A Simple Method for Modeling Changes Over Time

Uri Korn, FCAS

Abstract

Properly modeling changes over time is essential for forecasting and important for any model with data that spans multiple time periods. Regression models are probably the most commonly used for building predictive models in the insurance industry. These models do a fine job of fitting data and determining variable relationships, but are not meant for explaining how entities and relationships change over time, as time series models do. Time series models, on the other hand, have other drawbacks, depending on the type of model.

A method is presented to add time series components within a penalized regression framework so that these models are capable of handling everything a penalized generalized linear model can handle (distributional flexibility and credibility), as well as changes over time. Doing this, a subset of state space model functionality can be incorporated in a more familiar framework. The benefits of state space models in terms of their accuracy and intuitiveness are explained. This method can be useful for pricing models and detailed profitability studies, for example, as well as any other type of model with observations spanning multiple time periods.

Keywords. State Space Models, Penalized Regression, Elastic Net, Credibility, Forecasting, Time Series, Hierarchical Models

1. INTRODUCTION

Actuaries are frequently relied upon to make forecasts and predictions across time periods. This can be as a forecast for a future period, such as in ratemaking, as an interpretation of the past, as in reserving, or both together, as in profitability studies. Despite this, the most common type of model used, the linear regression model, is not equipped to handle changes over time, which is an important consideration when working with data that spans multiple time periods and when forecasting future periods.

This type of behavior is best modeled via a state space model, a flexible and powerful time series method. But besides being less familiar to many practitioners, they have other drawbacks as well, depending on how they are solved. A Bayesian model can be used, but these take extra time to build and fit and do not scale well to very large datasets. The other popular option is the Kalman Filter, but it is less accurate and does not provide the distributional flexibility that generalized linear models do.

This paper shows a method to add random walks and related state space model functionality to linear regression models. In a random walk, the complement of credibility for each period is the fitted value of the previous period. This is in contrast to including the time variable as a categorical variable, which would use the overall mean as the complement (if using a model that incorporates credibility, such as a mixed model). Besides being less intuitive since use as a categorical variable ignores the order of the periods, its performance is far inferior to the random walk, as is shown.

A simple method is shown that allows for a subset of state space model functionality, such as the described type of behavior, within a penalized linear regression framework that does not suffer the same drawbacks. This model can handle distributional flexibility and credibility, as well as time series components. With these modifications, these models are better equipped to deal with this type of data.

1.1 Research Context

State space models (SSMs) and penalized regression methods will be explained and used throughout this paper. On the SSM side, De Jong and Zehnwirth 1983 were the first to introduce their use into the actuarial literature and use them to smooth development patterns. Zehnwirth 1996 and Wuthrich and Merz 2008 both use SSMs to smooth reserving estimates and Evans and Schmid 2007 use them to smooth trend estimates. De Jong 2005 gives a nice overview of SSMs and also shows examples of their use in mortality modeling and claims reserving. Korn 2016 uses a simplified SSM to smooth loss ratios by year.

On the penalized regression side, see Hastie, et al. 2009. Hastie and Qian 2014 give a nice overview of these models and their use in the R modeling language. Williams et al. 2015 show the benefit of using these models for variable selection. And recently, Frees and Gee 2016 showed how these models can be used to price policy endorsements. These lists are not meant to be comprehensive; refer to the mentioned papers for further references.

1.2 Objective

The goal of this paper is to show an approach that fits within a regression framework, is capable of handling time series effects, works well with volatile data having relatively few periods, is capable of handling big data, and that produces results suitable for presentation. The proposed method will be referred to as a (linear) regression based state space model, or RSSM for short.

1.3 Outline

Section 2 gives an overview of SSMs. Section 3 discusses some alternative methods for handling changes over time and estimates their performance using simulation results. Section 4 explains how to implement a random walk using the RSSM. Section 5 discusses standardization of time series components, something that is necessary for penalized regression models. Sections 6 and 7 discuss more SSM functionality that can be implemented with this approach, including changing trends and momentum. Section 8 discusses some practical implementation issues, and section 9 demonstrates a use case involving yearly loss ratios.

2. AN OVERVIEW OF STATE SPACE MODELS

State space models (SSMs) are a commonly used methodology to model how different phenomena change over time. They are expressed as a series of related equations. Their flexibility and ease of interpretation make them a common modeling choice. They are usually solved for using either a Kalman Filter or a Bayesian type model (which are both discussed in the next section). The proposed approach provides another means of solving for a subset of SSMs within a GLM framework.

A simple linear trend model (known as "drift" in SSM terminology), for example, can be expressed as follows: (Kim and Nelson 1999)

$$Y_t = X_t + e_t$$

$$X_t = X_{t-1} + u$$

The first equation (also known as the measurement or observation equation) relates the actual data (*Y*) to the fitted values (*X*) with an error term (*e*). In the second (also known as the state or transition equation), the fitted values are increased by the trend (*u*) each period.

Another type of SSM is a random walk, which is a way to model gradual changes that can occur over time. The complement of credibility for each period is the fitted result of the previous period, which is an intuitive way to model changes over time. Such a model balances goodness of fit to the data versus having smaller or smoother changes from period to period. A random walk could be represented as follow:

$$Y_t = X_t + e_t$$

$$X_t = X_{t-1} + r_t$$

These equations are equivalent to the above except that in the second, the fitted values, instead of increasing by the same amount each period, are increased by varying amounts (*r*). This variable is another error term whose values are also minimized. The result is a model that balances goodness of fit to the data with as little change as possible, depending on the ratio of the error terms. The first term (*e*) represents the volatility of the data, while the second (*r*) represents the variance or average magnitude of the period-to-period changes.

A model where the trend (or drift) itself changes via a random walk can be modeled as well. An example is as follows:

$$Y_t = X_t + e_t$$

$$X_t = X_{t-1} + u_t$$

$$u_t = u_{t-1} + r_t$$

Here, the third equation allows for the trend itself (*u*) to follow a random walk. Both *e* and *r* are error terms that are minimized.

More types of models are discussed as well. Even though the proposed models are based on state space models, they can still be used without a complete familiarity of SSMs.

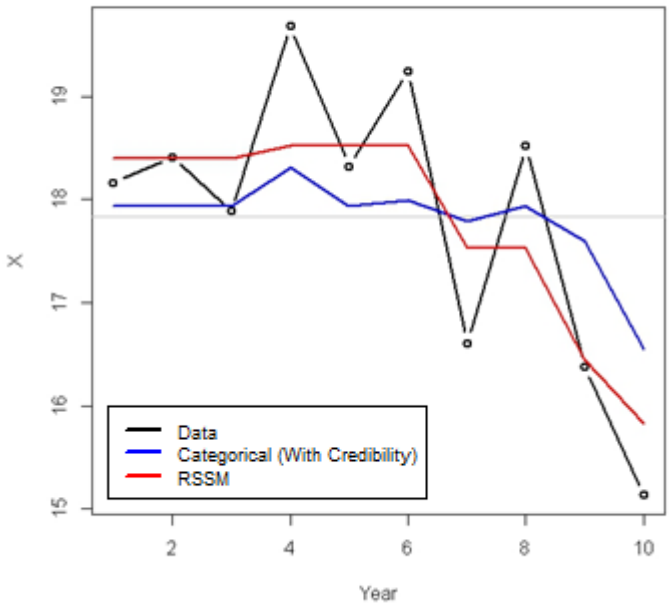
3. COMPARISON WITH EXISTING METHODS

3.1 An Overview

Perhaps the most common way that actuaries use to control for changes over time is to model the year within a GLM as a categorical variable. If a mixed model or penalized regression is used, credibility weighting is performed against the overall mean. But such an approach ignores the relationships between consecutive years. The complement of credibility for each year should be the fitted value of the previous year, which is much more intuitive than the overall mean. Figure 1 illustrates this point[1]. (A penalized regression model was used so that credibility is taken into account. This is similar to adding the year as a random effect in a mixed model.) Note the behavior in years 7 and 8, for example; even though the fitted curve should most likely be decreasing in this range, this does not occur since it is constrained to fall in between the data points and the overall mean. It can be seen that using an RSSM results in more intuitive behavior. Note how the former is also further off in the latest period making forecasts of future periods less accurate.

[1] All of these models were fit using an elastic net with 3-fold cross validation repeated 20 times.

Figure 1: Time as a categorical variable versus RSSM



State space models are another way to model changes over time. The most common methods of solving for an SSM are the Kalman Filter[2] and Bayesian Markov Chain Monte Carlo (MCMC) modeling. The Kalman Filter uses formulas to calculate the amount of credibility to be assigned to each period, using the previous period's prediction as the complement of credibility. The model requires three parameters, which are estimated via maximum likelihood: the value of the first period and two variance parameters that help determine the credibility. The calculations are made easier by assuming that both the distribution of the errors in the data as well as that of the period-to-period changes are normally distributed. (This model is essentially the time series equivalent of Buhlmann-Straub credibility.) For a more thorough review of the Kalman Filter, refer to Korn 2016.

One problem with using the Kalman Filter to model insurance data is its lack of distributional flexibility such as a GLM provides. Errors are assumed to be normally distributed and changes additive. There are some ways of fixing these issues (see Taylor and McGuire 2007 and Korn 2016), but these solutions are still not as robust, flexible, and/or as simple as the proposed. Using the Kalman Filter to fit the example data produces a fitted result with no changes, equal to the overall mean[3]. This is because this model requires more than just ten data points to adequately adapt to and fit the data.

