random variables into $\text{Exp}(\lambda)$-distributed random variables. The method can be further simplified by using the observation that U and $1 - U$ have the same distribution: if $U \sim \mathcal{U}[0, 1]$, then $-\log(U)/\lambda \sim \text{Exp}(\lambda)$.

JWST355-c01 JWST355-Voss Printer: Yet to Come July 27, 2013 13:27 Trim: 229mm × 152mm

Example 1.16 The *Rayleigh distribution* with parameter $\sigma > 0$ has density

$$f(x) = \begin{cases} \frac{x}{\sigma^2} e^{-x^2/2\sigma^2} & \text{if } x \geq 0 \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

For this distribution we find

$$F(a) = \int_0^a \frac{x}{\sigma^2} e^{-x^2/2\sigma^2}\, dx = -e^{-x^2/2\sigma^2}\Big|_{x=0}^{a} = 1 - e^{-a^2/2\sigma^2}$$

for all $a \geq 0$. Solving the equation $u = F(x) = 1 - e^{-a^2/2\sigma^2}$ for x we find the inverse $F^{-1}(u) = x = \sqrt{-2\sigma^2 \log(1-u)}$. By proposition 1.14 we know that $X = \sqrt{-2\sigma^2 \log(1-U)}$ has density f if we choose $U \sim \mathcal{U}[0, 1]$. As in the previous example, we can also write U instead of $1 - U$.

Example 1.17 Let X have density

$$f(x) = \begin{cases} 3x^2 & \text{for } x \in [0, 1] \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$F(a) = \int_{-\infty}^{a} f(x)\, dx = \begin{cases} 0 & \text{if } a < 0 \\ a^3 & \text{if } 0 \leq a < 1 \text{ and} \\ 1 & \text{for } 1 \leq a. \end{cases}$$

Since F maps $(0, 1)$ into $(0, 1)$ bijectively, F^{-1} is given by the usual inverse function and consequently $F^{-1}(u) = u^{1/3}$ for all $u \in (0, 1)$. Thus, by proposition 1.14, if $U \sim \mathcal{U}[0, 1]$, the cubic root $U^{1/3}$ has the same distribution as X.

Example 1.18 Let X be discrete with $P(X = 0) = 0.6$ and $P(X = 1) = 0.4$. Then

$$F(a) = \begin{cases} 0 & \text{if } a < 0 \\ 0.6 & \text{if } 0 \leq a < 1 \text{ and} \\ 1 & \text{if } 1 \leq a. \end{cases}$$

Using the definition of F^{-1} we find

$$F^{-1}(u) = \begin{cases} 0 & \text{if } 0 < u \leq 0.6 \text{ and} \\ 1 & \text{if } 0.6 < u < 1. \end{cases}$$

By proposition 1.14 we can construct a random variable X with the correct distribution from $U \sim \mathcal{U}[0, 1]$, by setting

$$X = \begin{cases} 0 & \text{if } U \leq 0.6 \text{ and} \\ 1 & \text{if } U > 0.6. \end{cases}$$

The inverse transform method can always be applied when the inverse F^{-1} is easy to evaluate. For some distributions like the normal distribution this is not the case, and the inverse transform method cannot be applied directly. The method can be applied (but may not be very useful) for discrete distributions such as in example 1.18. The main restriction of the inverse transform method is that distribution functions only exist in the one-dimensional case. For distributions on $\mathbb{R}^d$ where $d > 1$, more sophisticated methods are required.

1.4 Rejection sampling

The rejection sampling method is a more advanced, and very popular, method for random number generation. Several aspects make this method different from basic methods such as inverse transform method discussed in the previous section. First, rejection sampling is not restricted to $\mathcal{U}[0, 1]$-distributed input samples. The method is often used in multi-stage approaches where different methods are used to generate samples of approximately the correct distribution and then rejection sampling is used to convert these samples to follow the target distribution exactly. Secondly, while we state the method here only for distributions on the Euclidean space $\mathbb{R}^d$, the rejection sampling method can be generalised to work on very general spaces. Finally, a random and potentially large number of input samples is required to generate one output sample in the rejection method. As a consequence, the efficiency of the method becomes a concern.

1.4.1 Basic rejection sampling

In this section we introduce the fundamental idea that all rejection algorithms are based on. We start by presenting the basic algorithm which forms the prototype of the methods presented later.

Algorithm 1.19 (basic rejection sampling)

```
input:
    a probability density $g$ (the *proposal density*),
    a function $p$ with values in [0,1] (the *acceptance probability*)
randomness used:
    $X_n$ i.i.d. with density $g$ (the *proposals*),
    $U_n \sim \mathcal{U}[0, 1]$ i.i.d.
```

JWST355-c01 JWST355-Voss Printer: Yet to Come July 27, 2013 13:27 Trim: 229mm × 152mm

output:
  a sequence of i.i.d. random variables with density

$$f(x) = \frac{1}{Z} p(x) g(x) \quad \text{where} \quad Z = \int p(x) g(x) \, dx. \tag{1.3}$$

```
1: for $n = 1, 2, 3, \ldots$ do
2:     generate $X_n$ with density $g$
3:     generate $U_n \sim \mathcal{U}[0, 1]$
4:     if $U_n \leq p(X_n)$ then
5:         output $X_n$
6:     end if
7: end for
```

The effect of the random variables U_n in the algorithm is to randomly decide whether to output or to ignore the value X_n: the value X_n is output with probability $p(X_n)$, and using the trick from lemma 1.9 we use the event $\{U \leq p(X_n)\}$ to decide whether or not to output the value. In the context of rejection sampling, the random variables X_n are called *proposals*. If the proposal X_n is chosen for output, that is if $U_n \leq p(X_n)$, we say that X_n is *accepted*, otherwise we say that X_n is *rejected*.

Proposition 1.20 For $k \in \mathbb{N}$, let X_{N_k} denote the kth output of algorithm 1.19. Then the following statements hold:

(a) The elements of the sequence $(X_{N_k})_{k\in\mathbb{N}}$ are i.i.d. with density f given by (1.3).

(b) Each proposal is accepted with probability Z; the number of proposals required to generate each X_{N_k} is geometrically distributed with mean $1/Z$.

Proof For fixed n, the probability of accepting X_n is

$$P\left(U_n \leq p(X_n)\right) = \int p(x) g(x) \, dx = Z, \tag{1.4}$$

where Z is the constant defined in equation (1.3). Since the decisions whether to accept X_n for different n are independent, the time until the first success is geometrically distributed with mean $1/Z$ as required. This completes the proof of the second statement.

For the proof of the first statement, first note that the indices $N_1, N_2, N_3, \ldots$ of the accepted X_n are random. If we let $N_0 = 0$, we can write

$$N_k = \min\left\{ n \in \mathbb{N} \;\middle|\; n > N_{k-1}, U_n \leq p(X_n) \right\}$$

for all $k \in \mathbb{N}$. If we consider the distribution of X_{N_k} conditional on the value of N_{k-1}, we find

$$\begin{aligned} &P\big(X_{N_k} \in A \,\big|\, N_{k-1} = n\big) \\ &\quad= \sum_{m=1}^{\infty} P\left(N_k = n+m, X_{n+m} \in A \,\big|\, N_{k-1} = n\right) \\ &\quad= \sum_{m=1}^{\infty} P\Big(U_{n+1} > p(X_{n+1}), \ldots, U_{n+m-1} > p(X_{n+m-1}), \\ &\qquad\qquad U_{n+m} \leq p(X_{n+m}), X_{n+m} \in A \,\Big|\, N_{k-1} = n\Big) \\ &\quad= \sum_{m=1}^{\infty} P\left(U_{n+1} > p(X_{n+1})\right) \cdots P\left(U_{n+m-1} > p(X_{n+m-1})\right) \cdot \\ &\qquad\qquad P\left(U_{n+m} \leq p(X_{n+m}), X_{n+m} \in A\right). \end{aligned}$$

Here we used the fact that all the probabilities considered in the last expression are independent of the value of N_{k-1}. Similar to (1.4) we find

$$P\left(U_n \leq p(X_n), X_n \in A\right) = \int_A p(x)g(x)\,dx$$

and consequently we have

$$\begin{aligned} &P\left(X_{N_k} \in A \,\big|\, N_{k-1} = n\right) \\ &\quad= \sum_{m=1}^{\infty} (1-Z)^{m-1} \int_A p(x)g(x)\,dx \\ &\quad= \frac{1}{Z} \int_A p(x)g(x)\,dx \end{aligned}$$

by the geometric series formula. Since the right-hand side does not depend on n, we can conclude

$$P\left(X_{N_k} \in A\right) = \frac{1}{Z} \int_A p(x)g(x)\,dx$$

and thus we find that X_{N_k} has density pg/Z.

