Solutions to Problems in Chapter 8 of
Simulation Modeling and Analysis, 5th ed., 2015, McGraw-Hill, New York
by Averill M. Law

8.1. The inverse-transform method can be used in each case.

(*a*) From elementary calculus, $F(x) = \{\tan^{-1}[(x - \alpha)/\beta]\}/\pi + 0.5$, so $X = F^{-1}(U) = \beta \tan[\pi(U - 0.5)] + \alpha$. Alternatively, the following special property can be used: If Y_1 and Y_2 are IID N(0, 1) random variables, then $X = \beta(Y_1/Y_2) + \alpha$ has the desired Cauchy distribution. If $\alpha = 1$ and $\beta = 1$, the Cauchy distribution is just the *t* distribution with 1 df. (For the empirical study below, we used the inverse-transform method.)

(*b*) $F(x) = \exp\left[-e^{-(x-\alpha)/\beta}\right]$, so $X = F^{-1}(U) = -\beta\ln(-\ln U) + \alpha$.

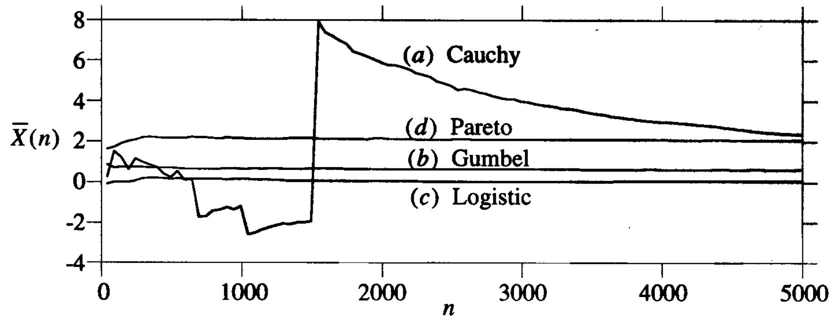
(*c*) $F(x) = 1\Big/\left[1 + e^{-(x-\alpha)/\beta}\right]$, so $X = F^{-1}(U) = -\beta\ln(1/U - 1) + \alpha$.

(*d*) $F(x) = 1 - (c/x)^{\alpha_2}$, so $X = F^{-1}(U) = c\big/(1-U)^{1/\alpha_2}$.

The expected values for these distributions are:

Distribution	(*a*) Cauchy	(*b*) Gumbel	(*c*) Logistic	(*d*) Pareto
Expectation (in general)	Does not exist	$\alpha - \beta\gamma$, where $\gamma = \Gamma'(1) = -0.577216$	α	$c\alpha_2/(\alpha_2 - 1)$, for $\alpha_2 > 1$
Expectation (in this case)	Does not exist	0.58	0	2

We used stream 1 of the generator in App. 7A to generate all 20,000 variates [5000 for each of parts (*a*) through (*d*)], without reseeding the generator between parts. The results, plotted for $n = 50, 100, 150, \ldots, 5000$, are



Clearly, the Cauchy sample means are wild; since this distribution does not have an expected value, the strong law of large large numbers does not apply (the sample means do not have anything to converge to). The expectations of the other distributions do exist, and the sample-mean curves settle down to them accordingly.

8.2. $N = 1$ if and only if the search terminates after the first comparison, i.e., if and only if $U \le 0.05$; in this case we set $X = 1$. Similarly, $N = 2$ if and only if we do not terminate the search after the first comparison, but do terminate it after the second comparison; in this case, $0.05 < U \le 0.1$, and $X = 2$. The other cases are similar, and in fact $N = X$ on a U-for-U basis, so they clearly have the same expectation $E(N) = E(X) = 4.45$.

