Reality & Randomness

Richard Duhrkopf
Department Editor

***Editor's Note:** This is the first in a series of occasional columns on using random numbers in simulations.*

I began a column from a suggestion for a computer program in a recent text. That column grew into two columns. After further consideration, I decided to write a third column to introduce a topic necessary for the other two. I want to discuss using random numbers in programs.

In the past, I have complained about too much drill/review software. Drill/review software isn't bad; it can be an effective tool. In fact, I have written several tutorial and drill/review programs. But I think we have spent enough time developing this type of software; we need to complement it with simulations. And, simulation software requires a different approach.

There are two different types of simulations: deterministic and stochastic. Deterministic simulations use a specific mathematical model to simulate an event. For example, in a genetic simulation, a student chooses between a variety of potential parents. The simulation generates offspring based upon the genotypes of those parents. For two heterozygous parents, the offspring would be 0.25 AA, 0.50 Aa and 0.25 aa. The deterministic simulation always returns the same offspring frequencies from those parental genotypes. This is a good example of a useful deterministic simulation. The simulation starts with 10 individuals of various phenotypes. The student is told nothing about how the trait is inherited. Parents are chosen and the program provides the results of the crosses. The student must determine how the trait is inherited and the genotypes of all individuals.

Richard Duhrkopf, editor of the Computer Center, is an assistant professor in the **department of biology at Baylor University, Waco, TX 76798.** He teaches introductory biology for majors and nonmajors and advanced genetics and also serves as director of Biological Computing. He has a B.S. in zoology and a M.S. and Ph.D. in genetics, all from Ohio State University. He has been active in the development of a wide variety of educational software.

Stochastic simulations provide more realism by adding the element of randomness. Reality involves a random component. If we are going to simulate reality, we must simulate that random component. In our example, in a group of real offspring generated from two heterozygous parents, the frequency of the AA genotype might be 0.30 by nothing more than random events.

Random number generators do what the name implies: they generate numbers in a random sequence. But why should we expect random number generators to be different from anything else connected with computers? Each language and each computer has its own random number generator. When I started programming, IBM's PL/I language had several different options with respect to random numbers.They could be generated from a uniform distribution (each number has an equal chance of being drawn) between specified values, or they could be generated from a normal distribution with a specified mean and variance. Most random number generators now generate only from a uniform distribution.

You also need to consider the nature of the numbers. Some random number generators return real (decimal) numbers between 0 and 1. Microsoft's MBASIC language works with real numbers. The MBASIC statement X = RND generates a real number between 0 and 1. Others generate random integers between 1 and N or between 0 and N (where N is specified by the program). For example, in ZBASIC the statement is X = RND(N) (where N is any integer) produces a random integer between 1 and N. The documentation for your language tells you how to draw random numbers and their nature.

Once a random number is drawn, it can be easily converted from one form to the other. If a real number between 0 and 1 is drawn, it can be converted to an integer. In MBASIC, the statement X = INT(RND*10) generates random integers between 0 and 10. If an integer is drawn, it can be converted to a real number. In ZBASIC, the statement X = (RND(11)-1)/10.0 generates a real number between 0 and 1 (providing X is declared as a real variable). However, this demonstrates a potential problem. The ZBASIC statement above begins by generating integers between 0 and 10. Dividing those numbers by 10 gives us only 10 different possible real numbers with only one decimal place (0, 0.1, 0.2, 0.3, etc.) A better technique would be to generate larger integers. That statement would be X = (RND(100001)-1)/100000.00000. That statement generates real numbers with 5 decimal places. Converting numbers from a uniform distribution to a normal distribution is a little harder and we will discuss that later.

There is one final point about random number generators that we must consider: seeding them. Most generators will always start with the same number unless they are told to do otherwise. We might question the logic of a random number generator that generates the same sequence of numbers. However, it helps in debugging programs for the random number generator to do just that. Once we know the program works, we can make the random number generator select randomly. That is done in most languages by the statement RANDOMIZE. This is analogous to the old method of using a table of random numbers. The numbers in that table are in a random sequence. But, if we always started with the first number in the table, we would always get the same sequence of random numbers. We must randomly choose a number in the table to start the sequence. That is what the RANDOMIZE statement does.