To see that the X_{N_k} are independent we need to show that

$$P\left(X_{N_1} \in A_1, \ldots, X_{N_k} \in A_k\right) = \prod_{i=1}^{k} P\left(X_{N_i} \in A_i\right)$$

for all sets $A_1, \ldots, A_k$ and for all $k \in \mathbb{N}$. This can be done by summing up the probabilities for the cases $N_1 = n_1, \ldots, N_k = n_k$, similar to the first part of the proof, but we omit this tedious calculation here. □

Example 1.21 Let $X \sim \mathcal{U}[-1, +1]$ and accept X with probability

$$p(X) = \sqrt{1 - X^2}.$$

Then, by proposition 1.20, the accepted samples have density

$$f(x) = \frac{1}{Z} p(x) g(x) = \frac{1}{Z} \cdot \sqrt{1 - x^2} \cdot \frac{1}{2} \mathbb{1}_{[-1,+1]}(x),$$

where we use the indicator function notation from equation (A.7) to get the abbreviation

$$\mathbb{1}_{[-1,+1]}(x) = \begin{cases} 1 & \text{if } x \in [-1, +1] \text{ and} \\ 0 & \text{otherwise} \end{cases}$$

and

$$Z = \int_{\mathbb{R}} \sqrt{1 - x^2} \cdot \frac{1}{2} \mathbb{1}_{[-1,+1]}(x)\, dx = \frac{1}{2} \int_{-1}^{1} \sqrt{1 - x^2}\, dx = \frac{1}{2} \cdot \frac{\pi}{2} = \frac{\pi}{4}.$$

Combining these results, we find that the density f of accepted samples is given by

$$f(x) = \frac{2}{\pi} \sqrt{1 - x^2}\, \mathbb{1}_{[-1,+1]}(x).$$

The graph of the density f forms a semicircle and the resulting distribution is known as Wigner's semicircle distribution.

One important property of the rejection algorithm 1.19 is that none of the steps in the algorithm makes any reference to the normalisation constant Z. Thus, we do not need to compute the value of Z in order to apply this algorithm. We will see that this fact, while looking like a small detail at first glance, is extremely useful in practical applications.

1.4.2 Envelope rejection sampling

The basic rejection sampling algorithm 1.19 from the previous section is usually applied by choosing the acceptance probabilities p so that the density f of the output values, given by (1.3), coincides with a given target distribution. The resulting algorithm can be written as in the following.

Algorithm 1.22 (envelope rejection sampling)

```
input:
    a function $f$ with values in $[0, \infty)$ (the non-normalised *target density*),
    a probability density $g$ (the *proposal density*),
    a constant $c > 0$ such that $f(x) \leq c\, g(x)$ for all $x$
randomness used:
    $X_n$ i.i.d. with density $g$ (the *proposals*),
    $U_n \sim \mathcal{U}[0, 1]$ i.i.d.
output:
    a sequence of i.i.d. random variables with density
        $$\tilde{f}(x) = \frac{1}{Z_f} f(x) \quad \text{where} \quad Z_f = \int f(x)\, dx$$

1: for $n = 1, 2, 3, \ldots$ do
2:     generate $X_n$ with density $g$
3:     generate $U_n \sim \mathcal{U}[0, 1]$
4:     if $cg(X_n)U_n \leq f(X_n)$ then
5:         output $X_n$
6:     end if
7: end for
```

The assumption in the algorithm is that we can already sample from the distribution with probability density g, but we would like to generate samples from the distribution with density $\tilde{f}$ instead. Normally, f will be chosen to be a probability density and in this case we have $\tilde{f} = f$, but in some situations the normalising constant Z_f is difficult to obtain and due to the distinction between f and $\tilde{f}$, in these situations the algorithm can still be applied. The rejection mechanism employed in algorithm 1.22 is illustrated in Figure 1.3. The function cg is sometimes called an 'envelope' for f.

Proposition 1.23 Let X_{N_k} for $k \in \mathbb{N}$ denote the kth output value of algorithm 1.22 with (non-normalised) target density f. Then the following statements hold:

(a) The elements of the sequence $(X_{N_k})_{k\in\mathbb{N}}$ are i.i.d. with density $\tilde{f}$.

(b) Each proposal is accepted with probability Z_f/c; the number $M_k = N_k - N_{k-1}$ of proposals required to generate each X_{N_k} is geometrically distributed with mean $E(M_k) = c/Z_f$.

Proof Algorithm 1.22 coincides with algorithm 1.19 where the acceptance probability p is chosen as

$$p(x) = \begin{cases} \frac{f(x)}{cg(x)} & \text{if } g(x) > 0 \text{ and} \\ 1 & \text{otherwise.} \end{cases}$$

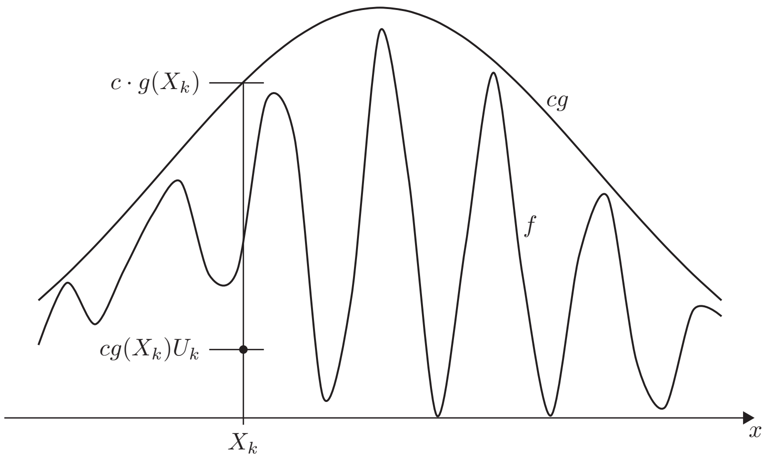


Figure 1.3 Illustration of the envelope rejection sampling method from algorithm 1.22. The proposal (X_k, $cg(X_k)$ U_k) is accepted, if it falls into the area underneath the graph of f. In Section 1.4.4 we will see that the proposal is distributed uniformly on the area under the graph of cg.

