Random Variate Generation

Christos Alexopoulos and Dave Goldsman

Georgia Tech

30 March 2020

Outline

Goal: Use $\mathcal{U}(0,1)$ numbers to generate observations (variates) from other distributions, and even stochastic processes.

- Discrete distributions, including Bernoulli, Binomial, Poisson, and empirical.
- Continuous distributions including exponential, normal, and empirical.

Outline

Inverse Transform Method

Inverse Transform Theorem: Let X be a random variable with cdf $F(x)$. We define the inverse cdf by

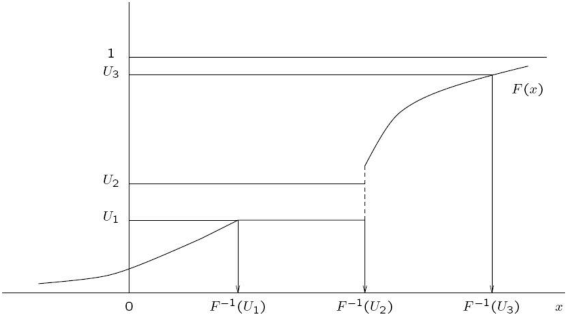
$$F^{-1}(u) = \min\{x : F(x) \geq u\}, \quad u \in [0, 1].$$

Then

(a) $U = F(X) \sim \mathcal{U}(0, 1)$.

(b) Let $U \sim \mathcal{U}(0, 1)$. Then the random variable $Y = F^{-1}(U)$ has the same distribution as X.

Remark: Mathematical precision requires the use of "infimum" in place of "min".


1
U_3
$F(x)$
U_2
U_1
0
$F^{-1}(U_1)$
$F^{-1}(U_2)$
$F^{-1}(U_3)$
x

Proof

(a) Notice that $F(x) \leq u$ is equivalent to $x \leq F^{-1}(u)$. If we replace the real value x with the random variable X, we obtain the following equality of events:

$$\{F(X) \leq u\} = \{X \leq F^{-1}(u)\}.$$

The c.d.f. of U is

$$\begin{aligned} P(U \leq u) &= P\{F(X) \leq u\} \\ &= P\{X \leq F^{-1}(u)\} = F(F^{-1}(u)) = u, \quad 0 \leq u \leq 1. \end{aligned}$$

(b) Since $u \leq F(x)$ is equivalent to $F^{-1}(x) \leq u$, we have the following equality of events:

$$\{U \leq F(x)\} = \{F^{-1}(U) \leq x\}$$

and

$$P\{Y \leq x\} = P\{F^{-1}(U) \leq x\} = P\{U \leq F(x)\} = F(x). \quad \square$$

Discrete Example: Suppose

$$X \equiv \begin{cases} -1 & \text{w.p. } 0.6 \\ 2.5 & \text{w.p. } 0.3 \\ 4 & \text{w.p. } 0.1 \end{cases}$$

Then we have

x	$P(X=x)$	$F(x)$	$\mathcal{U}(0,1)$'s
−1	0.6	0.6	[0.0,0.6]
2.5	0.3	0.9	(0.6,0.9]
4	0.1	1.0	(0.9,1.0]

Thus, if $U = 0.63$, we get $X = \min\{x : F(x) \geq U\} = 2.5$.

The Simio expression is `Random.Discrete(-1,0.6,2.5,0.9,4,1)`.

Example: The $\mathcal{U}(a, b)$ distribution.

$F(x) = (x - a)/(b - a)$, $a \le x \le b$.

Solving $(X - a)/(b - a) = U$ for X, we get $X = a + (b - a)U$. $\Box$

Example: The discrete uniform distribution on $\{a, a + 1, \dots, b\}$ with

$$P(X = k) = \frac{1}{b - a + 1}, \quad x = a, a + 1, \dots, b.$$

Check that

$$X = a + \lfloor (b - a + 1)U \rfloor,$$

where $\lfloor \cdot \rfloor$ is the floor function. $\Box$

Example: The exponential distribution.

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

Solving $F(X) = U$ for X,

$$X = -\frac{1}{\lambda}\ell\mathrm{n}(1-U) \quad \text{or} \quad X = -\frac{1}{\lambda}\ell\mathrm{n}(U). \quad \square$$

Example: The Weibull distribution.