Let's see how we might use random numbers in a simple example and move to one that is more complex. Our simple case involves a couple having a baby. What sex will that baby be? We can determine that by drawing a random number. In MBASIC, we would use:

```
X=RND.
IF X >=0.5 THEN SEX$="MALE"
ELSE SEX$="FEMALE"
```

In ZBASIC, the statements would be:

```
X=RND(2)
IF X=1 THEN SEX$="MALE" ELSE
SEX$="FEMALE"
```

We can expand this example to simulate families of a specific size by putting these statements in a loop. If we wanted to use ZBASIC to create a family of five children, the statements would be:

```
FOR I% = 1 TO 5
X=RND(2)
IF X=1 THEN SEX$="MALE" ELSE
SEX$="FEMALE"
PRINT SEX$
NEXT I%
```

Let's program the example we mentioned above. We can generate 100 offspring from a cross of two heterozygous parents. The statements for the ZBASIC program would be as follows:

```
A1%=0
A2%=0
A3%=0
FOR I% = 1 TO 100
X=RND(4)
IF X=1 THEN A1%=A1%+1 ELSE
IF X <=3 A2%=A2%+1 ELSE
A3%=A3%+1
NEXT I%
PRINT "The number of AA off-
spring would be ";A1%
PRINT "The number of Aa offspring
would be ";A2%
PRINT "The number of aa offspring
would be ";A3%
```

If we wanted to convert those genotypic ratios to phenotypic ratios, for complete dominance, the number of offspring with the dominant phenotype would be A1%+A2%. The number of recessive offspring would be A3%.

Random numbers from a uniform distribution work nicely in such simulations. However, there are times when we might want to draw numbers from a normal distribution. For example, let's say we wanted to produce 1000 offspring and determine their sex. We could do it with a loop which would draw 1000 random numbers as we did above. There is nothing wrong with this approach except it takes time. It would be better to draw a single random number which would give us the frequency of one of the sexes.

We might be tempted to draw a real number between 0 and 1 and use that as the frequency of one of the sexes. The problem with that approach is the distribution of the frequencies of the two sexes is not uniform. Our simulation has the same probability of setting the frequency of one sex to be 0.1 as it does of setting the frequency at 0.5. This is obviously unrealistic. The distribution of the sex of children is normally distributed with a mean of 0.5 and a certain standard deviation. We must draw a random number from that distribution to provide us with the frequency of one of the sexes. There is a way of converting a random number drawn from a uniform distribution into one from a normal distribution. I learned this several years ago when I was at Grinnell College. I needed random numbers from a normal distribution and BASIC on the VAX generated numbers only from a uniform distribution. The director of the computer center, Tom Moberg, showed me how to do the conversion. I must admit I do not understand exactly how it works, but I will show you that it does.

First, draw a number from the uniform distribution of real numbers between 0 and 1. One note here is to draw it from a distribution with large boundaries if you are drawing an integer and converting it to a real number. I usually draw a random integer between 1 and 10,000 and convert it to a real number by dividing by 10,000 (producing a number with four decimal places). That number (we will refer to it as X) then gets converted to a Z value by the following formula:

$$Z = [X^{0.135} - (1-X)^{0.135}] / 0.1975$$

This generates a random variable from a pseudo-normal distribution with a mean of 0 and a standard deviation of 1. To transform the number to a distribution with a different mean and standard deviation, multiply the Z value by the standard deviation and add that to the mean of the distribution. For example, if we wanted to draw a random number from a normal distribution with a mean of 0.5 and a

