SOCIETY OF ACTUARIES
PREDICTIVE ANALYTICS AND FUTURISM SECTION

PREDICTIVE ANALYTICS AND FUTURISM
APRIL 2020

Like Peas in a Pod: Ideas in Cluster Analysis

By Michael Niemerg

Cluster analysis is easy to grasp conceptually. You simply "group like things together." The fundamental algorithms like k-means and hierarchical clustering are also relatively easy to understand and don't require much background besides a little understanding of algebra. Despite such an intuitive premise, things can quickly become sophisticated. To that end, let's explore some extensions of cluster analysis beyond the vanilla approaches to understand ways we can improve our analysis and to get a broader sense of what's possible.

First, let's define more clearly what cluster analysis is. Cluster analysis is an unsupervised data-mining technique used to assign data into groupings whereby observations within a grouping (i.e., cluster) will be similar to each other. It is unsupervised because the algorithm does not have access to any label or response information when training. Instead, the model learns the relationships from the data itself without having access to any sort of "correct" answer (usually—we'll come back to this point).

Without a correct answer to judge the results by, how do we determine what makes a good cluster in the first place? This turns out to be a surprisingly tricky question. We are trying to create groupings that exhibit similar characteristics. Two natural consequences of this are that observations within a cluster should be similar to each other and observations within different clusters should be less similar to each other. While there are a variety of evaluation metrics in cluster analysis, most are ways of measuring and comparing these two ideas. The silhouette score is an illustrative example. For each data point, it measures the distance between the current data point and all the other observations within the same cluster and compares that to the average distance between the current data point and all observations for the nearest neighboring cluster. The closer the current data point is to the other observations within its own cluster center, and the farther away it is from the nearest cluster's observations, the better.

While in cluster analysis you don't use labels in training your model (with exceptions as noted later), you can be in the lucky situation of having access to a cluster assignment label to evaluate the model. If this is the case, there are a few methods that can evaluate cluster quality, many of which are closely related to the types of error metrics you would look at for classification tasks. Even if you don't have access to true cluster labels, another way to use this idea, called supervised clustering, is by using proxy features that you think the cluster model should be able to do a good job of classifying. For instance, you might want to group people with similar health profiles together. If you know what their health care claim costs are, you might be able to use this as a proxy of health.

Now that we've established some concepts behind cluster evaluation, we'll discuss some ideas that might be useful in practice. The ideas that follow are presented roughly in order of practicality.

DATA PREPROCESSING

Modeling is always about the data and its representation. Data is the foundation upon which everything rests. One of the first things to try, even before creating a model, is to make sure you preprocess your data in a meaningful way. Because clustering relies on the distance between features, you will often want to normalize all the data so that it is all represented on a similar scale. One way to do that is to standardize the features to be

within the range [-1,1]. This should effectively give every feature in your data set an equal "vote" in the model-building process. Without normalization, features with large absolute values will be overweighted. You might also do the opposite and transform the scales of certain columns to have more influence in the clustering algorithm. In adjusting the weights, you can express a preference for how large of a vote each feature gets in the model.

Another step that is beneficial is to perform label encoding for categorical features, i.e., represent your categories as numbers. To do this, you need to be a little thoughtful and treat ordinal and categorical data differently. If you have three distinct categories and you assign them to values 1, 2, and 3, then the clustering algorithm will treat two records coded with 1 and 2 as more similar than two records coded with 1 and 3. This might make sense if your data is naturally ordinal. If not, you might try one-hot encoding, where you create a single indicator feature for each category in the original feature.

Lastly, you can perform impact encoding. The idea here is that you replace each categorical value in a feature with a numerical value derived from some outcome of interest. For instance, if you have ZIP code as a feature it will have high cardinality and may not be useful in a model—it takes on too many values, mostly non-credible. You could instead create a feature that encodes the average income, mortality rate, or per member per month (PMPM) health care claim cost within that ZIP code.

One possible way to improve clustering model results is simply by trying out different model types.

ALGORITHMIC POTPOURRI

One possible way to improve clustering model results is simply by trying out different model types. A myriad of different algorithms exist out there for clustering—literally hundreds. Yet, just like in the supervised learning case, the number of really useful algorithms is much smaller. K-means and hierarchical clustering are the obvious choices. Gaussian mixture models, density-based spatial clustering of applications with noise (DBSCAN), spectral clustering, and clustering large applications (CLARA) are some more common alternatives. To select the right algorithm there are two options—learn about the differences between them and carefully select the appropriate algorithm based upon the problem at hand (for instance, k-medoids is more robust to outliers than k-means while spectral clustering is more resistant to noise), or simply try out multiple algorithms and evaluate the results. Given the lack of a ground-truth label, knowing when one technique is performing better than another requires considerable thought. It is not as straightforward as the supervised learning case.

FEATURE AND CLUSTER SELECTION

Irrelevant features in cluster analysis can slow down model training and unnecessarily bloat model size. Worse, they have the potential to degrade model performance. Feature selection can be tricky though and is a bit of a catch-22. If you already know what features matter, you would have already applied that knowledge and wouldn't need to do feature selection. And if you don't, and you lack a ground-truth label, how do you judge whether a feature matters?

Of course, the simplest form of feature selection is simply applying domain understanding. While qualitative, this form of feature selection is highly valuable. We can pick features that likely correlate to behaviors we are interested in and remove redundant, uninteresting, or highly correlated features. Barring that, there are also specialized algorithms that deal in feature selection. For instance, a feature selection algorithm might pick the features that most contribute to cluster compactness.

Intelligently selecting the number of clusters (for algorithms that don't do this automatically) is another way to improve the analysis. Several different metrics are useful, including the gap statistic, average silhouette score, and the elbow method. Conversely, the number of clusters selected may not involve any of these methods and may be informed by the use case. As an example: fifty clusters might be too unwieldy from an implementation perspective but five clusters might not be granular enough to give interesting insights.

CONSENSUS CLUSTERING

Consensus clustering is a way to (potentially) improve cluster analysis by creating multiple cluster models and then combining them. Effectively, consensus clustering is a way to create ensembles, similar to how we might for supervised learning (not surprisingly, this approach is also called ensemble clustering). The reason it works is similar to the intuition for ensembles in the supervised case, with a few new wrinkles added in. One of those wrinkles is that different clustering methods divide the feature space in very different ways geometrically, so combining algorithms can be a bit risky and isn't as easy as it might be in the supervised learning case.

There are several ways to create consensus models. Some possible methods include relabeling/voting, a co-occurrence matrix, and median partition methods. My experience is primarily with relabeling/voting methods so I'll expound upon that method for illustrative purposes.

One thing to remember with consensus clustering is that, unlike in supervised learning, there is no response value in clustering,

so the output of a cluster model is an arbitrary label, usually an integer. That is, there is no guarantee in any set of clustering models that the fifth cluster in one model corresponds to the fifth in another model. The label is just a placeholder, so you can't simply look up the fifth cluster in two different models and expect them to have any correspondence with each other.

Relabeling/voting takes care of this problem for us. First, it determines label correspondence between the base models. It does this by checking the labels and switching them so that the labels in all the base models of the ensemble refer to the same cluster. With all our labels on a consistent basis, we can now create the ensemble's prediction with either a hard or soft (probabilistic) classification of cluster membership for each observation. To do this we give each base cluster in the ensemble a vote as to which cluster it thinks the data point should belong to. For instance, if we had five different models with cluster predictions of 2, 2, 2, 3, and 2, then by way of voting we would label this instance as belonging to the second cluster for a hard classification. For a soft classification, we would view this point as having a 0.8 weight or probability of being in the second cluster and a corresponding weight of 0.2 of it being in the third cluster.

PREPROCESSING WITH SHAP VALUES

Let's now take things back a few steps, before applying any algorithms, to when we preprocessed our data. Instead of using the preprocessed data "as is," let's explore a new possibility: Shapley additive explanations (SHAP) values. SHAP values are derived from a classification or regression model. To explain how we can use SHAP values in clustering, let's first dive into what they are.