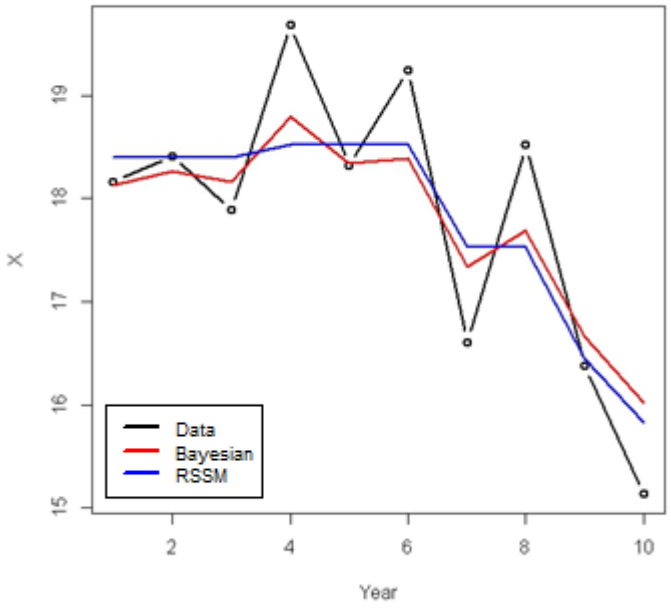
A more flexible framework is provided by Bayesian models, which are capable of modeling SSMs,

[2] The results of the Holt-Winters method, also known as exponential smoothing, should be roughly similar but less accurate than the Kalman Filter and will not be expanded upon here.

[3] The same was true when using the "bagging" method described in Korn 2016.

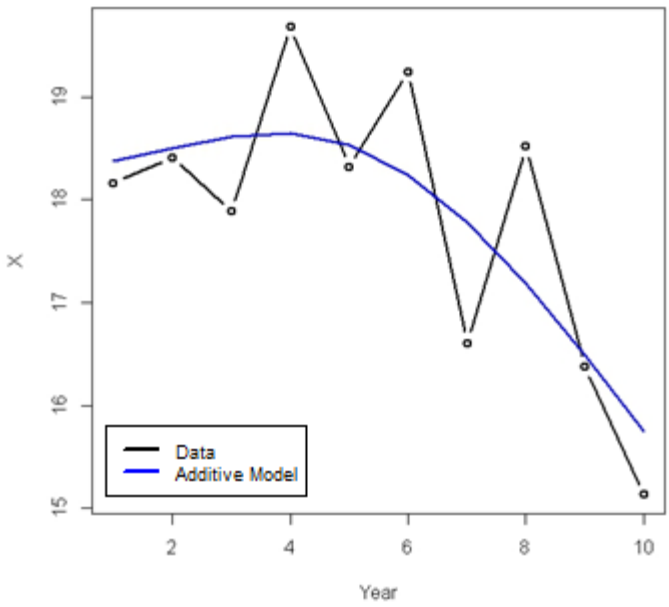
such as a random walk, as well as incorporating any type of distribution assumption. A Bayesian model implementing a random walk has parameters for every single period unlike the Kalman Filter that only has a parameter for the first period. Parameters are typically solved via MCMC techniques, which are simulations that are guided by the overall likelihood of the model. The downsides to these models are the specialized expertise required as well as the time needed to build and run each model. These models also do not scale well to large datasets or to a large number of parameters. As shown in Figure 2, running a Bayesian model on the example dataset performs satisfactory, although produces a much bumpier line than the proposed approach. This makes it more difficult to interpret and not as suitable for presentation. As can be seen, multiple changes are shown before year 6, despite little support for this in the data. The RSSM shows a decreasing trend starting from year 6, which seems to be the general trend of the data; the Bayesian line is still bumpy after this point.

Figure 2: Bayesian model versus RSSM



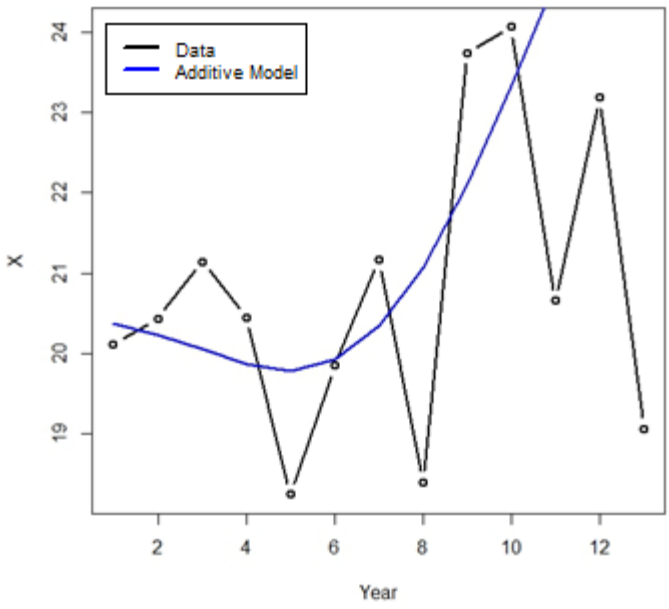
Another approach is to use an additive model, which uses a smoothing function, often a cubic spline, to adapt to the data. This type of model does a good job of fitting to the example data (using the mgcv package in R) as shown in Figure 3.

Figure 3: Additive model fit on the example data



A problem with splines is that even though the historical data may seem to fit well, they often show high trends at the end points, implying historical and prospective patterns that may not exist. Related to this, small changes in the data or a few new data points can often result in large changes. They are also very susceptible to outliers as shown in the next section. Figure 4 shows another example that demonstrates these issues. And finally, they are also difficult to blend with credibility techniques.

Figure 4: Issues with the additive model fit



Finally, there are ARIMA models. Besides for not being as intuitive as the methods discussed, they lack the flexibility of SSMs (Carlin 1992). Because of these issues, they will not be elaborated upon further.

3.2 Simulation Results

Simulations were conducted over a ten year period to compare the various methods to each other and to the proposed method. The first simulation exercise was fairly straightforward and did not attempt to mimic real data by including outliers, etc. The second simulation was meant to be more realistic and used t distributions instead of Gaussian and included occasional outliers to account for the fact that distribution fits are usually not exact. The results are shown in Table 1. The code used to run the simulations can be found in Appendix A.

Table 1: Simulation results of various time series methods

	Simulation 1 – No Outliers		Simulation 2 – Outliers		
Method	**RMSE[4] of Fitted Data**	**RMSE Relative to the Mean**	**RMSE of Fitted Data**	**RMSE Relative to the Mean**	**Difference in Percentages**
Mean	0.8414	0.0%	1.296	0.0%	
Elastic Net With Year as Factor	0.8118	-3.5%	1.412	+9.0%	+12.5%
Kalman Filter	0.8103	-3.7%	1.252	-3.3%	+0.4%
Kalman Filter with Bagging (See Korn 2016)	0.7660	-9.0%	1.196	-7.6%	+1.3%
Bayesian Model	0.6405	-23.9%	1.081	-16.6%	+7.3%
Additive Model[5]	0.6905	-17.9%	1.176	-9.2%	+8.7%
RSSM	0.6517	-22.5%	1.054	-18.7%	+3.9%
RSSM with 25% Momentum (See section 6.3)	0.6439	-23.5%	1.036	-20.0%	+3.4%

The Bayesian model was the best performing when there were no outliers, and the proposed method was the best with outliers, but both methods performed well. Note the poor performance of using the year as a factor (even with credibility being taken into account, as was the case here); when outliers are present, it is even worse than taking a simple average. The differences in the RMSE amounts are shown to the right as a rough measure of the robustness to outliers of each of the methods. Both the additive and Bayesian models are more susceptible to outliers than the proposed method. This is because these models depend on various formulas and assumptions to estimate the appropriate credibility, while the proposed uses cross validation and determines the best credibility by testing on the data itself.

[4] Root mean square error

[5] Note that this method is not completely compatible with the others, as it also has a trend component, which would improve the performance of the other models as well.

4. IMPLEMENTING A RANDOM WALK

4.1 Dummy Encodings

The proposed approach uses a GLM framework to implement a subset of SSM functionality, such as random walks. This provides a relatively simple and familiar modeling environment and also allows for distributional flexibility and credibility.

To explain the approach, when a categorical variable is added to a GLM, dummy encodings are created, such as those shown in Table 2. (The data values are shown on the left hand side, and the created model variables are shown across the top).

Table 2: Default dummy encodings for a year categorical variable

	2014	2015	2016
2013	0	0	0
2014	1	0	0
2015	0	1	0
2016	0	0	1

To implement a random walk, dummy encodings like those shown in Table 3 can be used instead.

Table 3: Initial dummy encodings for a random walk

	2014	2015	2016
2013	0	0	0
2014	1	0	0
2015	1	1	0
2016	1	1	1

With these, the coefficient value for 2014 affects not only that year, but the subsequent years, 2015 and 2016, as well. Likewise the coefficient value for 2015 effects both 2015 and the next year, 2016. If some form of credibility is applied (which is discussed in the next section), the starting point for each year is the previous year's fitted value. This allows for the fitted value of each year to be used as the complement of credibility for the following year. So, for example, if the 2015 coefficient is 0, its fitted value will match the 2014 fitted value.

Relating this back to SSMs, it can be seen that doing this is equivalent to the random walk, where *r* is the coefficient value for each year. The first equation (that relates the empirical data to the fitted values) is identical as well, except that here, the distribution of the error term (*e*) is determined by the GLM family.

$$Y_t = X_t + e_t$$

$$X_t = X_{t-1} + r_t$$

Implementing the approach in this fashion keeps the solution linear, which makes solving for the optimal parameters much easier. Non-linear problems are difficult to solve and even when solved, it is hard to determine if only a local maximum has been reached. Most statistical packages have methods for modifying the default dummy encodings of certain variables. Appendixes A and B show an example of doing this in R.