The second algorithm is clearly valid; for instance, we set $X = 4$ if and only if $0.7 < U \le 0.8$, which has probability $0.1 = p(4)$. The expected number of comparisons in this algorithm is

$$\begin{aligned} E(N') &= 1 \times 0.6 + 2 \times (0.7 - 0.6) + 3 \times (0.8 - 0.7) \\ &\quad + 4 \times (0.9 - 0.8) + 5 \times (0.95 - 0.9) + 6 \times (1 - 0.95) \\ &= 2.05 \end{aligned}$$

We coded both algorithms in Think C 4.0 on an Apple Macintosh IIcx, and used them to generate 10,000 variates; the first algorithm took 10 seconds and the second algorithm took 7 seconds. The speedup in the second algorithm is not as dramatic as was indicated by comparing the expected number of comparisons, since the second algorithm has to do an additional table lookup [to get $i'(i)$ rather than just i], and both algorithms have to generate a random number as well as control the loop index.

8.3. (*a*) Since $F(a) \le V \le F(b)$, we get $a \le F^{-1}(V) \le b$, i.e., X has the correct range, $[a, b]$. Now for $a \le x \le b$, the definitions of V and X imply

$$\begin{aligned} P(X \le x) &= P\left[F^{-1}(V) \le x\right] \\ &= P\left(F^{-1}\{F(a) + [F(b) - F(a)]U\} \le x\right) \\ &= P\{F(a) + [F(b) - F(a)]U \le F(x)\} \\ &= P\left[U \le \frac{F(x) - F(a)}{F(b) - F(a)}\right] \\ &= \frac{F(x) - F(a)}{F(b) - F(a)} \\ &= F^*(x) \end{aligned}$$

as desired. Note that going from the second line in the above to the third line is justified by our assumption that F is continuous and strictly increasing, i.e., is invertible. Going from the fourth to the fifth line is valid since $U \sim \mathrm{U}(0, 1)$.

(*b*) Reasoning as in App. 8A, X is defined only in the event that $F(a) \le U \le F(b)$ in step 2. Thus, $X = F^{-1}(U)$ in this case will clearly have the correct range, $[a, b]$. For $a \le x \le b$,

$$\begin{aligned} P(X \le x) &= P\left[F^{-1}(U) \le x \mid F(a) \le U \le F(b)\right] \\ &= \frac{P\left[F^{-1}(U) \le x,\ F(a) \le U \le F(b)\right]}{P[F(a) \le U \le F(b)]} \\ &= \frac{P[U \le F(x),\ F(a) \le U \le F(b)]}{F(b) - F(a)} \\ &= \frac{P[F(a) \le U \le F(x)]}{F(b) - F(a)} \qquad \text{(since } x \le b\text{)} \\ &= \frac{F(x) - F(a)}{F(b) - F(a)} \\ &= F^*(x) \end{aligned}$$

If $F(a) \approx 0$ and $F(b) \approx 1$, the algorithm in (*b*) will terminate on the first trial most of the time, but involves two comparisons and, in general, a random number of U's. The algorithm in (*a*) always needs exactly one U, but requires a multiply and add to obtain V, in addition to computing F^{-1}. If $F(b) - F(a)$ is small, the algorithm in (*a*) would probably be faster.

8.4. Note first that if F is continuous with range that extends outside $[a, b]$, then $\tilde{F}$ is mixed, since a variate from $\tilde{F}$ could be exactly equal to a [with probability $F(a)$] or be exactly equal to b [with probability $1 - F(b)$]; thus, our algorithm should reflect this.

1. Generate Y with distribution F.
2. If $Y < a$, return $X = a$. Otherwise, go on to step 3.
3. If $Y > b$, return $X = b$. Otherwise, go on to step 4.
4. Return $X = Y$.

[If F is continuous, we terminate with $X = a$ in step 2 in the event that $Y < a$, which has probability $F(a)$; similarly, we return $X = b$ from step 3 if $Y > b$, which has probability $1 - F(b)$.] To show in general that X has distribution function $\tilde{F}$, begin by noting that in any case $a \le X \le b$, so that X has the correct range $[a, b]$. Now for $a \le x < b$,

