How do we generate random variables?

- Sampling from continuous distributions
- Sampling from discrete distributions
- Random-number generators (Sampling from the U(0, 1) distribution)

Sampling from continuous distributions

Inverse Transform Method:

Let the random variable X have a continuous and increasing distribution function F. Denote the inverse of F by F^{-1} . Then X can be generated as follows:

- Generate U from U(0, 1);
- Return $X = F^{-1}(U)$.

If F is not continuous or increasing, then we have to use the *generalized* inverse function

$$F^{-1}(u) = \min\{x : F(x) \ge u\}.$$



Examples:

- X = a + (b a)U is uniform on (a, b);
- $X = -\ln(U)/\lambda$ is exponential with parameter λ ;
- $X = (-\ln(U))^{1/a}/\lambda$ is Weibull, parameters a and λ .

Unfortunately, for many distribution functions we do not have an easy-touse (closed-form) expression for the inverse of F.



Composition method:

This method applies when the distribution function F can be expressed as a mixture of other distribution functions F_1, F_2, \ldots ,

$$F(x) = \sum_{i=1}^{\infty} p_i F_i(x),$$

where

$$p_i \ge 0, \qquad \sum_{i=1}^{\infty} p_i = 1$$

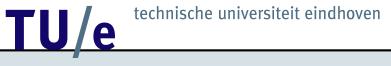
The method is useful if it is easier to sample from the F_i 's than from F. The algorithm is as follows:

• First generate an index *I* such that

$$P(I=i) = p_i, \qquad i = 1, 2, \dots$$

• Generate a random variable X with distribution function F_I .

department of mathematics and computing science



Examples:

• Hyper-exponential distribution:

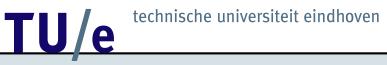
$$F(x) = p_1 F_1(x) + p_2 F_2(x) + \dots + p_k F_k(x), \qquad x \ge 0,$$

where $F_i(x)$ is the exponential distribution with parameter μ_i , $i = 1, \ldots, k$.

• Double-exponential (or Laplace) distribution:

$$f(x) = \begin{cases} \frac{1}{2}e^x, & x < 0; \\ \\ \frac{1}{2}e^{-x}, & x \ge 0, \end{cases}$$

where f denotes the density of F.



Convolution method:

In some case X can be expressed as a sum of independent random variables Y_1,\ldots,Y_n , so

 $X = Y_1 + Y_2 + \dots + Y_n.$

where the Y_i 's can be generated more easily than X.

Algorithm:

- Generate independent Y_1, \ldots, Y_n , each with distribution function G;
- Return $X = Y_1 + \cdots + Y_n$.



Example:

If X is Erlang distributed with parameters n and μ , then X can be expressed as a sum of n independent exponentials Y_i , each with mean $1/\mu$.

Algorithm:

- Generate *n* exponentials Y_1, \ldots, Y_n , each with mean μ ;
- Set $X = Y_1 + \dots + Y_n$.

More efficient algorithm:

- Generate n uniform (0, 1) random variables U_1, \ldots, U_n ;
- Set $X = -\ln(U_1U_2\cdots U_n)/\mu$.

Acceptance-Rejection method:

Denote the density of X by f. This method requires a function g that majorizes f,

$$g(x) \ge f(x)$$

for all x. Now g will not be a density, since

$$c = \int_{-\infty}^{\infty} g(x) dx \ge 1.$$

Assume that $c < \infty$. Then h(x) = g(x)/c is a density.

Algorithm:

- ı. Generate Y having density h;
- 2. Generate U from U(0, 1), independent of Y;
- 3. If $U \leq f(x)/g(x)$, then set X = Y; else go back to step 1.

The random variable X generated by the above algorithm has density f.

/ department of mathematics and computing science



Validity of the Acceptance-Rejection method:

Note

$$P(X \le x) = P(Y \le x | Y \text{accepted}).$$

Now,

$$P(Y \le x, Y \text{accepted}) = \int_{-\infty}^{x} \frac{f(y)}{g(y)} h(y) dy = \frac{1}{c} \int_{-\infty}^{x} f(y) dy,$$

and thus, letting $x \to \infty$ gives

$$P(\text{Yaccepted}) = \frac{1}{c}.$$

Hence,

$$P(X \le x) = \frac{P(Y \le x, Y \text{accepted})}{P(Y \text{accepted})} = \int_{-\infty}^{x} f(y) dy.$$

/ department of mathematics and computing science



Note that the number of iterations is geometrically distributed with mean c.

