Paper SAS2388-2018

Robust Principal Component Analysis: Two Analyses in One

Anthony Salis, State Farm Insurance Companies

Zohreh Asgharzadeh and Kyungduck Cha, SAS Institute Inc.

ABSTRACT

Home telematics is a new and growing field in the insurance industry. Whereas traditional rating variables, such as amount of insurance and type of home construction, describe a home by its structure, telematics variables describe how the home is used. Telematics variables are derived from sensors placed in the home, such as thermometers, motion detectors, and smoke detectors. The data from these sensors are summarized in many different ways in order to provide potential model variables. For insurance companies, the ultimate goal is to use these data to predict future insurance losses for a group of homeowner's policies. However, such high-frequency, high-dimensional data are usually full of outliers, so the data require preprocessing. Two objectives of this preprocessing are to determine which homes behave as outliers and to reduce the dimensionality of the predictor variables.

The RPCA procedure in SAS® Visual Data Mining and Machine Learning software uses robust principal component analysis to accomplish these objectives. Principal component analysis (PCA) is a common approach to reducing the dimensionality of a matrix. The "robust" part of this analysis involves splitting the original data matrix into a low-rank matrix and a sparse matrix before performing PCA. The low-rank matrix contains the "normal" portion of the variables, and the sparse matrix contains the "outlier" portion of the variables. You can use both these outputs from PROC RPCA to get a better understanding of the data and to prepare the data for modeling.

INTRODUCTION

State Farm is a leading provider of homeowner's insurance. Because each home is unique, the insurance risk that it presents is also unique. To price the insurance for a particular home, we build a model by using rating variables that describe the unique characteristics of the home. Traditional rating variables, such as amount of insurance and construction type, describe the home according to its structure. However, this does not present a complete picture of the risk of loss; how the home is used is also important. Home telematics devices provide this important information about how the home is used. The homeowner places sensors in the home that record activity such as temperature changes, motion of people in the home, and when the front door is locked or unlocked. We can summarize the data collected over the course of the year in many different ways to create potential model variables. Then we can use these characteristics along with the traditional rating variables to predict future insurance losses for a set of policies.

With the large possible number of sensors and many ways to summarize the data, telematics data can present their own challenges. Such high-frequency and high-dimensional data usually contain many outliers, and preprocessing the data is a necessary step. In this paper, we address two major data challenges: how to determine which homes behave as outliers and how to reduce the dimensionality of the predictor variables.

The first challenge is dealing with outlier observations. As part of normal exploratory data analysis, it is important to understand the data and which data elements might present difficulties in modeling. An outlier has the potential to skew the modeling results, because it has characteristics that are not similar to those of the other homes in the model. Especially with so many new and unknown variables collected by telematics, if you can quickly and accurately assess and remove outlier observations, you will be able to build a starter exploratory model much more efficiently.

The second challenge is reducing the dimensionality of the predictor variables. When you have hundreds of new variables, it can be difficult to sort through them to figure out which ones might be useful in predicting loss. In addition, many of these variables might have strong correlations with one another and thus provide redundant information. Reducing the dimensionality can speed up the process of finding the true predictive power of the data.

The RPCA procedure uses robust principal component analysis to achieve these goals. The outputs from this procedure can provide you with a better understanding of the data and help you prepare the data for modeling. In this paper, we walk you through an example of using home telematics data to quickly build a predictive model, using PROC RPCA to preprocess the data.

DATA

The data that we use in this analysis are simulated and not actual policyholder data. The variables were created in such a way that they would reflect traits similar to those of actual homeowner’s insurance and home telematics data. So although the data are not real, they provide an opportunity to demonstrate the power of the RPCA procedure.

The data set for this analysis consists of 150,000 homes. Each observation in the data set is representative of the characteristics and insurance loss experience of one home in one year. We randomly split the data into training and validation data sets: 60% of the observations belong to the training data set, and 40% belong to the validation data set.

The predictor variables are classified as either traditional variables or telematics variables. Traditional variables include things like amount of insurance, type of construction, square footage of the home, and policy deductible. These are characteristics of the home that insurance companies have traditionally used to rate the homeowner’s policy—that is, to charge an appropriate premium for the risk of loss to the home. These characteristics can describe the home itself (such as construction type and square footage), or they can describe the terms of the insurance policy (such as amount of insurance and deductible). There are 25 such variables in the data set.