SHAP values were initially introduced as a model interpretation method, where we allocate how much each feature in a supervised learning task contributes to the difference between the prediction for that particular observation and the overall average. As an example, the SHAP value in a linear regression is actually the coefficient for that feature multiplied by the feature value—because this value in a regression equation tells us how to move that record's prediction away from the overall average, conditional on all the other features. Things get more complicated in the nonlinear case of a gradient-boosted machine or random forest. That is where SHAP values really add insight. SHAP values are extremely useful, so if this has piqued your interest, I highly suggest you read more about them (for instance, see Lundberg 2017).

So how can we use SHAP values in clustering? Well, because SHAP values deconstruct each individual prediction into the contribution from each feature, a data set with n observations and m features would generate a matrix of n x m SHAP values. This means that we have a new matrix that can effectively replace every feature in our data set with a new feature for our clustering model. Why is this interesting? Because it puts everything on a consistent interpretation and scale. It can also help us preprocess our data in desirable ways when dealing with categorical data, especially if the feature has high cardinality (in a way similar to impact encoding). Additionally, to the extent that the supervised model's response corresponds to the types of behaviors you wish to use your cluster model on, it could result in tighter clusters.

The clear challenge with this approach is the requirement to first build a supervised model from which to get the SHAP values. This is a luxury that we often don't have, but if we do, we should exploit it.

SEMI-SUPERVISED CLUSTERING

There has been promising research on incorporating outside knowledge into the clustering process. Semi-supervised clustering is an approach that can be used either when we have partially labeled data or when we want to enforce prior knowledge into the model. Typically, this prior knowledge is represented by specifying a constraint that two observations must be in the same cluster or conversely that two observations can't be in the same cluster. Semi-supervised clustering can be particularly useful when labeling instances is expensive or when you have a strong desire to enforce constraints based upon the business context of the model. How this works is that the algorithm takes user-provided input in terms of labels or constraints and accounts for it in the model-building process. As an example, imagine building a clustering model using age bands. You might want to enforce a constraint that records with younger age bands should not be in the same cluster as records with older age bands while still desiring to use age as a way to measure similarity between records.

CONCLUSION

In many clustering problems, simple approaches like k-means can provide "good enough" results, and in my anecdotal experience they often provide strong baselines that can be hard to improve upon. However, by digging a little deeper you may find ways to take your results to the next level.

Although clustering is not always as clear-cut as classification or regression problems, it doesn't mean that all answers are equally good. More often than not, in the real world you will be creating clusters as a tool to achieve a goal, and a good clustering model should be measured by how well it helps you achieve that goal. There may not be a way to unambiguously measure whether or not you have "the best" model, but the end goal should inform all your modeling decisions. ■

Michael Niemerg, FSA, MAAA, is predictive modeling manager at Milliman IntelliScript. He can be contacted at *michael.niemerg@milliman.com*.

BIBLIOGRAPHY

Aggarwal C., Reddy C. Data Clustering: Algorithms and Applications, 1st Edition (2013).

Awasthi, P., Zadeh, R.B., Supervised Clustering (2010). NIPS'10 Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1.

Bair, E., Semi-supervised clustering methods (2013). Wiley Interdisciplinary Reviews: Computational Statistics 5(5):349-361.

Lundberg S., Lee S., A unified approach to interpreting model predictions (2017). Advances in Neural Information Processing Systems, 4765-4774.

Get Plugged in–New InsurTech Partnership

The SOA and Plug and Play relationship will allow InsurTech start-ups to validate their technology and modeling processes with actuaries. In turn, SOA members will have an exclusive look inside the world of emerging technologies. These efforts will help with the development of fair and financially sound insurance products to better serve consumers.

The strategic partnership with Plug and Play demonstrates the SOA's commitment to providing its members with dynamic learning experiences, rewarding volunteer opportunities, and collaborative events where they can learn from the experiences and ideas of peers around the world. Through this partnership SOA members and start-ups can share best practices and advance ideas for the benefit of the insurance industry, regulators and the public. The SOA and Plug and Play officially announce this partnership to support an exchange of knowledge between actuaries and start-ups. ■

Toiling in the Actuarial Vineyards: Accelerating Traditional Experience Analysis With GLM Trees

By Philip Adams

The ever-increasing volume and diversity of data available to an actuary is both exciting and terrifying. Exciting because of the amazing and unexpected findings waiting to be discovered, and terrifying because of the drudgery and disappointment to be endured along the way. And time is of the essence.

Technology rewards us with a mess of data and helps us sift through it with numerous open-source solutions.

BRIEF HISTORY OF ANALYSIS

Actuaries have traditionally analyzed data manually. Even with technology, an actuary is manually reviewing, evaluating and judging the fitness of data for its intended purposes. Generally, the traditional approach often follows a similar recipe for mortality analysis:

1. Look through the dimensions of the data to find statistically significant factors driving mortality.
   a. One dimension is considered at a time.
   b. Optionally, two or more can be considered at the same time.
   c. Filters can be introduced at any point.
2. Develop a set of factors for that dimension/combination of dimensions.
3. Do one of the following:
   a. Adjust experience with the new factors.
   b. Don't adjust.
4. Repeat 1–3 as needed.
5. Finally, check for reasonableness and fit.

The recipe works, but not without shortcomings.

1. Sifting through combinations of factors is labor intensive, and it is probable that some important factors and combinations will be missed.

2. When to stop looking for factors is left to judgment. The risk is that the modeler could either stop too early and miss important factors, or stop too late and waste valuable time.

3. Factors are not fit simultaneously. If there are dependencies, estimates may change when using its related factors. For example, if smoking status isn't distributed identically by gender, the Smoker factor may pick up some signal belonging to Gender.

4. It can happen that different types of models suit different parts of the data. For example, an interaction of dimensions in one section of the data might be unneeded in another.

The manual recipe does not scale with volume. Automated approaches are needed to fill the gap. A variety of predictive models can offer relief, including GLMs (generalized linear models), GAMs (generalized additive models), decision trees/forests, elastic nets, gradient boosting, etc.

This paper introduces a hybrid approach, the GLM tree, to an actuarial audience.

While GLM trees are not the only tool that can deal with these issues, they have the advantage of being intuitive and easy to explain. This cannot be said of random forest, elastic net regression, or boosting methods. As you will see, they get an actuary to an answer more efficiently than other methods all while remaining explainable.

In the early days of my mortality modeling, GLMs and GAMs were the best available tools.

DESSERT BEFORE DINNER

In the fall of 2018, the Society of Actuaries issued a challenge whereby the Individual Life Experience Committee (ILEC) released experience data from 2009 to 2015 and invited parties to submit the best data analysis solution. The top three entrants were awarded fabulous cash prizes.

Consider the following findings for term experience:

1. The mortality experience for term business having at most one preferred class (two-class) deteriorated significantly over the study period.

2. Experience for 3- and 4-risk class systems improved significantly.

3. Term experience with face amounts below $100,000 deteriorated.

GLM TREES

In the early days of my mortality modeling, GLMs and GAMs were the best available tools. In some cases, today's algorithms had not been invented yet (or at least not revealed to a wide audience).

Model fitting with GAMs can be a chore. Stepwise feature selection as implemented for GLMs does not work with splines. Instead, I searched for methods for automated feature/interaction detection. One can find many approaches, including chi-square automatic interaction detection (CHAID) and other decision tree types. CHAID had many of the features I wanted yet appeared to be incompatible with the A/E and qx analyses.

In 2008, Achim Zeileis, Torsten Hothorn and Kurt Hornik introduced a rigorous theoretical framework built on research into combining parametric models such as GLMs with decision tree models. The algorithm relies heavily on **parameter fluctuation tests** (method to detect whether there is unmodeled variation in the residuals).

If you were building a model by hand, you might check how well the model fits the data by examining one or more dimensions. For example, if you fit a constant percentage and then look at fluctuations of actual-to-model mortality, you might note some residual variation for some dimensions. The visual variation may look like noise, or it might show some patterns in the actual-to-model mortality.