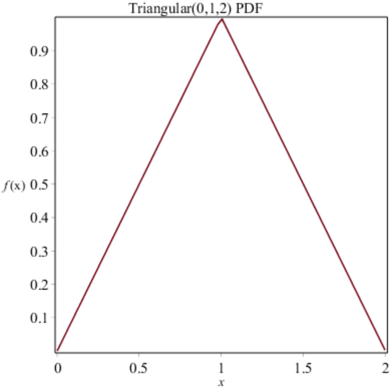
$F(x) = 1 - e^{-(\lambda x)^\alpha}, x > 0.$

Solving $F(X) = U$ for X,

$$X = \frac{1}{\lambda}\left[-\ell\mathrm{n}(1-U)\right]^{1/\alpha} \quad \text{or} \quad X = \frac{1}{\lambda}\left[-\ell\mathrm{n}(U)\right]^{1/\alpha}. \quad \square$$

Example: The triangular$(0, 1, 2)$ distribution has pdf

$$f(x) = \begin{cases} x & \text{if } 0 \leq x < 1 \\ 2 - x & \text{if } 1 \leq x \leq 2. \end{cases}$$



CDF Computation

We have two cases:

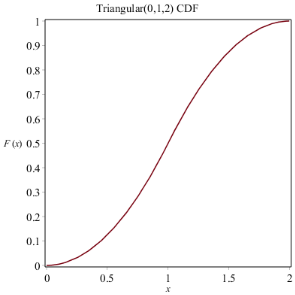
(a) For $0 \le x < 1$,

$$F(x) = \int_0^x t\,dt = x^2/2.$$

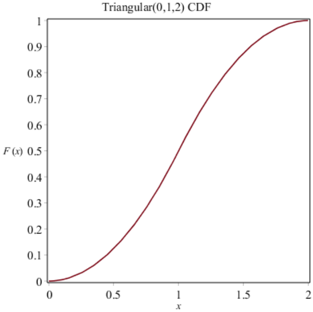
(b) For $1 \le x \le 2$,

$$F(x) = F(1) + \int_1^x (2-t)\,dt = \frac{1}{2} + \frac{1}{2} - \frac{1}{2}(2-x)^2 = 1 - \frac{1}{2}(2-x)^2.$$

Notice that $F(0) = 0$, $F(1) = 1/2$, and $F(2) = 1$.



CDF Inversion



(a) If $U < 1/2$, we solve $X^2/2 = U$ to get $X = \sqrt{2U}$.
(b) If $U \geq 1/2$, the only root of $1 - (2-X)^2/2 = U$ in $[1,2]$ is

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

Thus, for example, if $U = 0.4$, we get $X = \sqrt{0.8}$.

Remark: Do not replace U by $1-U$ here!

Example: The standard normal distribution. Unfortunately, the inverse cdf $\Phi^{-1}(\cdot)$ does not have an analytical form. *This is often a problem with the inverse transform method.*

Easy solution: Do a table lookup. E.g., If $U = 0.975$, then $Z = \Phi^{-1}(U) = 1.96$.

Crude portable approximation (Banks et al.): The following approximation gives at least one decimal place of accuracy for $0.00134 \le U \le 0.98865$:

$$Z \;=\; \Phi^{-1}(U) \;\approx\; \frac{U^{0.135} - (1-U)^{0.135}}{0.1975}.$$

A better approximation with absolute error $\leq 0.45 \times 10^{-3}$:

$$Z = \text{sign}(U - 1/2)\left(t - \frac{c_0 + c_1 t + c_2 t^2}{1 + d_1 t + d_2 t^2 + d_3 t^3}\right),$$

where $\text{sign}(x) = 1, 0, -1$ if x is positive, zero, or negative, respectively,

$$t = \{-\ell\text{n}[\min(U, 1 - U)]^2\}^{1/2},$$

and

$$\begin{aligned} c_0 &= 2.515517, \quad c_1 = 0.802853, \quad c_2 = 0.010328, \\ d_1 &= 1.432788, \quad d_2 = 0.189269, \quad d_3 = 0.001308. \end{aligned}$$

In any case, if $Z \sim \text{Nor}(0, 1)$ and you want $X \sim \text{Nor}(\mu, \sigma^2)$, just take $X \leftarrow \mu + \sigma Z$.

