Random Variates Generation

Chapter 4. Generating Discrete Random Variates

Chapter 5. Generating Continuous Random Variates

Inverse Transform Method

Theorem: Let $F(\cdot)$ be the cdf of the r.v. X, then

1. $U = F(X) \sim U(0,1)$.
2. Let $F^{-1}(u) = \inf\{x : F(x) > u\}, 0 \le u \le 1$. Then $F^{-1}(U) \sim X$, where $U \sim U(0,1)$.

Proof: See Casella and Berger (2002). □

Based on 2, one can generate $U(0,1)$ random numbers, then **take the inverse** of F to get random variates with distribution F.

Inverse Transform Method

Theorem: Let $F(\cdot)$ be the cdf of the r.v. X, then

1. $U = F(X) \sim U(0,1)$.
2. Let $F^{-1}(u) = \inf\{x : F(x) > u\}, 0 \le u \le 1$. Then $F^{-1}(U) \sim X$, where $U \sim U(0,1)$.

Proof: See Casella and Berger (2002). □

Based on 2, one can generate $U(0,1)$ random numbers, then **take the inverse** of F to get random variates with distribution F.

Inverse Transform Method

Theorem: Let $F(\cdot)$ be the cdf of the r.v. X, then

1. $U = F(X) \sim U(0,1)$.
2. Let $F^{-1}(u) = \inf\{x : F(x) > u\}, 0 \leq u \leq 1$. Then $F^{-1}(U) \sim X$, where $U \sim U(0,1)$.

Proof: See Casella and Berger (2002). $\square$

Based on 2, one can generate $U(0,1)$ random numbers, then **take the inverse** of F to get random variates with distribution F.

Continuous Case: $F \uparrow$ on the support of X, so F^{-1} exists.

Ex. 1. $X \sim Exp(1)$, $f(x) = e^{-x}, x > 0$; $F(x) = 1 - e^{-x}, x > 0$.

(i) Find F^{-1}:

Let $u = 1 - e^{-x} \Rightarrow x = -\log(1-u), 0 < u < 1$.

(ii) $U \sim U(0,1)$, then $X = -\log(1-U) \sim Exp(1)$.

Note that $1 - U$ and U are both $U(0,1)$, hence

$$\mathbf{X = -\log U \sim Exp(1)} \quad \text{if} \quad U \sim U(0,1).$$

Continuous Case: $F \uparrow$ on the support of X, so F^{-1} exists.

Ex. 1. $X \sim Exp(1)$, $f(x) = e^{-x}, x > 0$; $F(x) = 1 - e^{-x}, x > 0$.

(i) Find F^{-1}:

Let $u = 1 - e^{-x} \Rightarrow x = -\log(1-u), 0 < u < 1$.

(ii) $U \sim U(0,1)$, then $X = -\log(1-U) \sim Exp(1)$.

Note that $1 - U$ and U are both $U(0,1)$, hence

$$\mathbf{X} = -\log \mathbf{U} \sim \mathbf{Exp(1)} \text{ if } U \sim U(0,1).$$

Continuous Case: $F \uparrow$ on the support of X, so F^{-1} exists.

Ex. 1. $X \sim Exp(1)$, $f(x) = e^{-x}, x > 0$; $F(x) = 1 - e^{-x}, x > 0$.

(i) Find F^{-1}:

Let $u = 1 - e^{-x} \Rightarrow x = -\log(1-u), 0 < u < 1$.

(ii) $U \sim U(0,1)$, then $X = -\log(1-U) \sim Exp(1)$.

Note that $1 - U$ and U are both $U(0,1)$, hence

$$\mathbf{X = -\log U \sim Exp(1)} \text{ if } U \sim U(0,1).$$

Continuous Case: $F \uparrow$ on the support of X, so F^{-1} exists.

Ex. 1. $X \sim Exp(1)$, $f(x) = e^{-x}, x > 0$; $F(x) = 1 - e^{-x}, x > 0$.

(i) Find F^{-1}:

Let $u = 1 - e^{-x} \Rightarrow x = -\log(1-u), 0 < u < 1$.

(ii) $U \sim U(0,1)$, then $X = -\log(1-U) \sim Exp(1)$.

Note that $1 - U$ and U are both $U(0,1)$, hence

$$\mathbf{X} = -\log \mathbf{U} \sim \mathbf{Exp(1)} \quad \text{if } U \sim U(0,1).$$

$\square$

Ex. 2. $Y \sim Exp(\lambda)$, $f(y) = \lambda e^{-\lambda y}, y > 0, \lambda > 0$.

Here $1/\lambda$ is a *scale* parameter, i.e. $Y/(\frac{1}{\lambda}) \sim Exp(1) \equiv X$, thus,

$$\mathbf{Y} = \frac{X}{\lambda} = -\frac{\mathbf{1}}{\lambda} \log \mathbf{U}, \quad \text{where } U \sim U(0,1).$$

□

Exercise: Use Inverse Transform for Y directly.