In the case of model-based recursive partitioning, that last part about residual variation looking like noise is the key. For an ordered dimension (like age), under the null hypothesis where the residuals are independent, identically distributed random variables, the running sum of the residuals is distributed as a Brownian bridge (a random walk starting and ending at 0), subject to appropriate scaling. In the unordered case (e.g., categories), a Chi-square goodness-of-fit test is applied to the residuals for that dimension.

The dimension with the most variation wins. The algorithm searches for the best binary subdivision for that dimension. For an ordered dimension, each break point in the data is tested sequentially. For an unordered dimension having n levels, the algorithm tries all of the binary subdivisions of the dimensions. There are 2^{n-1}-1 possible subset breaks to check. In both cases, the partition that maximizes the likelihood the most wins.

Now that the data have been broken into subsets, the algorithm starts the process over on the smaller pieces. Eventually, the procedure stops, either because there is nothing to improve or the analyst specified a stopping rule.

A CONCRETE EXAMPLE

Since mortality trend is among the most important topics for life insurance, I attempted several model types, both regression and tree-based. As it happens, only GLM trees were able to discover what subsets of the data had meaningful differences in mortality levels and trends.

I used the Poisson regression model:

$$Number\ of\ Deaths \sim \beta\_0+\beta\_1\ Experience\ Year+log(Expected\ Claims\ 2015VBT)$$

The partitioning variables are everything else. Because some variables are insurance plan specific, I carried out the analysis separately for term and perm products. This example follows the algorithm/results for the term analysis. Since this is an exploratory exercise, no training/test split is performed.

Note that I emphasize intuitive understanding in my example and not technical understanding. Therefore, I am combining the parameter fluctuation test with the subset testing.

The algorithm starts by fitting the regression model to the data (Fig. 1). Mean trend is semi-significantly positive.

There are 12 variables to test. We demonstrate the first three and stop with face amount band. The first variable is gender. In Figure 2, we see the models for a potential split. While there is unmodeled trend variation for females, there is less for males. The mean mortality level has small variation. The second is age basis. In Figure 3, this appears to be a promising split, with ANB showing deterioration yet small variation for mean mortality.