$$\begin{aligned}
P(X \le x) &= \int_{-\infty}^{\infty} P\big(X \le x \mid Y = y\big)\, dF(y) \\
&= \int_{-\infty}^{x} P\big(X \le x \mid Y = y\big) dF(y) \qquad (\text{for } a \le x < b,\ Y > x \Rightarrow X > x) \\
&= \int_{-\infty}^{a} P\big(X \le x \mid Y = y\big) dF(y) + \int_{a}^{x} P(X \le x \mid Y = y)\, dF(y) \\
&= \int_{-\infty}^{a} P(a \le x)\, dF(y) + \int_{a}^{x} P\big(Y \le x \mid Y = y\big) dF(y) \\
&= \int_{-\infty}^{a} 1\, dF(y) + \int_{a}^{x} 1\, dF(y) \\
&= \int_{-\infty}^{x} 1\, dF(y) \\
&= F(x) \\
&= \tilde{F}(x) \qquad (\text{for } a \le x < b)
\end{aligned}$$

If F is continuous with corresponding density f, the symbol $dF(y)$ in the above integrals can be replaced by the more familiar $f(y)dy$ throughout; using the above generalized (Stieltjes) integrals allows us to take care of the discrete case at the same time.

8.5. Let H denote the distribution function of the beta(i, n–i+1) distribution. Then

$$P(X \le x) = P[F^{-1}(V) \le x] = P[V \le F(x)] = H[F(x)]$$

since $V \sim$ beta(i, n–i+1). On the other hand, if $Y_1, Y_2, \ldots, Y_n$ are IID with distribution function F, then

$$P[Y_{(i)} \le x] = P\{F[Y_{(i)}] \le F(x)\} = P[U_{(i)} \le F(x)] = H[F(x)]$$

since $F(Y_i) \sim$ U(0, 1), the ith smallest of the $F(Y_i)$'s is $F[Y_{(i)}]$, and $U_{(i)} \sim$ beta(i, n–i+1). Thus, X and $Y_{(i)}$ have the same distribution, $H[F(x)]$.

For $i = n$, the beta(n, 1) density is nx^{n-1}, so has distribution function x^n, which is easily inverted to obtain $V = U^{1/n}$. By symmetry (about 0.5) of the beta distribution (see Sec. 8.3.8), if we set $V = 1 - U^{1/n}$, then $V \sim$ beta(1, n).

8.6. Integrating $f(x)$, the distribution function is

$$F(x) = \begin{cases} 0.5e^{x} & \text{if } x \le 0 \\ 1 - 0.5e^{-x} & \text{if } x > 0 \end{cases}$$

which has inverse

$$x = F^{-1}(u) = \begin{cases} \ln(2u) & \text{if } u \le 0.5 \\ -\ln[2(1-u)] & \text{if } u > 0.5 \end{cases}$$

leading to the inverse-transform algorithm

1′. Generate $U \sim U(0, 1)$.

2′. If $U \le 0.5$, return $X = \ln(2U)$. Otherwise, return $X = -\ln[2(1 - U)]$.

The inverse-transform algorithm always requires one U, one compare, and one logarithm. The composition algorithm requires two U's, one compare, and one logarithm. Thus, the inverse-transform approach would be preferable. (Note that in step 2′ of the inverse-transform algorithm it is *not* valid to replace the $1 - U$ under the logarithm by U in the case that $U > 0.5$.)

8.7. If we integrate $f_R(x)$, then we get $F_{R(a,b)}(x) = [(x-a)/(b-a)]^2$ for $a \le x \le b$. Therefore, $F_{R(0,1)}(x) = x^2$ for $0 \le x \le 1$.

(*a*) Let $X \sim \text{RT}(0,1)$ and $X' = a + (b-a)X$. Then

$$\begin{aligned} P(X' \le x) &= P(a + (b-a)X \le x) \\ &= P(X \le (x-a)/(b-a)) \\ &= F_{R(0,1)}((x-a)/(b-a)) \\ &= [(x-a)/(b-a)]^2 \\ &= F_{R(a,b)}(x) \end{aligned}$$

