Simulation and Modelling

NATIONAL UNIVERSITY
of Computer & Emerging Sciences

Spring 2024
CS4056

Muhammad Shahid Ashraf

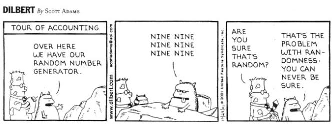
Notes

Random-Number Generation

Notes

Overview

- Discuss characteristics and the generation of random numbers.
- Subsequently, introduce tests for randomness:
  - Frequency test
  - Autocorrelation test



Notes

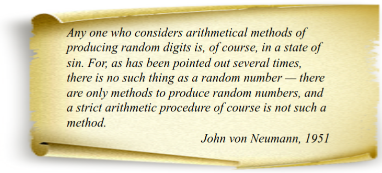
Overview

- Historically
  - Throw dices
  - Deal out cards
  - Draw numbered balls
  - Use digits of π
  - Mechanical devices (spinning disc, etc.)
  - Electric circuits
  - Electronic Random Number Indicator (ERNIE)
  - Counting gamma rays
- In combination with a computer
  - Hook up an electronic device to the computer
  - Read-in a table of random numbers

Notes

Pseudo-Random Numbers

- Approach: Arithmetically generation (calculation) of random numbers
- "Pseudo", because generating numbers using a known method removes the potential for true randomness.



- Goal: To produce a sequence of numbers in [0,1] that simulates, or imitates, the ideal properties of random numbers (RN).

Notes

Pseudo-Random Numbers

- Important properties of good random number routines
  - Fast
  - Portable to different computers
  - Have sufficiently long cycle
  - Replicable
  - Verification and debugging
  - Use identical stream of random numbers for different systems
- Closely approximate the ideal statistical properties of
  - uniformity and
  - independence

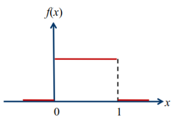
Notes

Pseudo-Random Numbers: Properties

- Two important statistical properties:
  - Uniformity
  - Independence
- Random number R_i must be independently drawn from a uniform distribution with PDF:

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

$$E(R) = \int_0^1 x\,dx = \left.\frac{x^2}{2}\right|_0^1 = \frac{1}{2}$$



PDF for random numbers

Notes

Pseudo-Random Numbers

- Problems when generating pseudo-random numbers
  - The generated numbers might not be uniformly distributed
  - The generated numbers might be discrete-valued instead of continuous-valued
  - The mean of the generated numbers might be too high or too low
  - The variance of the generated numbers might be too high or too low
- There might be dependence:
  - Autocorrelation between numbers
  - Numbers successively higher or lower than adjacent numbers
  - Several numbers above the mean followed by several numbers below the mean

Notes

Generating Random Numbers

- Midsquare method
- Linear Congruential Method (LCM)
- Combined Linear Congruential Generators (CLCG)
- Random-Number Streams

Notes

Midsquare method

- First arithmetic generator: Midsquare method
  - von Neumann and Metropolis in 1940s
- The Midsquare method:
  - Start with a four-digit positive integer Z_0
  - Compute: $Z_0^2 = Z_0 \times Z_0$ to obtain an integer with up to eight digits
  - Take the middle four digits for the next four-digit number

i	Z_i	U_i	$Z_i \times Z_i$
0	7182	-	51581124
1	5811	0.5811	33767721
2	7677	0.7677	58936329
3	9363	0.9363	87665769
…			

Notes

Midsquare method

- Problem: Generated numbers tend to 0

i	Z_i	U_i	$Z_i \times Z_i$
0	7182	-	51581124
1	5811	0,5811	33767721
2	7677	0,7677	58936329
3	9363	0,9363	87665769
4	6657	0,6657	44315649
5	3156	0,3156	09960336
6	9603	0,9603	92217609
7	2176	0,2176	04734976
8	7349	0,7349	54007801
9	78	0,0078	00006084
10	60	0,006	00003600
11	36	0,0036	00001296
12	12	0,0012	00000144
13	1	0,0001	00000001
14	0	0	00000000
15	0	0	00000000

