SOCIETY OF ACTUARIES

Article from:

CompAct

July 2009 – Issue 32

SOCIETY OF ACTUARIES Technology Section

CompAct

ISSUE 32 | JULY 2009

Cluster Modeling: A New Technique To Improve Model Efficiency

By Avi Freedman and Craig Reynolds

A new approach to actuarial modeling can solve a familiar problem for life insurance companies.

Life insurance companies around the world employ actuarial models to use in applications such as financial forecasting, product pricing, embedded value, risk management, and valuation. Depending on the company and the application, such models might be seriatim or might reflect some degree of compression.

Despite the enormous increase in processing power over the past decade or two, the ability of many companies to run models in a timely fashion has arguably worsened, primarily due to the large number of stochastic scenarios required to properly evaluate many types of insurance liabilities. For some purposes, nested stochastic scenarios need to be used, increasing runtime even more.

Typically, actuaries manage runtime in one of three ways:

- improving software efficiency;
- getting more or better hardware;
- reducing cell count via mapping techniques.

The first option can provide incremental payoffs but is unlikely to provide the order-of-magnitude performance improvements that the actuary might desire. The second option can help materially. Today it is not uncommon to see companies with grid farms of hundreds of computers. But the cost and maintenance efforts associated with such grid farms can be significant, and we have seen from experience that the complexity of the runs needed

CONTINUED ON **PAGE 4**

Actuaries
Risk is Opportunity.®

always seems to grow to match or exceed the capacity of the grid available.

Classic mapping techniques used to reduce cell counts and manage runtime can work reasonably well for some types of business, but they do not work as well when applied to products with many moving parts and heavy optionality, making it difficult to use them to create models even of moderate size, let alone of a smaller size to enable running against a large number of scenarios in a reasonable time.

Milliman has developed a technique called cluster modeling, a variant of cluster analysis techniques used in various scientific fields, to allow for an automated grouping of liabilities, assets or scenarios. This technique is described, with several liability examples, in our paper "Cluster analysis: A spatial approach to actuarial modeling."[1] The paper includes a number of case studies that illustrate excellent fit from cluster techniques with cell compression ratios (actual policy count/model cell count) of 1,000–1 or more.

We believe that cluster modeling will be a valuable tool that will allow companies to reduce their runtime for stochastic models …

In this article, we briefly describe the technique, and then offer some comments on implementation of the algorithm and on certain decisions that need to be made in considering how best to apply cluster modeling.

We believe that cluster modeling will be a valuable tool that will allow companies to reduce their runtime for stochastic models by orders of magnitude with minimal reduction in model accuracy. This process will generally be easier to implement than other processes for improving model efficiency and has material advantages over the use of replicating portfolios—another common technique to improve model efficiency.

DESCRIPTION OF THE TECHNIQUE

For purposes of exposition, we assume that we are using cluster modeling to compress a set of policies (perhaps 1,000,000) to a much smaller set of cells (perhaps 500).

The following are defined for each policy:

- An arbitrary number of *location variables*. A location variable is a variable whose value you would like your compressed model to be able to closely reproduce. Some location variables can be statically known items (e.g., starting reserves per unit); others can be projection results from a small number of *calibration scenarios* (typically one to three), such as:
  - reserves, cash value, account value, or premium per unit as of the projection date;
  - present value of GMB claims per unit;
  - sum of the premiums paid in the first five years of the projection per unit;
  - first-year liability cash flow per unit;
  - present value of profit (PVP) per unit.
- A size variable to represent the importance of a given policy. This ensures that large policies are not mapped away as readily as small policies, all other things being equal. For example, the size variable would typically be represented by face amount for life insurance or account value for deferred annuities.
- A *segment*; the program will not map across segment boundaries. Segments might be plan code, issue year, GAAP era, or any other dimension of interest. Reasons for using segments include:
  - To decrease calculation time, which is roughly proportional to the sum of the squares of the number of policies in each segment; a group run as one segment will take approximately 10 times as long as the same business split into 10 equal-sized segments. Assuming that the segments serve to separate policies that would be unlikely to be mapped together in any case, the results would be essentially the same.
  - For reporting, reconciliation, or similar reasons, where you might wish to keep policies from one

FOOTNOTES

[1] Available at http://www.milliman.com/expertise/life-financial/publications/rr/pdfs/cluster-analysis-a-spatial-rr08-01-08.pdf

segment of business from being mapped into another segment.
- Whenever the location variables by themselves do not, in your judgment, sufficiently distinguish policies in different segments.

The program then proceeds as follows:

1. The *distance* between any two policies is calculated using an n-dimensional sum-of-squares approach, as if the n location variables defined a location in n-dimensional space. Thus, as an example, with three location variables, *Var1, Var2* and *Var3*, the distance between policy 1 and policy 2 could be measured as:

$$\sqrt{(Var1_1 - Var1_2)^2 + (Var2_1 - Var2_2)^2 + (Var3_1 - Var3_2)^2}$$

In this definition, the location variables must be appropriately scaled. Each of the location variables is normalized by dividing each one by the size-weighted standard deviation of the associated variable. Users can also introduce weights to place different priorities on matching different distance variables.