It can be shown that $F_{L(a,b)}(x) = 1 - [(b-x)/(b-a)]^2$ for $a \le x \le b$. Let $X \sim \text{LT}(0,1)$ and $X' = a + (b-a)X$. Then

$$\begin{aligned} P(X' > x) &= P(a + (b-a)X > x) \\ &= P(X > (x-a)/(b-a)) \\ &= 1 - F_{L(0,1)}((x-a)/(b-a)) \\ &= \left(1 - \frac{x-a}{b-a}\right)^2 \\ &= \left(\frac{b-x}{b-a}\right)^2 \\ &= 1 - F_{L(a,b)}(x) \end{aligned}$$

(*b*) $P(1 - X \le x) = P(X \ge 1 - x) = 1 - F_{R(0,1)}(1-x) = 1 - (1-x)^2 = F_{L(0,1)}(x)$

(*c*) For RT(0,1), set $X = \sqrt{U}$. For LT(0,1), set $X = 1 - \sqrt{1-U}$.

(*d*)
$$\begin{aligned} P(\max\{U_1, U_2\} \le x) &= P(U_1 \le x, U_2 \le x) \\ &= P(U_1 \le x)P(U \le x) && \text{since } U_1 \text{ and } U_2 \text{ are independent} \\ &= x^2 && \text{since } U_1 \text{ and } U_2 \text{ are U(0,1)} \\ &= F_{R(0,1)}(x) \end{aligned}$$

This method involves generating two random numbers and doing one comparison. This is probably faster than generating one random number and computing the square root in the inverse-transform method. This would have to be traded off against the advantages of the inverse-transform method discussed in Sec. 8.2.1.

8.8. (*a*) $P(X > x) = P(U_1 > x, U_2 > x) = P(U_1 > x)P(U_2 > x) = (1 - x)^2$ (by independence and uniformity), so $F(x) = 1 - (1 - x)^2$. Thus, we obtain the inverse-transform formula $X = 1 - \sqrt{1 - U}$.

(*b*) $P(X \leq x) = P(U_1 \leq x, U_2 \leq x) = P(U_1 \leq x)P(U_2 \leq x) = x^2$, which leads to the inverse-transform formula $X = \sqrt{U}$.

(*c*) $P(X > x) = P(Y_1 > x, Y_2 > x) = P(Y_1 > x)P(Y_2 > x) = \left(e^{-x/\beta}\right)^2 = e^{-x/(\beta/2)}$ [by independence and the Y_i's being expo(β)], so $X \sim$ expo($\beta/2$). Thus, the (literal) inverse-transform formula is $X = -(\beta/2)\ln(1 - U)$.

For (*a*) and (*b*), these one-U algorithms are probably slower than actually generating U_1 and U_2 and then doing the appropriate comparison; square-root calculations are slow. (The one-U methods do have the advantage, though, of being the inverse-transform method.) For (*c*), however, the one-U method is clearly preferable to explicit generation of Y_1 and Y_2, since it *is* the inverse-transform method, and requires one (rather than two) random number, one (rather than two) logarithmic computation, and one (rather than two) multiplication.

8.9. Let A denote the event that acceptance occurs in step 3′. Then for $i = 0, \pm 1, \pm 2, \ldots,$

$$P(X = x_i) = P(Y = x_i \mid A) = \frac{P(A \mid Y = x_i)P(Y = x_i)}{P(A)}$$

For any j,

$$P(A \mid Y = x_j) = P\left[U \le \frac{p(Y)}{t(Y)} \,\middle|\, Y = x_j\right] = \frac{p(x_j)}{t(x_j)}$$

so

$$P(A) = \sum_{j=-\infty}^{\infty} P(A \mid Y = x_j)P(Y = x_j) = \sum_{j=-\infty}^{\infty} \frac{p(x_j)}{t(x_j)} r(x_j) = \sum_{j=-\infty}^{\infty} \frac{p(x_j)}{t(x_j)} \frac{t(x_j)}{c} = \frac{1}{c} \sum_{j=-\infty}^{\infty} p(x_j) = \frac{1}{c}$$