Notes

Linear Congruential Method (LCM)

- To produce a sequence of integers $X_1, X_2, \ldots$ between 0 and m-1 by following a recursive relationship:

$$X_{i+1} = (aX_i + c) \bmod m, \quad i = 0,1,2,\ldots$$

The multiplier
The increment
The modulus

- Assumption: $m > 0$ and $a < m$, $c < m$, $X_0 < m$
- The selection of the values for a, c, m, and X_0 drastically affects the statistical properties and the cycle length
- The random integers X_i are being generated in $[0, m\text{-}1]$

Notes

Linear Congruential Method (LCM)

- Convert the integers X_i to random numbers

$$R_i = \frac{X_i}{m}, \qquad i = 1, 2, \ldots$$

- Note:
- $X_i \in \{0, 1, \ldots, m-1\}$
- $R_i \in [0, \frac{m-1}{m}]$
- Use $X_0 = 27, a = 17, c = 43$, and $m = 100$.
- The X_i and R_i values are:

$X_1 = (17 \times 27 + 43) \bmod 100 = 502 \bmod 100 = 2$ ➡ $R_1 = 0.02$
$X_2 = (17 \times 2 + 43) \bmod 100 = 77$ ➡ $R_2 = 0.77$
$X_3 = (17 \times 77 + 43) \bmod 100 = 52$ ➡ $R_3 = 0.52$
$X_4 = (17 \times 52 + 43) \bmod 100 = 27$ ➡ $R_3 = 0.27$
...

Notes

Linear Congruential Method (LCM)

- Use $a = 13, c = 0$, and $m = 64$
- The period of the generator is very low
- Seed X_0 influences the sequence

i	X_i X_0=1	X_i X_0=2	X_i X_0=3	X_i X_0=4
0	1	2	3	4
1	13	26	39	52
2	41	18	59	36
3	21	42	63	20
4	17	34	51	4
5	29	58	23	
6	57	50	43	
7	37	10	47	
8	33	2	35	
9	45		7	
10	9		27	
11	53		31	
12	49		19	
13	61		55	
14	25		11	
15	5		15	
16	1		3	

Notes

Characteristics of a good Generator

- Maximum Density
  - The values assumed by R_i, i=1,2,... leave no large gaps on [0,1]
  - Problem: Instead of continuous, each R_i is discrete
  - Solution: a very large integer for modulus m
    - Approximation appears to be of little consequence
- Maximum Period
  - To achieve maximum density and avoid cycling
  - Achieved by proper choice of a, c, m, and X_0
- Most digital computers use a binary representation of numbers
  - Speed and efficiency are aided by a modulus, m, to be (or close to) a power of 2.
- The LCG has full period if and only if the following three conditions hold (Hull and Dobell, 1962):
  1. The only positive integer that (exactly) divides both m and c is 1
  2. If q is a prime number that divides m, then q divides a-1
  3. If 4 divides m, then 4 divides a-1

Notes

Proper choice of parameters

- For m a power 2, m=2^b, and $c \neq 0$
  - Longest possible period P=m=2^b is achieved if c is relative prime to m and a=1+4k, where k is an integer
- For m a power 2, m=2^b, and c=0
  - Longest possible period P=m/4=2^{b-2} is achieved if the seed X_0 is odd and a=3+8k or a=5+8k, for k=0,1,...
- For m a prime and c=0
  - Longest possible period P=m-1 is achieved if the multiplier a has property that smallest integer k such that a^k-1 is divisible by m is k = m-1

Notes

General Congruential Generators

- Linear Congruential Generators are a special case of generators defined by:

$$X_{i+1} = g(X_i, X_{i-1}, \ldots) \bmod m$$

- where $g()$ is a function of previous X_i's
  - $X_i \in [0, m\text{-}1]$, $R_i = X_i/m$