Example: The geometric distribution with pmf

$$P(X = k) = (1 - p)^{k-1} p,\ k = 1, 2, \ldots.$$

Using the familiar notation $q = 1 - p$, we write the cdf as

$$\begin{aligned}
F(k) &= P(X \le k) = 1 - P(X > k) \\
&= 1 - [P(X = k + 1) + P(X = k + 2) + \cdots] \\
&= 1 - [q^k p + q^{k+1} p + \cdots] \\
&= 1 - pq^k [1 + q + q^2 + \cdots] \\
&= 1 - pq^k \frac{1}{1 - q} \\
&= 1 - q^k.
\end{aligned}$$

CDF Inversion

Since

$$X = \min\left\{k : 1 - q^k \geq U\right\},$$

we have

$$F(X-1) < U \leq F(X).$$

Solving the inequalities

$$1 - q^{X-1} < U \leq 1 - q^X$$

for X we get

$$X - 1 < \frac{\ell\text{n}(1-U)}{\ell\text{n}(1-p)} \leq X \quad \text{or} \quad X = \left\lceil \frac{\ell\text{n}(1-U)}{\ell\text{n}(1-p)} \right\rceil,$$

where $\lceil \cdot \rceil$ is the "ceiling" function.

For instance, if $p = 0.3$ and $U = 0.72$, we obtain

$$X = \left\lceil \frac{\ell\text{n}(0.28)}{\ell\text{n}(0.7)} \right\rceil = 4.$$

Outline

Cutpoint Method (Fishman and Moore)

Suppose we want to generate from the discrete distribution

$$P(X = k) \;=\; p_k, \quad k = a, a+1, \ldots, b$$

with large $b - a$. Index the cdf as

$$q_k \;=\; P(X \le k), \quad k = a, a+1, \ldots, b.$$

In principle, we can use the inverse-transform method to compute $X = \min\{k : q_k \ge U\}$, but the average number of comparisons, C, inside the curly brackets may be large.

Indeed, $P(C = 1) = p_a$, $P(C = 2) = p_{a+1}$, ... Hence the mean

$$\begin{aligned} E(C) &= 1 \cdot p_a + 2 \cdot p_{a+1} + 3 \cdot p_{a+2} + \cdots \\ &= (a+1-a)p_a + (a+2-a)p_{a+1} + (a+3-a)p_{a+2} + \cdots \\ &= E(X+1) - a(p_a + p_{a+1} + p_{a+2} + \cdots) \\ &= E(X) + 1 - a \end{aligned}$$

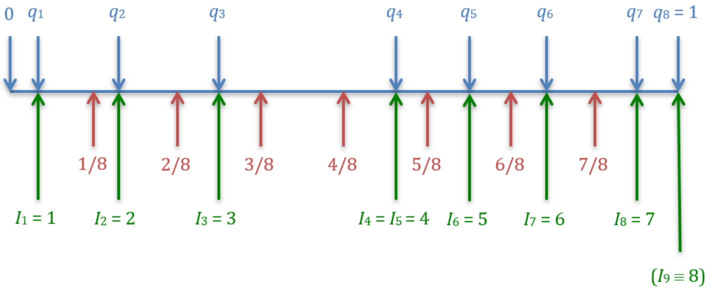
grows linearly with $E(X)$.

The cutpoint method of Fishman and Moore selects an integer $m > 1$ and stores the pointers (cutpoints)

$$I_j \;=\; \min\left\{k : q_k > \frac{j-1}{m}\right\}, \quad j = 1, \ldots, m.$$

These cutpoints help us scan through the list of possible k-values much more quickly than the classical inverse transform method.

Illustration with $a = 1$, $b = 8$, and $m = 8$:



$$I_j = \min\left\{k : q_k > \frac{j-1}{8}\right\}, j = 1, \ldots, 8$$

Intuition:

- we select an interval defined by the red "arrows" (uniformly), and then
- search only between the first two green pointers to the right of the left endpoint of the red interval.

Here is the algorithm that computes the cutpoints:

Algorithm CMSET

$j \leftarrow 0$, $k \leftarrow a - 1$, and $A \leftarrow 0$

While $j < m$:

  While $A \leq j$:

    $k \leftarrow k + 1$

    $A \leftarrow mq_k$

  $j \leftarrow j + 1$

  $I_j \leftarrow k$

Once the cutpoints are computed, we can use the cutpoint method for repeated sampling.