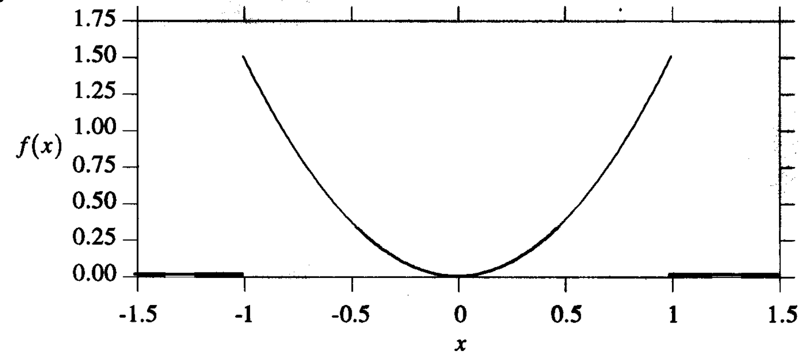
and thus

$$P(X = x_i) = \frac{\frac{p(x_i)}{t(x_i)} r(x_i)}{\frac{1}{c}} = \frac{\frac{p(x_i)}{t(x_i)} \frac{t(x_i)}{c}}{\frac{1}{c}} = p(x_i)$$

as desired. As in the continuous case, it is important that we be able to generate rapidly and easily from $\{r(x_i)\}$ and that $\{t(x_i)\}$ fit snugly down on top of $\{p(x_i)\}$; the probability of acceptance in step 3′ is $1/c$, as before.

8.10. Since $P(\text{acceptance}) = 1/c$ independently on each pass through the algorithm, the number of rejections has a geom$(1/c)$ distribution. Thus, the expected number of rejections is $c - 1$.

8.11. (*a*) The density looks like this:



Inverse transform. For $-1 \le x \le 1$, the distribution function is $F(x) = (x^3 - 1)/2$, leading to the inverse-transform formula $X = \sqrt[3]{2U - 1}$.

Composition. A natural decomposition is vertically at $x = 0$, i.e., write

$$f(x) = 0.5\, I_{[-1,0)}(x)\, 3x^2 + 0.5\, I_{[0,1]}(x)\, 3x^2$$

The composition algorithm is thus

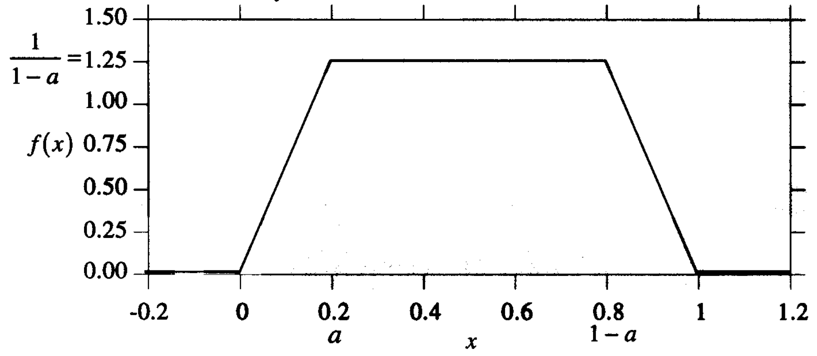
1. Generate U_1 and U_2 IID U(0, 1).
2. If $U_1 \le 0.5$, return $X = \sqrt[3]{-U_2}$. Otherwise, return $X = \sqrt[3]{U_2}$.

Acceptance-rejection. A simple majorizing function is the uniform $t(x) = I_{[-1,1]}(x)\,(1.5)$, which is clearly $\ge f(x)$ for all x. Thus, $c = 3$, $r(x)$ is the U(−1, 1) density, and the algorithm is

1. Generate U' ~U(0, 1) and let $Y = 2U' - 1$.
2. Generate U ~ U(0, 1), independent of U'.
3. If $U \le Y^2$, return $X = Y$. Otherwise, go back to step 1 and try again.