In this situation, the normalisation constant Z from (1.3) is given by:

$$Z = \int p(x)g(x)\,dx = \int \frac{f(x)}{cg(x)}\,g(x)\,dx = \frac{1}{c}\int f(x)\,dx = Z_f/c.$$

From proposition 1.20 we then know that the output of algorithm 1.19 is an i.i.d. sequence with density

$$\frac{1}{Z}pg = \frac{c}{Z_f}\frac{f}{cg}g = \frac{1}{Z_f}f$$

and that the required number of proposals to generate one output sample is geometrically distributed with mean $1/Z = c/Z_f$. □

Example 1.24 We can use rejection sampling to generate samples from the half-normal distribution with density

$$f(x) = \begin{cases} \frac{2}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) & \text{if } x \geq 0 \text{ and} \\ 0 & \text{otherwise.} \end{cases} \tag{1.5}$$

If we assume that the proposals are Exp(λ)-distributed, then the density of the proposals is

$$g(x) = \begin{cases} \lambda \exp(-\lambda x) & \text{if } x \geq 0 \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

In order to apply algorithm 1.22 we need to determine a constant $c > 0$ such that $f(x) \leq cg(x)$ for all $x \in \mathbb{R}$. For $x < 0$ we have $f(x) = g(x) = 0$. For $x \geq 0$ we have

$$\frac{f(x)}{g(x)} = \frac{2}{\sqrt{2\pi}\lambda} \exp\left(-\frac{x^2}{2} + \lambda x\right).$$

It is easy to check that the quadratic function $-x^2/2 + \lambda x$ attains its maximum at $x = \lambda$. Thus we have

$$\frac{f(x)}{g(x)} \leq c^*$$

for all $x \geq 0$, where

$$c^* = \frac{2}{\sqrt{2\pi}\lambda} \exp\left(-\frac{\lambda^2}{2} + \lambda \cdot \lambda\right) = \sqrt{\frac{2}{\pi\lambda^2}} \exp\left(\lambda^2/2\right).$$

Consequently, any $c \geq c^*$ satisfies the condition $f \leq cg$. From proposition 1.23 we know that the average number of proposals required for generating one sample, and thus the computational cost, is proportional to c. Thus we should choose c as small as possible and $c = c^*$ is the optimal choice.

Given our choice of g and c, the acceptance criterion from algorithm 1.22 can be simplified as follows:

$$\begin{aligned} & cg(x)\,U \leq f(x) \\ \iff \quad & \sqrt{\frac{2}{\pi\lambda^2}} \exp\left(\frac{\lambda^2}{2}\right) \lambda \exp(-\lambda x)\,U \leq \frac{2}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \\ \iff \quad & U \leq \exp\left(-\frac{x^2}{2} + \lambda x - \frac{\lambda^2}{2}\right) \\ \iff \quad & U \leq \exp\left(-\frac{1}{2}(x-\lambda)^2\right). \end{aligned}$$

This leads to the following algorithm for generating samples from the half-normal distribution:

```
1: for $n = 1, 2, 3, \ldots$ do
2:     generate $X_n \sim \mathrm{Exp}(\lambda)$
3:     generate $U_n \sim \mathcal{U}[0, 1]$
```

```
4:    if $U_n \leq \exp(-\frac{1}{2}(X_n - \lambda)^2)$ then
5:        output $X_n$
6:    end if
7: end for
```

Finally, since the density f is the density of a standard-normal distribution conditioned on being positive, and since the normal distribution is symmetric, we can generate standard normal distributed values by randomly choosing X_n or $-X_n$, both with probability $1/2$, for each accepted sample.

In algorithm 1.22, we can choose the density g of the proposal distribution in order to maximise efficiency of the method. The only constraint is that we need to be able to find the constant c. This condition implies, for example, that the support of g cannot be smaller than the support of f, that is we need $g(x) > 0$ whenever $f(x) > 0$. The average cost of generating one sample is given by the average number of proposals required times the cost for generating each proposal. Therefore the algorithm is efficient, if the following two conditions are satisfied:

(a) There is an efficient method to generate the proposals X_i. This affects the choice of the proposal density g.

(b) The average number c/Z_f of proposals required to generate one sample is small. This number is influenced by the value of c and, since the possible choices of c depend on g, also by the proposal density g.

1.4.3 Conditional distributions

The conditional distribution $P_{X|X\in A}$ corresponds to the remaining randomness in X when we already know that $X \in A$ occurred (see equation (A.4) for details). Sampling from a conditional distribution can be easily done by rejection sampling. The basic result is the following.

Algorithm 1.25 (rejection sampling for conditional distributions)

```
input:
  a set $A$ with $P(X \in A) > 0$
randomness used:
  a sequence $X_n$ of i.i.d. copies of $X$ (the *proposals*)
output:
  a sequence of i.i.d. random variables with distribution $P_{X|X\in A}$
1: for $n = 1, 2, 3, \ldots$ do
2:    generate $X_n$
3:    if $X_n \in A$ then
4:        output $X_n$
5:    end if
6: end for
```

Proposition 1.26 Let X be a random variable and let A be a set. Furthermore, let X_{N_k} for $k \in \mathbb{N}$ denote the kth output value of algorithm 1.25. Then the following statements hold:

(a) The elements of the sequence $(X_{N_k})_{k\in\mathbb{N}}$ are i.i.d. and satisfy

$$P(X_{N_k} \in B) = P(X \in B \mid X \in A)$$

for all $k \in \mathbb{N}$ and all sets B.

(b) The number $M_k = N_k - N_{k-1}$ of proposals required to generate each X_{N_k} is geometrically distributed with mean $E(M_k) = 1/P(X \in A)$.

Proof Algorithm 1.25 is a special case of algorithm 1.19 where the acceptance probability is chosen as $p(x) = \mathbb{1}_A(x)$. For this choice of p, the decision whether or not to accept the proposal given the value of X_n is deterministic and thus we can omit generation of the auxiliary random variables U_n in the algorithm.

Now assume that the distribution of X has a density g. Using equation (1.3) we then find

$$Z = \int \mathbb{1}_A(x) g(x)\, dx = P(X \in A)$$

and by proposition 1.20 we have

$$P(X_{N_k} \in B) = \frac{\int_B \mathbb{1}_A(x) g(x)\, dx}{Z} = \frac{P(X \in B \cap A)}{P(X \in A)} = P(X \in B \mid X \in A).$$

A similar proof gives the result in the case where X does not have a density. This completes the proof of the first statement of the proposition. The second statement is a direct consequence of proposition 1.20. □

The method presented in algorithm 1.25 works well if $p = P(X \in A)$ is not too small; the time required for producing a single output sample is proportional to $1/p$.

Example 1.27 We can use algorithm 1.25 to generate samples $X \sim \mathcal{N}(0, 1)$, conditioned on $X \geq a$. We simply have to repeat the following two steps until enough samples are output:

(a) generate $X \sim \mathcal{N}(0, 1)$;

(b) if $X \geq a$, output X.

The efficiency of this method depends on the value of a. The following table shows the average number $\mathbb{E}(N_a)$ of samples required to generate one output sample for different values of a, rounded to the nearest integer:

a	1	2	3	4	5	6
$\mathbb{E}(N_a)$	6	44	741	31 574	3 488 556	1 013 594 692

The table shows that the method will be slow even for moderate values of a. For $a \geq 5$ the required number of samples is so large that the method will likely be no longer practical.

For conditions with very small probabilities, rejection sampling can still be used to generate samples from the conditional distribution, but we have to use the full rejection sampling algorithm 1.22 instead of the simplified version from algorithm 1.25. This is illustrated in the following example.

Example 1.28 We can use algorithm 1.22 to generate samples from the conditional distribution of $X \sim \mathcal{N}(0, 1)$, conditioned on $X \geq a > 0$. The density of the conditional distribution is

$$\tilde{f}(x) = \frac{1}{Z} \exp(-x^2/2) \mathbb{1}_{[a,\infty]}(x) = \frac{1}{Z} f(x),$$

where Z is the normalising constant (we have included the pre-factor $1/\sqrt{2\pi}$ into Z to simplify notation).