In short, Algorithm CM below selects an index $L = \lfloor mU \rfloor + 1$ at random among $\{1, \ldots, L\}$ and searches within the set $\{I_L, \ldots, I_{L+1}\}$. Its correctness results from the fact that

$$P(I_L \le X \le I_{L+1}) = 1 \quad (I_{m+1} = b).$$

Algorithm CM

Generate U from $\mathcal{U}(0,1)$

$L \leftarrow \lfloor mU \rfloor + 1$ (select a red interval at random)

$X \leftarrow I_L$ (start search from the first green pointer after the left endpoint)

While $U > q_X$: $X \leftarrow X + 1$

Let $E(C_m)$ be the expected number of comparisons until Algorithm CM terminates. Given L, the maximum number of required comparisons is $I_{L+1} - I_L + 1$. Hence,

$$\begin{aligned} E(C_m) &\leq (I_2 - I_1 + 1)\frac{1}{m} + (I_3 - I_2 + 1)\frac{1}{m} + \cdots + (I_{m+1} - I_m + 1)\frac{1}{m} \\ &= \frac{b - I_1 + m}{m}. \end{aligned}$$

Note that if $m \geq b$, then $E(C_m) \leq (2m-1)/m \leq 2$.

Example: Consider the distribution

k	1	2	3	4	5	6	7	8
p_k	.01	.04	.07	.15	.28	.19	.21	.05
q_k	.01	.05	.12	.27	.55	.74	.95	1

For $m = 8$, we have the following cutpoints:

$$\begin{aligned}
I_1 &= \min[i : q_i > 0] = 1 \\
I_2 &= \min[i : q_i > 1/8] = 4 \\
I_3 &= \min[i : q_i > 2/8] = 4 \\
I_4 &= \min[i : q_i > 3/8] = 5 = I_5 \\
I_6 &= \min[i : q_i > 5/8] = 6 \\
I_7 &= \min[i : q_i > 6/8] = 7 = I_8 \\
I_9 &= 8
\end{aligned}$$

For $U = 0.219$, we have $L = \lfloor 8(0.219) \rfloor + 1 = 2$, and

$$X = \min[i : I_2 \le i \le I_3,\ q_i \ge 0.219] = 4.$$

Outline

Actual Data

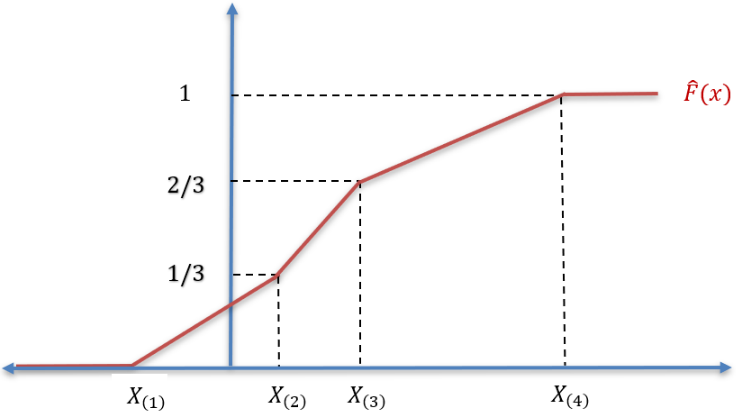
Suppose we have the sample $\{X_1, X_2, \ldots, X_n\}$ from a continuous distribution.

Sort the data in increasing order, and let $X_{(i)}$ denote the ith smallest observation, so that $X_{(1)} \le X_{(2)} \le \cdots \le X_{(n)}$.

In this case we will use the *interpolated* empirical cdf

$$\hat{F}(x) = \begin{cases} 0 & \text{if } x < X_{(1)} \\ \frac{i-1}{n-1} + \frac{x - X_{(i)}}{(n-1)(X_{(i+1)} - X_{(i)})} & \text{if } X_{(i)} \le x < X_{(i+1)} \\ & \quad \text{for } i = 1, \ldots, n-1 \\ 1 & \text{if } x \ge X_{(n)}. \end{cases}$$

Illustration with $n = 4$



Notice that $\hat{F}(x)$ rises most rapidly over the ranges of x that contain more observations.