4.2 Penalized Regression and Cross Validation

Penalized regression will be used as the credibility technique for both the random walk and other coefficients. This works by imposing a penalty to the likelihood the more a coefficient deviates from zero, thus reducing the coefficient values. This pushes the fitted values back towards the intercept, which is the overall mean, and thus credibility is applied.

Unlike mixed models, for example, which use likelihood based formulas and a number of assumptions to determine the magnitude of the penalty parameters, penalized regression uses k-fold cross validation. K-fold cross validation works as follows: the data is randomly divided into k chunks, or "folds". The model is fit on $k - 1$ of these folds using different values for the penalty parameter, and then each of these fitted models is tested on the remaining fold. This process is repeated k times, each time using a different fold for the validation. The penalty parameter that performs best on the test data is chosen. For smaller data sets, this process can even be repeated multiple times and the average penalty parameter selected. This procedure is implemented in many statistical packages.

As mentioned, this approach differs from Bayesian models, mixed models, and the Kalman Filter, all of which use different assumptions and formulas to estimate the penalty parameters. Another benefit of cross validation is that it provides an excellent framework for testing the results of the model as the cross validated predictions (that is, the predictions made on the holdout or test folds) can be compared against the actual data to calculate various metrics that are unaffected by any possible overfitting.

Another benefit is their ability to fit on a large number of variables, even with large datasets. This is because of the efficient fitting algorithm: the model is initially fit using a large penalty value that causes most of the coefficients to be near zero, making the model easy to solve for. This penalty is then gradually decreased and the model is refit, each time using the results of the previous model as the starting point for the coefficients. Because of this, changes to coefficient values are small at each step making it easier for the fitting procedure to find the optimal values (Friedman et al. 2009). Mixed models and Bayesian methods often do not scale as well with large datasets having a large number of

parameters. The run time for mixed models, especially, deteriorates rapidly as the number of variables or data points increases.

4.3 Types of Penalized Regression Methods

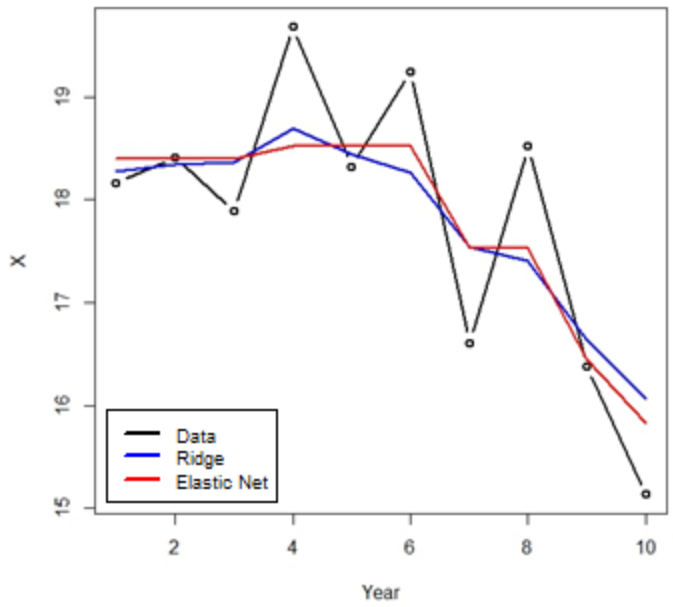
Penalized regression methods apply a penalty to the coefficient values in order to stabilize them. There are two types of penalty functions frequently used. Ridge regression is based on the squared value of the coefficients, also known as L2 regularization. This is similar to a mixed model or to using a normal distribution as the prior in a Bayesian setting.

The other type is the lasso, which penalizes based on the absolute value of the coefficients. A benefit of this type of model is that it can aid in variable selection. This is because the absolute value penalty will approach zero much faster than the squared values and so some coefficient values are set to zero, thus taking out their effect in the model. The downside of this model is that it does not handle correlated variables well. A compromise model called an elastic net provides the benefits of both by imposing a weighted average of both types of penalties (Zou and Hastie 2005). Such a model can handle correlated variables and perform variable selection.

There is another benefit of the elastic net for time series models. Because ridge regression does not shrink its coefficients down to zero, when using it to model a random walk, it will often show small changes in each year, even if it seems that there have not been any real changes. But the lasso cannot be used, since time series variables have correlations, thus making the elastic net the ideal choice[6]. A comparison of using ridge regression and the elastic net is shown in Figure 5. Note how ridge regression produces the bumpier line.

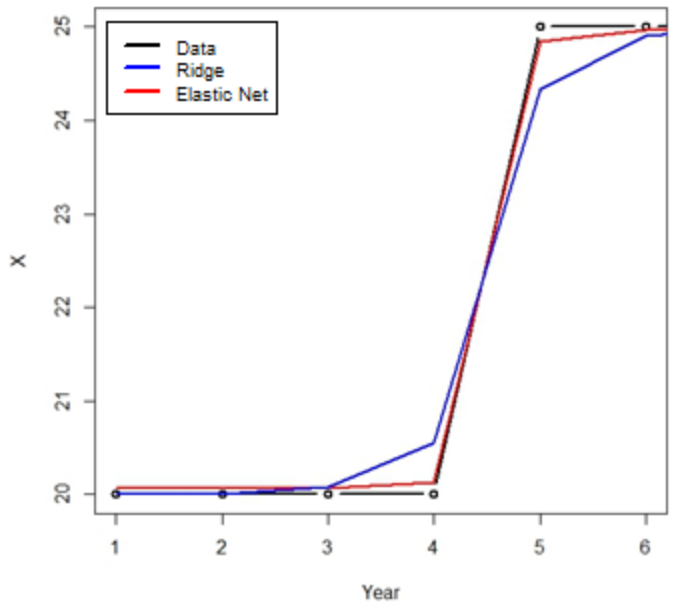
[6] Because of the first reason, it is suggested to give more weight to the lasso penalty.

Figure 5: Elastic net and ridge regression comparison



Related to this, because the square of a large number is an even larger number, if the data has a large jump in a single year, ridge regression will often model smaller changes over the course of several years. An elastic net model typically handles this scenario much better, as shown in Figure 5. The ridge model shows the decreasing trend as starting from year 4, where in reality, it seems that the decreasing trend did not start until year 6. A starker example is shown in Figure 6. It can be seen that a one-time increase is modeled over a period of three years.

Figure 6: Ridge regression comparison of a large change



For these reasons, the elastic net is the recommended model to use when fitting RSSMs.

4.4 Multiple Segmentations

Using the modified dummy encodings shown to model a random walk within a GLM framework makes it easy to not only model the overall changes, but also the changes by various segmentations. This can be done by including an interaction term between the segmentation and the random walk variable. Including a random walk variable by itself as well as an interaction term between the segmentation and the random walk produces a model that credibility weights each segment's changes using the overall average changes as the complement. This can be a powerful tool for handling yearly or quarterly data in a hierarchical fashion, much more detailed than simply modeling on an average trend.

A problem exists, however, when using the simple random walk dummy encoding shown above (Table 3) to model multiple segmentations. The issue is demonstrated in Figures 7 and 8. In this example, several segments are fit with their changes modeled using a random walk (using an interaction term as described) as well as a term for each segment.

Figure 7: A segment moving away from the mean

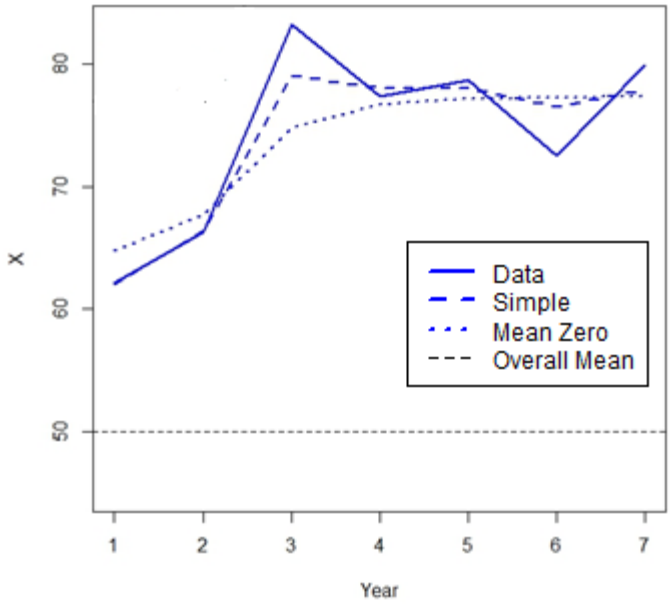
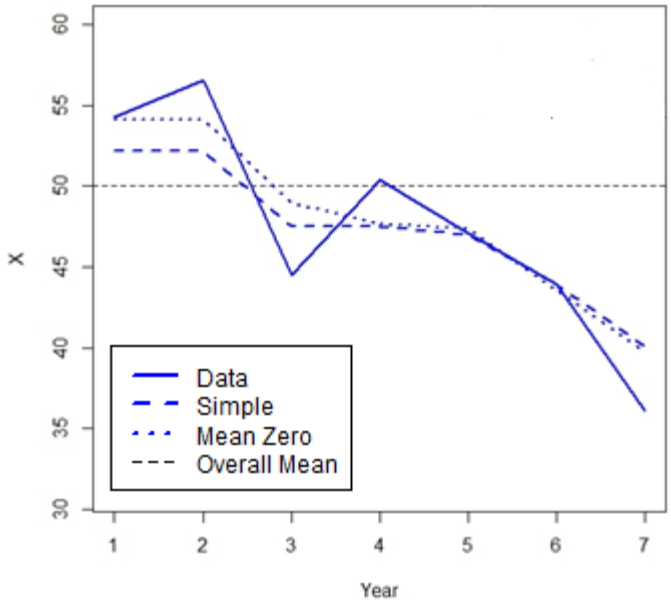


Figure 8: A segment moving towards the mean



The segment shown in Figure 7 is moving away from the overall mean, and in Figure 8, is moving towards the mean, as can be seen. Note how the fits using the simple encodings gives rise to a steeper curve than expected when moving away from the mean, and a flatter than expected curve when

moving towards the mean. This is because when using the simple encodings, the first point of each segment is determined by the value of the segment coefficient (since all of the random walk variables are zero at this point). The subsequent points are determined from the interaction between the segment and the random walk variable, all of which are shrunken towards zero due to the penalty. Because of this, the model will often "take a shortcut" and gradually approach the data points over several periods in order to reduce the total value of this penalty, which results in the pattern mentioned.