Inverse transform and composition both require exactly one cube-root evaluation, so composition would not appear to have any advantage over inverse transform here. Acceptance-rejection never requires any cube roots, but has only a probability of 1/3 of acceptance on a given pass; the expected number of random-number generations is thus 6. [Another majorizing function is $t(x) = I_{[-1,1]}(x)\,(1.5)\,|x|$, for which c = 1.5, cutting in half the expected number of iterations. Generating from the corresponding $r(x)$ involves composition to decide on LT(−1, 0) or RT(0, 1); if this is done as in Prob. 8.7(*d*), the expected number of random-number generations is still 6, and more comparisons are needed. Thus, the decrease in c might be offset by the increased effort to generate an observation on Y.]

(*b*) When a = 0.2, for instance, the density looks like this:



(Problem 8.11, continued from previous page)

Inverse transform. For $0 \le x \le 1$, the distribution function is

$$F(x) = \begin{cases} \dfrac{x^2}{2a(1-a)} & \text{if } 0 \le x \le a \\ \dfrac{2x-a}{2(1-a)} & \text{if } a \le x \le 1-a \\ 1 - \dfrac{(x-1)^2}{2a(1-a)} & \text{if } 1-a \le x \le 1 \end{cases}$$

Solving $U = F(X)$ leads to the inverse-transform formula

$$X = \begin{cases} \sqrt{2a(1-a)U} & \text{if } 0 < U < \dfrac{a}{2(1-a)} \\ \dfrac{a}{2} + (1-a)U & \text{if } \dfrac{a}{2(1-a)} \le U \le \dfrac{2-3a}{2(1-a)} \\ 1 - \sqrt{2a(1-a)(1-U)} & \text{if } \dfrac{2-3a}{2(1-a)} \le U < 1 \end{cases}$$

Composition. $f(x)$ is naturally decomposed at $x = a$ and $x = 1 - a$, i.e., for $0 \le x \le 1$, $f(x) = p_1 f_1(x) + p_2 f_2(x) + p_3 f_3(x)$, where $p_1 = a/[2(1-a)] = p_3$, $p_2 = (1-2a)/(1-a)$, f_1 is the RT(0, a) density, f_2 is the U(a, $1-a$) density, and f_3 is the LT($1-a$, 1) density. The algorithm is

1. Generate $U \sim$ U(0, 1).
2. If $U \le p_1$, return $X \sim$ RT(0, a). Otherwise go on to step 3.
3. If $U \le p_1 + p_2$, return $X \sim$ U(a, $1-a$). Otherwise go on to step 4.
4. Return $X \sim$ LT($1-a$, 1).

Acceptance-rejection. A natural majorizing function is the uniform

$$t(x) = I_{[0,1]}(x)\frac{1}{1-a}$$

so $c = 1/(1-a)$ and r is just the U(0, 1) density. Since $t(x) = f(x)$ for $a \le x \le 1-a$, the algorithm is

1. Generate $Y \sim$ U(0, 1).
2. If $a < Y < 1-a$, return $X = Y$. Otherwise go to step 3.
3. Generate $U \sim$ U(0, 1) independent of Y.
4. If $Y < a$ go to step 5. Otherwise go to step 6.
5. If $U \le Y/a$ return $X = Y$. Otherwise go back to step 1 and try again.
6. If $U \le (1-Y)/a$ return $X = Y$. Otherwise go back to step 1 and try again.

Acceptance-rejection is attractive since it is relatively simple and never requires square-root calculations. Square roots might be required for inverse transform, depending on where U falls. If generation from the RT and LT distributions is done as in Prob. 8.7(*d*), square roots are avoided, again at the expense of extra random-number generation. As a becomes smaller, all three algorithms should become faster; for small a, $f(x)$ is close to the U(0, 1) density. It seems difficult to predict which of the three algorithms would be fastest for a given value of a.

