Selecting random number seeds in practice

Kleijnen, J.P.C.

Published in:
Simulation: Technical journal of the Society for Computer Simulation

Publication date:
1986

SIMULATION 47:1, 15-17.

ISSN 0037-5497/86 $2.00+.10
Printed in the United States of America

TECHNICAL NOTE

Selecting random number seeds in practice

Jack P. C. Kleijnen
Professor of Simulation and Information Systems
Department of Information Systems and Accountancy (ISA)
School of Business and Economics
Catholic University Tilburg (Katholieke Hogeschool Tilburg)
5000 LE Tilburg, The Netherlands

Key words: random number generation, sampling, statistical analysis, variance reduction

ABSTRACT

A pseudorandom number generator, such as a multiplicative congruential generator, depends on its initial value or seed. The computer may select a seed using its internal clock. Alternatively the simulation analyst may use Fishman's tables with seeds spaced 100 000 apart. Theoretically, the concepts of sampling with and without replacement are involved. Practical problems arise if consecutive simulation runs cannot be made in a single terminal session. The user may select the simplest option. Runs for different systems may use common seeds to decrease the variability; then the internal clock is an impractical source since its internal (binary) representation must be saved.

INTRODUCTION

This note was inspired by the following practical questions. How should the seed of a pseudorandom number generator be specified? Should that seed be sampled or should it be taken from a table with seeds "a 100 000 apart" as Fishman[1] proposed? If the seed is to be sampled, how can this statistical concept be realized? And if Fishman's proposal is followed, should not his tables be extended to pseudorandom number generators implemented on different (micro)computers? The literature does not address these questions explicitly and concentrates on the design and analysis of pseudorandom generators, treating seeds as a minor detail. This note does not pretend to present new concepts. It does try to provide practical answers; it also sketches theoretical concepts that others may wish to develop further.

THE PROBLEM

The most popular pseudorandom number generators are variants of the *multiplicative congruential method.* In the following equation the symbol a denotes a (constant) multiplier, b an additive constant, m the modulo, and these three parameters are nonnegative integers (b may be zero):

$$x_i = (a\,x_{i-1} + b) \mod m \quad (i = 1, 2, \ldots) \tag{1}$$

Numbers between zero and one ($0 \leqslant r_i < 1$) result from:

$$r_i = \frac{x_i}{m} \tag{2}$$

The literature discusses the specification of the parameters a, b, and m in great detail, including statistical tests of the independence of successive pseudorandom numbers (a posteriori, empirical analysis) and mathematical analysis (a priori, deductive analysis based on number theory). For a recent review see Ripley.[5]

In practice the simulationist uses a computer with a given generator, i.e., the computer comes with given values for the parameters a, b, and m. The user, however, does have control over the initial value or *seed* x_0. For example, one wants to make a number of "runs" for a gas station modeled as a queuing system with random arrival and service times; simulation can cover n_1 days ($n_1 \geqslant 1$); next one more pump can be added to the existing system and its operation is simulated during n_2 days ($n_2 \geqslant 1$), and so on. How should the analyst select the seeds $x_0^{(j)}$ in run j with $j = 1,\ldots,n_1$ and $x_0^{(k)}$ in run k with $k = 1,\ldots,n_2$, and so on?

ALTERNATIVE SOURCES FOR THE SEED

Default options

The user may decide to let the computer select the seed. Usually the computer generates a seed through its *internal clock*, i.e., the computer looks at its "digital watch" and uses the numbers representing time to fix the seed.[5] Some computer systems have a different default option, i.e., they always start with the same seed, for example, 12345678×2^{31}. This second option means that all simulation experiments are statistically dependent. Consequently, if the experiments investigate variations of a system, then the conclusions are less general. Also, all experiments with the second option, use only part of the total pseudorandom number stream; this part may happen to have bad statistical properties. With the first option (internal clock) the bad and good parts of the stream have the same chance, assuming the internal clock generates random seeds;

a "good" random number generator should have "bad" parts (if a good die is thrown long enough, then a hundred 6s in a row do result).