To fix this issue, the random walk dummy encodings can be modified so that the mean of each year is zero (which can be done by simply subtracting the mean from each column). If these encodings are used instead, the segment coefficients now represent the average value of each segment, since the net effect of the random walk parameters is zero. Doing this fixes the tendency to "take a shortcut" and results in behavior that is more intuitive, as can be seen

On another note, it is worth mentioning that both the random walk and the segment parameters share the same penalty value. This does not mean that they will receive the same amount of credibility since this depends on other factors as well. But still, this should not be a concern as it is consistent with the penalized regression methodology, which uses the same penalty value for all parameters. If the variables are on the same scale (which they should be – see section 5), this will give them equal treatment in the model. An equal amount of explanation is penalized the same amount regardless of which variable it comes from. However, if desired, a different penalty can still be used for the random walk components by setting the penalty of these variables to a factor of the overall penalty and using another round of cross validation to determine this factor, although it is usually not necessary to do so.

4.5 Using Cross Validation with Panel Data

Panel data is the term used to describe data that both uses explanatory variables and has multiple observations across time periods, such as the data being described here. One of the assumptions of cross validation is that the folds are not correlated with each other. But this may be violated for this type of data, since the same entity may exist in different folds at different time periods, and these are correlated with each other.

To address this issue, instead of randomly selecting individual rows for inclusion in each fold as normally done, the entities themselves can be randomly assigned to folds along with all of their corresponding rows. This will reduce the correlation across folds when using panel data with cross validation. (Note that this is only necessary when a corresponding variable is not included in the model.)

4.6 Numeric Variables

This section illustrated how a random walk can be used to allow the coefficients of categorical variables to change over time. It is possible to do the same with numerical variables as well. Instead of including interactions, as done above, a new variable can be created and added to the model that is the product of the random walk and that variable. This works since:

$$\text{Coef1} \times V + \text{Coef2} \times V \times RW = V \times (\text{Coef1} + \text{Coef2} \times RW)$$

Where *V* is the desired variable, *RW* is a random walk variable, and *Coef1* and *Coef2* are two model coefficients. This allows for modeling how the relationships of numerical variables can change over time.

5. STANDARDIZATION

Before delving into more types of state space models, it is first necessary to discuss the standardization of different time series components. Since the same penalty is applied to every model variable, they should be on the same scale so that they receive equal treatment. Otherwise, equal coefficient values will cause greater changes to the variable with higher values. If one variable is stated in pounds and another in ounces, for example, the one stated in pounds has a larger scale and is likely to have greater effect on the fitted results. The most common approach is to normalize each variable by subtracting the mean and dividing by the standard deviation. This applies to numerical variables only. When dealing with both numerical and categorical variables, Gelman 2008 suggests dividing each numerical variable by twice the standard deviation instead. This is because a binary variable with fifty percent ones has a standard deviation of 0.5.

None of these approaches are designed for handling time series variables, however. These also need to be adjusted so that they can receive equal treatment. The following rules will be used to standardize time series variables to put them on the same scale as the other variables in the model and as each other:

1. A random walk variable (with no momentum, which is discussed later on) does not need to be adjusted, since it is similar to a categorical variable, which will not be modified. Since this random walk variable is not adjusted, all other time series variables can be compared to it for calculating their relative scaling factor.
2. Instead of comparing the standard deviations from the mean, the scale of a time series variable should be determined by calculating the square root of the average squared

differences between each time period. This is similar in nature to the common practice of using the standard deviation, but more properly reflects the nature of these variables.

When calculating these averages, since the denominator is the same for all time series variables (equal to the number of time periods), it will cancel out when being compared and can be ignored. This means that instead of comparing the average squared differences, the sum of the squared differences will be used instead. This quantity is equal to one for a plain random walk (with no momentum), which is the point of comparison. Therefore, the standardization divisor for each variable is simply equal to the square root of the sum of the squared differences[7].

Applying these rules to a simple trend (or drift) variable, which is a numerical sequence from one to the number of time periods, this variable would be divided by the square root of one less than the number of time periods. So, for example, a ten year series would be divided by three and a twenty six year series would be divided by five. (Note that the longer series receives a greater divisor. To explain, if both a trend and a random walk variable are in a model, the total penalty for using the random walk equals: (n – 1) times the average change, where *n* is the number of time periods (assuming a lasso penalty for simplicity). The penalty for using the standardized trend variable equals: $\sqrt{(n-1)}$ times the selected trend. Using the trend instead of the random walk can result in a lower penalty, but is also less flexible than the random walk. So, since the total penalty for using the random walk grows with the length of the sequence, to put the variables on equal footing, it is necessary for the trend penalty to do the same. Also, as the number of data points grows, a trend parameter is capable of having a greater impact on the likelihood, and so can withstand a larger penalty value.)

Both Appendix A and Appendix B show R code that use these standardization rules. When using in a penalized regression model, it is recommended to manually standardize all variables as described and to make sure that the penalized regression function used does not apply any additional standardization by default.

To recap, categorical variables should not be adjusted, numerical variables should be divided by twice the standard deviation, and time series variables should be divided by the square root of the sum of the squared differences.

[7] Note that twice this amount is not used, similar to how numerical variables are divided by twice their standard deviation, since dividing by this divisor already puts the variables on the same scale as a plain random walk. By contrast, it is necessary to divide numerical variables by twice their standard deviation to put them on the same scale as a categorical variable.

6. EXTENDING THE RANDOM WALK

6.1 Random Walk with Drift

The random walk model discussed above in section 4 assumes that the expected change for each period is zero, and this serves as the complement of credibility. If not the case, a trend (or drift) term can be added, and this value will serve as the effective complement of credibility instead. Such a term can be added to a GLM by including the time period as a numerical variable. (It is a good idea to set the mean of this variable to zero for the reasons mentioned in section 4.4. This variable should also be standardized as illustrated in section 5.)

6.2 Modeling a Changing Trend

All of the discussion above focused on a random walk on the level of a variable. It is also possible to model a changing trend (or drift) by using dummy encodings like those shown in Table 4.

Table 4: Example dummy encodings for a random walk on the slope

	2014	2015	2016
2013	0	0	0
2014	1	0	0
2015	2	1	0
2016	3	2	1

Using these, the 2014 coefficient, for example, will cause increases in years 2014 to 2016, and the 2015 coefficient will cause increases in years 2015 and 2016. This will cause a change to the slope. A trend term should also be added for the starting slope unless it is assumed to be zero. (Once again, coefficients that sum to zero should be used. The variable should also be standardized, as illustrated in section 5.)

6.3 Mean Reversion and Momentum

It is also possible to build a model that uses the concept of mean reversion. Allowing for mean reversion on the trend, for example, allows the trend to change, but also causes any changes to gradually decay over time and revert back to the long term average trend. This can be used to model shorter term changes in the trend that gradually revert back towards a long term average value. An example of dummy encodings with 25% mean reversion is shown in Table 5:

Table 5: Example dummy encodings with 25% mean reversion

	2014	2015	2016
2013	0	0	0
2014	1	0	0
2015	1 + 0.75	1	0
2016	$1 + 0.75 + 0.75^2$	1 + 0.75	1

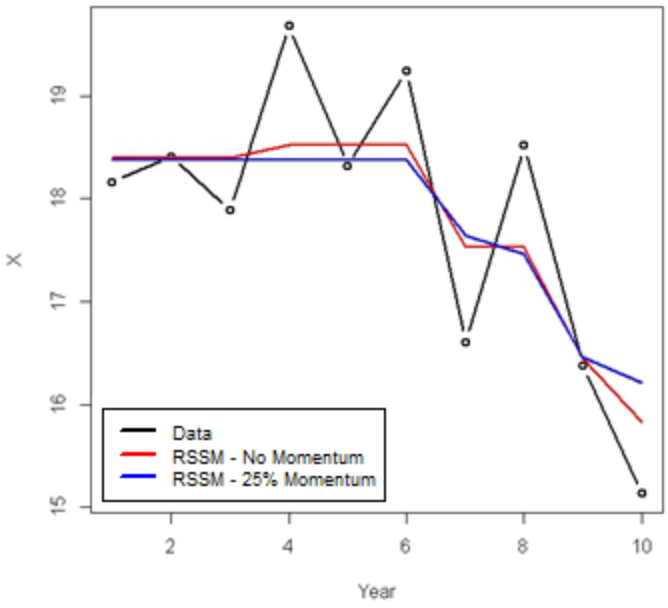
As can be seen, instead of adding one to each subsequent year, the added amounts decay exponentially. (As mentioned, after these dummy encoding are created, the mean should be subtracted from each column so that they all equal zero. They should be standardized as well as discussed in section 5. Those rules can be applied to standardize each column, which will receive slightly different penalties each.)