8.12. Let A denote the event of "acceptance" on a given iteration through the algorithm. Then

$$\begin{aligned}
P(A) &= P\left(V_1^2 + V_2^2 \le 1\right) \\
&= \int_{-1}^{1} P\left(V_1^2 \le 1 - x^2 \mid V_2 = x\right) 0.5\,dx \\
&= \int_{-1}^{1} P\left(-\sqrt{1-x^2} \le V_1 \le \sqrt{1-x^2}\right) 0.5\,dx \\
&= \int_{-1}^{1} \sqrt{1-x^2}\; 0.5\,dx \\
&= 0.25\left[x\sqrt{1-x^2} + \sin^{-1} x\right]_{x=-1}^{x=1} \\
&= \frac{\pi}{4}
\end{aligned}$$

If we let N = the number of "rejections," then $N \sim \text{geom}(\pi/4)$. The number of executions of step 1 is $N + 1$, so has expectation $E(N) + 1 = 4/\pi \approx 1.27$.

8.13. For $0 \leq x \leq 1$, the triang(0, 1, c) density can be expressed as $f(x) = cf_1(x) + (1 - c)f_2(x)$, where $f_1(x)$ is the RT(0, c) density and $f_2(x)$ is the LT(c, 1) density. Thus, an algorithm would be

1. Generate $U \sim$ U(0, 1).

2. If $U \leq c$, return $X \sim$ RT(0, c). Otherwise return $X =$ LT(c, 1).

If we generate from RT and LT as in Prob. 8.7(d), square roots are avoided at the expense of extra random-number generation. Whether inverse transform or composition is faster would depend on the speed of the random-number generator relative to the square-root operation.

8.14. (*a*) For $i = 0, 1, 2, \ldots,$

$$\begin{aligned} P\left\{\left\lfloor \frac{\ln U}{\ln(1-p)} \right\rfloor = i\right\} &= P\left[i \le \frac{\ln U}{\ln(1-p)} < i+1\right] \\ &= P\left[(i+1)\ln(1-p) < \ln U \le i\ln(1-p)\right] \\ &= P\left[(1-p)^{i+1} < U \le (1-p)^i\right] \\ &= (1-p)^i - (1-p)^{i+1} \\ &= (1-p)^i p \\ &= p(i) \end{aligned}$$

so the algorithm is valid. A direct implementation of the inverse-transform method would return $X = 0$ if and only if $U \le p$, i.e., $1 - p < 1 - U$; for $i = 1, 2, \ldots,$ it would set $X = i$ if and only if

$$\sum_{j=0}^{i-1}(1-p)^j p < U \le \sum_{j=0}^{i}(1-p)^j p$$

which is equivalent to $(1-p)^{i+1} \le 1 - U < (1-p)^i$. Thus, looking at the third line in the above set of equations and ignoring the probability-zero events resulting from the discrepancy between $<$ and $\le$ with regard to U and $1 - U$, the algorithm in Sec. 8.4.5 is identical to the inverse-transform method if U and $1 - U$ are interchanged.

(*b*) Let $U_0, U_1, U_2, \ldots$ be a sequence of IID U(0, 1) random variables. Then if X is the random variable given by this algorithm, $P(X = 0) = P(U_0 \le p) = p$, and for $i = 1, 2, \ldots,$

$$\begin{aligned} P(X = i) &= P(U_0 > p, U_1 > p, \ldots, U_{i-1} > p, U_i \le p) \\ &= (1-p)^i p \\ &= p(i) \end{aligned}$$

where the second line is justified since the U_j's are IID.

8.15. A simple (and obvious) algorithm is

1. Generate Y from the corresponding unshifted distribution.
2. Return $X = \gamma + Y$.

8.16. Let p_1 (= 0.155) be the area of the leftmost triangle in Fig. 8.12 and p_2 (= 0.792) be the area of the rectangle. Then the composition-based algorithm is

1. Generate $U \sim$ U(0,1).
2. If $U < p_1$, return $X \sim$ RT(0, 0.36) (see Prob. 8.7). Otherwise, go on to Step 3.
3. If $U < p_1 + p_2$, return $X \sim$ U(0.36, 0.84). Otherwise, go on to Step 4.
4. Return $X \sim$ LT(0.84, 1) (see Prob. 8.7).