Ex. 2. $Y \sim Exp(\lambda)$, $f(y) = \lambda e^{-\lambda y}, y > 0, \lambda > 0$.

Here $1/\lambda$ is a *scale* parameter, i.e. $Y/(\frac{1}{\lambda}) \sim Exp(1) \equiv X$, thus,

$$\mathbf{Y} = \frac{X}{\lambda} = -\frac{\mathbf{1}}{\lambda} \log \mathbf{U}, \quad \text{where } U \sim U(0,1).$$

□

Exercise: Use Inverse Transform for Y directly.

Ex. 2. $Y \sim Exp(\lambda)$, $f(y) = \lambda e^{-\lambda y}, y > 0, \lambda > 0$.

Here $1/\lambda$ is a *scale* parameter, i.e. $Y/(\frac{1}{\lambda}) \sim Exp(1) \equiv X$, thus,

$$\mathbf{Y} = \frac{X}{\lambda} = -\frac{\mathbf{1}}{\lambda} \log \mathbf{U}, \quad \text{where } U \sim U(0,1).$$

$\square$

Exercise: Use Inverse Transform for Y directly.

Ex. 2. $Y \sim Exp(\lambda)$, $f(y) = \lambda e^{-\lambda y}, y > 0, \lambda > 0$.

Here $1/\lambda$ is a *scale* parameter, i.e. $Y/(\frac{1}{\lambda}) \sim Exp(1) \equiv X$, thus,

$$\mathbf{Y} = \frac{X}{\lambda} = -\frac{\mathbf{1}}{\lambda} \log \mathbf{U}, \quad \text{where } U \sim U(0,1).$$

$\square$

Exercise: Use Inverse Transform for Y directly.

[**Discrete Case**]: $P(X = x_i) = p_i > 0, i = 1, 2, \ldots,$ such that $\sum_i p_i = 1$; $F(x) = \sum_{i: x_i \le x} p_i$. For $0 \le y < 1$,

$$F^{-1}(y) = \begin{cases} x_1 & 0 \le y < p_1 \\ x_2 & p_1 \le y < p_1 + p_2 \\ \vdots & \vdots \\ x_i & \sum_{j=1}^{i-1} p_j \le y < \sum_{j=1}^{i} p_j \\ \vdots & \vdots \end{cases}$$

Discrete Case: $P(X = x_i) = p_i > 0, i = 1, 2, \ldots,$ such that

$\sum_i p_i = 1; \quad F(x) = \sum_{i: x_i \le x} p_i$. For $0 \le y < 1$,

$$F^{-1}(y) = \begin{cases} x_1 & 0 \le y < p_1 \\ x_2 & p_1 \le y < p_1 + p_2 \\ \vdots & \vdots \\ x_i & \sum_{j=1}^{i-1} p_j \le y < \sum_{j=1}^{i} p_j \\ \vdots & \vdots \end{cases}$$

Discrete Case: $P(X = x_i) = p_i > 0, i = 1, 2, \ldots,$ such that $\sum_i p_i = 1; \quad F(x) = \sum_{i:x_i \le x} p_i$. For $0 \le y < 1$,

$$F^{-1}(y) = \begin{cases} x_1 & 0 \le y < p_1 \\ x_2 & p_1 \le y < p_1 + p_2 \\ \vdots & \vdots \\ x_i & \sum_{j=1}^{i-1} p_j \le y < \sum_{j=1}^{i} p_j \\ \vdots & \vdots \end{cases}$$

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

2. If $0 \leq U < p_1$, **set** $X = x_1$

$p_1 \leq U < p_1 + p_2$, **set** $X = x_2$

$\vdots$

$p_1 + \ldots + p_{i-1} \leq U < p_1 + \ldots + p_i$, **set** $X = x_i$

$\vdots$

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

2. If $0 \leq U < p_1$, **set** $X = x_1$

$p_1 \leq U < p_1 + p_2$, **set** $X = x_2$

$\vdots$

$p_1 + \ldots + p_{i-1} \leq U < p_1 + \ldots + p_i$, **set** $X = x_i$

$\vdots$

Justification: $X \sim$??

Answer:

$$\begin{aligned} \mathbf{P}(\mathbf{X} = \mathbf{x_i}) &= P\left(\sum_{j=1}^{i-1} p_j \le U < \sum_{j=1}^{i} p_j\right) \\ &= \sum_{j=1}^{i} p_j - \sum_{j=1}^{i-1} p_j \\ &= \mathbf{p_i}, \end{aligned}$$

$i = 1, 2, \ldots$. Okay!

Justification: $X \sim$??