We can sample from this distribution using proposals of the form $X = \tilde{X} + a$ where $\tilde{X} \sim \text{Exp}(\lambda)$. This proposal distribution has density

$$g(x) = \lambda \exp\left(-\lambda(x - a)\right) \mathbb{1}_{[a,\infty]}(x)$$

and we need to find a constant $c > 0$ such that $f(x) \leq cg(x)$ for all $x \geq a$. Also, we can still choose the parameter λ and, in order to maximise efficiency of the method, we should choose a value of λ such that the shape of g is as similar to the shape of f as possible. In order to achieve this, we choose c and λ so that at $x = a$ both the values and the derivatives of f and cg coincide (see Figure 1.4 for illustration). This leads to the conditions $\mathrm{e}^{-a^2/2} = f(a) = cg(a) = c\lambda$ and $-a\mathrm{e}^{-a^2/2} = f'(a) = cg'(a) = -c\lambda^2$ and solving these two equations for the two unknowns c and λ gives $\lambda = a$ and $c = \mathrm{e}^{-a^2/2}/a$.

Figure 1.4 indicates that for this choice of λ and c, condition $f \leq cg$ will be satisfied. Indeed we find

$$\begin{aligned}\frac{f(x)}{cg(x)} &= \frac{\exp(-x^2/2)}{1/a\ \exp(-a^2/2) \cdot a \exp\left(-a(x-a)\right)} \\ &= \exp\left(-x^2/2 + ax - a^2/2\right) = \exp\left(-\frac{(x-a)^2}{2}\right) \\ &\leq 1\end{aligned}$$

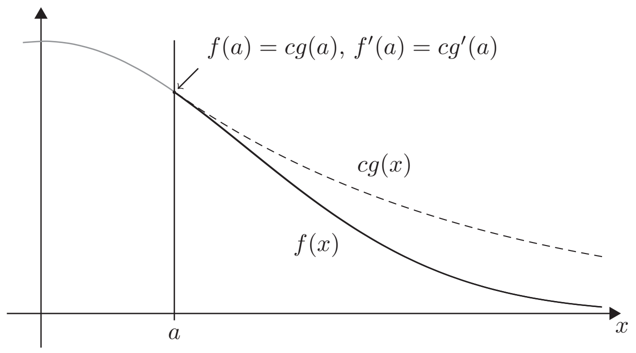


Figure 1.4 Illustration of the rejection mechanism from example 1.28. The graph shows the (scaled) proposal density cg, enveloping the (non-normalised) target density f.

and thus $f(x) \leq cg(x)$ for all $x \geq a$. Thus, we can apply algorithm 1.22 with proposal density g to generate samples from the distribution with density $\tilde{f}$. The resulting method consists of the following steps:

(a) generate $\tilde{X} \sim \mathrm{Exp}(a)$ and $U \sim \mathcal{U}[0, 1]$;

(b) let $X = \tilde{X} + a$;

(c) if $U \leq \exp(-(X-a)^2/2)$, output X.

From proposition 1.23 we know that the average number M_a of proposals required to generate one sample is

$$\mathbb{E}(M_a) = \frac{c}{\int_{\mathbb{R}} f(x)\,dx} = \frac{\exp(-a^2/2)/a}{\int_a^\infty \exp(-x^2/2)\,dx} = \frac{\exp(-a^2/2)}{a\sqrt{2\pi}\,(1-\Phi(a))},$$

where

$$\Phi(a) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{a} \exp(-x^2/2)\,dx$$

is the CDF of the standard normal distribution. The following table lists the value of $\mathbb{E}(M_a)$, rounded to three significant digits, for different values of a:

a	1	2	3	4	5	6
$\mathbb{E}(M_a)$	1.53	1.19	1.09	1.06	1.04	1.03

The table clearly shows that the resulting algorithm works well for large values of a: the steps required to generate one proposal are more complicated than for the method from example 1.27, but significantly fewer proposals are required.

1.4.4 Geometric interpretation

The rejection sampling method can be applied not only to the generation of random numbers, but also to the generation of random objects in arbitrary spaces. To illustrate this, in this section we consider the problem of sampling from the uniform distribution on subsets of the Euclidean space $\mathbb{R}^d$. We then use the resulting techniques to give an alternative proof of proposition 1.23, based on geometric arguments.

We write $|A|$ for the d-dimensional volume of a set $A \subseteq \mathbb{R}^d$. Then the cube $Q = [a, b]^3 \subseteq \mathbb{R}^3$ has volume $|Q| = (b - a)^3$, the unit circle $C = \{x \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 1\}$ has two-dimensional 'volume' π (area) and the line segment $[a, b] \subseteq \mathbb{R}$ has one-dimensional 'volume' $b - a$ (length). For more general sets A, the volume can be found by integration: we have

$$|A| = \int_{\mathbb{R}^d} \mathbb{1}_A(x)\, dx = \int \cdots \int \mathbb{1}_A(x_1, \ldots, x_d)\, dx_d \cdots dx_1.$$

Definition 1.29 A random variable X with values in $\mathbb{R}^d$ is uniformly distributed on a set $A \subseteq \mathbb{R}^d$ with $0 < |A| < \infty$, if

$$P(X \in B) = \frac{|A \cap B|}{|A|}$$

for all $B \subseteq \mathbb{R}^d$. As for real intervals, we use the notation $X \sim \mathcal{U}(A)$ to indicate that X is uniformly distributed on A.

The intuitive meaning of X being uniformly distributed on a set A is that X is a random element of A, and that all regions of A are hit by X equally likely. The probability of X falling into a subset of A only depends on the volume of this subset, but not on the location inside A.

Let $X \sim \mathcal{U}(A)$. From the definition we can derive simple properties of the uniform distribution: first we have

$$P(X \in A) = \frac{|A \cap A|}{|A|} = 1$$

and if A and B are disjoint we find

$$P(X \in B) = \frac{|A \cap B|}{|A|} = \frac{|\emptyset|}{|A|} = 0.$$

For general $B \subseteq \mathbb{R}^d$ we get

$$\begin{aligned} P(X \notin B) &= P(X \in \mathbb{R}^d \setminus B) \\ &= \frac{|A \cap (\mathbb{R}^d \setminus B)|}{|A|} = \frac{|A \setminus B|}{|A|} = \frac{|A| - |A \cap B|}{|A|} = 1 - \frac{|A \cap B|}{|A|}. \end{aligned}$$

Lemma 1.30 Let $A \subseteq \mathbb{R}^d$ be a set with volume $0 < |A| < \infty$. Then the uniform distribution $\mathcal{U}(A)$ has probability density $f = \mathbb{1}_A/|A|$ on $\mathbb{R}^d$.

Proof Let $X \sim \mathcal{U}(A)$. For $B \subseteq \mathbb{R}^d$ we have

$$P(X \in B) = \frac{|A \cap B|}{|A|} = \frac{1}{|A|} \int_{\mathbb{R}^d} \mathbb{1}_{A\cap B}(x)\,dx = \int_{\mathbb{R}^d} \mathbb{1}_B(x) \frac{\mathbb{1}_A(X)}{|A|}\,dx$$

and thus X has the given density f. □

Lemma 1.31 Let X be uniformly distributed on a set A, and let B be a set with $|A \cap B| > 0$. Then the conditional distribution $P_{X|X\in B}$ of X conditioned on the event $X \in B$ coincides with the uniform distribution on $A \cap B$.

Proof From the definition of the uniform distribution we get

$$\begin{aligned} P(X \in C \,|\, X \in B) &= \frac{P(X \in B \cap C)}{P(X \in B)} \\ &= \frac{|A \cap B \cap C|/|A|}{|A \cap B|/|A|} = \frac{|(A \cap B) \cap C|}{|A \cap B|}. \end{aligned}$$

Since this is the probability of a $\mathcal{U}(A \cap B)$-distributed random variable to hit the set C, the statement is proved. □

By combining the result of lemma 1.31 with the method given in algorithm 1.25, we can sample from the uniform distribution of every set which can be covered by a (union of) rectangles. This is illustrated in the following example.