8.17. It suffices to show that $I \sim \mathrm{DU}(0, n)$, $U' \sim \mathrm{U}(0, 1)$, and that they are also independent. Since $V \sim \mathrm{U}(0, n+1)$, clearly $I \sim \mathrm{DU}(0, n)$. Now $U' = V - I = V - \lfloor V \rfloor$ = the decimal part of V, so $0 \le U' < 1$. For $i \in \{0, 1, \ldots, n\}$ and $0 < x < 1$,

$$\begin{aligned} P(I = i, U' \le x) &= P(\lfloor V \rfloor = i, V - \lfloor V \rfloor \le x) \\ &= P(\lfloor V \rfloor = i, V \le i + x) \\ &= P(i \le V < i+1, V \le i + x) \\ &= P(i \le V < i + x) \\ &= \frac{(i+x) - i}{n+1} \\ &= \frac{x}{n+1} \end{aligned}$$

where the fifth line follows since $V \sim \mathrm{U}(0, n+1)$. Thus, the marginal distribution function of U' is

$$\begin{aligned} P(U' \le x) &= \sum_{i=0}^{n} P(I = i, U' \le x) \\ &= \sum_{i=0}^{n} \frac{x}{n+1} \\ &= (n+1)\frac{x}{n+1} \\ &= x \end{aligned}$$

so $U' \sim \mathrm{U}(0, 1)$. Moreover, we see that

$$P(I = i, U' \le x) = \frac{x}{n+1} = \frac{1}{n+1}x = P(I = i)P(U' \le x)$$

showing that I and U' are independent as well.

8.18. (*a*) If $F_i = 1$, then whenever we generate $I = i$ we would return $X = I = i$ for sure; the value L_i of the alias of i is never used and so would not even have to be defined. If instead we defined $L_i = i$ and set $F_i = 0$, then generating $I = i$ would result in returning $X = L_i = i$ for sure, which of course amounts to the same thing.

(*b*) For $i = 0, 1, \ldots, n$, let $A_i = L_i + F_i$. Since $0 \le F_i < 1$ for all i, $L_i = \lfloor A_i \rfloor$ and $F_i = A_i - \lfloor A_i \rfloor$, so all the information in the L_i and F_i arrays is in A_i. The algorithm's step 1 is unchanged, and step 2 becomes

2. If $U \le A_i - \lfloor A_i \rfloor$, return $X = I$. Otherwise, return $X = \lfloor A_i \rfloor$.

Clearly, this is more computation, but does cut the storage in half.

8.19. Suppose that $u(z) = u$, where $z = v/u$ for $0 < z < \infty$. Then from the definition of *S*, we get that $u(z) \leq \sqrt{pf(z)}$. If we let $v(z) = zu(z)$, then $v(z) \leq z\sqrt{pf(z)}$.

8.20. $S = \{(u,v) : 0 \le u \le \frac{v}{u}, 0 \le \frac{v}{u} \le 1\}$
$= \{(u,v) : 0 \le u^2 \le v, 0 \le v \le u\}$

In order for a point (u,v) to satisfy the two conditions, it must be in the region bounded above and below by the curves $v = u$ and $v = u^2$. Note that the first curve lies strictly above the second curve for $0 < u < 1$, and the two curves coincide for $u = 0$ and $u = 1$.

8.21. We will use a right triangle whose density function is defined by (see Problem 8.7)

$$f(x) = \begin{cases} 2x & \text{if } 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases}$$

Then $s/t = (1/6)/(1/2) = 1/3$, as compared to 1/6 in Example 8.8.

The following algorithm can be used to generate points (u, v) randomly in *T*:

1. Generate u from $\text{U}(0,1)$ and set $u^* = u$.
2. Generate v from $\text{U}(0, u^*)$.

8.22. To be supplied

8.23. If we integrate $f(x)$, then we get the following expression for the distribution function $F(x)$:

$$F(x) = 1 - (1 - x^{\alpha_1})^{\alpha_2}$$

If we solve the equation $U = F(X)$ for X, then we get the following inverse-transform algorithm for generating random variates from the Kumaraswamy distribution:

1. Generate $U \sim \mathrm{U}(0,1)$.

2. Set $X = \left[1 - (1 - U)^{1/\alpha_2}\right]^{1/\alpha_1}$.