Although an inverse-transform algorithm might appear to involve some kind of search, the fact that "corners" of $\hat{F}$ occur precisely at levels $0, \frac{1}{n-1}, \frac{2}{n-1}, \ldots, \frac{n-2}{n-1}, 1$ allows us to avoid an explicit search.

Algorithm EIT

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

$P \leftarrow (n-1)U$, $P \leftarrow (n-1)U$, $I \leftarrow \lfloor P \rfloor + 1$

$X \leftarrow X_{(I)} + (P - I + 1)(X_{(I+1)} - X_{(I)})$

Notice that the sorted data must be stored.

A limitation of this method is that all generated values will be between $X_{(1)}$ and $X_{(n)}$. This can be overcome using kernel density estimates (to be discussed later).

Grouped Data

Often the data are grouped into k adjacent intervals $[a_0, a_1), [a_1, a_2), \ldots, [a_{k-1}, a_k]$, so that the ith interval contains n_i observations ($n_1 + n_2 + \cdots + n_k = n$).

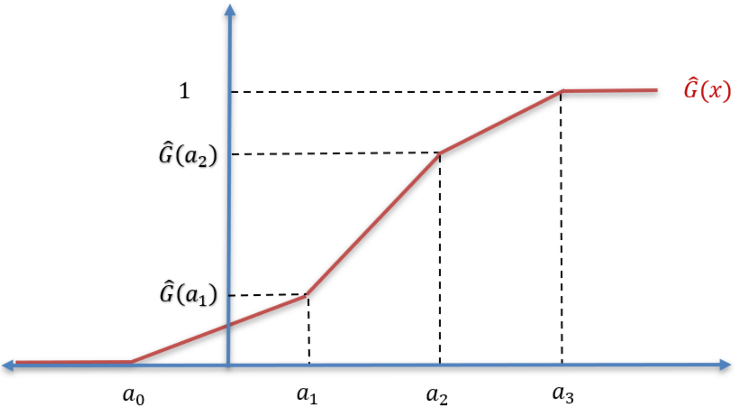
An empirical distribution can be defined by first setting

$$\hat{G}(a_0) = 0, \quad \hat{G}(a_i) = \frac{n_1 + \cdots + n_i}{n} \quad \text{for } i = 1, \ldots, k$$

followed by linear interpolation between the endpoints a_i:

$$\hat{G}(x) = \begin{cases} 0 & \text{if } x \le a_0 \\ \hat{G}(a_{i-1}) + \frac{x - a_{i-1}}{a_i - a_{i-1}}[\hat{G}(a_i) - \hat{G}(a_{i-1})] & \text{if } a_{i-1} \le x < a_i \\ & \quad \text{for } i = 1, \ldots, k \\ 1 & \text{if } x \ge a_k. \end{cases}$$

Illustration with $k = 3$ Intervals



Notice that $\hat{G}(x)$ rises most rapidly over the ranges of x that contain more observations.

Algorithm EGIT

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

Find the index I $(1 \le I \le k-1)$ such that $\hat{G}(a_i) \le U < \hat{G}(a_{i+1})$

$X \leftarrow a_I + \frac{a_{I+1}-a_I}{\hat{G}(a_{I+1})-\hat{G}(a_I)}[U - \hat{G}(a_I)]$

The search for the index I can be performed using the cutpoint method.

A limitation of this method is that all generated values will be between a_0 and a_k. This can be overcome using kernel density estimates (to be discussed later).

Example 8.3 from Banks set al.

$n = 100$ repair times (in hours) are grouped into 4 adjacent intervals.

Interval i $[a_{i-1}, a_i)$	Frequency n_i	Cumulative Freq. $\hat{G}(a_i)$
$[0.25, 0.5)$	31	$31/100 = 0.31$
$[0.5, 1.0)$	10	$41/100 = 0.41$
$[1.0, 1.5)$	25	$66/100 = 0.66$
$[1.5, 2.0)$	34	$100/100 = 1.00$

Let $U = 0.83$. Since $\hat{G}(a_3) \le U < \hat{G}(a_4)$, we have $I = 3$ and

$$X \leftarrow a_3 + \frac{a_4 - a_3}{\hat{G}(a_4) - \hat{G}(a_3)}[U - \hat{G}(a_3)] = 1.5 + \frac{2.0 - 1.5}{1.0 - 0.66}(0.83 - 0.66) = 1.75.$$

Remark: The respective Simio expression for generating realizations is `Random.Continuous(0.25,0,0.5,0.31,1,0.41,1.5,0.66,2,1)`.