Figure 1
First Regression Model on Term Data

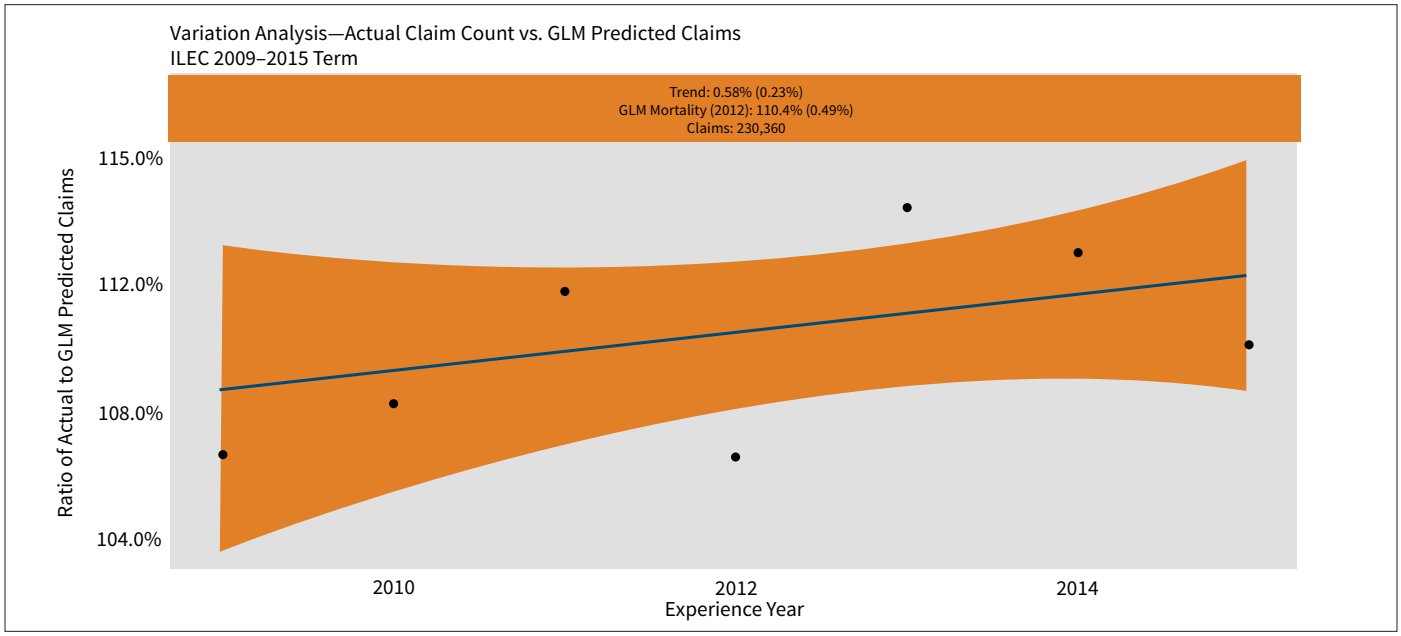


Figure 2
Candidate Models for Gender Split

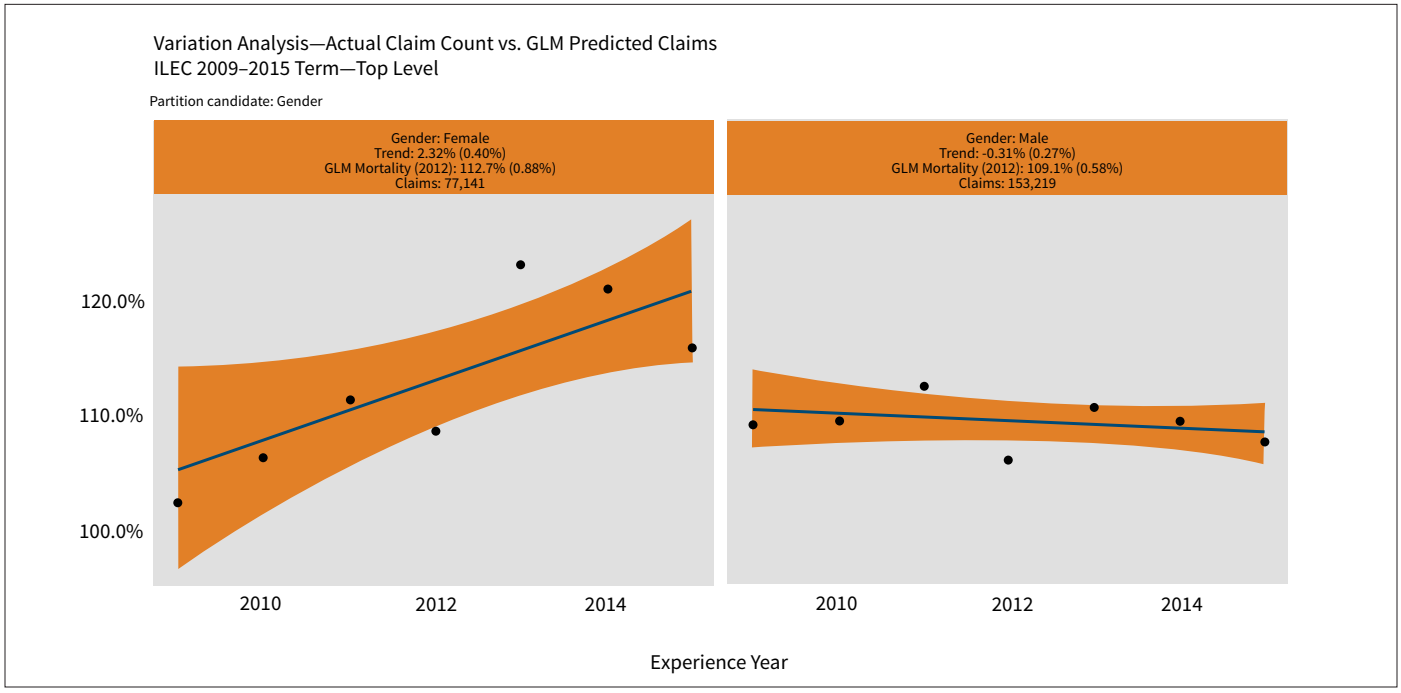


Figure 3
Candidate Models for Age Basis

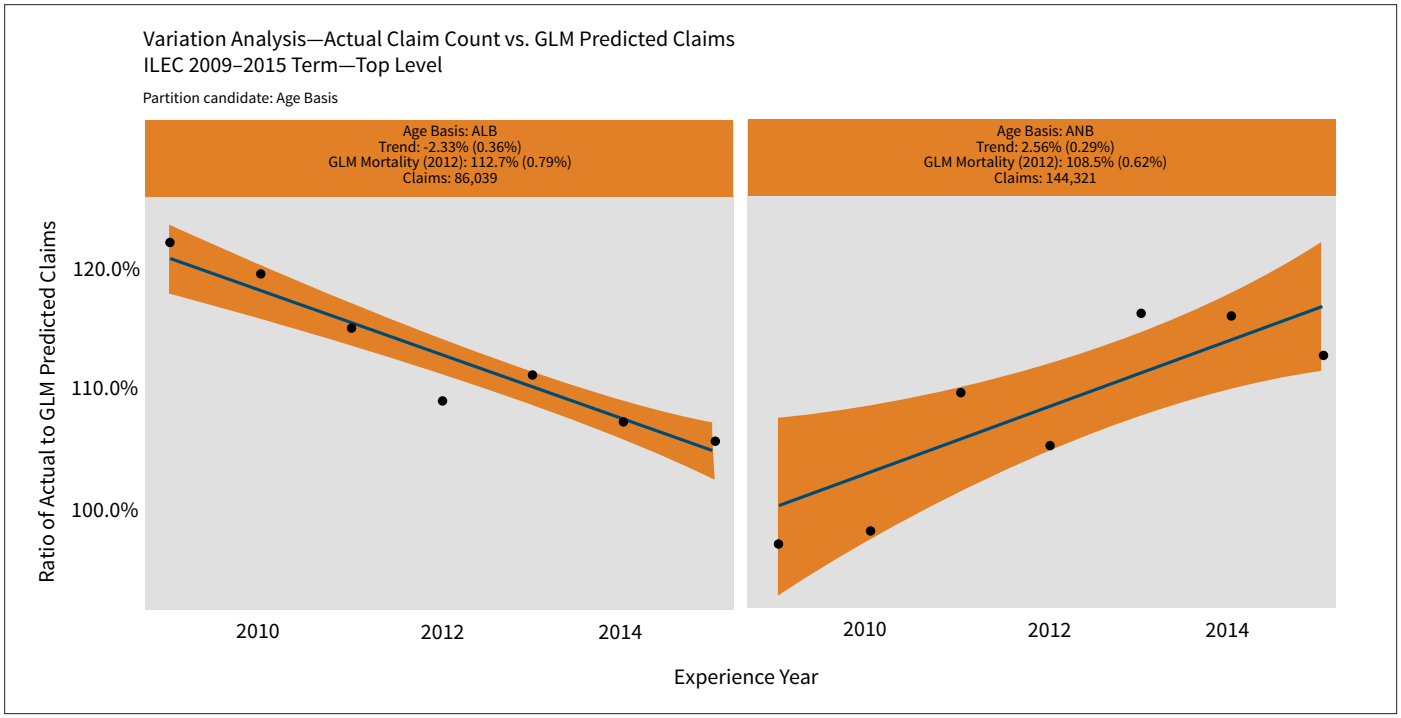
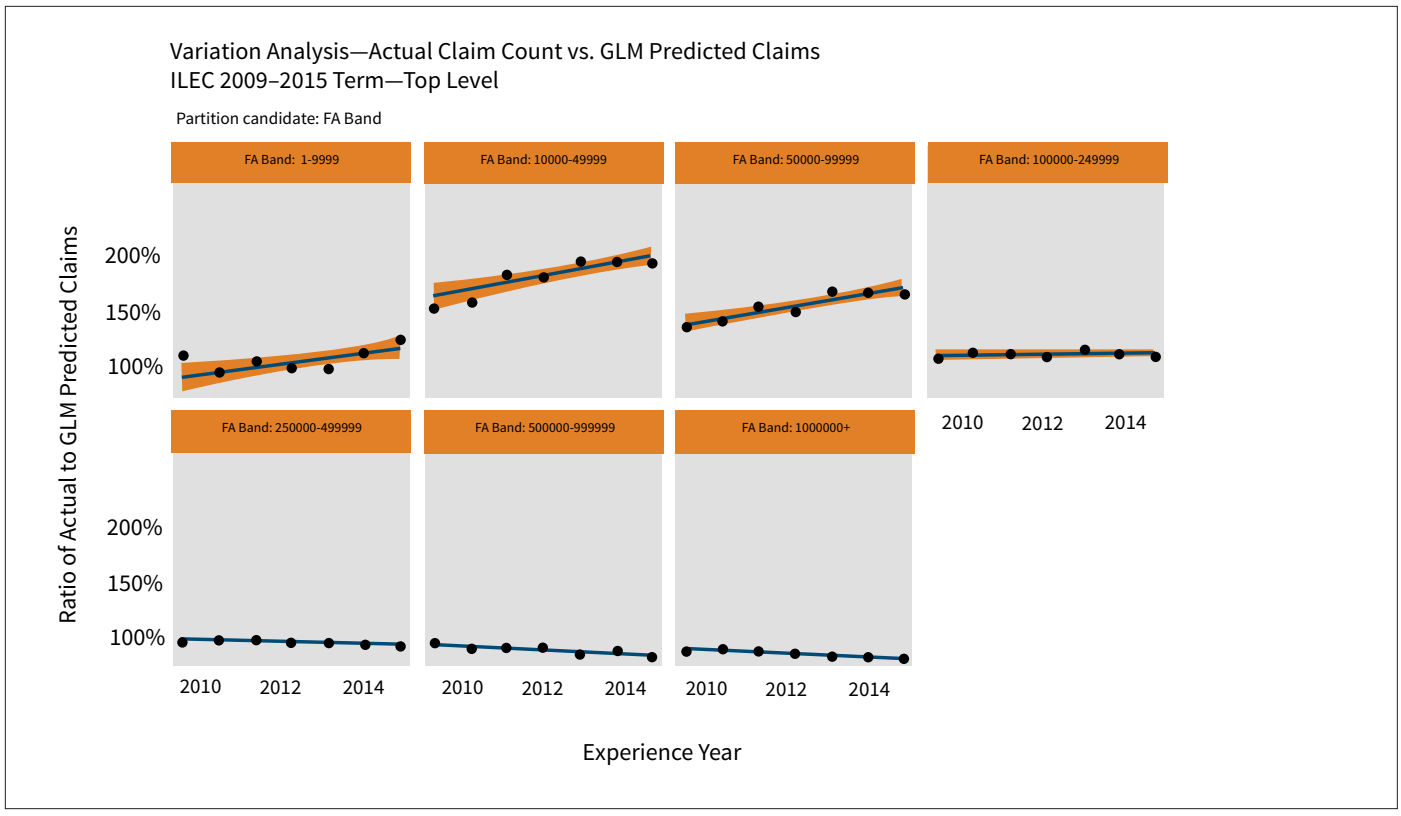