External sources for seeds

Instead of letting the computer select a seed, the user may select a seed. In practice many users pick the same simple seed (such as $x_0 = 123$) in all their simulation experiments. Selecting a common seed in all simulation experiments has disadvantages, as discussed above. These disadvantages can be eliminated as follows.

Fishman (pp. 481-487) gives tables with seeds "spaced 100 000 apart", say $x_0^{(1)}$, $x_0^{(2)}$,... If a user initializes with $x_0^{(1)}$ then 100 000 calls to the generator yield $x_0^{(2)}$, and so on. If the user selects two seeds from these tables, then these two simulation runs have *non-overlapping* streams of pseudorandom numbers. This overlap becomes of interest if several runs, each with its own seed, are considered (see the next section). First note, however, that Fishman gives these tables for only three generators. If the user prefers Fishman's option and the user has a different computer, then such a table must be constructed for the specific generator.

MORE SEEDS IN ONE EXPERIMENT

The user certainly makes more than one run. The present section will show how the simulationist may select the seeds of the n_1 runs with the same simulation program (in the preceding example, the user simulated n_1 days of the existing gas station).

The situation is simplest if the user makes these n_1 runs in a *single* terminal session or batch (after run #1 terminates, run #2 starts with the next pseudorandom number). For example, if run #1 stops after 500 numbers are used, then run #2 uses x_{500} as seed in Eq. 1.

The situation is different if the user makes the n_1 runs in more than one session. There are then several sources:

(1) The internal clock (see the preceding section).

(2) Tables with seeds 100 000 apart (again see that last section).

(3) A computer log, i.e., the computer saves the last pseudorandom number used in the previous session. This option has one major practical drawback, namely, internally the computer uses more bits than presented externally (on the screen or on paper). So the user cannot feed in the externally presented number as can be done with Fishman's tabulated numbers. If the user prefers option (3), it will be necessary to "dig" into the microcomputer or turn to the system analyst of the computer center. Statistically, options (2) and (3) are the same. The differences between the options (1) and (2) are investigated in the next subsection.

Sampling with and without replacement

If the analyst uses the internal clock (option 1) to specify the seed for the next run, then overlap between runs may occur, whereas tabulated seeds (option 2) guarantee nonoverlapping streams of pseudorandom numbers. Many authors consider this overlap as undesirable; see References 1; 4 (p. 30); and 6 (p. 507). Theoretically, however, sampling *with* replacement always means that sampling the same values is possible (for example, if from an urn with 1 000 000 balls a first sample of 5 balls is taken and then a second sample of 5 balls is taken, the second sample may contain one or more individual balls of the first sample, provided the first sample was replaced). Sampling *without* replacement yields a certain *dependence* (the balls of the first sample cannot be sampled in the second sample). The difference between sampling with and without replacement becomes smaller with increased population size. In the urn example the probability laws are known as the *binomial* and the *hypergeometric* distributions respectively. These distributions have the same mean; however, the variance of the hypergeometric distribution is:

$$\text{var}\left(\sum_{i=1}^{n} x_i\right) = n\,p\,(1-p)\,\frac{N-n}{N-1} \qquad (3)$$

where

x_i is zero or one (white or red ball); n is the sample size ($n=5$ in the example); N is the population size ($N=1\,000\,000$); and $p=p(x_1=1)$. As N increases, Eq. 3 approaches $n\,p\,(1-p)$, the variance of the binomial distribution. So, as the population gets bigger, the difference between sampling without and with replacement becomes smaller.

Sampling with replacement is generally advocated in statistical handbooks, since such a procedure creates independence and simplifies the statistical analysis! In the simulation literature, however, many authors implicitly advocate sampling without replacement. In simulation practice, the difference between the two procedures is negligible, since the psudorandom number generator has a very long period (big population size). Therefore the user may choose the simplest procedure. The user may let the computer select the seed through the internal clock, instead of using Fishman's tables or creating a new table. An exception arises if the user wants to apply the variance reduction techniques discussed in the next section.