<u>Answer</u>:

$$\begin{aligned} \mathbf{P(X = x_i)} &= P\left(\sum_{j=1}^{i-1} p_j \le U < \sum_{j=1}^{i} p_j\right) \\ &= \sum_{j=1}^{i} p_j - \sum_{j=1}^{i-1} p_j \\ &= \mathbf{p_i}, \end{aligned}$$

$i = 1, 2, \ldots$. Okay!

Justification: $X \sim$??

Answer:

$$\mathbf{P(X = x_i)} = P\left(\sum_{j=1}^{i-1} p_j \le U < \sum_{j=1}^{i} p_j\right) = \sum_{j=1}^{i} p_j - \sum_{j=1}^{i-1} p_j = \mathbf{p_i},$$

$i = 1, 2, \ldots$. Okay!

Justification: $X \sim$??

Answer:

$$\begin{aligned}
\mathbf{P(X = x_i)} &= P\left(\sum_{j=1}^{i-1} p_j \le U < \sum_{j=1}^{i} p_j\right) \\
&= \sum_{j=1}^{i} p_j - \sum_{j=1}^{i-1} p_j \\
&= \mathbf{p_i},
\end{aligned}$$

$i = 1, 2, \ldots.$ Okay!

Justification: $X \sim$??

Answer:

$$\begin{aligned} \mathbf{P(X = x_i)} &= P(\sum_{j=1}^{i-1} p_j \leq U < \sum_{j=1}^{i} p_j) \\ &= \sum_{j=1}^{i} p_j - \sum_{j=1}^{i-1} p_j \\ &= \mathbf{p_i}, \end{aligned}$$

$i = 1, 2, \ldots$. Okay!

More efficient procedure:

1. **Order** $p_{(1)} \geq p_{(2)} \geq p_{(3)} \geq \cdots$ with associated $x_{(1)}, x_{(2)}, x_{(3)} \cdots$ etc.
2. Generate $U \sim U(0,1)$.
3. $X = \begin{cases} x_{(1)} & 0 \leq U < p_{(1)} \\ x_{(2)} & p_{(1)} \leq U < p_{(1)} + p_{(2)} \\ \vdots & \vdots \end{cases}$.

Note: 1. The order is determined by p_i, not x_i.

2. $x_{(1)}$ is indeed the **mode** of the distribution, so the algorithm begins with the mode.

More efficient procedure:

1. **Order** $p_{(1)} \geq p_{(2)} \geq p_{(3)} \geq \cdots$ with associated $x_{(1)}, x_{(2)}, x_{(3)} \cdots$ etc.
2. Generate $U \sim U(0,1)$.
3. $X = \begin{cases} x_{(1)} & 0 \leq U < p_{(1)} \\ x_{(2)} & p_{(1)} \leq U < p_{(1)} + p_{(2)} \\ \vdots & \vdots \end{cases}$.

Note: 1. The order is determined by p_i, not x_i.

2. $x_{(1)}$ is indeed the **mode** of the distribution, so the algorithm begins with the mode.

More efficient procedure:

1. **Order** $p_{(1)} \geq p_{(2)} \geq p_{(3)} \geq \cdots$ with associated $x_{(1)}, x_{(2)}, x_{(3)} \cdots$ etc.
2. Generate $U \sim U(0,1)$.
3. $X = \begin{cases} x_{(1)} & 0 \leq U < p_{(1)} \\ x_{(2)} & p_{(1)} \leq U < p_{(1)} + p_{(2)} \\ \vdots & \vdots \end{cases}$.

Note: 1. The order is determined by p_i, not x_i.

2. $x_{(1)}$ is indeed the **mode** of the distribution, so the algorithm begins with the mode.

Special Cases

Ex. 1. **Uniform r.v.**, i.e. $P(X = i) = 1/n$, $i = 1, \ldots, n$.

So $x_i = i$ and $p_i = 1/n$, for $i = 1, \ldots, n$; $\sum_{j=1}^{i} p_i = i/n$.

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

$$X = \begin{cases} 1 & 0 \le U < \frac{1}{n} \\ 2 & \frac{1}{n} \le U < \frac{2}{n} \\ \vdots & \\ n & \frac{n-1}{n} \le U < 1 \end{cases} \iff \begin{cases} 1 & 0 \le nU < 1 \\ 2 & 1 \le nU < 2 \\ \vdots & \\ n & n-1 \le nU < n \end{cases}$$

Hence, $X = [nU] + 1 = \textbf{Int}(nU) + 1$. $\square$

Special Cases

Ex. 1. **Uniform r.v.**, i.e. $P(X = i) = 1/n$, $i = 1, \ldots, n$.

So $x_i = i$ and $p_i = 1/n$, for $i = 1, \ldots, n$; $\sum_{j=1}^{i} p_i = i/n$.