- Quadratic congruential generator
  - Defined by: $g(X_i, X_{i-1}) = aX_i^2 + bX_{i-1} + c$
- Multiple recursive generators
  - Defined by: $g(X_i, X_{i-1}, \ldots) = a_1 X_i + a_2 X_{i-1} + \cdots + a_k X_{i-k}$
- Fibonacci generator
  - Defined by: $g(X_i, X_{i-1}) = X_i + X_{i-1}$

Notes

Combined Congruential Generators

- Reason: Longer period generator is needed because of the increasing complexity of simulated systems.
- Approach: Combine two or more multiplicative congruential generators.

- Let $X_{i,1}, X_{i,2}, \ldots, X_{i,k}$ be the i-th output from k different multiplicative congruential generators.
  - The j-th generator $X_{\bullet j}$:

$$X_{i+1,j} = (a_j X_i + c_j) \bmod m_j$$

  - has prime modulus m_j, multiplier a_j, and period m_j-1
  - produces integers $X_{i,j}$ approx ~ Uniform on $[0, m_j - 1]$
  - $W_{i,j} = X_{i,j} - 1$ is approx ~ Uniform on integers on $[0, m_j - 2]$

Notes

Combined Congruential Generators

- Suggested form:

$$X_i = \left(\sum_{j=1}^{k} (-1)^{j-1} X_{i,j} \right) \bmod m_1 - 1 \qquad \text{Hence, } R_i = \begin{cases} \dfrac{X_i}{m_1}, & X_i > 0 \\[2ex] \dfrac{m_1 - 1}{m_1}, & X_i = 0 \end{cases}$$

- The maximum possible period is: $P = \dfrac{(m_1 - 1)(m_2 - 1)\ldots(m_k - 1)}{2^{k-1}}$

Notes

Combined Congruential Generators

- Example: For 32-bit computers, combining $k = 2$ generators with $m_1 = 2147483563$, $a_1 = 40014$, $m_2 = 2147483399$ and $a_2 = 40692$.
  The algorithm becomes:
  Step 1: Select seeds
  $X_{0,1}$ in the range $[1, 2147483562]$ for the 1st generator
  $X_{0,2}$ in the range $[1, 2147483398]$ for the 2nd generator
  Step 2: For each individual generator,
  $X_{i+1,1} = 40014 \times X_{i,1} \bmod 2147483563$
  $X_{i+1,2} = 40692 \times X_{i,2} \bmod 2147483399$
  Step 3: $X_{i+1} = (X_{i+1,1} - X_{i+1,2}) \bmod 2147483562$
  Step 4: Return

$$R_{i+1} = \begin{cases} \dfrac{X_{i+1}}{2147483563}, & X_{i+1} > 0 \\[2ex] \dfrac{2147483562}{2147483563}, & X_{i+1} = 0 \end{cases}$$

  Step 5: Set $i = i+1$, go back to step 2.
  - Combined generator has period: $(m_1 - 1)(m_2 - 1)/2 \sim 2 \text{ x } 10^{18}$

Notes

Combined Congruential Generators

- In Excel 2003 and 2007 new Random Number Generator

$$X, Y, Z \in \{1, ..., 30000\}$$
$$X = X \cdot 171 \bmod 30269$$
$$Y = Y \cdot 172 \bmod 30307$$
$$Z = Z \cdot 170 \bmod 30323$$
$$R = \left(\frac{X}{30269} + \frac{Y}{30307} + \frac{Z}{30323}\right) \bmod 1.0$$