Outline

Convolution Method

Convolution refers to adding things up.

Example: Binomial(n, p).

Suppose $X_1, \ldots, X_n$ are iid Bernoulli(p). Then $Y = \sum_{i=1}^{n} X_i \sim$ Bin(n, p).

So how do you get Bernoulli RV's?

Suppose $U_1, \ldots, U_n$ are iid $\mathcal{U}(0,1)$. If $U_i \leq p$, set $X_i = 1$; otherwise, set $X_i = 0$. Repeat for $i = 1, \ldots, n$. $\Box$

Example: Erlang(n, λ).

Suppose $X_1, \ldots, X_n$ are iid Expo(λ). Then $Y = \sum_{i=1}^n X_i \sim$ Erlang(n, λ).

Notice that by inverse transform,

$$\begin{aligned} Y &= \sum_{i=1}^n X_i \\ &= \sum_{i=1}^n \left[-\frac{1}{\lambda} \ell\mathrm{n}(U_i) \right] \\ &= -\frac{1}{\lambda} \sum_{i=1}^n \ell\mathrm{n}(U_i). \quad \Box \end{aligned}$$

Example: A crude "desert island" Nor(0,1) generator.

Suppose that $U_1, \ldots, U_n$ are iid $\mathcal{U}(0,1)$, and let $Y = \sum_{i=1}^n U_i$.

Note that $E(Y) = n/2$ and $\text{Var}(Y) = n/12$.

By the Central Limit Theorem, for large n:

$$Y \approx \text{Nor}(n/2, n/12).$$

In particular, let's choose $n = 12$, and assume that it's "large." Then

$$Y - 6 = \sum_{i=1}^{12} U_i - 6 \approx \text{Nor}(0, 1).$$

It sort of works OK, but I wouldn't use it in real life.

Other convolution-related tidbits:

Did you know. . . ?

$U_1 + U_2 \sim$ triangular$(0, 1, 2)$.

If $X_1, \ldots, X_n$ are iid geometric(p), then $\sum_{i=1}^n X_i \sim$ NegBin(n, p).

If $Z_1, \ldots, Z_n$ are iid Nor(0,1), then $\sum_{i=1}^n Z_i^2 \sim \chi_n^2$.

Outline

Acceptance-Rejection Method

Baby Example: Generate a $\mathcal{U}$(2/3,1) RV. (You would usually do this via inverse transform, but what the heck!)

Here's the A-R algorithm:

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

2. If $U \geq 2/3$, ACCEPT $X \leftarrow U$. Otherwise, REJECT and go to 1.

Remark: This method is actually much deeper — more-thorough treatment is a subject of ISyE 4045.

Example: Consider a Poisson process $\{N(t) : t \geq 0\}$ with rate λ and interarrival times $\{A_i, i = 1, 2, \ldots\}$. If $X = N(1)$, then X has the Poisson distribution with rate λ, that is,

$$P(X = n) = e^{-\lambda} \frac{\lambda^n}{n!}, \quad n = 0, 1, \ldots$$

and

$$\begin{aligned}
X = n \quad & \Leftrightarrow \quad \text{exactly } n \text{ arrivals by time } t = 1 \\
& \Leftrightarrow \quad \sum_{i=1}^{n} A_i \leq 1 < \sum_{i=1}^{n+1} A_i \\
& \Leftrightarrow \quad \sum_{i=1}^{n} \left[-\frac{1}{\lambda} \ell\text{n}(U_i) \right] \leq 1 < \sum_{i=1}^{n+1} \left[-\frac{1}{\lambda} \ell\text{n}(U_i) \right] \\
& \Leftrightarrow \quad -\frac{1}{\lambda} \ell\text{n} \left(\prod_{i=1}^{n} U_i \right) \leq 1 < -\frac{1}{\lambda} \ell\text{n} \left(\prod_{i=1}^{n+1} U_i \right) \\
& \Leftrightarrow \quad \prod_{i=1}^{n} U_i \geq e^{-\lambda} > \prod_{i=1}^{n+1} U_i . \qquad (1)
\end{aligned}$$

The following A-R algorithm samples $\mathcal{U}(0,1)$'s until (1) holds, i.e., until the first time n such that $e^{-\lambda} > \prod_{i=1}^{n+1} U_i$.