Example 1.32 (uniform distribution on the circle) Let $X_n, Y_n \sim \mathcal{U}[-1, +1]$ be i.i.d. By exercise E1.10 the pairs (X_n, Y_n) are then uniformly distributed on the square $A = [0, 1] \times [0, 1]$. Now let $(Z_k)_{k\in\mathbb{N}}$ be the subsequence of all pairs (X_{n_k}, Y_{n_k}) which satisfy the condition

$$X_n^2 + Y_n^2 \leq 1.$$

Then $(Z_k)_{k\in\mathbb{N}}$ is an i.i.d. sequence, uniformly distributed on the unit circle $B = \{x \in \mathbb{R}^2 \mid |x| \leq 1\}$. The probability p to accept each sample is given by

$$p = P((X_n, V_n) \in B) = \frac{|B|}{|A|} = \frac{\pi 1^2}{2^2} = \frac{\pi}{4} \approx 78.5\%$$

and the number of proposals required to generate one sample is, on average, $1/p \approx 1.27$.

To conclude this section, we give an alternative proof of proposition 1.23. This proof is based on the geometric approach taken in this section and uses a connection between distributions with general densities on $\mathbb{R}^d$ and uniform distributions on $\mathbb{R}^{d+1}$, given in the following result.

Lemma 1.33 Let $f\colon \mathbb{R}^d \to [0,\infty)$ be a probability density and let

$$A = \bigl\{(x,y) \in \mathbb{R}^d \times [0,\infty) \bigm| 0 \leq y < f(x)\bigr\} \subseteq \mathbb{R}^{d+1}.$$

Then $|A| = 1$ and the following two statements are equivalent:

(a) (X,Y) is uniformly distributed on A.

(b) X is distributed with density f on $\mathbb{R}^d$ and $Y = f(X)U$ where $U \sim \mathcal{U}[0,1]$, independently of X.

Proof The volume of the set A can be found by integrating the 'height' $f(x)$ over all of $\mathbb{R}^d$. Since f is a probability density, we get

$$|A| = \int_{\mathbb{R}^d} f(x)\,dx = 1.$$

Assume first that (X,Y) is uniformly distributed on A and define $U = Y/f(X)$. Since $(X,Y) \in A$, we have $f(X) > 0$ with probability 1 and thus there is no problem in dividing by $f(X)$. Given sets $C \subseteq \mathbb{R}^d$ and $D \subseteq \mathbb{R}$ we find

$$\begin{aligned} P\,(X \in C, U \in D) &= P\left((X,Y) \in \bigl\{(x,y) \bigm| x \in C, y/f(x) \in D\bigr\}\right) \\ &= \left|A \cap \bigl\{(x,y) \bigm| x \in C, y/f(x) \in D\bigr\}\right| \\ &= \int_{\mathbb{R}^d} \int_0^{f(x)} \mathbb{1}_C(x)\mathbb{1}_D\left(y/f(x)\right)\,dy\,dx. \end{aligned}$$

Using the substitution $u = y/f(x)$ in the inner integral we get

$$\begin{aligned} P\,(X \in C, U \in D) &= \int_{\mathbb{R}^d} \int_0^1 \mathbb{1}_C(x)\mathbb{1}_D(u) f(x)\,du\,dx \\ &= \int_C f(x)\,dx \cdot \int_D \mathbb{1}_{[0,1]}(u)\,du. \end{aligned}$$

Therefore X and U are independent with densities f and $\mathbb{1}_{[0,1]}$, respectively.

For the converse statement assume now that the random variables X with density f and $U \sim \mathcal{U}[0, 1]$ are independent, and let $Y = f(X)U$. Furthermore let $C \subseteq \mathbb{R}^d$, $D \subseteq [0, \infty)$ and $B = C \times D$. Then we get

$$\begin{aligned} P\left((X, Y) \in B\right) &= P\left(X \in C, Y \in D\right) \\ &= \int_C P(Y \in D | X = x) f(x)\, dx \\ &= \int_C P\left(f(x)U \in D\right) f(x)\, dx \\ &= \int_C \frac{|D \cap [0, f(x)]|}{f(x)} f(x)\, dx \\ &= \int_C |D \cap [0, f(x)]|\; dx. \end{aligned}$$

On the other hand we have

$$\begin{aligned} |A \cap B| &= \int_{\mathbb{R}^d} \int_0^{f(x)} \mathbb{1}_B(x, y)\, dy\, dx \\ &= \int_{\mathbb{R}^d} \mathbb{1}_C(x) \int_0^{f(x)} \mathbb{1}_D(y)\, dy\, dx \\ &= \int_C |D \cap [0, f(x)]|\; dx \end{aligned}$$

and thus $P((X, Y) \in B) = |A \cap B|$. This shows that (X, Y) is uniformly distributed on A. □

An easy application of lemma 1.33 is to convert a uniform distribution of a subset of $\mathbb{R}^2$ to a distribution on $\mathbb{R}$ with a given density $f : [a, b] \to \mathbb{R}$. For simplicity, we assume first that f lives on a bounded interval $[a, b]$ and satisfies $f(x) \leq M$ for all $x \in [a, b]$. We can generate samples from the distribution with density f as follows:

(a) Let (X_k, Y_k) are be i.i.d., uniformly distributed on the rectangle $R = [a, b] \times [0, M]$.

(b) Consider the set $A = \{(x, y) \in R \bigm| y \leq f(x)\}$ and let $N = \min\{ k \in \mathbb{N} \mid X_k \in B \}$. By lemma 1.31, (X_N, Y_N) is uniformly distributed on A.

(c) By lemma 1.33, the value X_N is distributed with density f.

This procedure is illustrated in Figure 1.5.

In the situation of algorithm 1.22, that is when f is defined on an unbounded set, we cannot use proposals which are uniformly distributed on a rectangle anymore. A solution to this problem is to use lemma 1.33 a second time to obtain a suitable proposal distribution. This approach provides an alternative way of understanding algorithm 1.22: in the situation of algorithm 1.22, X_k has density g and U_k is uniformly distributed on $[0, 1]$. Then we know from lemma 1.33 that the

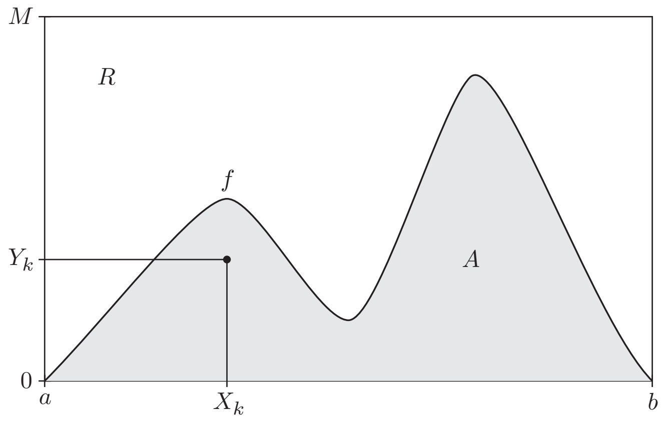


Figure 1.5 Illustration of the rejection sampling method where the graph of the target density is contained in a rectangle $R = [a, b] \times [0, M]$. In this case the proposals are uniformly distributed on the rectangle R and a proposal is accepted if it falls into the shaded region.

pair $(X_k, g(X_k)U_k)$ is uniformly distributed on the set $\{(x, v) \mid 0 \leq v < g(x)\}$. Consequently, $Z_k = (X_k, cg(X_k)U_k)$ is uniformly distributed on $A = \{(x, y) \mid 0 \leq y < cg(x)\}$. By lemma 1.31 and proposition 1.26, the accepted values are uniformly distributed on the set $B = \{(x, y) \mid 0 \leq y < f(x)\} \subseteq A$ and, applying lemma 1.33 again, we find that the X_k, conditional on being accepted, have density f. This argument can be made into an alternative proof of proposition 1.23.

