Chapter 6

Random-Number Generation
Contents

- Properties of Random Numbers
- Pseudo-Random Numbers
- Generating Random Numbers
  - Linear Congruential Method
  - Combined Linear Congruential Method
- Tests for Random Numbers
- Real Random Numbers
Overview

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

- Historically
  - Throw dices
  - Deal out cards
  - Draw numbered balls
  - Use digits of $\pi$
  - 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
Pseudo-Random Numbers
Pseudo-Random Numbers

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

---

*Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin. For, as has been pointed out several times, there is no such thing as a random number — there are only methods to produce random numbers, and a strict arithmetic procedure of course is not such a method.*

*John von Neumann, 1951*
Pseudo-Random Numbers

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

... probably ... can not be justified, but should merely be judged by their results. Some statistical study of the digits generated by a given recipe should be made, but exhaustive tests are impractical. If the digits work well on one problem, they seem usually to be successful with others of the same type.

*John von Neumann, 1951*
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
Pseudo-Random Numbers: Properties

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

\[
f(x) = \begin{cases} 
1, & 0 \leq x \leq 1 \\
0, & \text{otherwise} 
\end{cases}
\]

\[
E(R) = \int_{0}^{1} x \, dx = \frac{x^2}{2} \bigg|_{0}^{1} = \frac{1}{2}
\]
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
Generating Random Numbers
Generating Random Numbers

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

<table>
<thead>
<tr>
<th>$i$</th>
<th>$Z_i$</th>
<th>$U_i$</th>
<th>$Z_i \times Z_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7182</td>
<td>-</td>
<td>51581124</td>
</tr>
<tr>
<td>1</td>
<td>5811</td>
<td>0.5811</td>
<td>33767721</td>
</tr>
<tr>
<td>2</td>
<td>7677</td>
<td>0.7677</td>
<td>58936329</td>
</tr>
<tr>
<td>3</td>
<td>9363</td>
<td>0.9363</td>
<td>87665769</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Midsquare method

• Problem: Generated numbers tend to 0

<table>
<thead>
<tr>
<th>$i$</th>
<th>$Z_i$</th>
<th>$U_i$</th>
<th>$Z_i \times Z_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7182</td>
<td>-</td>
<td>51581124</td>
</tr>
<tr>
<td>1</td>
<td>5811</td>
<td>0,5811</td>
<td>33767721</td>
</tr>
<tr>
<td>2</td>
<td>7677</td>
<td>0,7677</td>
<td>58936329</td>
</tr>
<tr>
<td>3</td>
<td>9363</td>
<td>0,9363</td>
<td>87665769</td>
</tr>
<tr>
<td>4</td>
<td>6657</td>
<td>0,6657</td>
<td>44315649</td>
</tr>
<tr>
<td>5</td>
<td>3156</td>
<td>0,3156</td>
<td>09960336</td>
</tr>
<tr>
<td>6</td>
<td>9603</td>
<td>0,9603</td>
<td>92217609</td>
</tr>
<tr>
<td>7</td>
<td>2176</td>
<td>0,2176</td>
<td>04734976</td>
</tr>
<tr>
<td>8</td>
<td>7349</td>
<td>0,7349</td>
<td>54007801</td>
</tr>
<tr>
<td>9</td>
<td>78</td>
<td>0,0078</td>
<td>00006084</td>
</tr>
<tr>
<td>10</td>
<td>60</td>
<td>0,006</td>
<td>00003600</td>
</tr>
<tr>
<td>11</td>
<td>36</td>
<td>0,0036</td>
<td>00001296</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>0,0012</td>
<td>00000144</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>0,0001</td>
<td>00000001</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>00000000</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>00000000</td>
</tr>
</tbody>
</table>
... random numbers should not be generated with a method chosen at random. Some theory should be used.