Algorithm POIS1

$a \leftarrow e^{-\lambda}$; $p \leftarrow 1$; $X \leftarrow -1$

Until $p < a$

   Generate U from $\mathcal{U}(0,1)$

   $p \leftarrow pU$; $X \leftarrow X + 1$

Return X

Example (Banks et al.): Apply Algorithm POIS1 to obtain a Poisson(2) variate.

Sample until $e^{-\lambda} = 0.1353 > \prod_{i=1}^{n+1} U_i$.

n	U_{n+1}	$\prod_{i=1}^{n+1} U_i$	Stop?
0	0.3911	0.3911	No
1	0.9451	0.3696	No
2	0.5033	0.1860	No
3	0.7003	0.1303	Yes

Thus, we take $X = 3$.

Remark: Unfortunately, the expected number of U's required to generate one realization of X is $E(X) + 1 = \lambda + 1$.

If $\lambda \geq 20$, we can use the normal approximation

$$Z = \frac{X - \lambda}{\sqrt{\lambda}} \approx \text{Nor}(0,1).$$

Algorithm POIS2 (for $\lambda \geq 20$)

$\alpha \leftarrow \sqrt{\lambda}$

Generate Z from Nor$(0,1)$

Return $X = \max(0, \lfloor \lambda + \alpha Z + 0.5 \rfloor)$ (a "continuity correction")

Outline

Normal Distribution: Box–Muller Method

Here's a nice, easy way to generate standard normals.

Theorem: If U_1, U_2 are iid $\mathcal{U}(0,1)$, then

$$\begin{aligned} Z_1 &= \sqrt{-2\ell n(U_1)}\cos(2\pi U_2) \\ Z_2 &= \sqrt{-2\ell n(U_1)}\sin(2\pi U_2) \end{aligned}$$

are iid Nor(0,1).

Note that the trigonometric calculations must be done in radians.

Proof Someday soon. $\square$

Some interesting corollaries follow directly from Box–Muller.

Example: Note that

$$Z_1^2 + Z_2^2 \;\sim\; \chi_1^2 + \chi_1^2 \;\sim\; \chi_2^2.$$

But

$$\begin{aligned} Z_1^2 + Z_2^2 &= -2\ell\mathsf{n}(U_1)[\cos^2(2\pi U_2) + \sin^2(2\pi U_2)] \\ &= -2\ell\mathsf{n}(U_1) \\ &\sim \mathsf{Expo}(1/2). \end{aligned}$$

Thus, we've just proven that

$$\chi_2^2 \;\sim\; \mathsf{Expo}(1/2). \quad \square$$

Remark: The literature contains sophisticated algorithms for generating realizations from other distributions, including the gamma. These algorithms go beyond the scope of this class.

Beta and Pert Distributions

The pdf of the beta(α, β) distribution with shape parameters $\alpha > 0$ and $\beta > 0$ is

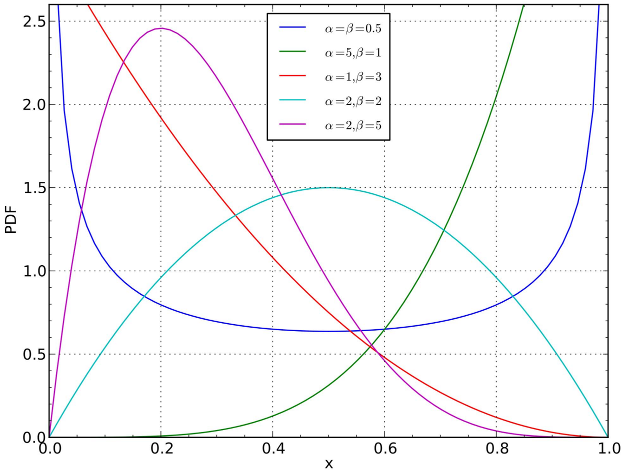
$$f(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \quad 0 < x < 1.$$

If $X \sim$ gamma(α, λ) and $Y \sim$ gamma(β, λ) are independent gamma RVs with shape parameters α and β and common scale parameter λ, then $X/(X+Y) \sim$ beta(α, β).

Mirror image property: $X \sim$ beta$(\alpha, \beta) \Leftrightarrow 1 - X \sim$ beta(β, α).