In contrast, telematics variables describe how the home is used. Sensors that the homeowner places around the home collect data continuously throughout the year. All this information is summarized in one observation for each home. The data from the sensors are summarized in many different ways in order to give depth to the information that each sensor provides. The telematics data are contained in 954 variables in the data set, for a total of 979 potential model variables.

The target variable is the loss ratio for the home. The loss ratio is the ratio of the amount of loss paid by State Farm to the amount of premium collected on that policy. On an individual basis, 95.6% of homes have a loss ratio of zero, because most homeowners do not experience an insured loss in a given one-year period. Those who do experience a loss generally have a very high loss ratio that overrepresents their actual average risk. This randomness inherent in the insurance process makes it is both impractical and impossible to accurately predict an individual homeowner’s loss in the following year. The goals of this model are to best separate groups of policies that are more or less likely to experience a loss and to better predict the amount of loss for the group as a whole.

OVERVIEW OF RPCA

The RPCA procedure, which is implemented in SAS Visual Data Mining and Machine Learning software, performs robust principal component analysis, taking into account that the data might be noisy or contain outliers. PROC RPCA decomposes the data M into a low-rank matrix, L, and a sparse matrix, S, where L is used to extract the “robust” principal components and S contains the outliers or bad data. This decomposition is done by solving the following optimization problem:

$$\begin{gathered}\text{minimize } \|\mathrm{L}\|_* + \lambda\|\mathrm{S}\|_1 \\ \text{subject to } M = L + S\end{gathered}$$

where $\|L\|_*$ is a nuclear norm of matrix L and $\|S\|_1$ is the $l_1$norm of matrix S (Candès et al. 2011). λ is a tuning parameter that balances the two terms in the preceding objective function. If λ is set to a larger value, S is more sparse. On the other hand, if λ is set to a relatively lower value, L is of a lower rank. Candès et al. show that under some minimal assumptions, if the rank of the low-rank component is not too large and the sparse component is reasonably sparse, setting λ as follows perfectly recovers the low-rank component of the input:

$$\lambda = \frac{1}{\sqrt{n}}$$

Here n is the number of observations in M. Throughout the analysis in this paper, we use the preceding formula to calculate the value of λ, which is the default value that PROC RPCA uses.

Once again, the assumption is that the input matrix M has a genuine low rank that was “polluted” by outliers or noise. Given this assumption, using PROC RPCA as a preprocessor to “clean” the data before doing further analysis can be helpful. Also, we can use PROC RPCA to obtain the principal components. Because in this procedure the singular value decomposition method is applied to the resulting low-rank matrix, the principal components are robust to the outliers and noise. Last but not least, we can use the sparse matrix to identify anomalies in the data. By identifying the significant values in the sparse matrix, we can find the corresponding values in the original data that are behaving as outliers.

The following code illustrates the syntax of PROC RPCA:

```
proc rpca
     data=mycas.home_data /* M matrix */
     center scale /* Standardize variables to help computation */
     decomp=svd /* Perform singular value decomposition on L matrix */
     outlowrank=mycas.lowrank_matrix /* L matrix */
     outsparse=mycas.sparse_matrix /* S matrix */
     ;
   input &inputvars.; /* All 979 potential predictor variables */
   id house_id; /* Observation identifier that is output */
   outdecomp svddiag=mycas.svddiag; /* Eigenvalues of L matrix */
   code file="&pca_score."; /* Output code to create principal components */
run;
```

ANALYSIS AND RESULTS

In the discussion that follows, the first section addresses the sparse matrix and a method of identifying outlier observations. The second section discusses the principal components of the low-rank matrix and how they can be used to help you model the data. The third section addresses the low-rank matrix and demonstrates the advantage of reducing dimensionality in the original variables.

The analysis uses the default value of λ throughout. Although λcan be tuned, using this default value is sufficient to demonstrate the power of the RPCA procedure.

DETECTING OUTLIERS

The RPCA procedure splits the input data set into a low-rank matrix and a sparse matrix. The sparse matrix consists of the “outlier” portion of each variable in the data set. One obvious way to detect outliers is to look at the nonzero values in the sparse matrix. However, because noise is present, many values in the sparse matrix are nonzero. The sparsity of the sparse matrix is 0.6506, which means that only 65% of the values in the sparse matrix are zero. It is hard to justify labeling 35% of all the data elements as outliers.

On closer inspection, you can see that many of these “nonzero” entries are actually very close to zero. Table 1 shows a data sample that illustrates this phenomenon.