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

$$X = \begin{cases} 1 & 0 \le U < \frac{1}{n} \\ 2 & \frac{1}{n} \le U < \frac{2}{n} \\ \vdots & \\ n & \frac{n-1}{n} \le U < 1 \end{cases} \iff \begin{cases} 1 & 0 \le nU < 1 \\ 2 & 1 \le nU < 2 \\ \vdots & \\ n & n-1 \le nU < n \end{cases}$$

Hence, $X = [nU] + 1 = \textbf{Int}(nU) + 1$. □

Special Cases

Ex. 1. **Uniform r.v.**, i.e. $P(X = i) = 1/n$, $i = 1, \dots, n$.

So $x_i = i$ and $p_i = 1/n$, for $i = 1, \dots, n$; $\sum_{j=1}^{i} p_i = i/n$.

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

$$X = \begin{cases} 1 & 0 \le U < \frac{1}{n} \\ 2 & \frac{1}{n} \le U < \frac{2}{n} \\ \vdots & \\ n & \frac{n-1}{n} \le U < 1 \end{cases} \Longleftrightarrow \begin{cases} 1 & 0 \le nU < 1 \\ 2 & 1 \le nU < 2 \\ \vdots & \\ n & n-1 \le nU < n \end{cases}$$

Hence, $X = [nU] + 1 = \textbf{Int}(nU) + 1$. □

Special Cases

Ex. 1. **Uniform r.v.**, i.e. $P(X = i) = 1/n$, $i = 1, \ldots, n$.

So $x_i = i$ and $p_i = 1/n$, for $i = 1, \ldots, n$; $\sum_{j=1}^{i} p_i = i/n$.

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

$$X = \begin{cases} 1 & 0 \leq U < \frac{1}{n} \\ 2 & \frac{1}{n} \leq U < \frac{2}{n} \\ \vdots & \\ n & \frac{n-1}{n} \leq U < 1 \end{cases} \iff \begin{cases} 1 & 0 \leq nU < 1 \\ 2 & 1 \leq nU < 2 \\ \vdots & \\ n & n-1 \leq nU < n \end{cases}$$

Hence, $X = [nU] + 1 = \textbf{Int}(nU) + 1$. $\square$

Application

1. **Random permutation**.

Method I:

1. choose one of the numbers $1, \ldots, n$ at random and then put that number in position n.
2. Then chooses at random one of the remaining $n-1$ numbers and puts that number in position $n-1$; and so on.

Application

1. **Random permutation.**

Method I:

1. choose one of the numbers $1, \ldots, n$ at random and then put that number in position n.
2. Then chooses at random one of the remaining $n-1$ numbers and puts that number in position $n-1$; and so on.

Application

1. **Random permutation**.

Method I:

1. choose one of the numbers $1, \ldots, n$ at random and then put that number in position n.
2. Then chooses at random one of the remaining $n-1$ numbers and puts that number in position $n-1$; and so on.

Application

1. **Random permutation** (Cont'd).

Method II: We do not have to consider exactly which of the numbers remain to be positioned, it is convenient and efficient to keep the numbers in an ordered list and then randomly choose the position of the number rather than the number itself.

Application

1. **Random permutation** (Cont'd).

1. Starting with any initial ordering $P_1, \ldots, P_n$, we pick one of the positions $1, \ldots, n$ at random, then **interchange the number** in that position with the one in position n.
2. Randomly choose one of the positions $1, \ldots, n-1$ and interchange the number in this position with the one in position $n-1$, and so on.

Application

1. **Random permutation** (Cont'd).

1. Starting with any initial ordering $P_1, \ldots, P_n$, we pick one of the positions $1, \ldots, n$ at random, then **interchange the number** in that position with the one in position n.
2. Randomly choose one of the positions $1, \ldots, n-1$ and interchange the number in this position with the one in position $n-1$, and so on.

Application

1. **Random permutation** (Cont'd).

Algorithm:

STEP 1: Let $P_1, P_2, \ldots, P_n$ be any permutation of $1, 2, \ldots, n$ (e.g., we can choose $P_j = j, j = 1, 2, \ldots, n$)

STEP 2: Set $k = n$.

STEP 3: Generate a random number U and let $I = \text{Int}(kU) + 1$.

Application

1. **Random permutation** (Cont'd).

Algorithm:

STEP 1: Let $P_1, P_2, \ldots, P_n$ be any permutation of $1, 2, \ldots, n$ (e.g., we can choose $P_j = j, j = 1, 2, \ldots, n$)

STEP 2: Set $k = n$.

STEP 3: Generate a random number U and let $I = \text{Int}(kU) + 1$.

Application

1. **Random permutation** (Cont'd).

Algorithm:

STEP 1: Let $P_1, P_2, \ldots, P_n$ be any permutation of $1, 2, \ldots, n$ (e.g., we can choose $P_j = j, j = 1, 2, \ldots, n$)

STEP 2: Set $k = n$.

STEP 3: Generate a random number U and let $I = \text{Int}(kU) + 1$.

STEP 4: Interchange the values of P_I and P_k.

STEP 5: Let $k = k - 1$ and if $k > 1$ go to **Step 3**.