The multiplicative generator (see Eqs. 1 and 2) implies that r_i does not equal $r_{i'}$ (unless the parameters a, b, and m are poorly chosen so that unacceptable statistical properties result), where $i \neq i'$ and $i, i' = 1, 2, \ldots, h$ and h denotes the cycle length or period. But this property means that sampling *within* the simulation run occurs *without* replacement! Theoretically, this property conflicts with the simulation model which specifies independent interarrival times. Practically, the dependence created by sampling without replacement, is negligible. Moreover, different pseudorandom numbers may generate identical variates (as in sampling with replacement), if the random input variable is discrete (counterexample: exponential interarrival times).

COMMON PSEUDORANDOM NUMBERS

In the gas station example, the two systems (namely the existing system and the system augmented with one more pump) may be simulated using the same pseudorandom arrival pattern of customers, i.e., using two identical seeds (for the moment, we assume that arrival times are the only random component of the model). Common seeds create common pseudorandom number streams. Common seeds tend to reduce the variances of the *differences* between simulation responses. Unfortunately, the statistical analysis of a simulation experiment with common seeds is controversial (for a recent survey see Reference 3).

An additional practical problem is that the user needs *multiple* seeds per run. The reason is that in order to create a strong positive correlation between the responses of different systems, it is advisable to use separate pseudorandom number streams *per* type of variable. For example, one stream for arrival times of customers and one stream for service times of pump operators. (In simple systems a single seed suffices, for example, in the gas station the odd pseudorandom numbers $r_1, r_3, r_5, \ldots$ are used for arrival times and the even numbers $r_2, r_4, r_6, \ldots$ for service times so that no "synchronization" problem arises (see Reference 2, p. 201). Considering $K \geqslant 1$ types of input variables per run, the sources for the K seeds are the same as in the situation with a single input variable, namely Fishman's tables (select K values from the table) and the internal clock (the clock increases with one tick per pulse so that by the time the simulation program asks for the next seed, the clock has been ticking away for a "long time"). However, if different systems are not run in parallel, then the state of the internal clock must be saved; the externally displayed clock is inaccurate. So it is simplest to realize common seeds through externally provided seeds like Fishman's seeds.

We add three notes: (1) Multiple seeds can be represented by a single seed, concatenating the K individual seeds (see Reference 4, p. 30), (2) Antithetic pseudorandom numbers is a different variance reduction technique which, however, involves the same issues as do common random numbers, and (3) Common seeds are also useful as a debugging device, i.e., the corrected simulation program uses the same seed as the original program did.

CONCLUSIONS

Statistical models are only approximations of reality, i.e., the uniform distribution is a model of a "good" pseudorandom number generator. Selecting seeds "randomly" and "100 000 apart" may be modelled as sampling with and without replacement respectively. The difference between these two models is negligible, if the population size is large, as is the case for pseudorandom number generators with long cycles. Therefore practical considerations may guide the simulation analyst, for example, can the internal representation of the computer clock be saved, and do Fishman's tables apply to the generator at hand?

REFERENCES

1 FISHMAN, G.S.
Principles of Discrete Event Simulation. John Wiley & Sons, Inc., New York (1978).

2 KLEIJNEN, J.P.C.
Statistical Techniques in Simulation. Volumes I and II. Marcel Dekker, Inc., New York. (Russian translation (1978): Publishing House 'Statistics,' Moscow) (1974/1975).

3 KLEIJNEN, J.P.C.
Statistical Tools for Simulation Practioners. Marcel Dekker, Inc., New York (in press, 1986).

4 MIHRAM, G.A.
"Simulation methodology: statistical aspects." *Proceedings, 1983 Winter Simulation Conference* (Roberts, S.; J. Banks, and B. Schmeiser, eds.). Published by IEEE, New York (1983), 27-36.

5 RIPLEY, B.D.
"Computer generation of random variables - a tutorial." *International Statistical Review* 51 (1983), 301-319.

6 SCHRUBEN, L.W. and MARGOLIN, B.H.
"Pseudorandom number assignment in statistically designed simulation and distribution sampling experiments." *Journal American Statistical Association* 73:363 (1978) 504-525.