- It is stated that this method produces more than 10^{13} numbers

Notes

Random-Numbers Streams

- The seed for a linear congruential random-number generator:
  - Is the integer value X_0 that initializes the random-number sequence
  - Any value in the sequence $(X_0, X_1, ..., X_P)$ can be used to "seed" the generator
- A random-number stream:
  - Refers to a starting seed taken from the sequence $(X_0, X_1, ..., X_P)$.
  - If the streams are b values apart, then stream i is defined by starting seed:

$$S_i = X_{b(i-1)} \qquad i = 1, 2, \ldots, \left\lfloor \frac{P}{b} \right\rfloor$$

  - Older generators: $b = 10^5$
  - Newer generators: $b = 10^{37}$
- A single random-number generator with k streams can act like k distinct virtual random-number generators
- To compare two or more alternative systems.
  - Advantageous to dedicate portions of the pseudo-random number sequence to the same purpose in each of the simulated systems.

Notes

Tests for Random Numbers

- The seed for a linear congruential random-number generator:
  - Is the integer value X_0 that initializes the random-number sequence
  - Any value in the sequence $(X_0, X_1, ..., X_P)$ can be used to "seed" the generator
- A random-number stream:
  - Refers to a starting seed taken from the sequence $(X_0, X_1, ..., X_P)$.
  - If the streams are b values apart, then stream i is defined by starting seed:

$$S_i = X_{b(i-1)} \qquad i = 1, 2, \ldots, \left\lfloor \frac{P}{b} \right\rfloor$$

  - Older generators: $b = 10^5$
  - Newer generators: $b = 10^{37}$
- A single random-number generator with k streams can act like k distinct virtual random-number generators
- To compare two or more alternative systems.
  - Advantageous to dedicate portions of the pseudo-random number sequence to the same purpose in each of the simulated systems.

Notes

Tests for Random Numbers

- Two categories:
  - Testing for **uniformity**:

$$H_0: \; R_i \sim U[0,1]$$
$$H_1: \; R_i \nsim U[0,1]$$

    - Failure to reject the null hypothesis, H_0, means that evidence of non-uniformity has not been detected.
  - Testing for **independence**:

$$H_0: \; R_i \sim \text{independent}$$
$$H_1: \; R_i \nsim \text{independent}$$

    - Failure to reject the null hypothesis, H_0, means that evidence of dependence has not been detected.
- Level of significance α, the probability of rejecting H_0 when it is true:

$$\alpha = P(\text{reject } H_0 \mid H_0 \text{ is true})$$

Notes

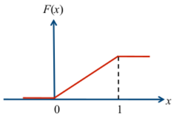
Tests for Random Numbers

- When to use these tests:
  - If a well-known simulation language or random-number generator is used, it is probably unnecessary to test
  - If the generator is not explicitly known or documented, e.g., spreadsheet programs, symbolic/numerical calculators, tests should be applied to many sample numbers.
- Types of tests:
  - Theoretical tests: evaluate the choices of m, a, and c without actually generating any numbers
  - Empirical tests: applied to actual sequences of numbers produced.
    - Our emphasis.

Notes

Frequency tests: Kolmogorov-Smirnov Test

- Compares the continuous CDF, $F(x)$, of the uniform distribution with the empirical CDF, $S_N(x)$, of the N sample observations.
  - We know: $F(x) = x, \quad 0 \le x \le 1$
  - If the sample from the RNG is $R_1, R_2, \ldots, R_N$, then the empirical CDF, $S_N(x)$ is:



$$S_N(x) = \frac{\text{Number of } R_i \text{ where } R_i \le x}{N}$$

- Based on the statistic: $D = max \mid F(x) - S_N(x)\mid$
  - Sampling distribution of D is known

Notes

Frequency tests: Kolmogorov-Smirnov Test

- The test consists of the following steps
  - **Step 1:** Rank the data from smallest to largest $R_{(1)} \le R_{(2)} \le \ldots \le R_{(N)}$
  - **Step 2:** Compute
    $$D^+ = \max_{1 \le i \le N} \left\{ \frac{i}{N} - R_{(i)} \right\}$$
    $$D^- = \max_{1 \le i \le N} \left\{ R_{(i)} - \frac{i-1}{N} \right\}$$
  - **Step 3:** Compute $D = \max(D^+, D^-)$
  - **Step 4:** Get D_α for the significance level α
  - **Step 5:** If $D \le D_\alpha$ accept, otherwise reject H_0