Donald E. Knuth, The Art of Computer Programming, Vol. 2
Generating Random Numbers
Linear Congruential Method
Linear Congruential Method

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

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

- 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 $[0, m-1]$
Linear Congruential Method

- Convert the integers $X_i$ to random numbers

\[ R_i = \frac{X_i}{m}, \quad i = 1, 2, \ldots \]

- Note:
  - $X_i \in \{0, 1, \ldots, m-1\}$
  - $R_i \in [0, (m-1)/m]$
Linear Congruential Method: Example

- Use $X_0 = 27$, $a = 17$, $c = 43$, and $m = 100$.
- The $X_i$ and $R_i$ values are:

\[
\begin{align*}
X_1 &= (17 \times 27 + 43) \mod 100 = 502 \mod 100 = 2 \\
R_1 &= 0.02 \\
X_2 &= (17 \times 2 + 43) \mod 100 = 77 \\
R_2 &= 0.77 \\
X_3 &= (17 \times 77 + 43) \mod 100 = 52 \\
R_3 &= 0.52 \\
X_4 &= (17 \times 52 + 43) \mod 100 = 27 \\
R_4 &= 0.27
\end{align*}
\]
Linear Congruential Method: Example

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

<table>
<thead>
<tr>
<th>$i$</th>
<th>$X_i$</th>
<th>$X_i$</th>
<th>$X_i$</th>
<th>$X_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$X_0=1$</td>
<td>$X_0=2$</td>
<td>$X_0=3$</td>
<td>$X_0=4$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>26</td>
<td>39</td>
<td>52</td>
</tr>
<tr>
<td>2</td>
<td>41</td>
<td>18</td>
<td>59</td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>42</td>
<td>63</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>34</td>
<td>51</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>29</td>
<td>58</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>57</td>
<td>50</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>37</td>
<td>10</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>33</td>
<td>2</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>45</td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td></td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>53</td>
<td></td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>49</td>
<td></td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>61</td>
<td></td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>25</td>
<td></td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Linear Congruential Method: 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.
Linear Congruential Method:
Characteristics of a good Generator

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 $a-1$
Linear Congruential Method:
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 $k=m-1$
Characteristics of a Good Generator
Characteristics of a Good Generator
Random-Numbers in Java

- Defined in java.util.Random

```java
private final static long multiplier = 0x5DEECE66DL; // 25214903917
private final static long addend = 0xBL; // 11
private final static long mask = (1L << 48) - 1; // 2^48-1 = 281474976710655

protected int next(int bits) {
    long oldseed, nextseed;
    ...
    oldseed = seed.get();
    nextseed = (oldseed * multiplier + addend) & mask;
    ...
    return (int)(nextseed >>> (48 - bits)); // >>> Unsigned right shift
}
```
General Congruential Generators

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

\[ X_{i+1} = g(X_i, X_{i-1}, \ldots) \mod m \]

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

• Quadratic congruential generator
  • Defined by: \( g(X_i, X_{i-1}) = aX_i^2 + bX_{i-1} + c \)

• Multiple recursive generators
  • Defined by: \( g(X_i, X_{i-1}, \ldots) = a_1X_i + a_2X_{i-1} + \cdots + a_kX_{i-k} \)

• Fibonacci generator
  • Defined by: \( g(X_i, X_{i-1}) = X_i + X_{i-1} \)
Combined Linear 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 \( X_{i,j} \):

\[
X_{i+1,j} = (a_j X_i + c_j) \mod m_j
\]

- has prime modulus \( m_j \), multiplier \( a_j \), and period \( m_j - 1 \)
- produces integers \( X_{i,j} \) approx \( \sim \) Uniform on \([0, m_j - 1]\)
- \( W_{i,j} = X_{i,j} - 1 \) is approx \( \sim \) Uniform on integers on \([0, m_j - 2]\)
Combined Linear Congruential Generators

- Suggested form:

\[ X_i = \left( \sum_{j=1}^{k} (-1)^{j-1} X_{i,j} \right) \mod m_1 - 1 \]

Hence, \( R_i = \begin{cases} \frac{X_i}{m_1}, & X_i > 0 \\ \frac{m_1 - 1}{m_1}, & X_i = 0 \end{cases} \)

- The maximum possible period is:

\[ P = \frac{(m_1 - 1)(m_2 - 1) \ldots (m_k - 1)}{2^{k-1}} \]
Combined Linear 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,
- $X_{i+1,1} = 40014 \times X_{i,1} \mod 2147483563$
- $X_{i+1,2} = 40692 \times X_{i,2} \mod 2147483399$

Step 3: $X_{i+1} = (X_{i+1,1} - X_{i+1,2}) \mod 2147483562$

Step 4: Return

$$R_{t+1} = \begin{cases} \frac{X_{i+1}}{2147483563}, & X_{i+1} > 0 \\ \frac{2147483562}{2147483563}, & X_{i+1} = 0 \end{cases}$$

Step 5: Set $i = i+1$, go back to step 2.