Home ID	Value in Original Matrix	Value in Low-Rank Matrix	Value in Sparse Matrix
1	1	0.944	0.056
2	3	1.299	1.701
3	1	1.247	–0.247
4	0	0	0

Table 1. Variable Values in Each Data Matrix

The magnitude of many of these nonzero entries in the sparse matrix is very small compared to the magnitude of the original value. For the purposes of detecting outliers, it is prudent to consider such values as close enough to zero that they can be restated as being zero. This would increase the number of zeros in the sparse matrix so that you can isolate the nonzero values. But what is the threshold for considering a value to be insignificantly different from zero? This question does not have an absolute answer, because some variables are larger in magnitude than others. In this case, the value of 0.056 in the sparse matrix for Home 1 is a small portion of the value, but for another variable this might be a very large value.

One method of determining what constitutes an outlier is to standardize the variables. Then the test of whether a value differs from zero corresponds to a statistical significance test. Under a normal distribution assumption, an observation has a probability of 0.9973 of falling within three standard deviations of the mean. Because it is so rare to fall outside three standard deviations from the mean, that threshold is frequently used in the context of outliers. Applying that threshold, we can consider a value in the sparse matrix to be different from zero if it falls more than three standard deviations away from zero. Any values that are within three standard deviations of zero can be restated as zero.

The standard deviation of the variable sounds easy, but it is not so easily defined. The data are now stated among three different data sets: the original matrix, the low-rank matrix, and the sparse matrix. Which data set should you use to calculate the standard deviation? The standard deviation from the original matrix is the largest and therefore will produce the most zeros in the sparse matrix. The standard deviation from the low-rank matrix is smaller and represents the variation of “normal” values of the variable. The standard deviation from the sparse matrix directly represents the variation of the values being tested. We chose to use the standard deviation from the original matrix to produce the greatest number of zeros in the sparse matrix. In the end, the choice of standard deviation and threshold is yours, and the end result is not significantly affected.

The following code shows the calculation of the restated sparse matrix:

```
/* Calculate standard deviation of each variable in the M matrix */
proc cas;
  simple.summary /
    inputs={"&inputvars."}
    subset={"std"}
    table={name="home_data"}
    casout={
      name="home_data_std",
      replace=true};
run;
  table.save /
    table={name="home_data_std"}
    name="home_data_std.sashdat"
```

```
      replace=true;
  run;
  quit;


  /* Transpose to a wide data set to prepare to merge */
  proc transpose data=mycas.home_data_std out=mycas.std_values prefix=s_;
    id _column_;
    var _std_;
  run;

  /* Get list of variables containing standard deviations */
  data _null_;
    call symput(
      'std_vars',
      strip(cat('std_',tranwrd("&inputvars.",' ',' std_')))
      );
  run;

  /* Add a dummy variable to the sparse matrix to prepare for the merge */
  data mycas.sparsematrix;
    _name_='_std_';
    set mycas.sparsematrix;
  run;

  /* Replace values in the S matrix that are "close" to 0 with a 0 */
  /* Count the "true" nonzero entries */
  data mycas.sparse_matrix_true;
    merge mycas.sparse_matrix mycas.std_values;
    by _name_;
    array inputvars[*] &inputvars.;
    array std_vars[*] &std_vars.;
    count=0;
    do v=1 to dim(inputvars);
      if abs(inputvars[v]/std_vars[v]) < 3 then inputvars[v]=0;
      count + (abs(inputvars[v]) > 0);
    end;
    drop _name_ &std_vars. v;
  run;
```

Table 2 shows how the values have changed after you restate the values in the sparse matrix.

Home ID	Value in Original Matrix	Value in Low-Rank Matrix	Value in Original Sparse Matrix	Value in Restated Sparse Matrix
1	1	0.944	0.056	0
2	3	1.299	1.701	1.701
3	1	1.247	–0.247	0
4	0	0	0	0

Table 2. Modified Sparse Matrix Values

With the restated sparse matrix, the sparsity has increased from 0.6506 to 0.9875. This has eliminated many nonzero values so that the more significant nonzero values can be isolated. The original task was to find any observations that behave as outliers. In order to do that, you need to summarize the values of the sparse matrix in such a way as to provide a single metric for each observation in the data set. We chose to count, for every observation, the number of variables that have nonzero values. Table 3 shows the frequency table of this count.