This concept of mean reversion can be used to relate a random walk on the level of a variable to a random walk on the trend. If the mean reversion is set to zero, no mean reversion will occur and the result is identical to a random walk on the trend. Alternatively, if the mean reversion is set to one, the changing trend will immediately revert back to its long term value after one period and so each change only affects a single period. This is identical to a random walk on the level of a variable. Any value in between zero and one can be viewed as a compromise of the two models.

One way of looking at these models (with perhaps a higher mean reversion value, although not necessarily) is as a random walk on the variable's level, but with momentum. In these models, the complement of credibility for the change of each period is a value between zero and the previous period's change. This will cause changes to continue in the same direction the following period, unless they are reversed. This is often a more realistic expectation since, quite often, changes display serial correlation over time.

Fitting the example data with this type of model produces the result shown in Figure 9. Note how this model both improves the fit to the data and results in a smoother, nicer looking curve.

Figure 9: Random walk with momentum



Cross validation can be used for choosing the optimal momentum for a model. Values can be tested in jumps of 5%, 10%, or 25%, etc., with all random walk variables sharing the same value. Or alternatively, multiple random walk variables with different momentum values can all be included and a grouped lasso penalty (which will cause each group of variables to either all be included or all be excluded) can be used to decide which are optimal, if supported by the statistical package being used.

The SSM equations for this mean reversion model are as follows (where the average long term trend is assumed to be zero for simplicity). It is easy to verify that these will produce identical results as using the dummy encodings shown above.

$$Y_t = X_t + e_t$$

$$X_t = X_{t-1} + u_t$$

$$u_t = au_{t-1} + r_t$$

It can be seen that if the mean reversion parameter (*a*) is set to one, these equations will be equivalent to those of a changing slope. If *a* is set to zero, then, $u_t = r_t$ and these equations are equivalent to a random walk on the level. If *a* is set to a value between zero and one, the trend will gradually decay back towards zero (or to the long term trend, if specified in the model).

6.4 Level Mean Reversion

It is worth mentioning that mean reversion can be used on the level of a variable as well, not just on the trend. This would cause any changes in the random walk to gradually decay, causing the level to revert back towards its long term average over time. If a trend or drift component is also included, the level will gradually revert back towards the trended long term average. This would be done by having the encodings of the random walk start at one as usual, but subsequent values are multiplied by a factor causing them to decay exponentially back towards zero over time. However, level mean reversion probably has less applicability to insurance modeling.

6.5 Extra Dispersion

It is also worth mentioning that another time series component can be added to provide some extra flexibility. A random walk models changes by period that are expected to continue in the next. In contrast, another component can be added for spikes and dips to the fitted values that occur only within a single period and which do not continue to the next. The SSM equations for this model are as follows:

$$Y_t = X_t + e_t$$

$$X_t = Z_t + d_t$$

$$Z_t = Z_{t-1} + r_t$$

Where Z_t is another state variable and d_t is this new component, which is an error term that is minimized. This component can be added to a GLM by including the year as a factor. However, doing so in addition to a random walk variable usually results in poorer performance (in the author's experience) and using it is not recommended, unless perhaps a greater penalty is applied to these variables. Another way to view this component is as a random walk with full level mean reversion.

7. MORE SSM COMPONENTS

7.1 Seasonality

If modeling on a period of less than a year, it may be necessary to account for the different levels of each month, quarter, or other unit, depending on the data. This can be accomplished simply by adding another categorical variable, adding another random walk, or by using splines[8].

7.2 Predictive Variables

Sometimes, the causes of yearly changes are understood and can be related to external variables. When this is the case, the variable can be incorporated in the model to help improve the predictions. This variable should be included as an index so that only changes in the variable affect the level each period and not the actual value of the variable (although if this variable is the same for every segment, the effect is identical).

To give an example, if yearly loss ratios by country are correlated with the interest rate, an index based on the interest rate can be used. This index can be created by dividing all values by the interest rate of the first year for that country. Note that the index was based on the interest rate itself and not the change in the interest rate so that changes in the interest rate will cause changes to the yearly loss ratios. If the interest rate has a lagged effect on the loss ratios, it is also possible to insert it lagged by the appropriate number of periods, which can be determined via another round of cross validation.

If, on the other hand, changes in the loss ratio trend are correlated with changes in the interest rate, for example, another variable can be added that is the product of the interest rate index and a numerical variable for the year (in addition to the year variable by itself to represent the average slope). This will work since:

$$\text{Coef1} \times Y + \text{Coef2} \times I \times Y = Y\,(\text{Coef1} + \text{Coef2} \times I)$$

Where *Y* is the year, *I* is the interest rate index, and *Coef1* and *Coef2* are two model coefficients. As can be seen, changes in the index will cause changes in the slope, the amount of which is determined by the value of *Coef2*. (Both the interest rate and the slope should be standardized, as discussed in section 5.)

[8] Note that the issues mentioned above regarding additive models do not apply here since periods are repeated multiple times.

7.3 Multidimensional Random Walks

A two (or more) dimensional random walk can be constructed as well by interacting two random walks with each other. Depending on the packages used, the columns may need to be constructed manually, however. (The columns should be multiplied together while still in ones and zeros, and they can be made to sum to zero afterwards.) This can be useful for geographical smoothing, for example.

7.4 Correlated time series

Correlated time series can also be modeled using this framework. As an example, consider the case discussed above (section 7.2) where changes in the interest rate affect the loss ratio. As an alternative, the interest rate and the loss ratio can be modeled as correlated time series. This can be done by, instead of including the interest rate as a variable in each row, the interest rates would be given their own rows. Then, a model can be fit with a random walk that affects both the loss ratios and the interest rates. If the random walk variables in the loss ratio rows are multiplied by 10%, for example, each modeled point change in the interest rate will cause a 10% change in the loss ratio. The difference between this model and the one discussed above is that what the model determines to be the "errors" in the interest rate will not affect the loss ratio, since these do not affect the path of the random walk. It is also possible to assign a separate random walk to the loss ratios to capture the uncorrelated changes. (Although, in this example, different distribution families may be needed for each component, which is not possible with most standard regression packages. A categorical variable is also needed to determine whether each row is a loss ratio or an interest rate so that each can receive proper treatment in the model formula.)

Another example of a correlated time series approach is a dynamic factors model, which is a method for modeling missing or unknown variables that change over time (Geweke 1977). The missing variables are modeled via a random walk but coefficients by entity control the magnitude of the change for each entity. So essentially, this dynamic factor is a random walk whose magnitude varies by entity. An example of this is the stock market beta for each industry and company. Various market effects drive the value of stocks up and down, but each industry and company is affected differently from these changes.

For the simple interest rate model, the correlation percentage can be determined via another round of cross validation. Or alternatively, it can be approximated by initially fitting the percentage using the interest rate as a predictive variable, as discussed, then fitting another model for the random walk using this percentage, and then refitting the percentage using the fitted (standardized) random walk as a variable in the model.

A (mostly) one-sided dynamic factors model (where all coefficients are assumed to have the same

sign) can be approximated in a similar fashion. A two-sided model can be approximated by initially fitting the random walk with a small penalty value so that it is not shrunken to zero. The penalty can then be gradually decreased as iterations between the random walk and the correlations are fit.

For more complicated models and for more exact solutions to these models, the EM algorithm (Expectation-Maximization, Dempster et al. 1977) can be used. Further discussion is outside the scope of this paper.

8. SOME NOTES ON IMPLEMENTATION

The method presented can be implemented using most existing elastic net packages. To do so, the default dummy encodings for random walk variables can be changed, as discussed. (See Appendix A and Appendix B for example R code.) If not possible, the random walk variables for each period can also be created and added manually.

Before a GLM solving algorithm is run, a matrix is created for the specified independent variables. Many GLM functions do this implicitly. Just as a separate column is created in this matrix for each possible value of a categorical variable, a separate column is also created for each period in a random walk variable, except for the first. If interactions with some segmentation are included, separate columns will be created for every combination of year and segment. Because of this, the resulting matrix can become quite large for models using a large number of predictors and that have a large number of data points. For most models, this is not an issue, but for very large ones, if encountering memory issues, instead of creating a standard matrix that utilizes memory for each cell value, a sparse matrix can be created instead, which only utilizes memory for non-zero cells. This can reduce the amount of memory required dramatically since most of the values in a typical modeling matrix are usually zero. (The example shown in Appendix A takes this approach.)

Some sparse matrix implementations (such as the "sparse.model.matrix" function in the "Matrix" package in R), when building a sparse matrix, will initially create a non-sparse matrix and only convert it to a sparse matrix at the very end. This can still create memory issues for very large models. If encountering issues, the matrix can be constructed manually one variable at a time so that sparse matrices can be used even during construction, which will save memory.

9. LOSS RATIO CASE STUDY

The proposed method will be demonstrated on an example involving yearly loss ratios. Three segments are used in the example, each having two sub-segments, making six sub-segments in all.