- Combined generator has period: $(m_1 - 1)(m_2 - 1)/2 \sim 2 \times 10^{18}$
Random-Numbers in Excel 2003

• In Excel 2003 and 2007 new Random Number Generator

\[
\begin{align*}
X, Y, Z & \in \{1, \ldots, 30000\} \\
X &= X \cdot 171 \mod 30269 \\
Y &= Y \cdot 172 \mod 30307 \\
Z &= Z \cdot 170 \mod 30323 \\
R &= \left( \frac{X}{30269} + \frac{Y}{30307} + \frac{Z}{30323} \right) \mod 1.0
\end{align*}
\]

• It is stated that this method produces more than \(10^{13}\) numbers

• For more info: [http://support.microsoft.com/kb/828795](http://support.microsoft.com/kb/828795)
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 $(X_0, X_1, \ldots, X_P)$ can be used to “seed” the generator

- A random-number stream:
  - Refers to a starting seed taken from the sequence $(X_0, X_1, \ldots, X_P)$.
  - If the streams are $b$ values apart, then stream $i$ is defined by starting seed:
    \[ S_i = X_{b(i-1)} \quad i = 1, 2, \ldots, \left\lfloor \frac{P}{b} \right\rfloor \]
  - 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.
Random Numbers in OMNeT++

- OMNeT++ releases prior to 3.0 used a linear congruential generator (LCG) with a cycle length of $2^{31}-2$.
- By default, OMNeT++ uses the Mersenne Twister RNG (MT) by M. Matsumoto and T. Nishimura.
- MT has a period of $2^{19937}-1$, and 623-dimensional equidistribution property is assured.
- This RNG can be selected from omnetpp.ini
- OMNeT++ allows plugging in your own RNGs as well. This mechanism, based on the cRNG interface.
Tests for Random Numbers
Tests for Random Numbers

- Two categories:
  - Testing for **uniformity**:
    
    
    \[ H_0: \ R_i \sim U[0,1] \]
    
    \[ H_1: \ R_i \sim U[0,1] \]
    
    - Failure to reject the null hypothesis, \( H_0 \), means that evidence of non-uniformity has not been detected.
  
  - Testing for **independence**:
    
    \[ H_0: \ R_i \sim \text{independent} \]
    
    \[ H_1: \ R_i \sim \text{independent} \]
    
    - Failure to reject the null hypothesis, \( H_0 \), means that evidence of dependence has not been detected.

- Level of significance \( \alpha \), the probability of rejecting \( H_0 \) when it is true:

\[ \alpha = P(\text{reject } H_0 \mid H_0 \text{ is true}) \]
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.
Tests for Random Numbers

Frequency tests: Kolmogorov-Smirnov Test
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: \( F(x) = x, \quad 0 \leq x \leq 1 \)

- If the sample from the RNG is \( R_1, R_2, ..., R_N \), then the empirical CDF, \( S_N(x) \) is:

\[
S_N(x) = \frac{\text{Number of } R_i \text{ where } R_i \leq x}{N}
\]

- Based on the statistic: \( D = \max | F(x) - S_N(x) | \)
- Sampling distribution of \( D \) is known
Kolmogorov-Smirnov Test

- The test consists of the following steps
  - **Step 1:** Rank the data from smallest to largest
    \[ R_1 \leq R_2 \leq \ldots \leq R_N \]
  - **Step 2:** Compute
    \[
    D^+ = \max_{1 \leq i \leq N} \left\{ \frac{i}{N} - R_i \right\}
    \]
    \[
    D^- = \max_{1 \leq i \leq N} \left\{ R_i - \frac{i-1}{N} \right\}
    \]
  - **Step 3:** Compute \( D = \max(D^+, D^-) \)
  - **Step 4:** Get \( D_\alpha \) for the significance level \( \alpha \)
  - **Step 5:** If \( D \leq D_\alpha \) accept, otherwise reject \( H_0 \)