How to choose g?

- \bullet Try to choose g such that the random variable Y can be generated rapidly;
- The probability of rejection in step 3 should be small; so try to bring c close to 1, which mean that g should be close to f.

Example:

The Beta(4,3) distribution has density

$$f(x) = 60x^3(1-x)^2, \qquad 0 \le x \le 1.$$

The maximal value of f occurs at x=0.6, where f(0.6)=2.0736. Thus, if we define

$$g(x) = 2.0736, \qquad 0 \le x \le 1,$$

then g majorizes f.

Algorithm:

I. Generate Y and U from U(0, 1);

2. If

$$U \le \frac{60Y^3(1-Y)^2}{2.0736},$$

then set X = Y; else reject Y and return to step 1.

Generating Normal random variables

Methods:

- Central Limit Theorem
- Acceptance-Rejection method
- Box-Muller method



Central Limit Theorem:

This is an approximation method.

The Central Limit Theorem states that for a sequence of iid random variables Y_1, Y_2, \ldots with mean μ and variance σ^2 , the distribution of

$$\frac{\sum_{i=1}^{n} Y_i - n\mu}{\sigma\sqrt{n}}$$

converges to the standard normal distribution as n tends to infinity. If we take $Y_i = U_i$, where U_i are from U(0, 1), then $\mu = 1/2$ and $\sigma^2 = 1/12$. Hence, we may approximate a standard normal random variable by

$$\sum_{i=1}^{12} U_i - 6$$

Acceptance-Rejection method:

If X is N(0,1), then the density of $\left|X\right|$ is given by

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

Now the function

$$g(x) = \sqrt{2e/\pi}e^{-x}$$

majorizes f. This leads to the following algorithm:

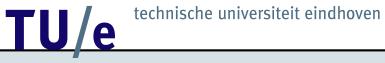
- ı. Generate an exponential Y with mean ı;
- 2. Generate U from U(0, 1), independent of Y;

3. If

$$U \le e^{-(Y-1)^2/2},$$

then accept Y; else reject Y and return to step 1.

4. Return X = Y or X = -Y, both with probability 1/2.



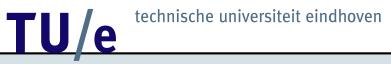
Box-Muller method:

If U_1 and U_2 are independent U(0,1) random variables, then

$$X_{1} = \sqrt{-2 \ln U_{1}} \cos(2\pi U_{2})$$

$$X_{2} = \sqrt{-2 \ln U_{1}} \sin(2\pi U_{2})$$

are independent standard normal random variables.



Sampling from discrete distributions

General method:

(Discrete version of Inverse Transform Method)

Let \boldsymbol{X} be a discrete random variable with probabilities

$$P(X = x_i) = p_i, \qquad i = 0, 1, \dots, \qquad \sum_{i=0}^{\infty} p_i = 1.$$

To generate a realization of X, we first generate U from U(0, 1) and then set $X = x_i$ if

$$\sum_{j=0}^{i-1} p_j \le U < \sum_{j=0}^{i} p_j.$$

So the algorithm is as follows:

- Generate U from U(0, 1);
- Determine the index *I* such that

$$\sum_{j=0}^{I-1} p_j \le U < \sum_{j=0}^{I} p_j$$

and return $X = x_I$.

The second step requires a *search*; for example, starting with I = 0 we keep adding I to I until we have found the (smallest) I such that

$$U < \sum_{j=0}^{I} p_j$$

Note: The algorithm needs exactly one uniform random variable U to generate X; this is a nice feature if you use variance reduction techniques.

/ department of mathematics and computing science



Fast methods when X has a finite support:

- Arraymethod. This method requires that p_i is exactly equal to a q-place decimal.
- Alias method.

These methods require some 'set-up'.

Array method

Suppose $p_i = k_i/100$, i = 1, ..., m, where k_i 's are integers with $0 \le k_i \le 100$

Construct array
$$A[i]$$
, $i = 1, ..., 100$ as follows:
set $A[i] = x_1$ for $i = 1, ..., k_1$
set $A[i] = x_2$ for $i = k_1 + 1, ..., k_1 + k_2$, etc.