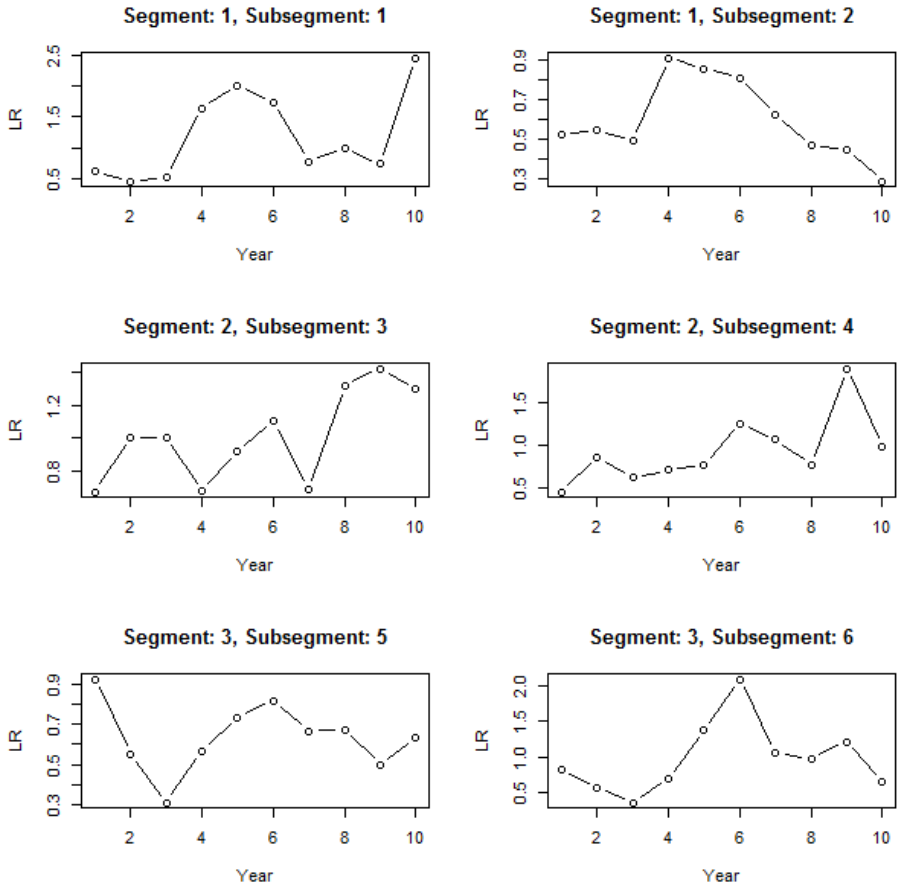
Each of these segments and sub-segments are affected by various changing factors, both on the

premium and on the loss side, which cause the loss ratios to vary by year. Premium is affected by rate changes, which are easily accounted for, but the recorded rate changes are usually not a hundred percent accurate and do not take everything into account, such as changes in policy wording. On the losses side, besides from a longer term trend that affects the losses by a (perhaps) similar amount each year, social, legal, economic, and other factors can cause changes over shorter periods. Some of these may affect the entire book while others can be limited to various segments or sub-segments.

Another consideration is that claims take time to be reported and settled and so our current snapshot of losses will develop over time. Our goal is to estimate the ultimate loss ratio for each year and for future years for reserving, rate making, profit study, or informational purposes.

The ultimate chain ladder loss ratios are shown in Figure 10. This example assumes that premiums have already been on-leveled for rate changes and that losses have been trended by the long term average trend (although it is possible to use the procedure to fit this as well). For simplicity, the on-level premiums for each sub-segment for each year are assumed to be $1,000, although varying premiums can be accommodated as well.

Figure 10: Trended, ultimate loss ratios by segment



To fit this data, an elastic net with variables for a random walk is used. The model will account for the different loss ratio levels for each segment and sub-segment as well as the changes to each by year, incorporating credibility for both the level and changes. A Tweedie family is used to fit the aggregated losses.

A Cape Cod-like approach is used to account for development where the loss ratios inputted into the model are the reported loss ratios multiplied by the loss development factors (i.e., the chain ladder loss ratios), and the premiums divided by the loss development factors (i.e., the "used premiums") are used as the regression weights. This procedure accounts for development while taking into account the extra volatility of the greener years. This causes the model to give less weight to these years, but all years are still taken into account for determining the fitted loss ratios for each year. (As noted in Korn 2016, if full credibility is given to the yearly changes, the final indications will equal the chain ladder. If no credibility is given to the yearly changes, the results will equal the weighted average, which is the Bornhuetter-Ferguson loss ratio with the Cape Cod loss ratios used as the a priori. Anywhere in between can be thought of as a credibility weighting between these two methods.)

The code used to generate and fit the data is shown in Appendix B. The regression formula used was as follows, where a colon is used to indicate interaction effects, *ult.lr* are the ultimate loss ratios, *intercept* is an intercept term, *seg* is the variable for the segment, *subseg* is the variable for the sub-segment, and *yr.rw* is a random walk variable:

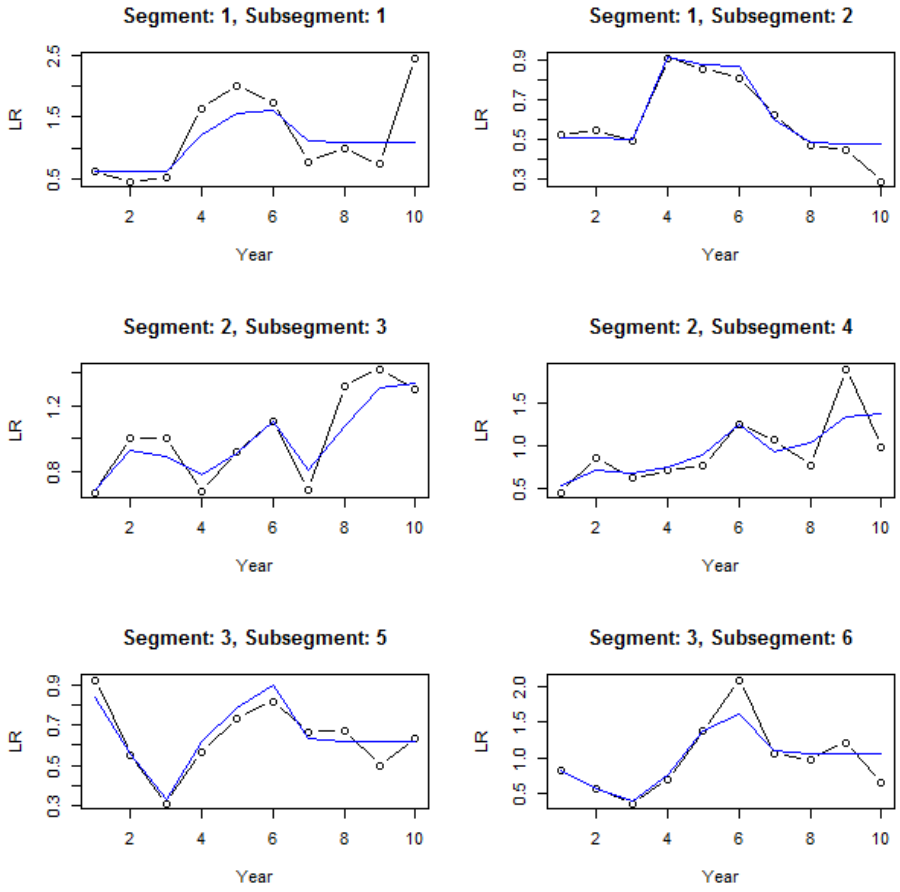
$$\log(\text{ult.lr}) = \text{intercept} + \text{seg} + \text{subseg} + \text{yr.rw} + \text{seg:yr.rw} + \text{subseg:yr.rw}$$

This is a hierarchical credibility model. The overall average level of the loss ratio is determined by the intercept. The average relativities for each segment are determined by the *seg* coefficients, and the additional relativities for each sub-segment are determined by the *subseg* coefficients. Since each coefficient value is penalized and pushed back towards zero, each level is credibility weighted back towards the previous, i.e., each segment is credibility weighted back towards the overall mean and each sub-segment is credibility weighted back towards the segment. The same can be said for the changes by year. The *yr.rw* variable creates a random walk on the intercept, which affects all segments and sub-segments. The interaction of this variable and the segment allows for additional changes that only affect particular segments, and these changes are credibility weighted back towards the overall changes by year. The same occurs at the sub-segment level, and these changes are credibility weighted back towards the indicated segment changes.

The model just discussed does a good job of fitting all segments and their changes by year but does not take into account any possible autocorrelation between years, i.e., momentum of the yearly changes. Stated another way, if a change is observed in a previous period, then instead of using no change as the complement of credibility for the next period, perhaps the complement should be set

to somewhere in between zero and the previous indicated change. This can be tested by redefining the random walk variable with different amounts of momentum and then refitting the model. The momentum value having the lowest cross validated error is then selected. The cross validated error (using the deviance as the error measure) for the model with no momentum is 0.0999. Testing in increments of 0.1, the next value tested is 0.1. This model has a cross validated error of 0.0984, which is better than the first model. Testing a momentum value of 0.2 yields an error of 0.1015, worse than the previous. So the selected value is 0.1. The final fitted results for each segment and sub-segment are shown in Figure 11.

Figure 11: Fitted trended, ultimate loss ratios by segment



Looking at this figure, some common trends can be seen. Most of the sub-segments start increasing at year 3 and decrease in year 7, although this increase is delayed a year in sub-segment 3. Common patterns can be seen by segment as well. Sub-segments in segment 1 show an initial increase followed by a decrease, segment 2 shows a generally increasing pattern, and segment 3 shows a decrease followed by an increase, and then no change from year 7. (All of this can be seen by looking directly at the fitted coefficients as well.) It is also apparent that less weight is given to the more recent years

due to the additional uncertainty of these immature years.

10. CONCLUSION

This paper showed a method for incorporating a subset of state space model functionality into a linear regression framework. Besides for the improved performance and ease of use of this method, the resulting models are intuitive and the corresponding parameters lend themselves easily to interpretation. This can be a useful tool when attempting to "dig deeper" and discover changes or trends that may be affecting particular segments or entities. It can make forecasts into future periods more accurate as well. The results are suitable for presentation, which is an important consideration since findings often need to be communicated to other parties. Lastly, it is incorporated in a framework that scales well to large datasets, an important consideration in the age of big data.