---

### Kolmogorov-Smirnov Critical Values

<table>
<thead>
<tr>
<th>Degrees of Freedom</th>
<th>( D_{0.10} )</th>
<th>( D_{0.05} )</th>
<th>( D_{0.01} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.950</td>
<td>0.975</td>
<td>0.995</td>
</tr>
<tr>
<td>2</td>
<td>0.776</td>
<td>0.842</td>
<td>0.892</td>
</tr>
<tr>
<td>3</td>
<td>0.642</td>
<td>0.708</td>
<td>0.782</td>
</tr>
<tr>
<td>4</td>
<td>0.564</td>
<td>0.624</td>
<td>0.733</td>
</tr>
<tr>
<td>5</td>
<td>0.510</td>
<td>0.565</td>
<td>0.669</td>
</tr>
<tr>
<td>6</td>
<td>0.470</td>
<td>0.521</td>
<td>0.618</td>
</tr>
<tr>
<td>7</td>
<td>0.438</td>
<td>0.486</td>
<td>0.577</td>
</tr>
<tr>
<td>8</td>
<td>0.411</td>
<td>0.457</td>
<td>0.543</td>
</tr>
<tr>
<td>9</td>
<td>0.388</td>
<td>0.432</td>
<td>0.514</td>
</tr>
<tr>
<td>10</td>
<td>0.368</td>
<td>0.410</td>
<td>0.490</td>
</tr>
<tr>
<td>11</td>
<td>0.352</td>
<td>0.391</td>
<td>0.468</td>
</tr>
<tr>
<td>12</td>
<td>0.338</td>
<td>0.375</td>
<td>0.450</td>
</tr>
<tr>
<td>13</td>
<td>0.325</td>
<td>0.361</td>
<td>0.433</td>
</tr>
<tr>
<td>14</td>
<td>0.314</td>
<td>0.349</td>
<td>0.418</td>
</tr>
<tr>
<td>15</td>
<td>0.304</td>
<td>0.338</td>
<td>0.404</td>
</tr>
<tr>
<td>16</td>
<td>0.295</td>
<td>0.328</td>
<td>0.392</td>
</tr>
<tr>
<td>17</td>
<td>0.286</td>
<td>0.318</td>
<td>0.381</td>
</tr>
<tr>
<td>18</td>
<td>0.278</td>
<td>0.309</td>
<td>0.371</td>
</tr>
<tr>
<td>19</td>
<td>0.272</td>
<td>0.301</td>
<td>0.363</td>
</tr>
<tr>
<td>20</td>
<td>0.264</td>
<td>0.294</td>
<td>0.356</td>
</tr>
<tr>
<td>25</td>
<td>0.24</td>
<td>0.27</td>
<td>0.32</td>
</tr>
<tr>
<td>30</td>
<td>0.22</td>
<td>0.24</td>
<td>0.29</td>
</tr>
<tr>
<td>35</td>
<td>0.21</td>
<td>0.23</td>
<td>0.27</td>
</tr>
</tbody>
</table>

Over \( \frac{D}{\sqrt{N}} \) for \( N = 35 \):

- \( 1.22 \)
- \( 1.36 \)
- \( 1.63 \)
Kolmogorov-Smirnov Test

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

<table>
<thead>
<tr>
<th>( i )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R(i) )</td>
<td>0.05</td>
<td>0.14</td>
<td>0.44</td>
<td>0.81</td>
<td>0.93</td>
</tr>
<tr>
<td>( i/N )</td>
<td>0.20</td>
<td>0.40</td>
<td>0.60</td>
<td>0.80</td>
<td>1.00</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{Step 1:} & \quad \text{Arrange } R(i) \text{ from smallest to largest} \\
\text{Step 2:} & \quad D^+ = \max\{i/N - R(i)\} \\
& \quad D^- = \max\{R(i) - (i-1)/N\}
\end{align*}
\]

\[
\begin{align*}
\text{Step 3:} & \quad D = \max(D^+, D^-) = 0.26 \\
\text{Step 4:} & \quad \text{For } \alpha = 0.05, \\
& \quad D_\alpha = 0.565 > D = 0.26
\end{align*}
\]

Hence, \( H_0 \) is not rejected.
Tests for Random Numbers

Frequency tests: Chi-square Test
Chi-square Test

- Chi-square test uses the sample statistic:

\[ \chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \]