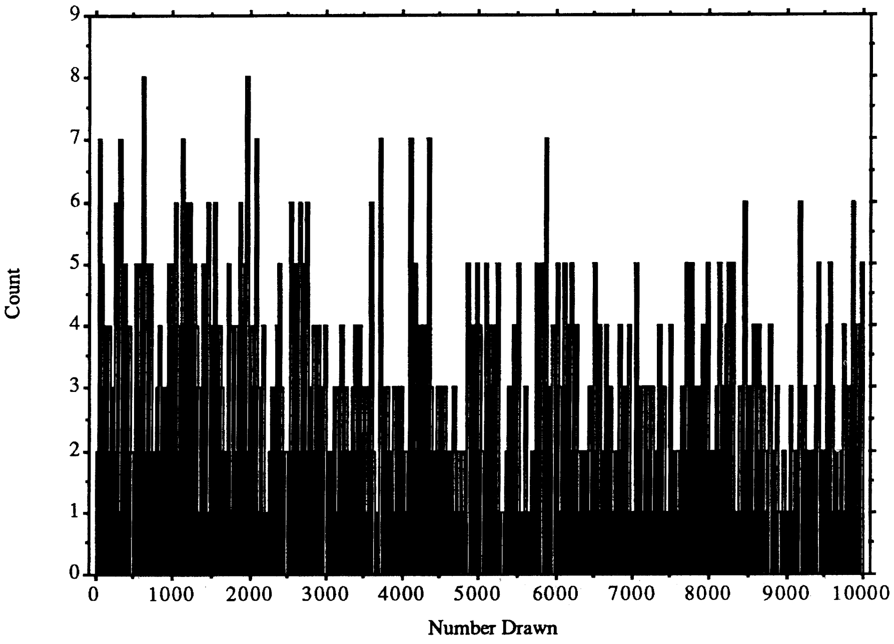

Figure 1. The uniform distribution of 1000 random integers between 1 and 10,000.

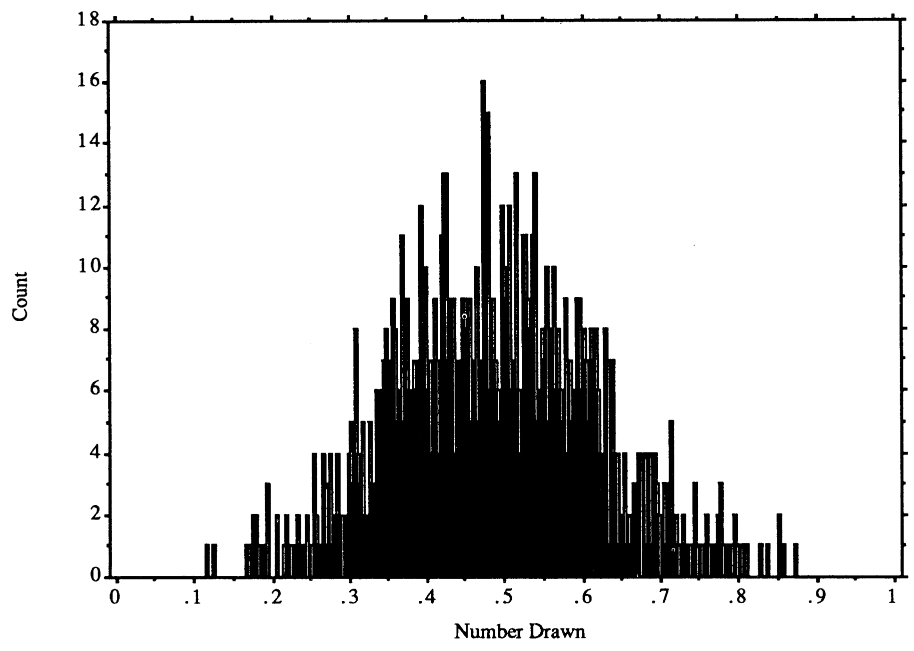

Figure 2. The distribution produced as a result of the normalizing transformation.

standard deviation of 0.125, we would use the following statements:

```
X=RND(10,000)/10,000.0000
Z=((X^.135) - ((1-X)^.135))/.1925
Fr=0.5 + (0.125*Z)
```

To show how this really works, I drew 1000 random integers between 1 and 10,000 and converted them to random real numbers from that normal distribution. Figure 1 shows the uniform distribution of the numbers drawn. Figure 2 shows the pseudo-normal distribution of the numbers after conversion.

This technique allows us to draw a single random number to be used for the frequency of one of the sexes in a simulation. There are two take-home messages this month. First, it is possible to do things if you are persistent enough. And second, it is great to have friends like Tom who know more than you do.

edvotek® inc.
The Biotechnology Education Company

EDVOTEK is more than a biotechnology supply company

- ***Education and training***
- ***Experiment modules***
- ***Biotechnology equipment***
- ***In-house technical service***

EDVOTEK biotechnology training courses: Bridging core technology and education

EDVOTEK, Inc. - P.O. Box 1232, West Bethesda, Maryland 20827-1232

1-800-EDVOTEK or (301) 251-5990