APPENDIX A: Simulation Code

```
library( glmnet )
library( mgcv )
library( compiler )
library( rstan )

# logit functions for later
logit <- function(x) log( x / ( 1 - x ) )
ilogit <- function(x) exp(x) / ( 1 + exp(x) )

# function for creating dummy variables that include a column for every single value (unlike the default behavior in GLMs that leave out one value).
# this is needed for penalized regression (i.e. credibility models), since every segment is credibility weighted back towards the intercept (mean)
contr.pen <- function( n, contrasts=TRUE, sparse=FALSE ) contr.treatment( n, contrasts=contrasts, sparse=sparse )
contr.pen.sparse <- function( n, contrasts=TRUE, sparse=TRUE ) contr.treatment( n, contrasts=contrasts, sparse=sparse )

# function for creating dummy variables for implementing random walks
contr.randwalk <- function( n, contrasts = TRUE, sparse = TRUE, momentum=0, rel.cred=1, stdize=TRUE ) {
  if (length(n) <= 1L) {
    if (is.numeric(n) && length(n) == 1L && n > 1L)
      levels <- seq_len(n)
    else stop("not enough degrees of freedom to define contrasts")
  } else { levels <- n }
  levels <- as.character(levels)
  if ( sparse ) {
    cont <- Matrix( c(0), nrow=length(levels), ncol=length(levels) - 1, sparse=TRUE )
  } else {
    cont <- matrix( c(0), nrow=length(levels), ncol=length(levels) - 1 )
  }
  for ( i in 2:n ) {
    cont[, i - 1] <- ifelse( 1:n < i, 0, ifelse( rep( momentum, n ) == 1, ( 1:n - i + 1 ), ( 1 - momentum ^ ( 1:n - i + 1 ) ) / ( 1 - momentum ) ) )
    cont[, i - 1] <- cont[, i - 1] - mean( cont[, i - 1] )
  }
  if (contrasts) {
    colnames(cont) <- levels[-1]
  }
  # standardize
  if ( stdize ) {
    for ( i in 1:ncol(cont) ) {
      cont[,i] <- cont[,i] / sum( diff( cont[,i] ) ^ 2 ) ^ 0.5
    }
  }

  cont <- cont * rel.cred
  cont
}

# fit kalman filter
kf.fit <- function( x.obs, prem=rep( 1, length(x.obs) ), incl=1:length(x.obs) ) {
  kf <- cmpfun( function( params, return.values=F ) {
    n <- length(x.obs)
```

```
        x.pred0 <- rep( 0, n )
        x.pred <- rep( 0, n )
        k <- rep( 0, n )
        p0 <- rep( 0, n )
        p <- rep( 0, n )
        f <- rep( 0, n )
        err <- rep( 0, n )

        Q.est <- exp( params[1] ) * ilogit( params[2] )
        R.est <- exp( params[1] ) * prem[1] * ( 1 - ilogit( params[2] ) )

        for ( i in 1:n ) {
          x.pred0[i] <- ifelse( i > 1, x.pred[i - 1], params[3] )
          err[i] <- x.obs[i] - x.pred0[i]
          p0[i] <- ifelse( i > 1, p[i - 1] + Q.est, 0 )
          f[i] <- p0[i] + R.est / prem[i]
          k[i] <- ifelse( ( ! i %in% incl ) | f[i] == 0, 0, p0[i] / f[i] )
          p[i] <- p0[i] * ( 1 - k[i] )
          x.pred[i] <- x.pred0[i] + k[i] * err[i]
        }
        loglik <- sum( sapply( 1:n, function(i) log( max( 1e-100, dnorm( err[i], 0, f[i] ^ 0.5 ) ) ) ) )
        if ( return.values ) {
          x.smooth <- rep( 0, n )
          x.smooth[n] <- x.pred[n]
          for ( i in (n - 1):1 ) x.smooth[i] <- x.pred[i] + ( p[i] / p0[i + 1] ) * ( x.smooth[i + 1] - x.pred[i] )
          list( loglik=loglik, Q=Q.est, R=R.est, x=x.smooth, x.pred=x.pred, f=f, k=k, p=p )
        } else {
          loglik
        }
      } )
      o <- optim( c( log( var(x.obs) / 2 ), -10, sum( prem * x.obs ) / sum( prem ) ), kf, control=list( fnscale=-1 ),
method='BFGS' )
      kf( o$par, return.values=T )
    }

    # calculate root mean square error
    rmse <- function( x, y, i=1:length(x) ) mean( ( x[i] - y[i] ) ^ 2 ) ^ 0.5

    # number of years
    n <- 10
    # innovation variance
    Q <- 0.002
    # volaility
    R <- 0.003
    # serial correlation
    years.corr <- 0.1

    # definitions for no outliers
    q.large.pct.change <- 0
    r.large.pct.change <- 0
    q.df.reg <- 250
    q.df.large <- 250
    r.df.reg <- 250
    r.df.large <- 250
    q.large.pct.change <- 0
    r.large.pct.change <- 0
    pct.change <- 0.5
```

```
Q.large <- 0
R.large <- 0

# definitions for with outliers
q.df.reg <- 6
q.df.large <- 3
r.df.reg <- 12
r.df.large <- 6
q.large.pct.change <- 0.05
r.large.pct.change <- 0.1
pct.change <- 0.5
Q.large <- Q * 5
R.large <- R * 5

num.iter <- 250

do.graph1 <- FALSE
print.cvm1 <- FALSE
do.bayes <- TRUE
# run the simulation
results <- sapply( 1:num.iter, function(iter) {
  if ( iter %% 10 == 0 ) print( iter )

  # simulate data
  q <- rt( n, q.df.reg ) * Q ^ 0.5
  # add serial correlation
  for ( i in 2:n ) {
    q[i] <- years.corr * q[i - 1] + sqrt( 1 - years.corr ^ 2 ) * q[i]
  }
  q <- ifelse( runif( n ) < pct.change, q, rep( 0, n ) )
  q.large <- ifelse( runif( n ) < q.large.pct.change, rt( n, q.df.large ) * Q.large ^ 0.5, rep( 0, n ) )
  q <- q + q.large
  q.cuml <- cumsum( q )
  a <- 20 * exp(q.cuml)
  r.large <- ifelse( runif( n ) < r.large.pct.change, rt( n, r.df.large ) * R.large ^ 0.5, rep( 0, n ) )
  o <- exp( log( a ) + rt( n, r.df.reg ) * R ^ 0.5 + r.large )

  # create the time series variables
  d <- data.frame( y=1:n, a=a, o=o )
  # standardize trend variable
  d$y.std <- d$y / sqrt( n - 1 )
  d$y.std <- d$y.std - mean( d$y.std )
  d$y.fac <- factor( d$y )
  contrasts( d$y.fac ) <- contr.pen( nrow(d), sparse=TRUE )
  d$y.rw <- d$y.fac
  contrasts( d$y.rw ) <- contr.randwalk( nrow(d), sparse=TRUE )
  d$y.rw.mom <- d$y.fac
  contrasts( d$y.rw.mom ) <- contr.randwalk( nrow(d), momentum=0.25, sparse=TRUE )

  if ( do.graph1 ) {
    plot( d$y, d$o, type='b', xlab='Year', ylab='X', lwd=2 )
    lines( d$y, d$a, type='b', lty=3 )
    abline( h=mean( d$o ), col='gray' )
  }

  # function to fit models
  fit.model <- function( form, num.folds=3, num.times=20, alpha=0.75, fit.incl=1:n, do.graph=do.graph1,
```

```
print.cvm=print.cvm1, col='blue', lty=1, lwd=1 ) {
        x <- sparse.model.matrix( form, d )
       # remove the intercept (first column) since the penalized regression function adds it automatically
        # (but create the matrix with the intercept, since sometimes the model.matrix function won't use the correct contrasts for the first term of the formula without an intercept)
        x <- x[, -1]

        fit <- glmnet( x[fit.incl,], log( d$o[fit.incl] ), family='gaussian', standardize=FALSE, alpha=alpha )  #, lambda.min.ratio=1e-10 )
        fit.cv <- lapply( 1:num.times, function(i) cv.glmnet( x[fit.incl,], log( d$o[fit.incl] ), family='gaussian', standardize=FALSE, alpha=alpha, nfolds=num.folds, lambda=fit$lambda ) )
        lambda <- mean( sapply( fit.cv, function(x) x$lambda.min ) )
        cvm <- mean( sapply( fit.cv, function(x) min( x$cvm ) ) )
        p <- exp( predict( fit, newx=x, type='response', s=lambda ) )
        if ( do.graph ) lines( d$y, p, col=col, lwd=lwd, lty=lty )
        if ( print.cvm ) print( cvm )
        p
      }

      d$p.mean <- mean( d$o )
      d$p.fac <- fit.model( ~ y.fac, col='blue', print.cvm=print.cvm1 )
      d$p.rw <- fit.model( ~ y.rw, col='red', lwd=2, print.cvm=print.cvm1 )
      d$p.rw.mom <- fit.model( ~ y.rw.mom, col='red', lty=2, lwd=2, print.cvm=print.cvm1 )

      # kalman filter
      d$p.kf <- exp( kf.fit( log( d$o ), incl=1:n )$x )
      if ( do.graph1 ) lines( d$y, d$p.kf, col='green' )

      # kalman filter with bagging
      fit.kf.bag <- rowSums( exp( sapply( 1:10, function(i) kf.fit( log( d$o ), incl=sample( 1:n, round( n * (2/3) ) ) )$x ) ) ) / 10
      d$p.kf.bag <- fit.kf.bag
      if ( do.graph1 ) lines( d$y, d$p.kf.bag, col='green', lty=2 )

      # spline
      fit.sp <- gam( log( o ) ~ s( y ), data=d )
      d$p.sp <- exp( predict( fit.sp, newdata=d ) )
      if ( do.graph1 ) lines( d$y, d$p.sp, col='gold', lwd=2 )

      # bayesian model
      if ( do.bayes ) {
        # model on log of observed instead of handling in model, so that comparable to other models
        data.stan <- list(
          x=log( d$o )
          , N=n
          , y_rw_scale=5
        )
        fit.bayes <- stan( 'rw_test.stan', data=data.stan, chains=3, iter=1000 )
        stan.obj <- extract( fit.bayes, permuted=TRUE )
        #traceplot( fit.bayes )
        #stan_dens( fit.bayes, separate_chains=TRUE )
        d$p.bayes <- exp( sapply( 1:n, function(i) mean( stan.obj[['y']][,i] ) ) )
        if ( do.graph1 ) lines( d$y, d$p.bayes, col='pink', lwd=2 )
      }

      # calculate prediction errors of each method
      rmse.pred <- c()
```

```
  for ( f in names(d)[ grep( 'p.', names(d), fixed=TRUE ) ] ) {
    rmse.pred <- c( rmse.pred, rmse( d[[f]], d$a, 1:n ) )
    names( rmse.pred )[length(rmse.pred)] <- f
  }
  rmse.pred
} )

data.frame( method=rownames( results ), rmse=sapply( 1:nrow(results), function(i) mean( results[i,] ) ) )
```