- Approximately the chi-square distribution with \( n-1 \) degrees of freedom
- For the uniform distribution, \( E_i \), the expected number in each class is:

\[ E_i = \frac{N}{n}, \quad \text{where } N \text{ is the total number of observations} \]
- Valid only for large samples, e.g., \( N \geq 50 \)
Chi-square Test: Example

• Example with 100 numbers from [0,1], \( \alpha = 0.05 \)
• 10 intervals
  • \( \chi^2_{0.05,9} = 16.9 \)
• Accept, since
  • \( X^2_0 = 11.2 < \chi^2_{0.05,9} \)

\[
\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}
\]

<table>
<thead>
<tr>
<th>Interval</th>
<th>Upper Limit</th>
<th>( O_i )</th>
<th>( E_i )</th>
<th>( O_i - E_i )</th>
<th>( (O_i - E_i)^2 )</th>
<th>( (O_i - E_i)^2/E_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
<td>9</td>
<td>10</td>
<td>-1</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
<td>5</td>
<td>10</td>
<td>-5</td>
<td>25</td>
<td>2.5</td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
<td>6</td>
<td>10</td>
<td>-4</td>
<td>16</td>
<td>1.6</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>16</td>
<td>10</td>
<td>6</td>
<td>36</td>
<td>3.6</td>
</tr>
<tr>
<td>6</td>
<td>0.6</td>
<td>13</td>
<td>10</td>
<td>3</td>
<td>9</td>
<td>0.9</td>
</tr>
<tr>
<td>7</td>
<td>0.7</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0.8</td>
<td>7</td>
<td>10</td>
<td>-3</td>
<td>9</td>
<td>0.9</td>
</tr>
<tr>
<td>9</td>
<td>0.9</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1.0</td>
<td>14</td>
<td>10</td>
<td>4</td>
<td>16</td>
<td>1.6</td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td>100</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>11.2</td>
</tr>
</tbody>
</table>

\( X^2_0 = 11.2 \)
Tests for Random Numbers
Tests for autocorrelation
Tests for Autocorrelation

- Autocorrelation is concerned with dependence between numbers in a sequence
- Example:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.12</td>
<td>0.01</td>
<td>0.23</td>
<td>0.28</td>
<td>0.89</td>
<td>0.31</td>
<td>0.64</td>
<td>0.28</td>
<td>0.83</td>
<td>0.93</td>
</tr>
<tr>
<td>0.99</td>
<td>0.15</td>
<td>0.33</td>
<td>0.35</td>
<td>0.91</td>
<td>0.41</td>
<td>0.60</td>
<td>0.27</td>
<td>0.75</td>
<td>0.88</td>
</tr>
<tr>
<td>0.68</td>
<td>0.49</td>
<td>0.05</td>
<td>0.43</td>
<td>0.95</td>
<td>0.58</td>
<td>0.19</td>
<td>0.36</td>
<td>0.69</td>
<td>0.87</td>
</tr>
</tbody>
</table>

- Numbers at 5-th, 10-th, 15-th, ... are very similar
- Numbers can be
  - Low
  - High
  - Alternating
Tests for Autocorrelation

- Testing the autocorrelation between every $m$ numbers ($m$ is a.k.a. the lag), starting with the $i$-th number
  - The autocorrelation $\rho_{i,m}$ between numbers: $R_i, R_{i+m}, R_{i+2m}, R_{i+(M+1)m}$
  - $M$ is the largest integer such that $i + (M + 1)m \leq N$

- Hypothesis:
  
  \[
  H_0 : \quad \rho_{i,m} = 0, \quad \text{if numbers are independent}
  \]
  
  \[
  H_1 : \quad \rho_{i,m} \neq 0, \quad \text{if numbers are dependent}
  \]

- If the values are uncorrelated:
  
  - For large values of $M$, the distribution of the estimator of $\rho_{i,m}$, denoted $\hat{\rho}_{i,m}$ is approximately normal.
Tests for Autocorrelation