Kolmogorov-Smirnov Critical Values

Degrees of Freedom (N)	$D_{0.10}$	$D_{0.05}$	$D_{0.01}$
1	0.950	0.975	0.995
2	0.776	0.842	0.929
3	0.642	0.708	0.828
4	0.564	0.624	0.733
5	0.510	0.565	0.669
6	0.470	0.521	0.618
7	0.438	0.486	0.577
8	0.411	0.457	0.543
9	0.388	0.432	0.514
10	0.368	0.410	0.490
11	0.352	0.391	0.468
12	0.338	0.375	0.450
13	0.325	0.361	0.433
14	0.314	0.349	0.418
15	0.304	0.338	0.404
16	0.295	0.328	0.392
17	0.286	0.318	0.381
18	0.278	0.309	0.371
19	0.272	0.301	0.363
20	0.264	0.294	0.356
25	0.24	0.27	0.32
30	0.22	0.24	0.29
35	0.21	0.23	0.27
Over 35	$\frac{1.22}{\sqrt{N}}$	$\frac{1.36}{\sqrt{N}}$	$\frac{1.63}{\sqrt{N}}$

Notes

Frequency tests: Kolmogorov-Smirnov Test

- Example: Suppose N=5 numbers: 0.44, 0.81, 0.14, 0.05, 0.93.

	i	1	2	3	4	5
Step 1:	$R_{(i)}$	0.05	0.14	0.44	0.81	0.93
	i/N	0.20	0.40	0.60	0.80	1.00
Step 2:	$i/N - R_{(i)}$	0.15	0.26	0.16	-	0.07
	$R_{(i)} - (i\text{-}1)/N$	0.05	-	0.04	0.21	0.13

Arrange $R_{(i)}$ from smallest to largest

$D^+ = max\{i/N - R_{(i)}\}$

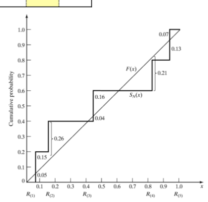
$D^- = max\{R_{(i)} - (i\text{-}1)/N\}$

Step 3: $D = \max(D^+, D^-) = 0.26$

Step 4: For $\alpha = 0.05$,

$D_\alpha = 0.565 > D = 0.26$

Hence, H_0 is not rejected.



Notes

Random-Variate Generation

Notes

Overview

- Develop understanding of generating samples from a specified distribution as input to a simulation model.
- Illustrate some widely-used techniques for generating random variates:
  - Inverse-transform technique
  - Acceptance-rejection technique
  - Special properties

Notes

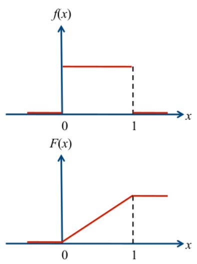
Preparation

- It is assumed that a source of uniform [0,1] random numbers exists. Linear Congruential Method (LCM)
- Random numbers $R, R_1, R_2, \ldots$ with
  - PDF

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

  - CDF

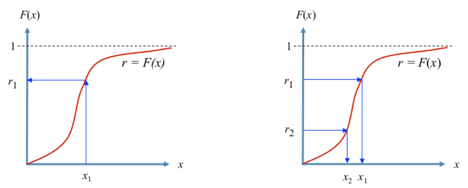
$$F_R(x) = \begin{cases} 0 & x < 0 \\ x & 0 \le x \le 1 \\ 1 & x > 1 \end{cases}$$