STEP 6: $P_1, \ldots, P_n$ is the desired random permutation. $\square$

Exercise: Generate a random subset of size r, of the integers $1, \ldots, n$.

Application

2. **Calculating Averages**

Suppose we want to approximate $\overline{a} = \sum_{i=1}^{n} a(i)/n$, where n is large and the value $a(i), i = 1, \ldots, n$, are complicated and not easily calculated.

<u>**Recall**</u>: If X is a discrete uniform random variable over the integers $1, \ldots, n$, then the random variable $a(X)$ has a mean given by

$$E[a(X)] = \sum_{i=1}^{n} a(i) P\{X = i\} = \sum_{i=1}^{n} \frac{a(i)}{n} = \overline{a}.$$

Application

2. **Calculating Averages**

Suppose we want to approximate $\overline{a} = \sum_{i=1}^{n} a(i)/n$, where n is large and the value $a(i), i = 1, \ldots, n$, are complicated and not easily calculated.

Recall: If X is a discrete uniform random variable over the integers $1, \ldots, n$, then the random variable $a(X)$ has a mean given by

$$E[a(X)] = \sum_{i=1}^{n} a(i)P\{X = i\} = \sum_{i=1}^{n} \frac{a(i)}{n} = \overline{a}.$$

Application

2. **Calculating Averages**

Suppose we want to approximate $\overline{a} = \sum_{i=1}^{n} a(i)/n$, where n is large and the value $a(i), i = 1, \ldots, n$, are complicated and not easily calculated.

Recall: If X is a discrete uniform random variable over the integers $1, \ldots, n$, then the random variable $a(X)$ has a mean given by

$$E[a(X)] = \sum_{i=1}^{n} a(i) P\{X = i\} = \sum_{i=1}^{n} \frac{a(i)}{n} = \overline{a}.$$

Algorithm:

STEP 1: Generate $U_i \sim U(0,1)$ and set $X_i = \text{Int}(nU_i) + 1$, for $i = 1, \ldots, k$. ($E[a(X_i)] = \overline{a}$.)

STEP 2: $\overline{a} \approx \sum_{i=1}^{k} \frac{a(X_i)}{k}$. $\square$

Note: As k is large (though much smaller than n) the average of these values should approximately equal $\overline{a}$.

Ex. 2. **Poisson r. v.**. $X \sim P(\lambda)$, $P(X = i) = \frac{e^{-\lambda}\lambda^i}{i!} = p_i$,

$i = 0, 1, 2, \ldots$; $F(x) = \sum_{i=0}^{[x]} \frac{e^{-\lambda}\lambda^i}{i!} = \sum_{i=1}^{[x]} p_i, x \geq 0 \ldots,\quad F^{-1} = ?$

X is a discrete r.v. and moreover,

$$p_{i+1} = \frac{e^{-\lambda}\lambda^{i+1}}{(i+1)!} = \frac{\lambda}{i+1} p_i, i = 0, 1, 2, \ldots$$

Ex. 2. **Poisson r. v.**. $X \sim P(\lambda)$, $P(X = i) = \frac{e^{-\lambda}\lambda^i}{i!} = p_i$,

$i = 0, 1, 2, \dots$; $F(x) = \sum_{i=0}^{[x]} \frac{e^{-\lambda}\lambda^i}{i!} = \sum_{i=1}^{[x]} p_i, x \geq 0 \dots, \quad F^{-1} = ?$

X is a discrete r.v. and moreover,

$$p_{i+1} = \frac{e^{-\lambda}\lambda^{i+1}}{(i+1)!} = \frac{\lambda}{i+1} p_i, i = 0, 1, 2, \dots$$

Ex. 2. **Poisson r.v.** (Cont'd).

Algorithm I: Start with p_0.

1. Generate $U \sim U(0,1)$.
2. $$X = \begin{cases} 0 & 0 \le U < p_0 = e^{-\lambda} \\ 1 & p_0 \le U < p_0 + p_1 \qquad (p_1 = \lambda p_0) \\ 2 & p_0 + p_1 \le U < p_0 + p_1 + p_2 \qquad (p_2 = \frac{\lambda}{2} p_1) \\ \vdots & \vdots \end{cases}.$$

$\square$

Ex. 2. **Poisson r.v.** (Cont'd).

Note: 1. The number of 'searches' in Step 2 is determined by U,

so it is a r.v., say N, with pmf

$$
\begin{aligned}
P(N=1) &= P(X=0) = p_0 \\
P(N=2) &= P(X=1) = p_1 \\
&\vdots
\end{aligned}
$$

So $P(N=k) = P(X=k-1) = p_{k-1}, k = 1, 2, \ldots.$

<u>Ex</u>. 2. **Poisson r.v.** (Cont'd).