- Correlation at lag \( j \)

\[
\rho_j = \frac{C_j}{C_0}
\]

\[
C_j = \text{Cov}(X_i, X_{i+j}) = E(X_i X_{i+j}) - E(X_i)E(X_{i+j})
\]

\[
C_0 = \text{Cov}(X_i, X_i) = E(X_i X_i) - E(X_i)E(X_i) = E(X_i^2) - [E(X_i)]^2 = \text{Var}(X_i)
\]

\[
\Rightarrow \rho_j = \frac{E(X_i X_{i+j}) - E(X_i)E(X_{i+j})}{\text{Var}(X_i)}
\]

- Assume \( X_i = U_i \)

\[
E(U_i) = \frac{1}{2} \quad \text{and} \quad \text{Var}(U_i) = \frac{1}{12}
\]

\[
\rho_j = \frac{E(U_i U_{i+j}) - \frac{1}{4}}{\frac{1}{12}} = 12E(U_i U_{i+j}) - 3
\]
Tests for Autocorrelation

• Test statistics is:
  \[ Z_0 = \frac{\hat{\rho}_{i,m}}{\hat{\sigma}_{\rho_{i,m}}} \]
  - \( Z_0 \) is distributed normally with mean = 0 and variance = 1, and:

  \[
  \hat{\rho}_{i,m} = \frac{1}{M + 1} \left[ \sum_{k=0}^{M} R_{i+km} \times R_{i+(k+1)m} \right] - 0.25
  \]

  \[
  \hat{\sigma}_{\rho_{i,m}} = \frac{\sqrt{13M + 7}}{12(M + 1)}
  \]

• After computing \( Z_0 \) do not reject the hypothesis of independence if \( -z_{\alpha/2} \leq Z_0 \leq -z_{\alpha/2} \)
Tests for Autocorrelation

- If $\rho_{i,m} > 0$, the subsequence has positive autocorrelation
  - High random numbers tend to be followed by high ones, and vice versa.

- If $\rho_{i,m} < 0$, the subsequence has negative autocorrelation
  - Low random numbers tend to be followed by high ones, and vice versa.
Example

• Test whether the 3\textsuperscript{rd}, 8\textsuperscript{th}, 13\textsuperscript{th}, and so on, for the numbers on Slide 38.
  • Hence, $\alpha = 0.05$, $i = 3$, $m = 5$, $N = 30$, and $M = 4$

\[
\hat{\rho}_{35} = \frac{1}{4+1} \left[ (0.23)(0.28) + (0.28)(0.33) + (0.33)(0.27) + (0.27)(0.05) + (0.05)(0.36) \right] - 0.25 = -0.1945
\]

\[
\sigma_{\hat{\rho}_{35}} = \frac{\sqrt{13(4) + 7}}{12(4+1)} = 0.128
\]

\[
Z_0 = -\frac{0.1945}{0.1280} = -1.516
\]

• $z_{0.025} = 1.96$

• Since $-1.96 \leq Z_0 = -1.516 \leq 1.96$, the hypothesis is not rejected.
Shortcomings

- The test is not very sensitive for small values of $M$, particularly when the numbers being tested are on the low side.
- Problem when “fishing” for autocorrelation by performing numerous tests:
  - If $\alpha = 0.05$, there is a probability of 0.05 of rejecting a true hypothesis.
  - If 10 independence sequences are examined:
    - The probability of finding no significant autocorrelation, by chance alone, is $0.95^{10} = 0.60$.
    - Hence, the probability of detecting significant autocorrelation when it does not exist = 40%
Real Random Numbers
Real Random Numbers

- There are also sources for real random numbers in the Internet
- www.random.org
  "RANDOM.ORG offers true random numbers to anyone on the Internet. The randomness comes from atmospheric noise, which for many purposes is better than the pseudo-random number algorithms typically used in computer programs. People use the numbers to run lotteries, draws and sweepstakes and for their games and gambling sites."

http://www.random.org/analysis/
Real Random Numbers

• http://www.randomnumbers.info/
  „It offers the possibility to download true random numbers generated using a quantum random number generator upon demand. “
Real Random Numbers

- Hardware based generation of random numbers
- http://www.comscire.com
Summary

- In this chapter, we described:
  - Generation of random numbers
  - Testing for uniformity and independence
  - Sources of real random numbers

- Caution:
  - Even with generators that have been used for years, some of which still in use, are found to be inadequate.
  - This chapter provides only the basics
  - Also, even if generated numbers pass all the tests, some underlying pattern might have gone undetected.