1.5 Transformation of random variables

Samples from a wide variety of distributions can be generated by considering deterministic transformations of random variables. The inverse transform method, introduced in Section 1.3, is a special case of this technique where we transform a uniformly distributed random variable using the inverse of a CDF. In this section, we consider more general transformations.

The fundamental question we have to answer in order to generate samples by transforming a random variable is the following: if X is a random variable with values in $\mathbb{R}^d$ and a given distribution, and if $\varphi : \mathbb{R}^d \to \mathbb{R}^d$ is a function, what is the distribution of $\varphi(X)$? This question is answered in the following theorem.

Theorem 1.34 (transformation of random variables) Let $A, B \subseteq \mathbb{R}^d$ be open sets, $\varphi : A \to B$ be bijective and differentiable with continuous partial derivatives, and let X be a random variable with values in A. Furthermore let $g : B \to [0, \infty)$ be a probability density and define $f : \mathbb{R}^d \to \mathbb{R}$ by

$$f(x) = \begin{cases} g(\varphi(x)) \cdot |\det D\varphi(x)| & \text{if } x \in A \text{ and} \\ 0 & \text{otherwise.} \end{cases} \tag{1.6}$$

Then f is a probability density and the random variable X has density f if and only if $\varphi(X)$ has density g.

The matrix $D\varphi$ used in the theorem is the Jacobian of φ, as given in the following definition.

Definition 1.35 Let $\varphi : \mathbb{R}^d \to \mathbb{R}^d$ be differentiable. Then the *Jacobian matrix* $D\varphi$ is the $d \times d$ matrix consisting of the partial derivatives of φ: for $i, j = 1, 2, \ldots, d$ we have $D\varphi(x)_{ij} = \frac{\partial \varphi_i}{\partial x_j}(x)$.

Theorem 1.34 is a consequence of the substitution rule for integrals. Before we give the proof of theorem 1.34, we first state the substitution rule in the required form.

Lemma 1.36 (substitution rule for integrals) Let $A, B \subseteq \mathbb{R}^d$ be open sets, $f : B \to \mathbb{R}$ integrable, and φ: $A \to B$ be bijective and differentiable with continuous partial derivatives. Then

$$\int_B f(y)\,dy = \int_A f(\varphi(x))\,|\det D\varphi(x)|\,dx$$

where $D\varphi$ denotes the Jacobian matrix of φ.

A proof of lemma 1.36 can, for example, be found in the book by Rudin (1987, theorem 7.26). Using lemma 1.36 we can now give the proof of the transformation rule for random variables.

Proof (of theorem 1.34). By definition, the function f is positive and using lemma 1.36, we get

$$\int_{\mathbb{R}^d} f(x)\,dx = \int_A g(\varphi(x)) \cdot |\det D\varphi(x)|\,dx = \int_B g(y)\,dy = 1.$$

Thus f is a probability density.

Now assume that X is distributed with density f and let $C \subseteq B$. Then, by equation (A.8):

$$\begin{aligned} P(\varphi(X) \in C) &= \int_A \mathbb{1}_C(\varphi(x))\, f(x)\,dx \\ &= \int_A \mathbb{1}_C(\varphi(x))\, g(\varphi(x)) \cdot |\det D\varphi(x)|\,dx. \end{aligned}$$

Now we can apply lemma 1.36, again, to transform the integral over A into an integral over the set B: we find

$$P(\varphi(X) \in C) = \int_B \mathbb{1}_C(y) g(y)\, dy.$$

Since this equality holds for all sets C, the random variable $\varphi(X)$ has density g. The converse statement follows by reversing the steps in this argument. □

While theorem 1.34 is most powerful in the multidimensional case, it can be applied in the one-dimensional case, too. In this case the Jacobian matrix is a 1×1 matrix, that is a number, and we have $|\det D\varphi(x)| = |\varphi'(x)|$.

Example 1.37 (two-dimensional normal distribution) Assume that we want to sample from the two-dimensional standard normal distribution, that is from the distribution with density

$$g(x, y) = \frac{1}{2\pi} \exp\left(-\frac{x^2 + y^2}{2}\right).$$

Since g depends on (x, y) only via the squared length $x^2 + y^2$ of this vector, we try to simplify g using polar coordinates. The corresponding transformation φ is given by

$$\varphi(r, \theta) = (r\cos(\theta), r\sin(\theta))$$

for all $r > 0$, $\varphi \in (0, 2\pi)$. Note that we define φ only on the open set $A = (0, \infty) \times (0, 2\pi)$ in order to satisfy the requirement from theorem 1.34 that φ must be bijective. The resulting image set is $B = \varphi(A) = \mathbb{R}^2 \setminus \{(x, y) \mid x \geq 0, y = 0\}$, that is B is strictly smaller than $\mathbb{R}^2$ since it does not include the positive x-axis. This is not a problem, since the two-dimensional standard normal distribution hits the positive x-axis only with probability 0 and thus takes values in the set B with probability 1.

The Jacobian matrix of φ is given by

$$D\varphi(r, \theta) = \begin{pmatrix} \frac{\partial}{\partial r}\varphi_1 & \frac{\partial}{\partial \theta}\varphi_1 \\ \frac{\partial}{\partial r}\varphi_2 & \frac{\partial}{\partial \theta}\varphi_2 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -r\sin(\theta) \\ \sin(\theta) & r\cos(\theta) \end{pmatrix}$$

and thus we get $|\det D\varphi(r, \theta)| = \left| r\cos(\theta)^2 + r\sin(\theta)^2 \right| = r$. Using theorem 1.34 we have reduced the problem of sampling from a two-dimensional normal distribution to the problem of sampling from the density

$$f(r, \theta) = g(\varphi(r, \theta)) \cdot |\det D\varphi(r, \theta)| = \frac{1}{2\pi} \exp(-r^2/2) \cdot r$$

on $(0, \infty) \times (0, 2\pi)$.

JWST355-c01 JWST355-Voss Printer: Yet to Come July 27, 2013 13:27 Trim: 229mm × 152mm

The density $f(r, \theta)$ does not depend on θ and we can rewrite it as the product $f(r, \theta) = f_1(\theta) f_2(r)$ where $f_1(\theta) = 1/2\pi$ is the density of $\mathcal{U}[0, 2\pi]$ and $f_2(r) = r \exp(-r^2/2)$. From example 1.16 we know how to sample from the density f_2: if $U \sim \mathcal{U}[0, 1]$, then $R = \sqrt{-2\log(U)}$ has density f_2. Consequently, we can use the following steps to sample from the density g:

(a) Generate $\Theta \sim \mathcal{U}[0, 2\pi]$ and $U \sim \mathcal{U}[0, 1]$ independently.

(b) Let $R = \sqrt{-2\log(U)}$.

(c) Let $(X, Y) = \varphi(R, \Theta) = (R\cos(\Theta), R\sin(\Theta))$.

Then (R, Θ) has density f and, by theorem 1.34, the vector (X, Y) is standard normally distributed in $\mathbb{R}^2$. This method for converting pairs of uniformly distributed samples into pairs of normally distributed samples is called the Box–Muller transform (Box and Muller, 1958).

