More Efficient Monte Carlo Simulations for Mortality Assumption Testing

by Douglas L. Robbins

Editor's Note: The section's Statutory Issues List Serve would be an appropriate forum for discussing concepts in this article.

The advent of Model Regulation XXX, now passed in most states, has imposed on the industry a need to quantitatively evaluate the anticipated mortality rates underlying many blocks of life insurance business. The most common method of doing this has been developing a probability distribution for total claims (given the anticipated mortality for the block being tested) and seeing where actual claims over a given period fall. If the actual result is at an unacceptably high percentile of the predicted claims distribution, then current anticipated mortality is rejected in favor of some higher set of rates.

Developing a closed-form distribution for aggregate claims, however, can be problematic when many policies of varying face amounts are involved. For that reason, the most common industry method of creating this distribution has been Monte Carlo testing. In the most common version of this methodology, each policy in the block is independently assigned a random number between zero and one—once per scenario. If the random number is lower than the anticipated mortality rate for that policy, that is considered a "death," and the face amount for that policy is tabulated. If the random number is higher, then that is considered a "survival," and zero is entered. For each scenario run, the face amounts of all "deaths" are summed to give a total claim amount. The resulting sample, over a large set of simulations, provides the empirical claims distribution.

This process can create the need for the generation of quite a large quantity of random numbers. For a block of 100,000 policies over 10,000 simulations, for example, the requirement would be for one billion random numbers. Occasionally, an actuary faced with such a requirement could run into a real time crunch. Is there a way to reduce this random number requirement, and thus computer run time?

There is, and it was first suggested to me by my father, Edward Robbins. (He in turn would credit reading from one of several good 20-plus-year old *Transactions* articles that deal with compounding of distributions in risk theory. Neither of us has been able to pinpoint exactly which article inspired his thinking on this.)

To see how run time can be reduced, it is helpful to think of the one billion required simulations as though they were done in a spreadsheet (despite the fact that the testing would doubtlessly *really* be done via some program). In that spreadsheet, the identifiers of the 100,000 policies being tested run down column A. The face amounts pertaining to each policy are then entered down the rows of column B.

Then, in each of the next 10,000 columns, for each of the 100,000 policies, there are a series of ones and zeros, with a one representing a death (the generated random number being less than the anticipated mortality rate for this policy), and a zero representing survival. Then at the bottom of each of these 10,000 columns, a sumproduct function is done between this column of zeros and ones and the face amount column, producing a total claim amount in dollars for that trial. The resulting 10,000 total claim amounts become the empirical distribution for analyzing the actual mortality results over the test period.

Now it is true that the above methodology is one correct way of forming the required empirical distribution. But it is not the only correct way. Distributionally, there is a way to fill in the 10,000 entries in each row, using far fewer random numbers.

In the above methodology, each entry in our one billion-cell grid is a Bernoulli trial. The random numbers we are drawing are tested against the Bernoulli probability density function (PDF) relevant to the given row's anticipated mortality rate, in order to assign a one or zero. However, it is clear that if we have an infinite series of repeated Bernoulli trials with the same probability of success (in this case death), a geometric distribution will provide the probability that our first success occurs on precisely trial number "n."

For example, say an established NFL quarterback can be assumed based on experience to have a 5 percent chance of throwing an interception on any given pass. It is then clear that at any time, the chance of his next pass being an interception (assuming we don't know anything else about the type of pass it will be, quality of defense, etc.) is 5 percent. The chance of his first interception coming on his second pass from now, must be 95 percent times 5 percent equals 4.75 percent. The chance of it being on his third pass is 95 percent (squared) times five percent equals 4.51 percent, and so on.

These values form the geometric PDF, which is defined as P(N=n) = (1-q)^(n-1)*q, where q is the probability of success (in our case death) on any given trial. The values of the pdf can be summed in order to form the cumulative distribution function (CDF).

Let us now go back to drawing random numbers. To start filling in our one billion-cell grid of ones and zeros under the *old* methodology, we used an initial random number and compared it to the Bernoulli PDF, filling in cell one. Instead, we now compare it to the CDF of the Geometric distribution. By the logic above, the cdf entry pertaining to that random number can be thought of as the timing of the first observed death in a series of independent trials, all with the same probability (the anticipated mortality for this row).

In other words, say that our q for this row is actually 1 percent. Our first random number drawn is 0.5. Under the *old* methodology, this would fill only one grid cell, with an entry of zero (survival). Under the *new* methodology, drawing 0.5 from the geometric CDF with a parameter q = 0.01, produces a result of 69. This results in us going across the first row of our grid, filling in 68 zeros and then a one in the 69th cell. (Verbal interpretation: the first observed death in a large number of identical trials came on trial number 69. All earlier trials resulted in survival.)

Only at the 70th cell do we then need a new draw of a random number. That number is used to fill in cells starting with the 70th entry on the top row. Say we draw 0.005. That is clearly a result of one from our CDF, so cell 70 also gets an entry of one, and then we start again with the 71st cell. The next draw might be 0.98, producing a result of 389 zeros and then a one in the 460th cell. The program would continue filling in values in this way through the end of the first grid row.

At the end of that row, the program would have to stop no matter what. The reason is that the compounding of Bernoulli trials into a Geometric distribution only works if the trials have identical q's. (And except in infrequent cases, the q will change when starting with a new policy.) Any CDF result taking things beyond the end of the row results in just filling in the remainder of that row with zeros and then starting over at the next row.

Once the entire grid of 1 billion cells is filled in with ones and zeros, the stochastic implications of the grid under this new methodology are the same as for the old one. Thus the interpretation of the column totals is stochastically exactly the same.

What is the expected result of all of this in terms of run time? Since the expectation for any geometric trial result is 1/q, the expected run time for any row would be about q times the run time required to fill the row in one cell at a time. In general, overall run time should be about the average q times the old run time!

Douglas L. Robbins, FSA, MAAA, is a consulting actuary at Tillinghast-Towers Perrin in Atlanta, Ga. He can be reached at doug.robbins@tillinghast.com.