Count of Nonzero Variables in Sparse Matrix	**Frequency**
0	5,631
1	5,911
2	7,592
. .	. .
127	1
129	1
251	1

Table 3. Frequency of Sparse Matrix Entries

Among 979 variables, it is still rare for an observation to have no nonzero values (fewer than 4% of the observations do). So it is not useful to treat any observation that has a nonzero value in the sparse matrix as an outlier. In fact, having all zeros in the sparse matrix could indicate a data irregularity! However, the number of variables with nonzero values provides more insight into the likelihood that the observation is an outlier. Thus it would be reasonable to create a cutoff that would signify how many nonzero values constitute an outlier.

As you can see from these data, the observation with the most nonzero variables has 251, and the observation with the second-most nonzero variables has 129. There is such a gradual increase leading up to the 129, and then such a large jump, that this observation is a logical candidate to label as an outlier. Remember, a given data set does not necessarily have many (if any) observations that are true outliers, so it is not necessary to arbitrarily choose a lower threshold to force a certain percentage of the observations to be labeled as outliers. This is, of course, a judgment call.

The outlier analysis of the sparse matrix described earlier can help you determine whether to remove an observation from the data set or perhaps make a transformation to the variables, such as capping their values. If we use the low-rank matrix to build the model, there are already adjustments to the observations to exclude the potential outliers; so the low-rank matrix can be used directly for modeling.

MODELING BY USING PRINCIPAL COMPONENTS

Principal component analysis (PCA) is a well-known method of reducing dimensionality. The principal components represent linear combinations of variables. They are designed so that each principal component is orthogonal to all other components. The components are ordered by their eigenvalues, which describe the amount of variation among the independent variable explained by the component. PCA enables you to reduce the dimensionality of the data set by considering only a subset of the components that explain a large percentage of the total variation in the data. You can then use these principal components to build a model.

It is tempting to think that the components that explain the most variation in the data are the most predictive variables. However, PCA does not consider the target—just the design matrix. Therefore, there is no guarantee that any individual component will exhibit a relationship with the target. However, it is still reasonable to remove the components that explain very little variation in the data. At a certain point, variables with little variation in the data are unlikely to affect a model prediction significantly, even though they might be well correlated with the target.

There are a couple of advantages when you use principal components to build a model. First, the principal components are orthogonal, which means they are linearly uncorrelated. The original variables could have some strong dependencies, making the model selection process more challenging. Second, with this method of dimension reduction, there is very little loss of data. Removing correlated variables from a model removes any additional information that they might contain. A drawback to modeling with principal components is that, because each component is a linear combination of all the variables, the interpretability of each component can be very low. For the purposes of this discussion, the interpretability is not of primary concern; the focus is instead on building a model quickly in order to show whether the data have predictive power.

We built two different regression models by using these data. The first model was built using the principal components of the original data matrix. The second model was built using the principal components of the low-rank matrix. We compared these two models to determine whether using the low-rank matrix would help provide a better fit to the model. We evaluated these models by using a holdout validation data set.

Each of the models was a generalized linear model that we produced using the GENSELECT procedure in SAS® Visual Statistics. The set of potential model variables consisted of the components that explained 99% of the variation in the data. We used forward selection to select model variables and the Schwarz Bayesian criterion (SBC) to select the final model. The final models were built using a log-link function and a Tweedie error distribution.

The Tweedie distribution is commonly used for insurance loss ratio models because it handles the zero-inflated target well. However, the Tweedie distribution is very complex to fit, and PROC GENSELECT currently does not support variable selection with this error distribution. An adaptation was made to first model with a logistic regression to reduce the number of variables and then use a simplified backward selection. The process is as follows:

1. Create a binary target variable (loss ratio is 0 or not). Use forward selection in a logistic regression to create a finer subset of potential model variables.

2. Fit a generalized linear model with a Tweedie error distribution on the original loss ratio target. Use the subset of model variables that are output from the logistic regression as predictors.

3. Identify the variables from the Tweedie model that are statistically significant. Refit the Tweedie model by using only these variables.