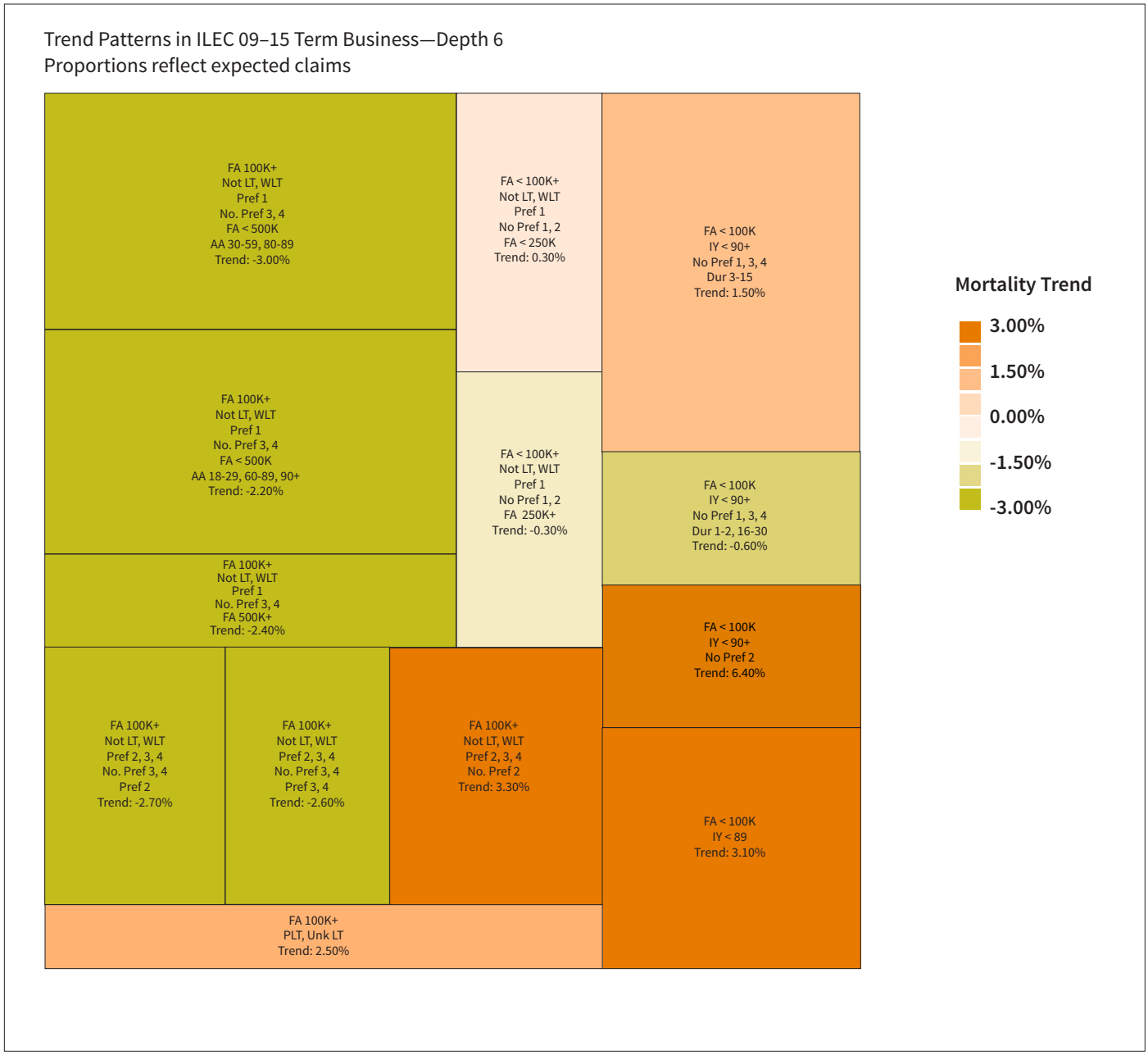
Figure 4
Candidate Models for Face Amount Band



In Figure 4, there is substantial variation for both trend and mean mortality with increasing face amount band. Trend ranges from positive to negative with increasing face amount, and with the exception of face amounts under 10,000, mean mortality declines with increasing face amount. After testing the other nine variables, face amount was the first dimension along which to split the data. Because it has seven levels, there are 63 splits. To lessen computation, face amount band is treated as an ordered factor, reducing this to six splits. The chosen split was at 100,000.

The algorithm then builds a tree recursively defined by split conditions with a GLM at each node/leaf of the tree. The partykit package expresses the results as a traditional tree. To get around the limitations of the default output, I expressed the results as a tree map as in Figure 5. The minimum node size was 10,000 expected claims.

Figure 5
Tree Map of GLM Output for Trend



If you let your eyes wander, some findings emerge:

1. Face amounts less than 100,000 exhibited deterioration (the right third of the square). In the instance of 2-class preferred systems, deterioration was 6.4 percent (SE 0.42 percent) on average per year. One possible exception was the light green block, but this is statistically not significant (SE 0.44 percent).

2. Face amounts 100,000 and higher witnessed improvement in general, with two exceptions.

    a. The lower left corner contains post-level term and unknown level-term business. The trend here is potentially contaminated with slope misalignment. The net deterioration is 2.5 percent (SE 0.28 percent) per year on average.

    b. The angry red block above it is residual standard of 2-class preferred systems including the non-level term and within level term business. Within level term dominates the block. There was substantial deterioration of 3.3 percent on average per year (SE 0.43 percent).

3. Right next to the angry red block is a very green block that contains residual standard of 3- and 4-class preferred systems. For these lives, there was substantial improvement on average of 2.6 percent per year (SE 0.32 percent), partly offsetting the deterioration of the 2-class systems.

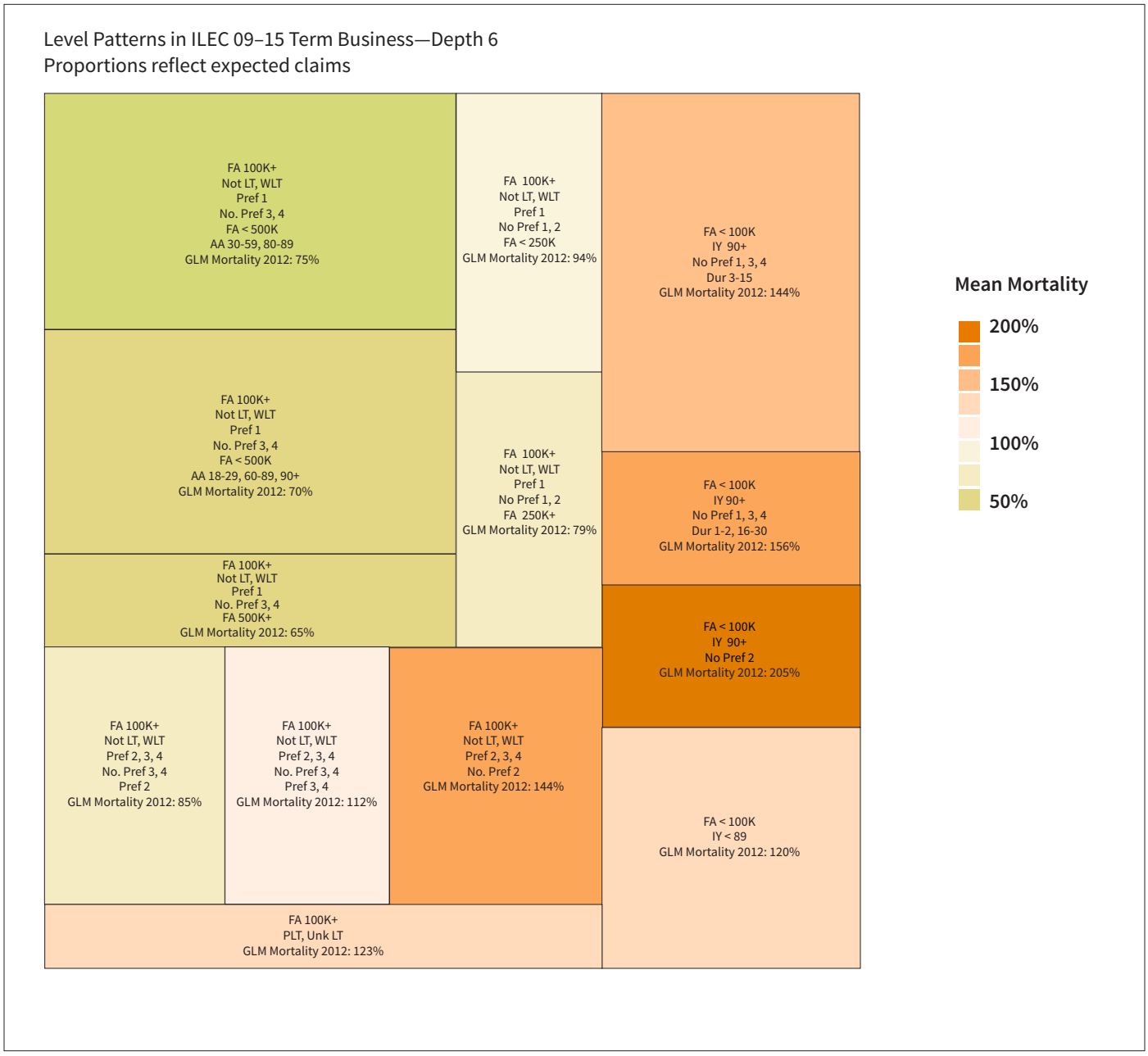
The same plot can be had for adjusted mean mortality (centering at 2012). In Figure 6, we see that many relationships are as we expect: higher face amounts have better mortality, better preferred has better mortality, post-level term has worse mortality. Standing out is the high mortality for 2-class preferred systems with face < $100,000.