<u>Note</u>: 2. $EN = \sum_{k=1}^{\infty} kP(N=k) = \sum_{k=1}^{\infty} kP(X=k-1)$

$$= \sum_{j=0}^{\infty}(j+1)P(X=j) = 1 + EX.$$

Thus, on the average, it needs $1 + EX = 1 + \lambda$ iterations

to generate a $P(\lambda)$ r.v. What about if λ is **large**?

3. For large λ, $\max p_i = \max\{p_{[\lambda]}, p_{[\lambda]+1}\}$,

i.e. the **mode** is around λ.

Ex. 2. **Poisson r.v.** (Cont'd).

Note: 2. $EN = \sum_{k=1}^{\infty} kP(N=k) = \sum_{k=1}^{\infty} kP(X=k-1)$

$= \sum_{j=0}^{\infty} (j+1)P(X=j) = 1 + EX$.

Thus, on the average, it needs $1 + EX = 1 + \lambda$ iterations

to generate a $P(\lambda)$ r.v. What about if λ is **large**?

3. For large λ, $\max p_i = \max\{p_{[\lambda]}, p_{[\lambda]+1}\}$,

i.e. the **mode** is around λ.

Ex. 2. **Poisson r.v.** (Cont'd).

Algorithm II: Start with $p_{(1)} = p_{[\lambda]}$, then $p_{(2)} = p_{[\lambda]+1}$,

$$p_{(3)} = p_{[\lambda]-1}, p_{(4)} = p_{[\lambda]+2}, \ldots, \text{ etc.} \qquad \square$$

<u>Ex</u>. 2. **Poisson r.v.** (Cont'd).

<u>Note</u>: 1. p_{i+1}, p_{i-1} can also be recursively calculated from p_i.

2. On the average, as λ large, **Simulation II** is faster than I.

In II, $N \leq 1 + 2|X - \text{Int}(\lambda)|$, so

$EN < 1 + 2E|X - \text{Int}(\lambda)|$.

If λ is large, $X \approx N(\lambda, \lambda)$, $(X - \lambda)/\sqrt{\lambda} \approx N(0,1) \equiv Z$,

so $E|X - \lambda| \approx \sqrt{\lambda}E|Z| = \sqrt{\lambda}\sqrt{2/\pi} < \sqrt{\lambda}$.

Hence, $EN << 1 + \lambda$.

3. For large λ, to avoid the computer round-off error, use

$p_m = \exp(-\lambda + m\ln\lambda - \sum_{k=1}^{m} \ln k)$, $m = [\lambda]$.

Ex. 2. **Poisson r.v.** (Cont'd).

Note: 1. p_{i+1}, p_{i-1} can also be recursively calculated from p_i.

2. On the average, as λ large, **Simulation II** is faster than I.

In II, $N \leq 1 + 2|X - \text{Int}(\lambda)|$, so

$EN < 1 + 2E|X - \text{Int}(\lambda)|$.

If λ is large, $X \approx N(\lambda, \lambda)$, $(X - \lambda)/\sqrt{\lambda} \approx N(0,1) \equiv Z$,

so $E|X - \lambda| \approx \sqrt{\lambda} E|Z| = \sqrt{\lambda}\sqrt{2/\pi} < \sqrt{\lambda}$.

Hence, $EN << 1 + \lambda$.

3. For large λ, to avoid the computer round-off error, use

$p_m = \exp(-\lambda + m \ln \lambda - \sum_{k=1}^{m} \ln k)$, $m = [\lambda]$.

Ex. 2. **Poisson r.v.** (Cont'd).

Note: 1. p_{i+1}, p_{i-1} can also be recursively calculated from p_i.

2. On the average, as λ large, **Simulation II** is faster than I.

In II, $N \leq 1 + 2|X - \text{Int}(\lambda)|$, so

$EN < 1 + 2E|X - \text{Int}(\lambda)|$.

If λ is large, $X \approx N(\lambda, \lambda)$, $(X - \lambda)/\sqrt{\lambda} \approx N(0,1) \equiv Z$,

so $E|X - \lambda| \approx \sqrt{\lambda}E|Z| = \sqrt{\lambda}\sqrt{2/\pi} < \sqrt{\lambda}$.

Hence, $EN << 1 + \lambda$.

3. For large λ, to avoid the computer round-off error, use

$p_m = \exp(-\lambda + m \ln \lambda - \sum_{k=1}^{m} \ln k)$, $m = [\lambda]$.

Ex. 2. **Poisson r.v.** (Cont'd).

Note: 1. p_{i+1}, p_{i-1} can also be recursively calculated from p_i.

2. On the average, as λ large, **Simulation II** is faster than I.

In II, $N \le 1 + 2|X - \text{Int}(\lambda)|$, so

$EN < 1 + 2E|X - \text{Int}(\lambda)|$.