2. The *importance* of each policy is defined by the cluster modeling process as the size times the distance to the nearest policy. Thus, a policy is unlikely to be mapped to another if it has a large size and is far away from others; however, a small policy or one that is very close to another is likely to be mapped to another policy.
3. At each step, the process finds the policy with the lowest importance and maps it to its nearest neighbor *(the destination policy)*, adjusting the size, and hence the importance, of the destination policy in the process. This step is repeated until the model has the desired number of model points.
4. At this point, only the user-specified target number of clusters remains. In the next step, the program finds the most representative policy in each cluster, which is the policy in each cluster that is closest to the average location (centroid) of all cells in the cluster. In general, each cell in the compressed in-force file will consist of a policy from the original in-force file, scaled up (i.e., with all variables that are logically additive grossed up by the size of the entire cell group over the size of the original policy, and all other variables taken from the original policy). For certain variables, you may prefer instead to sum the values from the various policies mapped into the cell, although this should be done sparingly because the distance methodology in essence assumes that cells will be scaled up.

The pictures below can help demonstrate the cluster modeling process. In it, we assume just two location variables that reflect two dimensions. The scatter plot in Figure 1 represents the value of each location variable by the point placement on the two-dimensional graph. The size of each dot represents the size of the policy. In Figure 2, each policy has been assigned to a cluster. Finally, the resulting four-point model is shown in Figure 3 with the size of the four model points appropriately grossed up:

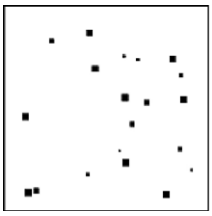

Figure 1

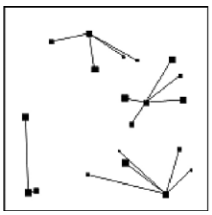

Figure 2

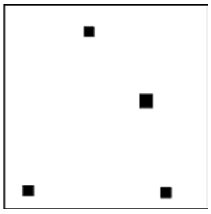

Figure 3

IMPLEMENTING THE TECHNIQUE

A naive implementation of the process, shown using R, might be as appears on page six. There we use an example involving six policies (from two segments), and three location variables representing nothing in particular.

CONTINUED ON **PAGE 6**

Avi Freedman, FSA, MAAA, is a consulting actuary with the life insurance consulting practice of Milliman in New York, NY. He may be reached at avi.freedman@milliman.com.

Craig W. Reynolds, FSA, MAAA, is a principal with the life insurance consulting practice of Milliman in Seattle, Wash. He may be reached at craig.reynolds@milliman.com.

```
1  ####################setup
2  loc0 <- read.table(textConnection("
3  23. 15. 13.
4  10. 20. 30.
5  24. 15. 13.
6  10. 26. 30.
7  25. 15. 13.
8  10. 20. 31.
9  "))
10 siz <- c(49, 25, 5, 50, 50, 100)
11 segments <- c(0, 1, 0, 1, 0, 1)
12 weights <- c(1, 1, 10)
13 endcells <- 3
14 ####################calculations
15 loc1=loc0*siz
16 avgloc <- apply(loc1,2,sum)/sum(siz)
17 vars <- (apply(loc1*loc1/siz,2,sum)/sum(siz))-avgloc*avgloc
18 scalars <- weights/sqrt(vars)
19 loc <- t(scalars * t(loc1/siz))
20 elements <- length(siz)
21 mappings <- elements - endcells
22 listel <- 1:elements
23 z1 <- rep(listel, elements)
24 z2 <- sort(z1)
25 use <- segments[z1] == segments[z2] & z1 != z2
26 z1 <- z1[use]
27 z2 <- z2[use]
28 diff <- loc[z1,]-loc[z2,]
29 dist <- sqrt(apply(diff*diff, 1, sum))
30 siz2 <- siz
31 dest <- listel
32 for (tt in 1:mappings) {
33    keep <- dest[z1]==z1 & dest[z2]==z2
34      z1 <- z1[keep]
35      z2 <- z2[keep]
36      dist <- dist[keep]
37      importance <- siz2[z1]*dist
38      index <- order(importance)[1]
39      tempfrom <- z1[index]
40      tempto <- z2[index]
41      print(c(tempfrom, tempto))
42      dest[dest == tempfrom] <- tempto
43     siz2[tempto] <- siz2[tempto] + siz2[tempfrom]
44     siz2[tempfrom] <- 0
45 }
46 print(dest)
```

Location variable, size and other information is given in lines two through13. Lines 15 through19 perform the normalization. Lines 20 through 27 limit comparisons to pairs of cells in the same segment, and 28 and 29 calculate distances. Lines 30 through 42 correspond to steps two and three. The above script does not implement step four; this is left as an exercise for interested readers.