WHAT ABOUT PERM? ANALYSIS BY AMOUNT?

Perm has been omitted from this paper for brevity. The claim count is nearly 10x as large as for term, with much longer issue year horizons and more insurance plan types.

Analysis by amount has a few differences. For parameters, the GLM family is changed to a Tweedie distribution with parameter 1.2. Minimum size depends on the specified weighting vector. The minimum size is set to 10,000 * $50,000, or $500,000,000, and the maximum depth tree depth is set to six. All but one of the resulting leaves has at least 10,000 claims.

Figure 6
Tree Map of GLM Output for Mean Mortality



LIMITATIONS

The intent of the analysis was to unravel some of the riddles around trend in the ILEC 2009–2015 dataset. Since the minimum claim size per node was so large, it is likely that more insights can be gained by allowing the algorithm to drill deeper or changing the GLM model used for each node.

I encountered a few problems when applying the GLM tree function in the partykit package to the data. The first was data sparsity with depth; there must be enough diversity in the data to support fitting a GLM within any proposed node. An early attempt at GLM trees was to have the GLM model template be an interaction between attained age group and duration group. The result would be an optimal subdivision of the data with a custom select-and-ultimate mortality table for each node of the tree. However, not every combination of age and duration will be available in every subset, or there may be no claims.

The second was weighting. The GLM tree function applies the same weights parameter to the GLM fitting and the parameter fluctuation tests. This is problematic when using an offset in a Poisson model. If a weights vector is specified, the resulting GLMs will be skewed. If no weight is specified, the individual GLMs are fine, but the parameter fluctuation tests will weight equally each row of the data. Thus, it was necessary to customize the code to allow separate weights for the GLM fitting steps.

The third was lack of accommodation for splines. I had attempted to build a "GAM tree" function where the models within each node were GAMs. Adapting the spline parameters to parameter fluctuation tests proved challenging, and I ultimately set the task aside for later research.

FUTURE DIRECTIONS

I offer GLM trees as a valuable tool that helps to bridge the gap between the needs of actuarial analysis and the potential of data science methods. As an exploratory tool, it can illuminate structures in datasets. Using the typical recipe with training and test data, it can be applied as a predictive model. It can also be a point of departure for additional analysis, such as exposing where to focus further analysis or as a point of departure for more sophisticated models. ■

Philip Adams, FSA, CERA, is an AVP and actuary, Biometric Research, for Munich American Reassurance Company. He can be contacted at *padams@munichre.com.*

Get Better Acquainted With Your Known Unknowns

By Dan Kim and Boyang Meng

Uncertainty of a predictive model is a fact of life that many insurers could be overlooking at their peril without a framework for assessing it.

Predictive analytics have become increasingly commonly used across the U.S. life insurance industry in areas such as mortality and policyholder experience analysis, automated pricing and underwriting, in-force management and fraud/claims analytics.

Predictive models are usually better at detecting signals from a large dataset and are more likely to be precise in making predictions compared to traditional approaches such as a tabular or one-way analysis. For example, U.S. life insurers often use actual-to-expected ratios in a tabular form to develop best estimate assumptions. Predictive models, like generalized linear models or tree models, may improve the traditional models, but the danger is that models become regarded as perfect and a silver bullet for decision making within the business.

For, as the renowned statistician George E. P. Box once said: "All models are wrong, but some are useful."

ZERO ERROR IS A PIPEDREAM

What he was referring to is that while any predictive model will (or should) be built to minimize the generalized error, the error will never practically be zero. So, the question insurers need to think about is how much the future will emerge differently from their predictions. To answer this question, having a framework to determine a level of model uncertainty can be invaluable.

Such understanding matters, because it can be fundamental to things such as whether an insurance applicant that may be below the underwriting criteria is falsely approved, how much capital and reserves companies need to hold, the chances of fraudulent insurance claims making it through the vetting process, and decisions about risk transfer.

FRAMEWORK GOALS

There are a few key issues to address in such a framework. Actuaries usually ask, "How credible is the data?" Instead, we can expand this into more specific, targeted questions. What is the confidence level of the model's average predictions? How significantly can reality differ from these predictions? The aim is to determine the degree to which your predictions may be uncertain so that you can augment your business strategy to minimize the impact from such uncertainty.

We can apply those questions to the simplest of predictive models—the outcomes of tossing a coin. Let's say 10 tosses of this coin have yielded six heads and four tails. Without knowledge of the fairness of this coin, what could the range of outcomes for 20 tosses be? What about 1,000 tosses?

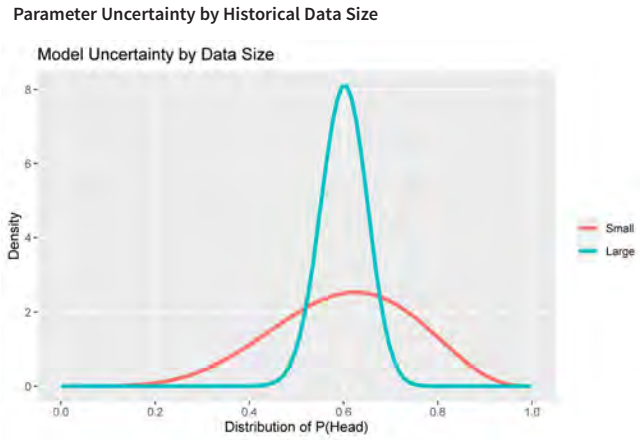
First, we must develop an assumption or a predictive model on the fairness of the coin. Then, we need to quantify the uncertainty of the model. Finally, given all these, we can understand the range and probability distribution of possible outcomes from more tosses.

An assumption about the fairness of the coin can be illustrated as a probability of showing a head. So, a reasonable assumption, based on experience, is 60 percent. We can estimate the uncertainty of that probability using binomial distribution as shown below to give an estimated standard deviation of 15.5 percent.

$$Standard\ deviation\ of\ P(Head) = \sqrt{\frac{P(Head) \times \{1 - P(Head)\}}{Number\ of\ Observations}} = \sqrt{\frac{0.6 \times 0.4}{10}} \approx 0.155$$

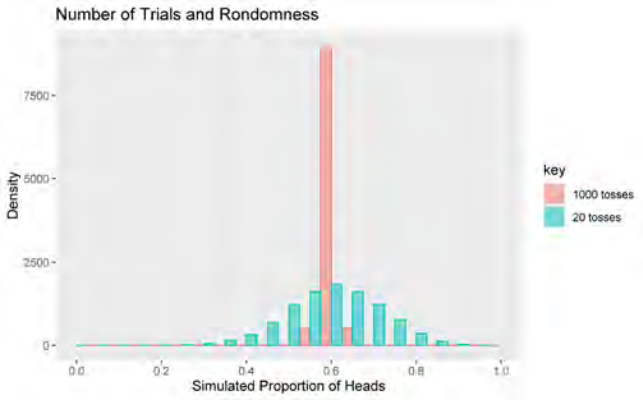
If we had more observations, for example, 60 heads from 100 observations, our belief about the model would be stronger. The more data that's available, the less the model uncertainty (see Figure 1).