If λ is large, $X \approx N(\lambda, \lambda)$, $(X - \lambda)/\sqrt{\lambda} \approx N(0,1) \equiv Z$,

so $E|X - \lambda| \approx \sqrt{\lambda}E|Z| = \sqrt{\lambda}\sqrt{2/\pi} < \sqrt{\lambda}$.

Hence, $EN << 1 + \lambda$.

3. For large λ, to avoid the computer round-off error, use

$p_m = \exp(-\lambda + m \ln \lambda - \sum_{k=1}^{m} \ln k)$, $m = [\lambda]$.

Ex. 3. **Bernoulli r.v.**. $X \sim Ber(p)$, i.e. $P(X=1) = p = 1 - P(X=0)$;

$x_1 = 1, x_2 = 0$, and $p_1 = p$.

Algorithm:

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

STEP 2: $X = 1$ if $0 \leq U < p$; otherwise, $X = 0$. $\square$

Ex. 4. **Binomial r.v.**. $X \sim bin(n, p)$.

Note that $X = \sum_{i=1}^{n} X_i$, X_i i.i.d. $Ber(p)$.

Algorithm:

STEP 1: $X = 0$.

STEP 2: Generate $U \sim U(0, 1)$.

STEP 3: If $U < p$, set $X = X + 1$.

STEP 4: Repeat 2, 3 n times; $X \sim bin(n, p)$. □

(See text for other alternatives.)

Ex. 4. **Binomial r.v.**. $X \sim bin(n,p)$.

Note that $X = \sum_{i=1}^{n} X_i$, X_i i.i.d. $Ber(p)$.

Algorithm:

STEP 1: $X = 0$.

STEP 2: Generate $U \sim U(0,1)$.

STEP 3: If $U < p$, set $X = X + 1$.

STEP 4: Repeat 2, 3 n times; $X \sim bin(n,p)$. $\square$

(See text for other alternatives.)

Ex. 5. **Geometric r.v.**. $X \sim Geo(p)$, i.e. $P(X = i) = (1-p)^{i-1}p$,

$i = 1, 2, \ldots$

Method I: $X =$ number of trials up to the *first success* in a sequence of $Ber(p)$ trials.

Algorithm:

STEP 1: Set $n = 0$.

STEP 2: Set $n = n + 1$.

STEP 3: Generate $U \sim U(0, 1)$.

STEP 4: If $U < p$, set $X = n$, stop; else, return to 2.

Ex. 5. **Geometric r.v.** (Cont'd).

Method II: Inverse Transform,

$$F(x) = 1 - (1-p)^i, i \leq x < i+1; i = 1, 2, \ldots.$$

For $\tilde{F}(x) = 1 - (1-p)^x, x > 0$, $\tilde{F}^{-1}(y) = \dfrac{\log(1-y)}{\log(1-p)}, 0 < y < 1$.

Ex. 5. **Geometric r.v.** (Cont'd).

Note that

$$
\begin{aligned}
x = F^{-1}(u) &= \inf\{x : F(x) > u\} \\
&= \inf\{j : j > \tilde{F}^{-1}(u), \text{ integer }\} \\
&= \text{smallest integer } j \text{ such that } j > \tfrac{\log(1-u)}{\log(1-p)} \\
&= \left[\frac{\log(1-u)}{\log(1-p)}\right] + 1, \quad 0 < u < 1.
\end{aligned}
$$

Recall that $1 - U \sim U \sim U(0,1)$, so $\log(1-U) \sim \log U$.

Ex. 5. **Geometric r.v.** (Cont'd).

Note that

$$
\begin{aligned}
x = F^{-1}(u) &= \inf\{x : F(x) > u\} \\
&= \inf\{j : j > \tilde{F}^{-1}(u), \text{ integer } \} \\
&= \textbf{smallest integer } j \text{ such that } j > \tfrac{\log(1-u)}{\log(1-p)} \\
&= \left[\frac{\log(1-u)}{\log(1-p)}\right] + 1, \quad 0 < u < 1.
\end{aligned}
$$

Recall that $1 - U \sim U \sim U(0,1)$, so $\log(1-U) \sim \log U$.

Ex. 5. **Geometric r.v.** (Cont'd).

Note that

$$
\begin{aligned}
x = F^{-1}(u) &= \inf\{x : F(x) > u\} \\
&= \inf\{j : j > \tilde{F}^{-1}(u),\ \text{integer}\ \} \\
&= \textbf{smallest integer } j \text{ such that } j > \tfrac{\log(1-u)}{\log(1-p)} \\
&= \left[\frac{\log(1-u)}{\log(1-p)}\right] + 1,\quad 0 < u < 1.
\end{aligned}
$$

Recall that $1 - U \sim U \sim U(0,1)$, so $\log(1-U) \sim \log U$.

Ex. 5. **Geometric r.v.** (Cont'd).

