

PSEUDO-RANDOM VARIABLE GENERATORS, CONT.*

Ewa Paszek

This work is produced by OpenStax-CNX and licensed under the Creative Commons Attribution License 2.0[†]

Abstract

This course is a short series of lectures on Introductory Statistics. Topics covered are listed in the Table of Contents. The notes were prepared by Ewa Paszek and Marek Kimmel. The development of this course has been supported by NSF 0203396 grant.

1 PSEUDO-RANDOM VARIABLE GENERATORS, cont.

1.1 A Shift-Register Generator

An alternative class of pseudo-numbers generators are **shift-register** or **Tausworthe generators**, which have their origins in the work of **Golomb (1967)**. These algorithms operate on **n-bit**, pseudo-random binary vectors, just as congruential generators¹ operate on pseudo-random integers. To return a uniform (0,1) variate, the binary vector must be converted to an integer and divided by one plus the largest possible number, 2^n .

1.2 Fibonacci Generators

The final major class of generators to be considered are **the lagged Fibonacci generators**, which take their name from the famous Fibonacci sequence $U_i = U_{i-1} + U_{i-2}$. This recursion is reminiscent of the congruential generators, which the added feature that the current value depends on the two previous values.

The integer generator based directly on the Fibonacci formula

$$2^n \tag{1}$$

has been investigated, but not found to be satisfactory random. A more general formulation can be given by the equation:

$$U_i = U_{i-r} \cdot U_{i-s}, r \geq 1, s \geq 1, r \neq s, \tag{2}$$

where the symbol ‘square’ represents an arbitrary mathematical operation. We can think of the $U_i = 0$ as either binary vectors, integers, or real numbers between 0 and 1, depending on the operation involved.

As examples:

1. The $U_i = 0$ are real and dot represents either mod 1 addition or subtraction.

*Version 1.4: Oct 8, 2007 3:53 pm -0500

[†]<http://creativecommons.org/licenses/by/2.0/>

¹"PSEUDO-NUMBERS" <http://cnx.org/content/m13103/latest/#para_3>

2. The $U_i = 0$ are $(n - 1)$ -bit integers and dot represents either mod 2^n addition, subtraction or multiplication.
3. The $U_i = 0$ are binary vectors and dot represents any of binary addition, binary subtraction, exclusive-or addition, or multiplication.

Other generators that generalize even further on the Fibonacci idea by using a linear combination of previous random integers to generate the current random integer are discussed in **Knuth (1981, Chap 3.2.2)**.

1.3 Combinations of Generators (Shuffling)

Intuitively, it is tempting to believe that “combining” two sequences of pseudo-random variables will produce one sequence with better uniformity and randomness properties than either of the two originals. In fact, even though good congruential², Tausworthe (Section 1.1: A Shift-Register Generator), and Fibonacci (Section 1.2: Fibonacci Generators) generators exist, combination generators may be better for a number of reasons. The individual generators with short cycle length can be combined intone with a very long cycle. This can be a great advantage, especially on computers with limited mathematical precision. These potential advantages have led to the development of a number of successful combination generators and research into many others.

One of such generator, is a combination of three congruential generators, developed and tested by **Wichmann and Hill (1982)**.

Another generator, **Super-Duper**, developed by G.Marsaglia, combines the binary form of the output from the multiplicative congruential generator with a multiplier $a=69.069$ and modulus $m = 2^{32}$ with the output of the 32-bit Tausworthe generator using a left-shift of 17 and a right shift of 15. This generator performs well, though not perfectly, and suffers from some practical drawbacks.

A third general variation, **a shuffled generator**, randomizes the order in which a generator’s variates are output. Specifically, we consider one pseudo-random variate generator that produces the sequence (U_1, U_2, \dots) of uniform $(0,1)$ variates, and a second generator that outputs random integers, say between 1 and 16.

The algorithm for the combined, shuffled generator is as follows:

1. Set up a “table” in memory of locations 1 through 16 and store the values U_1, U_2, \dots, U_{16} sequentially in the table.
2. Generate one value, \mathbf{V} , between 1 and 16 from the second generator.
3. Return the \mathbf{U} variate from location \mathbf{V} in the table as the desired output pseudo-random variate.
4. Generate a new \mathbf{U} variate and store it in the location \mathbf{V} that was just accessed.
5. If more random variates are desired, return to Step 2.

NOTE: the size of the table can be any value, with larger tables creating more randomness but requiring more memory allocation

This method of shuffling by randomly accessing and filling a table is due to **MacLaren and Marsaglia (1965)**. Another scheme, attributed to **M.Gentlemanin Andrews et al. (1972)**, is to permute the table of 128 random numbers before returning them for use. The use of this type of combination of generators has also been described in the contexts of simulation problems in physics by **Binder and Stauffer (1984)**.

²“PSEUDO-NUMBERS” <http://cnx.org/content/m13103/latest/#para_3>