When the lemma is used to find sampling methods, usually g will be the given density of the distribution we want to sample from. Our task is then to find a transformation φ so that the density f described by (1.6) corresponds to a distribution we can already sample from. In this situation, φ should be chosen so that it 'simplifies' the given density g. In practice, finding a useful transformation φ often needs some experimentation.

Example 1.38 Assume we want to sample from the distribution with density $g(y) = \frac{3}{2}\sqrt{y} \cdot \mathbb{1}_{[0,1]}(y)$. We can cancel the square root from the definition of g by choosing $\varphi(x) = x^2$. Then we can apply theorem 1.34 with $A = B = [0, 1]$ and, since $|\det D\varphi(x)| = \left|\varphi'(x)\right| = 2x$, we get

$$f(x) = g(\varphi(x)) \cdot |\det D\varphi(x)| = \frac{3}{2}x \cdot 2x = 3x^2$$

for all $x \in [0, 1]$. From example 1.17 we already know how to generate samples from this density: If $U \sim \mathcal{U}[0, 1]$, then $X = U^{1/3}$ has density f and, by theorem 1.34, $Y = \varphi(X) = X^2 = U^{2/3}$ has density g.

An important application of the transformation rule from theorem 1.34 is the case where X and $\varphi(X)$ are both uniformly distributed. From the relation (1.6) we see that if X is uniformly distributed and if $|\det D\varphi|$ is constant, then $\varphi(X)$ is also uniformly distributed. For example, using this idea we can sometimes transform the problem of sampling from the uniform distribution on an unbounded set to the easier problem of sampling from the uniform distribution on a bounded set. This idea is illustrated in Figure 1.6. Combining this approach with lemma 1.33 results in the following general sampling method.

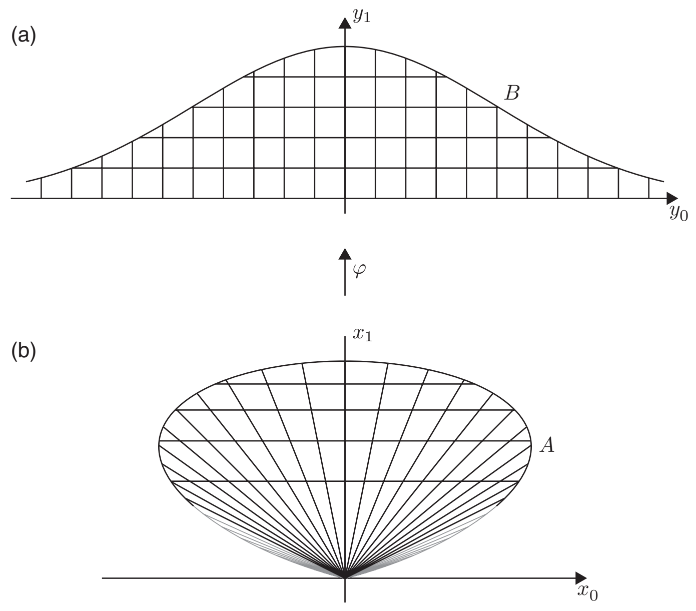


Figure 1.6 Illustration of the transformation used in the ratio-of-uniforms method. The map φ from equation (1.7) maps the bounded set shown in (b) into the unbounded set in (a). The areas shown in grey in (b) map into the tails in (a) (not displayed). Since φ preserves area (up to a constant), the uniform distribution on the set in (b) is mapped into the uniform distribution on the set in (a).

Theorem 1.39 (ratio-of-uniforms method) Let $f : \mathbb{R}^d \to \mathbb{R}_+$ be such that $Z = \int_{\mathbb{R}^d} f(x)\,dx < \infty$ and let X be uniformly distributed on the set

$$A = \left\{ (x_0, x_1, \ldots, x_d) \,\middle|\, x_0 > 0, \frac{x_0^{d+1}}{d+1} < f\left(\frac{x_1}{x_0}, \ldots, \frac{x_d}{x_0}\right) \right\} \subseteq \mathbb{R}_+ \times \mathbb{R}^d.$$

Then the vector

$$Y = \left(\frac{X_1}{X_0}, \ldots, \frac{X_d}{X_0}\right)$$

has density $\frac{1}{Z} f$ on $\mathbb{R}^d$.

Proof The proof is an application of the transformation rule for random variables. To see this, consider the set

$$B = \left\{ (y_0, y_1, \ldots, y_d) \,\middle|\, 0 < y_0 < f(y_1, \ldots, y_d)/Z \right\} \subseteq \mathbb{R}_+ \times \mathbb{R}^d$$

and define a transformation $\varphi : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}_+ \times \mathbb{R}^d$ by

$$\varphi(x_0, x_1, \ldots, x_d) = \left(\frac{Zx_0^{d+1}}{d+1}, \frac{x_1}{x_0}, \ldots, \frac{x_d}{x_0} \right). \tag{1.7}$$

We have $x \in A$ if and only if $\varphi(x) \in B$ and thus φ maps A onto B bijectively. Since the determinant of a triagonal matrix is the product of the diagonal elements, the Jacobian determinant of φ is given by

$$\det D\varphi(x) = \det \begin{pmatrix} Zx_0^d & & & \\ -\frac{x_1}{x_0^2} & \frac{1}{x_0} & & \\ \vdots & & \ddots & \\ -\frac{x_d}{x_0^2} & & & \frac{1}{x_0} \end{pmatrix} = Zx_0^d \frac{1}{x_0} \cdots \frac{1}{x_0} = Z$$

for all $x \in A$.

Since X is uniformly distributed on A, the density h of X satisfies

$$h(x) = \frac{1}{|A|} \mathbb{1}_A(x) = \frac{1}{Z|A|} \mathbb{1}_B(\varphi(x)) \cdot |\det D\varphi(x)|$$

and by theorem 1.34 the random variable $\varphi(X)$ then has density

$$g(y) = \frac{1}{Z|A|} \mathbb{1}_B(y)$$

for all $y \in \mathbb{R}_+ \times \mathbb{R}^d$. This density is constant on B and thus the random variable $\varphi(X)$ is uniformly distributed on B.

To complete the proof we note that the vector Y given in the statement of the theorem consists of the last d components of $\varphi(X)$. Using this observation, the claim now follows from lemma 1.33. □

Example 1.40 The Cauchy distribution has density

$$f(x) = \frac{1}{\pi(1 + x^2)}.$$

For this case, the set A from theorem 1.39 is

$$\begin{aligned} A &= \left\{ (x_0, x_1) \;\middle|\; x_0 > 0, \frac{x_0^2}{2} \le \frac{1}{\pi \left(1 + (\frac{x_1}{x_0})^2\right)} \right\} \\ &= \left\{ (x_0, x_1) \;\middle|\; x_0 > 0, \frac{\pi}{2} x_0^2 \le \frac{x_0^2}{x_0^2 + x_1^2} \right\} \\ &= \left\{ (x_0, x_1) \;\middle|\; x_0 > 0, x_0^2 + x_1^2 \le \frac{2}{\pi} \right\}, \end{aligned}$$

that is A is a semicircle in the x_0/x_1-plane. Since we can sample from the uniform distribution on the semicircle (see exercises E1.13 and E1.14), we can use the ratio-of-uniforms method from theorem 1.39 to sample from the Cauchy distribution. The following steps are required:

(a) Generate (X_0, X_1) uniformly on the semicircle A.

(b) Return $Y = X_1/X_0$.

Note that, since only the ratio between X_1 and X_0 is returned, A can be replaced by a semicircle with arbitrary radius instead of the radius $\sqrt{2/\pi}$ found above.