The following code demonstrates this modeling process:

```
/* Number of components to capture 99% of variation */
proc sql noprint;
  select sum(SingularVal) into :sum from mycas.svddiag;
quit;

data _null_;
  set mycas.svddiag;
  retain PctVar 0;
  PctVar = PctVar + SingularVal/&sum.;
  if PctVar > 0.99 then do;
    call symput('p',compress(put(_N_,3.)));
    stop;
  end;
run;

/* Score principal components on the L matrix */
data mycas.model_data;
  set mycas.lowrank_matrix;
  Target01 = (Target > 0); /* Create binary target variable */
  %inc "&pca_score."; /* Score principal components on the L matrix */
  keep house_id Train_Ind Target Target01 Prin1-Prin&p.;
run;
```

```
/* Preliminary variable selection on logistic model */
proc genselect data=mycas.model_data;
  partition role=Train_Ind(test='0' train='1');
  model Target_01 = Prin1-Prin&p. / center distribution=binomial;
  selection method=forward(select=sl sle=0.1 stop=sl choose=sbc);
  ods output ParameterEstimates=Parm_Est_Binary;
run;

/* Select variables from logistic model */
proc sql noprint;
  select Effect
    into :effects separated by ' '
    from Parm_Est_Binary
    where Effect^='Intercept';
quit;

/* Preliminary Tweedie model */
proc genselect data=mycas.model_data;
  partition role=Train_Ind(test='0' train='1');
  model Target = &effects. / center distribution=tweedie;
  ods output ParameterEstimates=Parm_Est_Tweedie;
run;

/* Select variables that are still significant in Tweedie model */
proc sql noprint;
  select Effect
    into :final_effects separated by ' '
    from Parm_Est_Tweedie
    where Effect^='Intercept' and cdf('chisquare',ChiSq,1) > 0.999;
quit;

/* Final model */
proc genselect data=&indata.;
  partition role=Train_Ind(test='0' train='1');
  model Target = &final_effects. / center distribution=tweedie;
  code file="&glm_code.";
  output out=mycas.home_data_pred copyvar=house_id p=pTarget;
  ods output ParameterEstimates=Parm_Est_Final;
run;
```

After the models were built, we calculated fit statistics for the training and validation data sets. The training data set had 90,000 observations, but only 4,000 had a nonzero target. The validation data set was similar, with 60,000 observations, only 2,600 of which had a nonzero target. This is a very small sample size for modeling insurance losses, so we made a couple of considerations in calculating the fit statistics. First, we binned the data into five groups of equal size. Second, we smoothed the actual losses over the five groups by using a cubic spline. These considerations were designed to remove random variation from the actual losses and to focus on the systematic variation in order to produce more reliable fit statistics. Table 4 shows the fit statistics for the training and validation data sets.

Data Set	Components Selected	Lift		MAPE	
		Training	Validation	Training	Validation
Original	4	1.72	1.13	0.058	0.186
Low-Rank	8	2.30	1.17	0.072	0.225

Table 4. Fit Statistics for Initial Models

Here you can see a bit of a mixed bag. In the validation data set, the lift of the two models is very similar, and the mean absolute percentage error (MAPE) is a little better in the original data set. Also, notice that the difference in the lift between the training and validation data sets is much greater for the model that was built using the low-rank data set. This tells you a couple of things. First, the model built using the low-rank matrix is able to grab more segmentation out of the training data set than the model built using the original matrix. SBC penalizes the model for each parameter, and it was still able to justify including these extra parameters and obtain larger segmentation. Second, the model that was built using the low-rank matrix is more severely overfit than the model built using the original matrix. The amount of deterioration in the fit statistics between the training and validation data sets tells you how well the model generalizes to new data. The sharper decrease in lift for the model that was built using the low-rank data tells you that, although adding these extra parameters allowed for better segmentation of the training data, it does not help you segment the new data any better than the model built using the original matrix with fewer parameters.

Because overfitting is an issue with these data, it is important to make a fair comparison between these two models. We refit the model that was built on the low-rank matrix by using only the four most significant components. This way both models are equally parameterized and the effect of RPCA is better isolated. Table 5 shows the resulting fit statistics.

Data Set	Components Selected	Lift		MAPE	
		Training	Validation	Training	Validation
Original	4	1.72	1.13	0.058	0.186
Low-Rank	4	2.18	1.40	0.073	0.168

Table 5. Fit Statistics for Final Models

Here the model built using the low-rank matrix has better validation than the model built using the original matrix. After the outlier information that is contained in the data has been removed, you can focus on the more normal data when fitting the model, and the resulting model performs better.

REDUCING THE DIMENSIONALITY OF THE ORIGINAL VARIABLES

This section presents a bit of bonus analysis of the low-rank matrix that ties into the modeling analysis in the previous section. The overall purpose of all this work is to explore the data and determine which variables predict the target. Using principal components enables you to quickly reduce the dimensionality and create a prediction of the target. However, the principal components can be very difficult to interpret, and PCA might not be a slam-dunk method of determining which original variables to tackle in the model-building process.