The Pert(a, b, c) distribution used in Simio (also denoted as beta-Pert) with minimum value a, most likely value b, and maximum value c is obtained from the following transformation (the value in <> is omitted):

$$X \sim \text{beta}\left(1 + 4\,\frac{b-a}{c-a}, 1 + 4\,\frac{c-b}{c-a}\right) \Leftrightarrow a + (c-a)X \sim \text{Pert}(a, b, c, <4>).$$



Beta densities for various combinations of (α, β). Source: `en.wikipedia.org/wiki/Beta_distribution`

Outline

Generating Stochastic Processes

Generating Poisson Arrivals

When the arrival rate is *constant*, say λ, the interarrival times of a PP(λ) are iid expo(λ), and we can generate the arrival times iteratively:

$$\begin{aligned} T_0 &\leftarrow 0 \\ T_i &\leftarrow T_{i-1} - \frac{1}{\lambda}\ell\text{n}(U_i), \quad i \geq 1. \end{aligned}$$

Suppose that we want to generate a *fixed number* n of PP(λ) arrivals in a *fixed time interval* $[a, b]$. To do so, we note a theorem stating that the joint distribution of the n arrivals is the same as the joint distribution of the order statistics of n iid $\mathcal{U}(a, b)$ RVs.

Generate iid $U_1, \ldots, U_n$ from $\mathcal{U}(0, 1)$

Sort the U_i's: $U_{(1)} < U_{(2)} < \cdots < U_{(n)}$

Set the arrival times to $T_i \leftarrow a + (b - a)U_{(i)}$

Nonstationary Poisson Process

Let

$$\begin{aligned}\lambda(t) &= \text{rate (intensity) function at time } t,\\ \Lambda(t) &= \int_0^t \lambda(u)\,du, \quad \text{(cumulative rate function)}\\ N(t) &= \text{number of arrivals during } [0,t].\end{aligned}$$

Then

$$\begin{aligned}N(s+t)-N(s) &\sim \text{Poisson}\Big(\int_s^{s+t} \lambda(u)\,du\Big))\\ &\sim \text{Poisson}\Big(\Lambda(s+t)-\Lambda(s)\Big).\end{aligned}$$

Incorrect NHPP Algorithm [it can "skip" intervals with large $\lambda(t)$]

$T_0 \leftarrow 0$; $i \leftarrow 0$

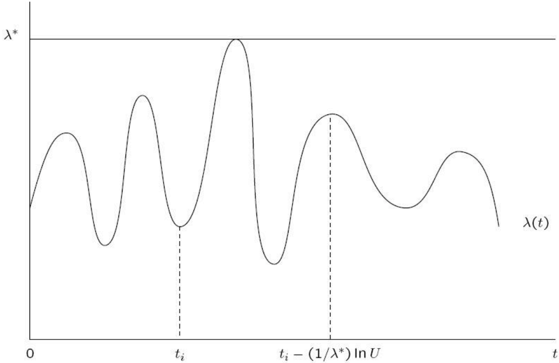
Repeat

Generate U from $\mathcal{U}(0,1)$

$T_{i+1} \leftarrow T_i - \frac{1}{\lambda(T_i)}\ell\mathrm{n}(U)$

$i \leftarrow i + 1$

Don't use this algorithm because it has the potential to "skip" periods of high arrival rates!


λ^*
$\lambda(t)$
0
t_i
$t_i - (1/\lambda^*)\ln U$
t

The following acceptance-rejection algorithm assumes that $\lambda^* \equiv \sup \lambda(t) < \infty$, generates potential arrivals with rate λ^*, and accepts a potential arrival at time t with probability $\lambda(t)/\lambda^*$.

Thinning Algorithm (used by Simio)

$T_0 \leftarrow 0$; $i \leftarrow 0$

Repeat

  $t \leftarrow T_i$

  Repeat

    Generate U, V from $\mathcal{U}(0,1)$

    $t \leftarrow t - \frac{1}{\lambda^*}\ell\mathsf{n}(U)$

  until $V \leq \lambda(t)/\lambda^*$

  $i \leftarrow i + 1$

  $T_i \leftarrow t$

Outline

Good Sources

- Here is an out-of-print book, whose length indicates the extent of the overall research area:
  http://www.nrbook.com/devroye/
- A plethora of codes can be found at
  http://statmath.wu.ac.at/arvag/software.html