1.6 Special-purpose methods

There are many specialised methods to generate samples from specific distributions. These are often faster than the generic methods described in the previous sections, but can typically only be used for a single distribution. These specialised methods (optimised for speed and often quite complex) form the basis of the random number generators built into software packages. In contrast, the methods discussed in the previous sections are general purpose methods which can be used for a wide range of distributions when no pre-existing method is available.

1.7 Summary and further reading

In this chapter we have learned about various aspects of random number generation on a computer. The chapter started by considering the differences between ‘pseudo random number generators’ (the ones considered in this book) and ‘real random number generators’ (which we will not consider further). Using the LCG as an example, we have learned about properties of pseudo number generators. In particular we considered the rôle of the ‘seed’ to control reproducability of the generated numbers. Going beyond the scope of this book, a lot of information about LCGs and about testing of random number generators can be found in Knuth (1981). The Mersenne Twister, a popular modern PRNG, is described in Matsumoto and Nishimura (1998).

Building on the output of pseudo number generators, the following sections considered various general purpose methods for generating samples from different distributions. The methods we discussed here are the inverse transform method, the rejection sampling method, and the ratio-of-uniforms method (a special case of the transformation method). More information about rejection sampling and its extensions can be found in Robert and Casella (2004, Section 2.3). A specialised method for generating normally distributed random variables can, for example, be found in Marsaglia and Tsang (2000). Specialised methods for generating random numbers from various distributions are, for example, covered in Dagpunar (2007, Chapter 4) and Kennedy and Gentle (1980, Section 6.5).

An expository presentation of random number generation and many more references can be found in Gentle *et al.* (2004, Chapter II.2).

Exercises

E1.1 Write a function to implement the LCG. The function should take a length n, the parameters m, a and c as well as the seed X_0 as input and should return a vector $X = (X_1, X_2, \ldots, X_n)$. Test your function by calling it with the parameters $m = 8$, $a = 5$ and $c = 1$ and by comparing the output with the result from example 1.3.

E1.2 Given a sequence $X_1, X_2, \ldots$ of $\mathcal{U}[0, 1]$-distributed pseudo random numbers, we can use a scatter plot of (X_i, X_{i+1}) for $i = 1, \ldots, n - 1$ in order to try to assess whether the X_i are independent.

(a) Create such a plot using the built-in random number generator of R:

```
X <- runif(1000)
plot(X[1:999], X[2:1000], asp=1)
```

Can you explain the resulting plot?

(b) Create a similar plot, using your function LCG from exercise E1.1:

```
m <- 81
a <- 1
c <- 8
seed <- 0
X <- LCG(1000, m, a, c, seed)/m
plot(X[1:999], X[2:1000], asp=1)
```

Discuss the resulting plot.

(c) Repeat the experiment from (b) using the parameters $m = 1024$, $a = 401$, $c = 101$ and $m = 2^{32}$, $a = 1\,664\,525$, $c = 1\,013\,904\,223$. Discuss the results.

E1.3 One (very early) method for pseudo random number generation is von Neumann's middle square method (von Neumann, 1951). The method works as follows: starting with $X_0 \in \{0, 1, \ldots, 99\}$, define X_n for $n \in \mathbb{N}$ to be the middle two digits of the four-digit number X_{n-1}^2. If X_{n-1}^2 does not have four digits, it is padded with leading zeros. For example, if $X_0 = 64$, we have $X_0^2 = 4096$ and thus $X_1 = 09 = 9$. In the next step, we find $X_1^2 = 81 = 0081$ and thus $X_2 = 08 = 8$.

(a) Write a function which computes X_n from X_{n-1}.

(b) The output of the middle square method has loops. For example, once we have $X_N = 0$, we will have $X_n = 0$ for all $n \geq N$. Write a program to find all cycles of the middle square method.

(c) Comment on the quality of the middle square method as a PRNG.

E1.4 Write a program which uses the inverse transform method to generate random numbers with the following density:

$$f(x) = \begin{cases} 1/x^2 & \text{if } x \geq 1 \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

To test your program, plot a histogram of 10 000 random numbers together with the density f.

E1.5 For $n \in \mathbb{N}$, let K_n denote the (random) number of accepted proposals among the first n generated proposals in algorithm 1.19. Show that, with probability 1, we have

$$\lim_{n\to\infty} \frac{1}{n} K_n = Z.$$

E1.6 Implement the rejection method from example 1.24 to generate samples from a half-normal distribution from Exp(1)-distributed proposals. Test your program by generating a histogram of the output and by comparing the histogram with the theoretical density of the half-normal distribution.

E1.7 In example 1.24 we have learned how rejection sampling can be used to convert Exp(λ)-distributed proposals into standard normally distributed samples.

(a) Extend the method to convert Exp(λ)-distributed proposals into $\mathcal{N}(0, \sigma^2)$-distributed samples.

(b) For given σ^2, determine the optimal value of the parameter λ.

E1.8 Consider algorithm 1.22 where the target distribution has density f/Z_f with

$$f(x) = \frac{1}{\sqrt{x}} \exp\left(-y^2/2x - x\right)$$

and $Z_f = \int_0^\infty f(\tilde{x})\, d\tilde{x}$, and where the proposals are Exp(1)-distributed. Find the optimal value for the constant c from algorithm 1.22 for this example.

E1.9 Let f and g be two probability densities and $c \in \mathbb{R}$ with $f(x) \leq cg(x)$ for all x. Show that $c \geq 1$ and that $c = 1$ is only possible for $f = g$ (except possibly on sets with volume 0).

E1.10 Let $X \sim \mathcal{U}[a, b]$ and $Y \sim \mathcal{U}[c, d]$ be independent. Using the definition of the uniform distribution on a set, show that (X, Y) is uniformly distributed on the rectangle $R = [a, b] \times [c, d]$.

E1.11 Without using rejection sampling, propose a method to sample from the uniform distribution on the set

$$A = ([0, 1] \times [0, 1]) \cup ([2, 4] \times [0, 1]).$$

Write a program implementing your method.

E1.12 Without using rejection sampling, propose a method to sample from the uniform distribution on the set

$$B = ([0, 2] \times [0, 2]) \cup ([1, 3] \times [1, 3]) .$$

Write a program implementing your method.

E1.13 Consider the uniform distribution on a semicircle.

(a) Explain how rejection sampling can be used to convert i.i.d. proposals $U_n \sim \mathcal{U}([-1, 1] \times [0, 1])$ into an i.i.d. sequence $(V_k)_{k\in\mathbb{N}}$ which is uniformly distributed on the semicircle $\{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1, y \geq 0\}$. Compute the acceptance probability of the method.

(b) Write a computer program which generates 1000 samples from the uniform distribution on the semicircle, using the method from (a). Create a scatter plot showing the random points. How many proposals were needed to generate 1000 samples?

E1.14 Propose a rejection method to sample from the uniform distribution on the semicircle

$$\left\{(x, y) \in \mathbb{R}^2 \,\middle|\, x^2 + y^2 \leq 1, y \geq 0\right\}$$

which has an acceptance probability of greater than 80%. Implement your method.

E1.15 Let (X, Y) be uniformly distributed on the semicircle

$$\left\{(x, y) \in \mathbb{R}^2 \,\middle|\, x^2 + y^2 \leq 1, y \geq 0\right\} .$$

Find the densities of X and Y, respectively.

E1.16 Let X be a random variable on $\mathbb{R}^d$ with density $f : \mathbb{R}^d \to [0, \infty)$ and let $c \neq 0$ be a constant. Determine the density of cX.

E1.17 Let $X \sim \mathcal{N}(0, 1)$. Determine the density of $Y = (X^2 - 1)/2$.

E1.18 Write a program to implement the ratio-of-uniforms method to sample from the Cauchy distribution with density

$$f(x) = \frac{1}{\pi(1 + x^2)}.$$