Notes

Inverse-transform Technique

- The concept:
  - For CDF function: $r = F(x)$
  - Generate r from uniform $(0,1)$, a.k.a $U(0,1)$
  - Find x, $x = F^{-1}(r)$



Notes

Inverse-transform Technique

- The inverse-transform technique can be used in principle for any distribution.
- Most useful when the CDF F(x) has an inverse $F^{-1}(x)$ which is easy to compute.
- Required steps
  - Compute the CDF of the desired random variable X.
  - Set F(X) = R on the range of X.
  - Solve the equation F(X) = R for X in terms of R.
  - Generate uniform random numbers $R_1, R_2, R_3, \ldots$ and compute the desired random variate by $X_i = F^{-1}(R_i)$

Notes

Inverse-transform Technique

- Exponential Distribution
- PDF

$$f(x) = \lambda exp(-\lambda x)$$

- CDF

$$F(x) = 1 - exp(-\lambda x)$$

- Simplification

$$X = -\frac{\ln R}{\lambda}$$

- Since R and (1-R) are uniformly distributed on $[0, 1]$

- To generate $X_1, X_2, X_3 \ldots$

$$1 - exp(-\lambda X) = R$$

$$exp(-\lambda X) = 1 - R$$

$$-\lambda X = \ln(1 - R)$$

$$X = -\frac{1}{\lambda}\ln(1 - R)$$

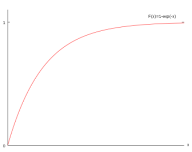
$$X = F^{-1}(R)$$

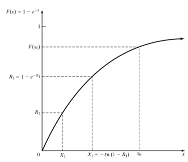
Notes

Inverse-transform Technique

Generation of Exponential Variates X_i with Mean 1, given Random Numbers R_i

i	1	2	3	4	5
R_i	0.1306	0.0422	0.6597	0.7965	0.7696
X_i	0.1400	0.0431	1.078	1.592	1.468



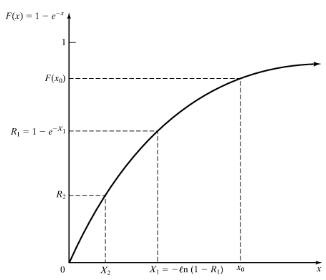


Notes

Inverse-transform Technique

Check: Does the random variable X_1 have the desired distribution?

$$P(X_1 \le x_0) = P(R_1 \le F(x_0)) = F(x_0)$$



Notes

Inverse-transform Technique

- Examples of other distributions for which inverse CDF works are:
  - Uniform distribution
  - Weibull distribution
  - Triangular distribution
- Random variable X uniformly distributed over $[a, b]$

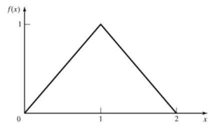
$$
\begin{aligned}
F(X) &= R \\
\frac{X-a}{b-a} &= R \\
X-a &= R(b-a) \\
X &= R(b-a)+a
\end{aligned}
$$

Notes

Inverse-transform Technique

NATIONAL UNIVERSITY of Computer & Emerging Sciences

The CDF of a Triangular Distribution with endpoints (0, 2) is given by



$$F(x)=\begin{cases}0 & x\le 0\\ \frac{x^2}{2} & 0<x\le 1\\ 1-\frac{(2-x)^2}{2} & 1<x\le 2\\ 1 & x>2\end{cases}$$

$$R(X)=\begin{cases}\frac{X^2}{2} & 0\le X\le 1\\ 1-\frac{(2-X)^2}{2} & 1\le X\le 2\end{cases}$$

X is generated by

$$X=\begin{cases}\sqrt{2R} & 0\le R\le \frac{1}{2}\\ 2-\sqrt{2(1-R)} & \frac{1}{2}<R\le 1\end{cases}$$

Notes

Inverse-transform Technique

NATIONAL UNIVERSITY of Computer & Emerging Sciences

The variate is

- The Weibull Distribution is described by
  - PDF

$$f(x)=\frac{\beta}{\alpha^{\beta}}x^{\beta-1}e^{-\left(\frac{x}{\alpha}\right)^{\beta}}$$

