STA 5106 Computational Methods in Statistics I

Department of Statistics

Florida State University

Class 16

October 29, 2019

Chapter 5
Simulating Probability Distributions

5.1 Introduction

Simulation

- Simulation of random variables using computers is among the fastest growing areas of computational statistics.
- Many statistical methods rely on simulating random variables.
- In many practical situations the probability distributions are far too complicated to analyze and often it is easier to simulate these distributions on computers and the resulting samples can be analyzed instead.
- In general, there are two steps to simulate a given distribution:
    1. Generate samples from a standard uniform probability, e.g. in [0, 1],
    2. Transform those samples appropriately to simulate the desired distribution.

5.2 Simulating Uniform Random Variable

Pseudorandom Numbers

- How can one simulate a uniform random variable?
- A basic idea is to use algebraic methods to generate sequences of numbers that mimic the behavior of a uniform random variable. These numbers are called **pseudorandom numbers**.
- **Definition 12** A **uniform pseudorandom number** generator is a mapping f that, starting from an initial value x_0, generates a sequence

$$x_0, f(x_0), f(f(x_0)), f(f(f(x_0))), \ldots .$$

- Since f is computed on a computer, it is a deterministic mapping. That is, given x_0 the remaining sequence is fixed every time the sequence is computed.

Properties

- The sequence should have the following properties:

1. The patterns between the numbers that appearing in a sequence should be minimized.

2. The correlation between the neighboring elements should be reasonably small.

3. The values should be distributed nearly uniformly over the whole range of possible values.

4. The sequences should have large periods, where a period is later defined to be duration after which a sequence repeats itself.

Properties

5. There exist a set of goodness-of-fit tests for testing the probability distributions associated with the observed random variables. The elements of a pseudorandom sequence should provide a reasonable performance in these goodness-of-fit tests.

- We study the following three techniques to generate pesudorandom sequences on digital computers:

1. Congruential Generators

2. Shift-Register Generators

3. Lagged Fibonacci Generators

Congruential Generators

- A common idea in pesudorandom methods utilizes modular arithmetic to formulate the congruential generators.

- First we introduce the notation for modular arithmetic. For a positive integer m and a given number b, ***a* is called the residue of *b* modulus *m*** if

$$a = b - [b/m]m,$$

  where $[\cdot]$ denotes the largest integer less than the argument.

- For example,

  if $m = 5$ and $b = 13$, then $a = 3$.

  if $m = 9$ and $b = 26$, then $a = 8$.

  if $m = 4$ and $b = 10$, then $a = 2$.

Congruential Generators

- We will use the notation that a is b mod m.
- It can be checked that a large number of numbers have the same residue.
- In fact, if we define two numbers to be equivalent if they have the same residue, then this relationship establishes an equivalence class on the space of numbers.
- We will restrict ourselves to non-negative integers, and the equivalence class for this set can be denoted by

$$\{0, 1, 2, 3, \ldots, m-1\} \; .$$

Congruential Generators

- Modular arithmetic can be used to generate sequences of pseudorandom numbers in the following way.

- Define the next element of the sequence is given by

$$x_{i+1} = (ax_i + c) \bmod m,$$

  where *a* is called the multiplier, *m* is the base and *c* is the additive constant.

- The generators based on this general formula are called **linear congruential generators**.

- If $c = 0$, then the resulting generator is called a **multiplicative generator**.

Congruential Generators

- The choice of a and m determines the properties of the resulting sequence.

- We start with an initial number, $x_0 \in \{1, 2, \ldots, m-1\}$, called the **seed** of the number generator.

- **The remaining sequence is determined by the relation**

$$x_{i+1} = (ax_i) \bmod m,$$

Then, the sequence in [0, 1] is x_i/m, $i = 1, 2, \ldots$

- This relation is completely deterministic, i.e. if a number repeats in the sequence then the following sequence also repeats.

Congruential Generators

- **Definition 13** The period of a sequence is defined as the number of elements in the sequence before the sequence repeats itself.

- For the base m, the maximum period of a sequence is $m-1$. For the same m, but different multipliers a' s, the resulting sequences can have very different periods.

- For example:

  $m = 31$, $a = 12$, $x_0 = 1$,

  then the sequence goes through all 30 numbers from 1 to 30.

  $m = 31$, $a = 9$, $x_0 = 1$,

  then the sequence goes through 15 numbers between 1 and 30.

Remark on the Base *m*

- m is often a **prime** number, in particular a **Mersenne prime**, which has the form $2^p - 1$ (most commonly used: $2^{31} - 1$).
- If (and only if) m is a prime and the multiplier, a, is a **primitive root** modulo m, then the generator has a maximal period of $m - 1$.
- A primitive root, a, modulo a prime, m, is a number such that $m - 1$ is the smallest k satisfying $1 = a^k \bmod m$

Reference:

Random Number Generation and Monte Carlo Methods, James E. Gentle, Springer, 1998.

Shift-Register Geneators

- This class of generators rely on high-speed implementation of a finite-memory device, that utilizes a weighted sum of the last *n* elements of the series to compute the next element.

- A general form can be described by the equation:

$$x_{i+1} = w_1 x_i + w_2 x_{i-1} + \ldots + w_n x_{i-n} + 1 .$$

- The advantage of this method is that it can be implemented rather fast in hardware.

Lagged Fibonacci Generators

- The general form of these generators is given by:

$$x_i = x_{i-r} \cdot x_{i-s} \,,$$

where $\cdot$ is a binary operation defined on the set in which the x_i' s take values.

- We will denote a continuous uniform random variable taking values between 0 and 1 by ***U*****[0, 1].**