Then, first sample a random index I from $1,\ldots,100{:}$ $I=1+\lfloor 100U \rfloor$ and set X=A[I]

Alias method

Set-up: Express the distribution $\{p_1, \ldots, p_m\}$ as an *equiprobable* mixture of m distributions, each living on (at most) two points in $\{1, \ldots, m\}$. So decompose $M = \{I, ..., m\}$ into m pairs $\{A_i, B_i\}, i = 1, \ldots, m$, such that

$$M = \bigcup_{i=1}^{m} \{A_i, B_i\}$$

and assign probabilities $P(A_i)$ and $P(B_i) = 1 - P(A_i)$ to each pair, such that

$$p_i = \frac{1}{m} \sum_{j=1}^m f_j(i)$$

where $f_j(i)$ is equal to $P(A_i)$ if $i = A_i$, $P(B_i)$ if $i = B_i$, and 0 otherwise.

Then, sample as follows:

- Generate a uniform random variable U_1 on (0, 1)and let $I = 1 + \lfloor mU_1 \rfloor$;
- Generate a uniform random variable U_2 on (0, 1); if $U_2 \leq P(A_I)$, then $X = A_I$, else $X = B_I$.



Sampling from special discrete distributions

Bernoulli

Two possible outcomes of X (success or failure):

$$P(X = 1) = 1 - P(X = 0) = p.$$

Algorithm:

- Generate U from U(0, 1);
- If $U \leq p$, then X = 1; else X = 0.



Discrete uniform

The possible outcomes of X are $m,m+1,\ldots,n$ and they are all equally likely, so

$$P(X = i) = \frac{1}{n - m + 1}, \qquad i = m, m + 1, \dots, n.$$

Algorithm:

• Generate U from U(0, 1);

• Set
$$X = m + \lfloor (n - m + 1)U \rfloor$$
.

Note: No search is required, and compute (n - m + 1) ahead.

Geometric

A random variable \boldsymbol{X} has a geometric distribution with parameter \boldsymbol{p} if

$$P(X = i) = p(1 - p)^{i}, \quad i = 0, 1, 2, ...;$$

X is the number of failures till the first success in a sequence of Bernoulli trials with success probability p.

Algorithm:

- Generate independent Bernoulli(*p*) random variables Y_1, Y_2, \ldots ; let *I* be the index of the first successful one, so $Y_I = 1$;
- Set X = I 1.

Alternative algorithm:

- Generate U from U(0, 1);
- Set $X = \lfloor \ln(U) / \ln(1-p) \rfloor$.

Binomial

A random variable X has a binomial distribution with parameters n and p if

$$P(X=i) = \binom{n}{i} p^{i} (1-p)^{n-i}, \qquad i = 0, 1, \dots, n;$$

X is the number of successes in n independent Bernoulli trials, each with success probability p.

Algorithm:

- Generate *n* Bernoulli(*p*) random variables Y_1, \ldots, Y_n ;
- Set $X = Y_1 + Y_2 + \dots + Y_n$.

Alternative algorithms can be derived by using the following results.

Let Y_1, Y_2, \ldots be independent geometric(*p*) random variables, and *I* the smallest index such that

$$\sum_{i=1}^{I+1} (Y_i + 1) > n.$$

Then the index I has a binomial distribution with parameters n and p.

Let Y_1, Y_2, \ldots be independent exponential random variables with mean 1, and I the smallest index such that

$$\sum_{i=1}^{I+1} \frac{Y_i}{n-i+1} > -\ln(1-p).$$

Then the index I has a binomial distribution with parameters n and p.



Negative Binomial

A random variable X has a negative-binomial distribution with parameters $n \mbox{ and } p$ if

$$P(X=i) = \binom{n+i-1}{i} p^n (1-p)^i, \qquad i = 0, 1, 2, \dots;$$

X is the number of failures before the n-th success in a sequence of independent Bernoulli trials with success probability p.

Algorithm:

- Generate n geometric(p) random variables Y_1, \ldots, Y_n ;
- Set $X = Y_1 + Y_2 + \dots + Y_n$.

Poisson

A random variable X has a Poisson distribution with parameter λ if

$$P(X = i) = \frac{\lambda^{i}}{i!}e^{-\lambda}, \qquad i = 0, 1, 2, \dots;$$

X is the number of events in a time interval of length 1 if the inter-event times are independent and exponentially distributed with parameter λ .

Algorithm:

• Generate exponential inter-event times Y_1, Y_2, \ldots with mean 1; let I be the smallest index such that

$$\sum_{i=1}^{I+1} Y_i > 1;$$

• Set X = I.



Or:

• Generate U(0,1) random variables U_1, U_2, \ldots ; let *I* be the smallest index such that

$$\prod_{i=1}^{I+1} U_i < e^{-1};$$

• Set X = I.