When run, the script will show cell three mapped into cell one, then five into one, then two into six. Examination of the data will show why this was done. Cells three and one are close, and cell three is very small. Cells five and one are close, and when cell three is mapped into cell one, cell one is larger than cell five, and so cell five is mapped into cell one. The other segment, consisting of cells two, four and six, is more spread out when the large weight on the third location variable is taken into account, so only now does a mapping occur in that segment. Understanding why two is closer to six than to four requires working out the standard deviations.

While the R code given above is helpful for understanding the algorithm, it is unsuitable for any real use, since it uses too much space and runs very slowly. In the production implementation, memory usage is reduced by not keeping around all of the distances, and recalculating missing distances as needed. Runtime is minimized by using C++, including compiler intrinsics for SSE instructions, instead of R; by distributing calculations over multiple cores where appropriate; and by giving careful consideration to when calculations may be deferred (and probably ultimately rendered unnecessary) without changing the results.

OPEN QUESTIONS

There are numerous questions that will need to be faced by companies adopting cluster modeling techniques. Because the techniques are still new, it is uncertain what choices companies will make in practice; these choices will depend on the specific circumstances.

How close is close enough?

We worked with a client on creating a cluster model to determine the market value of liabilities (MVL) averaged over multiple scenarios, both for a basic set of scenarios and for sets of shocked scenarios. Differences (compared to a near-seriatim model) were very small (within about five bp of starting account value) for the baseline and liability shocks, but somewhat larger for economic shocks (up to about 20 bp for a large downward equity shock). (SEE TABLE 1), Pg. 8

A company in that situation might well choose to use cluster models for the insurance shocks but continue to use the larger model for economic shocks (while searching for a set of location variables and calibration scenarios that would work better). Or it might choose to use cluster models for all of the shocks, and use the computing resources freed up to run larger sets of scenarios and reduce the sampling error caused by scenario selection.

How should cell clustering and scenario clustering be mixed?

The clustering technique can also be used to select scenarios (in this case, there is only one segment, and all scenarios have weight one). Location variables can be based on the scenario inputs (e.g., wealth factors) and/or on results from a model. The model itself may be a very small cluster model; the resulting set of scenarios may then be run against the full model or a relatively large cluster model. Alternatively, of course, companies can rely entirely on cell clustering and ignore the scenario clustering functionality. More information as to the benefits of the various approaches will develop as companies gain experience with the technique.

What model sizes are best?

Cluster models of 2,500 cells will, of course, take longer to run through various scenarios than models of 250 cells. On the other hand, the relatively larger models have their advantages. The advantage is not so much in being able to serve usefully across a wide range of stochastic scenarios under a single set of assumptions, but more from the increased robustness in the face of changes either to liability assumptions or to the assumptions underlying the scenario generator. In order to determine the optimal size, companies will need to consider how broadly they intend the models to be used and how large of a model they are willing to have.

CONTINUED ON **PAGE 8**

Table 1
Market Value of Liabilities
Calculated using full model and cluster model
Per $10,000 of starting account value
Under base assumptions and 11 shocks

	Full	Cluster	Full	Cluster
	MVL	MVL	Shock	Shock
Base	9,896	9,899		
1	9,847	9,851	-49	-48
2	9,928	9,930	32	31
3	9,555	9,557	-340	-342
4	10,134	10,140	239	241
5	9,824	9,827	-72	-73
6	9,961	9,966	65	67
7	9,872	9,875	-24	-25
8	9,923	9,926	27	27
9	9,967	9,979	71	80
10	7,501	7,487	-2,395	-2,412
11	10,317	10,328	421	429

What should the interplay be between clustering and replicating portfolios?

In recent years, replicating portfolio techniques have become popular avenues for addressing model efficiency concerns. In our view, this is something of a historical accident: Techniques that are a plausible solution to a different problem (communicating with people who do not understand and/or do not care about insurance liabilities) were, out of convenience, pressed into service to address a different concern (model efficiency) to which they are less well suited. Replicating portfolio techniques require a fairly large number of scenarios to be run with a large model; furthermore, for any sensitivity in the liability assumptions, another set of such scenarios must be rerun.

Our preference for cluster modeling techniques, however, may not be universally shared. Some companies may rely on replicating portfolios for most work where model efficiency is a concern, using cluster models, if at all, only for a quick assessment of the effect of liability assumption changes. Others will use cluster modeling for most work where model efficiency is a concern, using replicating portfolios, if at all, only for communication with investment personnel or where corporate staff wishes to assess liability results under different scenarios without using liability modeling software.

It should be noted that cluster modeling can enhance replicating portfolio techniques. Results from a cluster model for a large number of economic scenarios can be better as input to the replicating portfolio procedure than results from a full model against a smaller number of scenarios, as more data points should result in better replication fit. ■