  - CDF

$$F(X)=1-e^{-\left(\frac{x}{\alpha}\right)^{\beta}}$$

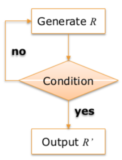
Notes

Acceptance-Rejection Technique

NATIONAL UNIVERSITY of Computer & Emerging Sciences

- Useful particularly when inverse CDF does not exist in closed form
  - Thinning
- Illustration: To generate random variates, $X \sim U(1/4, 1)$

> Procedure:
> Step 1. Generate $R \sim U(0,1)$
> Step 2. If $R \ge 1/4$, accept $X=R$.
> Step 3. If $R < 1/4$, reject R, return to Step 1



- R does not have the desired distribution, but R conditioned (R') on the event $\{R \ge 1/4\}$ does.
- Efficiency: Depends heavily on the ability to minimize the number of rejections.

Notes

Acceptance-Rejection Technique

- Probability mass function of a Poisson Distribution

$$P(N = n) = \frac{\alpha^n}{n!} e^{-\alpha}$$

- Exactly n arrivals during one time unit

$$A_1 + A_2 + \cdots + A_n \leq 1 < A_1 + A_2 + \cdots + A_n + A_{n+1}$$

- Since interarrival times are exponentially distributed we can set

$$A_i = \frac{-\ln(R_i)}{\alpha}$$

  - Well known, we derived this generator in the beginning of the class

- Procedure of generating a Poisson random variate N is as follows
  1. Set n=0, P=1
  2. Generate a random number R_{n+1}, and replace P by P x R_{n+1}
  3. If $P < \exp(-\alpha)$, then accept N=n
     - Otherwise, reject the current n, increase n by one, and return to step 2.

Notes

Acceptance-Rejection Technique

- Example: Generate three Poisson variates with mean α=0.2
  - $\exp(-0.2) = 0.8187$
- Variate 1
  - Step 1: Set $n = 0$, $P = 1$
  - Step 2: $R1 = 0.4357$, $P = 1 \times 0.4357$
  - Step 3: Since $P = 0.4357 < \exp(-0.2)$, accept $N = 0$
- Variate 2
  - Step 1: Set $n = 0$, $P = 1$
  - Step 2: $R1 = 0.4146$, $P = 1 \times 0.4146$
  - Step 3: Since $P = 0.4146 < \exp(-0.2)$, accept $N = 0$
- Variate 3
  - Step 1: Set $n = 0$, $P = 1$
  - Step 2: $R1 = 0.8353$, $P = 1 \times 0.8353$
  - Step 3: Since $P = 0.8353 > \exp(-0.2)$, reject $n = 0$ and return to Step 2 with $n = 1$
  - Step 2: $R2 = 0.9952$, $P = 0.8353 \times 0.9952 = 0.8313$
  - Step 3: Since $P = 0.8313 > \exp(-0.2)$, reject $n = 1$ and return to Step 2 with $n = 2$
  - Step 2: $R3 = 0.8004$, $P = 0.8313 \times 0.8004 = 0.6654$
  - Step 3: Since $P = 0.6654 < \exp(-0.2)$, accept $N = 2$

Notes

Acceptance-Rejection Technique

- It took five random numbers to generate three Poisson variates
- In long run, the generation of Poisson variates requires some overhead!

N	R_{n+1}	P	Accept/Reject	Result
0	0.4357	0.4357	P < exp(- α)	Accept	N=0
0	0.4146	0.4146	P < exp(- α)	Accept	N=0
0	0.8353	0.8353	P ≥ exp(- α)	Reject	
1	0.9952	0.8313	P ≥ exp(- α)	Reject	
2	0.8004	0.6654	P < exp(- α)	Accept	N=2

Notes

Notes