Note that

$$\begin{aligned} x = F^{-1}(u) &= \inf\{x : F(x) > u\} \\ &= \inf\{j : j > \tilde{F}^{-1}(u), \text{ integer } \} \\ &= \textbf{smallest integer } j \text{ such that } j > \tfrac{\log(1-u)}{\log(1-p)} \\ &= \left[\frac{\log(1-u)}{\log(1-p)}\right] + 1, \quad 0 < u < 1. \end{aligned}$$

Recall that $1 - U \sim U \sim U(0,1)$, so $\log(1-U) \sim \log U$.

Ex. 5. **Geometric r.v.**(Cont'd).

Algorithm:

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

STEP 2: $X = \textbf{Int}(\frac{\log U}{\log(1-p)}) + 1$. $\square$

Note: Method I is more time consuming, for it needs more comparisons and more random number generations.

Ex. Simulation of the **largest observation** in a random sample.

$X_1, \ldots, X_n \overset{i.i.d.}{\sim} F(\cdot)$. Let $Y_1 \leq Y_2 \leq \ldots \leq Y_n$ be the order statistics of the sample. How to simulate Y_1 or Y_n?

Method I:

1. Generate $X_1, \ldots, X_n$ from F.

2. Let $Y_1 = \min\{X_1, \ldots, X_n\}$ $(Y_n = \max\{X_1, \ldots, X_n\})$.

Ex. Simulation of the **largest observation** in a random sample.

$X_1, \ldots, X_n \overset{i.i.d.}{\sim} F(\cdot)$. Let $Y_1 \leq Y_2 \leq \ldots \leq Y_n$ be the order statistics of the sample. How to simulate Y_1 or Y_n?

Method I:

1. Generate $X_1, \ldots, X_n$ from F.
2. Let $Y_1 = \min\{X_1, \ldots, X_n\}$ ($Y_n = \max\{X_1, \ldots, X_n\}$).

Method II: Use Inverse Transform. Recall that

$$F_{Y_1}(y) = P(Y_1 \le y) = 1 - P(Y_1 > y) = 1 - \prod_{i=1}^{n} P(X_i > y) = 1 - (1 - F(y))^n.$$

Let $u = F_{Y_1}(y) = 1 - (1 - F(y))^n$.

Solve for y and get $y = F^{-1}(1 - (1 - u)^{1/n})$.

Thus, take $Y_1 = F^{-1}(1 - U^{1/n})$, where $U \sim U(0, 1)$.

Similarly, $Y_n = F^{-1}(U^{1/n})$. $\square$

Method II: Use Inverse Transform. Recall that

$$F_{Y_1}(y) = P(Y_1 \le y) = 1 - P(Y_1 > y) = 1 - \prod_{i=1}^{n} P(X_i > y) = 1 - (1 - F(y))^n.$$

Let $u = F_{Y_1}(y) = 1 - (1 - F(y))^n$.

Solve for y and get $y = F^{-1}(1 - (1 - u)^{1/n})$.

Thus, take $Y_1 = F^{-1}(1 - U^{1/n})$, where $U \sim U(0,1)$.

Similarly, $Y_n = F^{-1}(U^{1/n})$. □

Method II: Use Inverse Transform. Recall that

$$F_{Y_1}(y) = P(Y_1 \le y) = 1 - P(Y_1 > y) = 1 - \prod_{i=1}^{n} P(X_i > y) = 1 - (1 - F(y))^n.$$

Let $u = F_{Y_1}(y) = 1 - (1 - F(y))^n$.

Solve for y and get $y = F^{-1}(1 - (1-u)^{1/n})$.

Thus, take $Y_1 = F^{-1}(1 - U^{1/n})$, where $U \sim U(0,1)$.

Similarly, $Y_n = F^{-1}(U^{1/n})$. □

Method II: Use Inverse Transform. Recall that

$$F_{Y_1}(y) = P(Y_1 \le y) = 1 - P(Y_1 > y) = 1 - \prod_{i=1}^{n} P(X_i > y) = 1 - (1 - F(y))^n.$$

Let $u = F_{Y_1}(y) = 1 - (1 - F(y))^n$.

Solve for y and get $y = F^{-1}(1 - (1 - u)^{1/n})$.

Thus, take $Y_1 = F^{-1}(1 - U^{1/n})$, where $U \sim U(0, 1)$.

Similarly, $Y_n = F^{-1}(U^{1/n})$. $\square$

Method II: Use Inverse Transform. Recall that

$$F_{Y_1}(y) = P(Y_1 \le y) = 1 - P(Y_1 > y) = 1 - \prod_{i=1}^{n} P(X_i > y) = 1 - (1 - F(y))^n.$$

Let $u = F_{Y_1}(y) = 1 - (1 - F(y))^n$.

Solve for y and get $y = F^{-1}(1 - (1 - u)^{1/n})$.

Thus, take $Y_1 = F^{-1}(1 - U^{1/n})$, where $U \sim U(0, 1)$.

Similarly, $Y_n = F^{-1}(U^{1/n})$. $\square$