Figure 1
Probability of a Head (P(Head)) Density Distribution of the Coin and Credibility



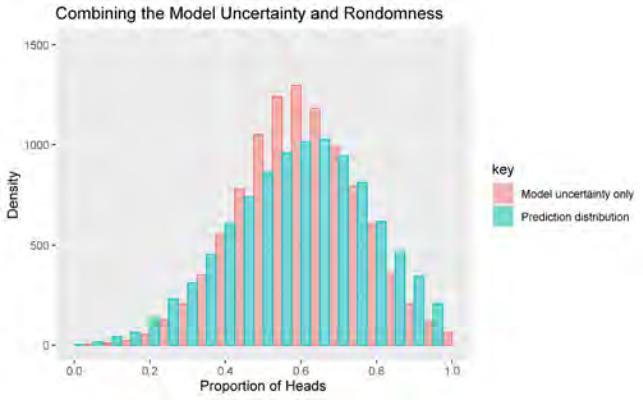
Some level of random noise would occur even if we have a model with a high degree of confidence in the underlying response. In our coin example, even if we are sure about the fairness of the coin, 20 tosses are expected to show a range of the number of heads due to random volatility. We may see 12 heads, but 11 or 13 would also be likely given a coin with 60 percent of probability of heads. More tosses would ensure the outcome will be close to 60 percent of heads (see Figure 2). This measure can be particularly relevant when making predictions for a small number of cases or exposures.

Figure 2
Impact of Random Noise



The overall uncertainty of the prediction is derived from both model uncertainty and randomness, as depicted in Figure 3. Even if the historical data indicated there is a 60 percent probability of heads with a high confidence, the future may surprise us (i.e., 10 percent or 95 percent heads).

Figure 3
The Prediction Uncertainty Combines Parameter Uncertainty and Randomness. (Based on 10,000 Simulations to Create the Density)

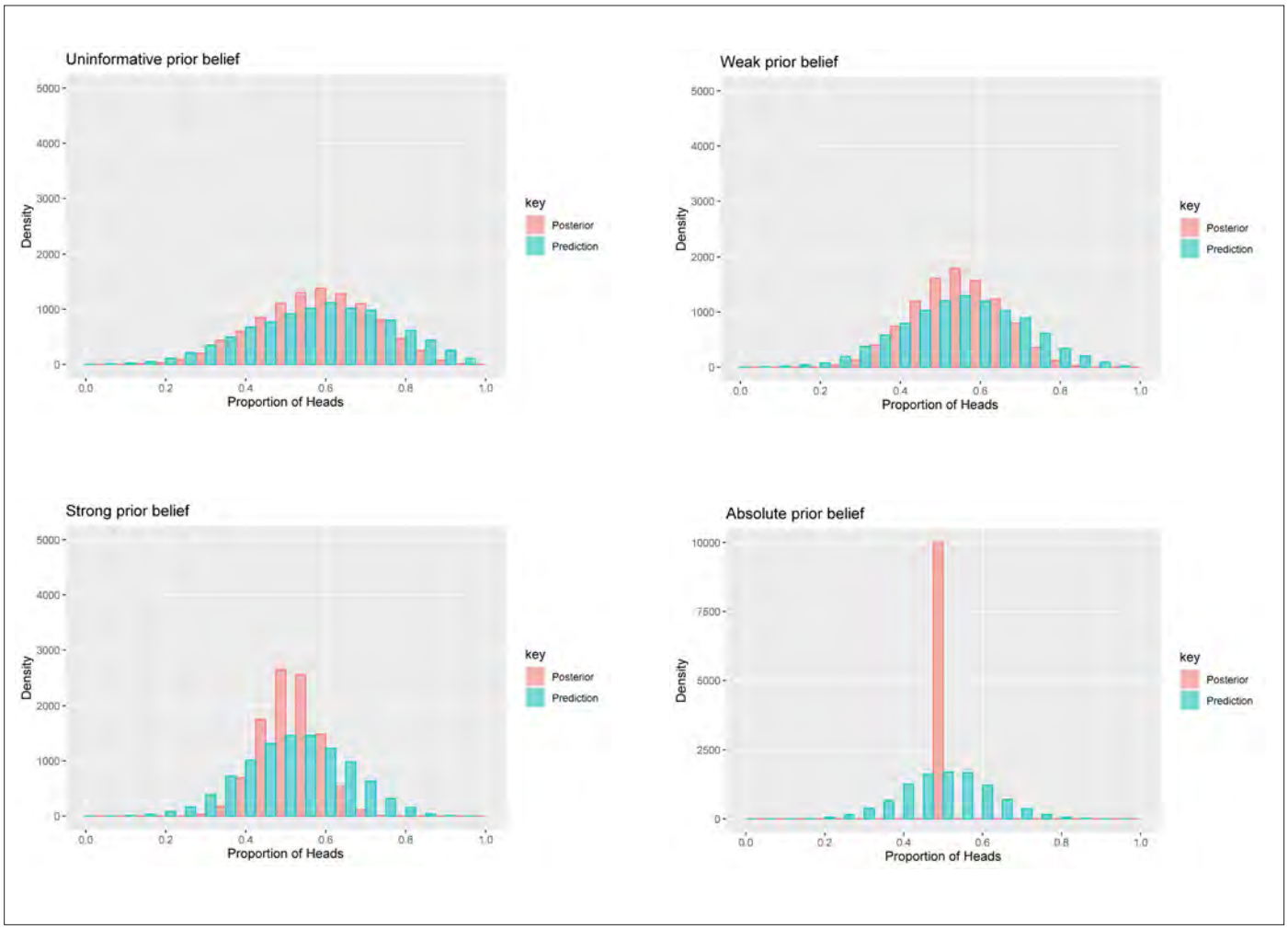


BAYESIAN APPROACH

So far, we have seen a frequentist view by developing a model on uncertainty relying solely on the historical data. Another approach is to use a Bayesian method that combines historical data and judgment (or a belief in the prior distribution). Depending on the prior belief, the view of the model and prediction uncertainties would differ, as illustrated in four charts in Figure 4. If there had been no or little prior knowledge, the posterior distribution would be more dispersed and fit to data similar to the frequentist view as shown in the chart indicated as uninformative or weak prior belief. A stronger prior belief that the coin is fair would produce a posterior distribution closer to 50 percent with less dispersion, putting less weight to the historical data. The selection of the prior distribution relies on qualitative subject matter expertise and intuition. This is a great way to combine the insights and domain expertise with the historical data, especially when the data is scarce.

Some level of random noise would occur even if we have a model with a high degree of confidence in the underlying response. In our coin example, even if we are sure about the fairness of the coin, 20 tosses are expected to show a range of the number of heads due to random volatility.

Figure 4
Model Uncertainty Combines Parameter Uncertainty and Randomness. Posterior and Prediction Distributions are Developed Based on 10,000 Simulations.