One way to reduce dimensionality in the original variables is to cluster the original variables directly. We did this by using the VARCLUS procedure in SAS/STAT® software on all the variables in both the original matrix and the low-rank matrix. The output from PROC VARCLUS, displayed in Figure 1, shows the proportion of variables that are explained by the first *k* clusters.

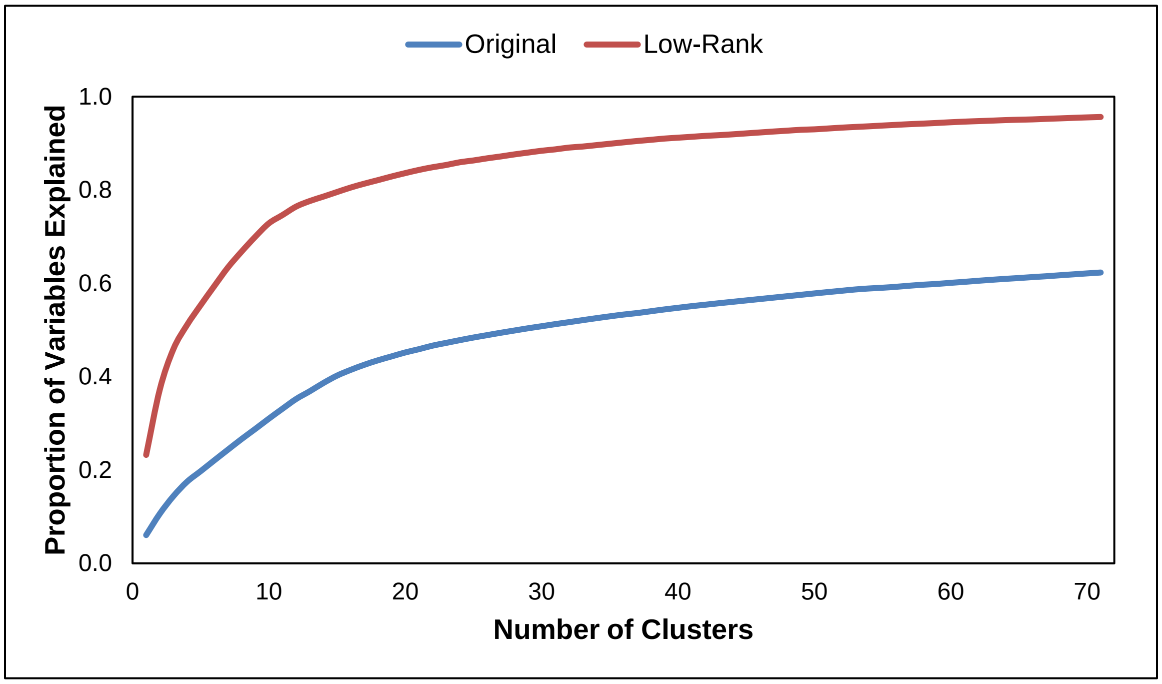

Figure 1. Proportion of Variables Explained by Clusters

For a given number of variable clusters, a greater portion of the variables in the low-rank matrix are explained than in the original matrix. This shouldn't be too surprising. The low-rank matrix has a data modification to remove the outlier portions from the variables. This creates variables that are less variant within themselves and more likely to be similar to other variables in the data set. When you remove some of this variation, the clustering can focus on the "normal" portions of these variables and capture the similarities between the variables that are not distorted by outliers. This can make it easier for you to focus your exploration on certain variables.

CONCLUSION

In this paper, we present some analysis of simulated homeowner's insurance data collected using home telematics devices. The analysis uses various outputs of the RPCA procedure in SAS Visual Data Mining and Machine Learning to explore a modeling data set. The sparse matrix is useful in detecting outliers in the data. Using the low-rank matrix removes the sensitivity of the model to outlier predictors and can produce better modeling results. Using principal components provides a model that quickly demonstrates the predictive power of the data.

The approach in this paper is a straightforward method of building a model quickly, without any data preparation. It is not meant as a substitute for the full model-building process. Instead, it provides a way to do an initial dive into the data to help you gather useful information about the data.

REFERENCE

Candès, E. J., Li, X., Ma, Y., and Wright, J. (2011). "Robust Principal Component Analysis?" *Association for Computing Machinery* 58:1–37.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Anthony Salis
State Farm Insurance Companies
Anthony.Salis.r8ja@statefarm.com

Zohreh Asgharzadeh
SAS Institute Inc.
Zohreh.Asgharzadeh@sas.com

Kyungduck Cha
SAS Institute Inc.
Kyungduck.Cha@sas.com