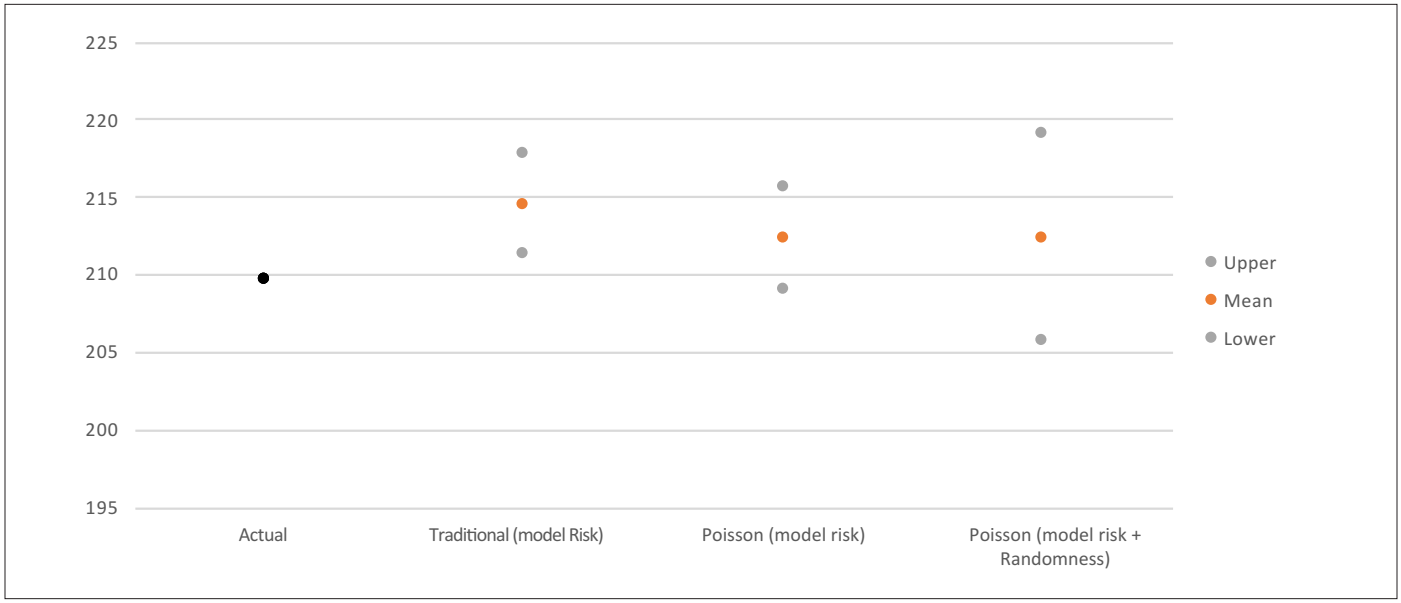
CASE STUDY: MORTALITY ANALYSIS

With this theory in hand, we can now look to the operations of a life insurer, a much more complex problem than coin tosses. We will apply the same framework illustrated using coins, to a mortality study, which is one of key assumptions for life insurers.

We first developed actual-to-expected (A/E) ratios on historical experience based on a classical approach. The A/E ratios were developed in a tabular form by gender, smoking status and substandard. As a comparison, we performed a Poisson regression, which is one of the popular generalized linear models. We used the Bayesian approach, assuming we are quite confident with the base expected mortality table. The prior distribution is assumed to be 100 percent of the table with 5 percent of standard deviation.

We then developed predictions from a separate hold-out dataset, which was not used to develop the models. We then developed the mean and the 95 percent intervals of the predictions of the hold-out data. The predictions of the mortality rate (per 10,000 lives) of the group is shown in the chart in Figure 5. The actual mortality rate of the hold-out data set was 210 per 10,000 while the traditional model predicted 215. The Bayesian-Poisson model prediction of 212 was closer to the actual compared to the traditional model that overfit to the training data set. The lower bound of confidence interval of the traditional model was 211, which is still higher than the actual (the second chart). A better way to view this is to compare the actual to the prediction interval. The Bayesian-Poisson model expects the actual mortality would be between 206 and 219 with 95 percent confidence, which includes the actual mortality (the fourth chart). The actual data isn't a surprise given the Bayesian analysis.

Figure 5
Predictions (Mean, and Upper and Lower Bounds Based on 95 Percent Confidence Level)



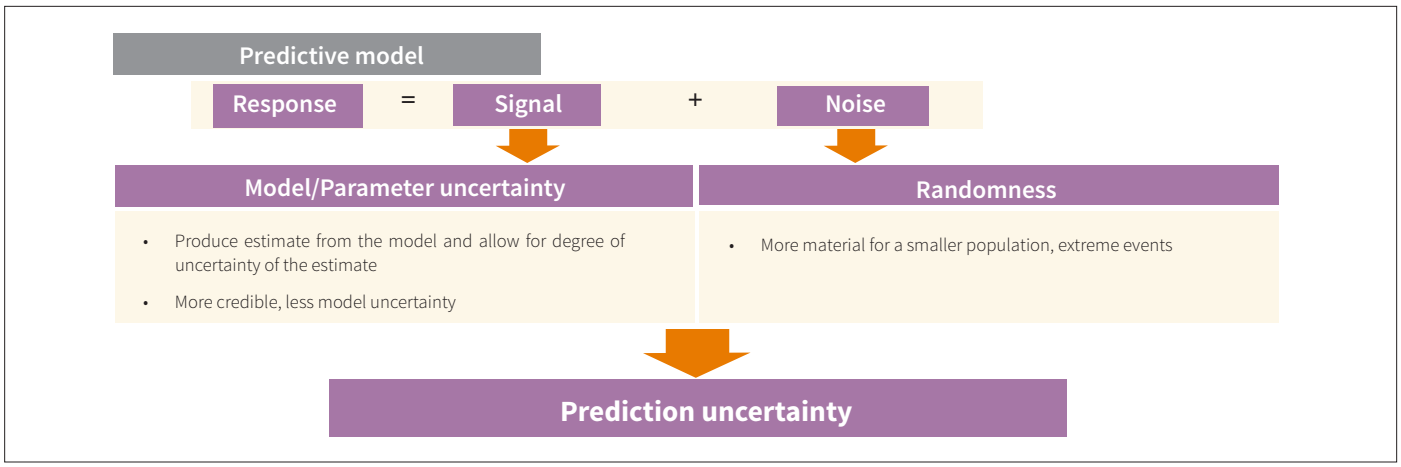
From this we can see that the Bayesian approach is less prone to overfitting, and this was also the case when we reviewed the results in a more granular subgroup level with less credible data. Additionally, the Bayesian framework allowed us to combine historical data and actuarial judgment and helps us directly address the question of how the model is uncertain through its posterior distribution. We could add randomness to the posterior distribution to create the prediction distribution. We believe the Bayesian approach is one of the most effective quantitative analysis tools to inform how the model can deviate from reality and support risk management strategy.

THE SIGNAL AND THE NOISE

If insurers are truly going to get to grips with the known unknowns in their businesses, such as inevitable variances between predictions and actual experience, their use of predictive models needs to accommodate the inherent prediction uncertainty.

Figure 6 illustrates a framework for doing that and, consequently, avoiding the tendency to accept models without a critical thought. When we develop a prediction model, we try to remove the noise and capture the signal. Ultimately, any model is a generalization of the complex and seemingly chaotic reality; but still useful as an approximation. With that understanding of the model in hand, the business should not ignore the noise, which is and always will be part of the reality. The Bayesian framework provides a way to address the uncertainties associated both with determining the model used for capturing the signal and understanding the possible noise (randomness) that would undermine the accuracy of the prediction.

Figure 6
Illustrative Framework for Assessing Prediction Uncertainty



As much as predictive models have improved actuaries' abilities to make more accurate and precise projections and assumptions, our foresight will never be 20/20. That much we know, so building our knowledge of the prediction uncertainty in our models is an essential part of fully understanding them and making sound business decisions based on them. ■

Dan Kim is a director in Willis Towers Watson's Insurance Consulting and Technology business, based in Atlanta. He can be contacted at *dan.kim@willistowerswatson.com*

Boyang Meng is a senior consultant in Willis Towers Watson's Insurance Consulting and Technology business, based in Atlanta. He can be contacted at *boyang.meng@willistowerswatson.com*