Save in separate file names "rw_test.stan":

```
data {
  int N;
  vector<lower=0>[N] x;
  real<lower=0> y_rw_scale;
}

parameters {
  vector<lower=-2, upper=2>[N - 1] y_rw;
  real<lower=0, upper=2> y_sd;
  real<lower=0> y_rw_sd;
  real y1;
}

transformed parameters {
  vector[N] y;

  y[1] = y1;
  for ( i in 2:N ) {
    y[i] = y[i - 1] + y_rw[i - 1];
  }
}


model {
  y1 ~ uniform( 1, 5 );
  y_rw_sd ~ cauchy( 0, y_rw_scale );
  y_rw ~ normal( 0, y_rw_sd );
  x ~ normal( y, y_sd );
}
```

APPENDIX B: Loss Ratio Example

```
library( HDtweedie )  # elastic net for Tweedie family
library( Matrix )  # for building sparse matrices
library( ggplot2 )

# function for creating dummy variables that include a column for every single value (unlike the default behavior in GLMs that leave out one value).
# this is needed for penalized regression (i.e. credibility models), since every segment is credibility weighted back towards the intercept (mean)
contr.pen <- function( n, contrasts=TRUE, sparse=FALSE ) contr.treatment( n, contrasts=contrasts, sparse=sparse )
contr.pen.sparse <- function( n, contrasts=TRUE, sparse=TRUE ) contr.treatment( n, contrasts=contrasts, sparse=sparse )

# function for creating dummy variables for implementing random walks
contr.randwalk <- function( n, contrasts = TRUE, sparse = TRUE, momentum=0, rel.cred=1, stdize=TRUE ) {
  if (length(n) <= 1L) {
    if (is.numeric(n) && length(n) == 1L && n > 1L)
      levels <- seq_len(n)
    else stop("not enough degrees of freedom to define contrasts")
  } else { levels <- n }
  levels <- as.character(levels)
  if ( sparse ) {
    cont <- Matrix( c(0), nrow=length(levels), ncol=length(levels) - 1, sparse=TRUE )
  } else {
    cont <- matrix( c(0), nrow=length(levels), ncol=length(levels) - 1 )
  }
  for ( i in 2:n ) {
    cont[, i - 1] <- ifelse( 1:n < i, 0, ifelse( rep( momentum, n ) == 1, ( 1:n - i + 1 ), ( 1 - momentum ^ ( 1:n - i + 1 ) ) / ( 1 - momentum ) ) )
    cont[, i - 1] <- cont[, i - 1] - mean( cont[, i - 1] )
  }
  if (contrasts) {
    colnames(cont) <- levels[-1]
  }
  # standardize
  if ( stdize ) {
    for ( i in 1:ncol(cont) ) {
      cont[,i] <- cont[,i] / sum( diff( cont[,i] ) ^ 2 ) ^ 0.5
    }
  }

  cont <- cont * rel.cred
  cont
}

num.segs <- 3
num.subsegs.each <- 2
num.subsegs <- num.segs * num.subsegs.each
# for finding the appropriate segment for each subsegment
seg.map <- c( sapply( 1:num.segs, function(i) rep( i, num.subsegs.each ) ) )
num.yrs <- 10
years.corr <- 0.2
ldf <- 3 ^ ( 0.65 ^ ( ( num.yrs - 1 ):0 ) )
```

```
# --- simulate losses ---
overall.avg <- log( 700 )
# segment relativities
seg.rel <- rnorm( num.segs, 0, 0.1 )
# subsegment relativies
subseg.rel <- rnorm( num.subsegs, 0, 0.1 )
# random walk coefs
rw.chg.init <- rnorm( num.yrs, 0, 0.2 )
rw.seg.chg.init <- sapply( 1:num.segs, function(i) rnorm( num.yrs, 0, 0.15 ) )
rw.subseg.chg.init <- sapply( 1:num.subsegs, function(i) rnorm( num.yrs, 0, 0.1 ) )
# add correlation (momentum) to the changes
rw.chg <- rw.chg.init
rw.seg.chg <- rw.seg.chg.init
rw.subseg.chg <- rw.subseg.chg.init
for ( i in 2:num.yrs ) {
  rw.chg[i] <- years.corr * rw.chg.init[i - 1] + sqrt( 1 - years.corr ^ 2 ) * rw.chg.init[i]
  for ( j in 1:num.segs ) rw.seg.chg[i, j] <- years.corr * rw.seg.chg.init[i - 1, j] + sqrt( 1 - years.corr ^ 2 ) * rw.seg.chg.init[i,
j]
  for ( j in 1:num.subsegs ) rw.subseg.chg[i, j] <- years.corr * rw.subseg.chg.init[i - 1, j] + sqrt( 1 - years.corr ^ 2 ) *
rw.subseg.chg.init[i, j]
}

# using simulated values, produce the data, with error. (note this is a crude simulation)
for ( subseg in 1:num.subsegs ) {
  seg <- seg.map[subseg]
  x <- exp( overall.avg + seg.rel[seg] + subseg.rel[subseg]
    + cumsum( rw.chg ) + cumsum( rw.seg.chg[,seg] ) + cumsum( rw.subseg.chg[,subseg] )
    + rnorm( num.yrs, 0, 0.2 * ldf ) ) / ldf
  d.add <- data.frame( seg=seg, subseg=subseg, yr=1:num.yrs, loss=x, ldf=ldf, ep=1000 )
  if ( subseg == 1 ) {
    d <- d.add
  } else {
    d <- rbind( d, d.add )
  }
}
d$seg <- factor( d$seg )
d$subseg <- factor( d$subseg )

# cape cod-like method to develop losses and weight years
d$ult.lr <- d$loss * d$ldf / d$ep
# this will give the greener years less weight
d$used.ep <- d$ep / d$ldf

# --- build the model ---
# create random walk variable
# set different values of the momentum (and relative credibility if want) here
rw.momentum <- 0
rw.rel.cred <- 1
# rw.momentum <- 0.1
d$yr.rw <- factor( d$yr )
contrasts( d$yr.rw ) <- contr.randwalk( num.yrs, sparse=TRUE, momentum=rw.momentum, rel.cred=rw.rel.cred )

# create the modeling  matrix that describes the independent variables of the model (use a sparse matrix)
# change the default contrasts (i.e. dummy variable encodings) to create columns for every level of a variable
options( contrasts = c( 'contr.pen.sparse', 'contr.pen.sparse' ) )
# (no intercept in formula since glmnet function adds automatically)
x <- sparse.model.matrix( ~ seg + subseg + yr.rw + seg:yr.rw + subseg:yr.rw, d )
```

```
    # remove the intercept (first column) since the penalized regression function adds it automatically
    # (but create the matrix with the intercept, since sometimes the model.matrix function won't use the correct contrasts for the first term of the formula without an intercept)
    x <- x[, -1]
    options( contrasts = c( 'contr.treatment', 'contr.poly' ) )

    # fit the model
    # (HDtweedie doesn't support sparse matrices unlist glmnet, so need to convert back to unsparse matrix)
    fit <- HDtweedie( as.matrix( x ), d$ult.lr, weights=d$used.ep, p=1.9, alpha=0.75, standardize=FALSE, lambda.factor=1e-3 )
    # cross validate to get the optimal penalty parameter
    set.seed( 1112 )
    fit.xval <- cv.HDtweedie( as.matrix( x ), d$ult.lr, weights=d$used.ep, p=1.9, alpha=0.75, standardize=FALSE, lambda.factor=1e-3, nfolds=5 )
    # cross validated deviance (error) of the model - use this to test the momentum parameter
    min( fit.xval$cvm )
    lambda <- fit.xval$lambda.min
    # check that the chosen penalty isn't at the ends
    which( lambda == fit.xval$lambda )
    length( fit.xval$lambda )

    # make predictions
    d$ult.lr.pred <- predict( fit, as.matrix( x ), type='response', s=lambda )[,1]

    # --- graph results ---
    # one at a time
    par( mfrow=c( 3, 2 ) )
    for ( subseg in 1:num.subsegs ) {
      plot( 1:num.yrs, d$ult.lr[d$seg == seg.map[subseg] & d$subseg == subseg], type='b', xlab='Year', ylab='LR'
        , main=paste( 'Segment: ', seg.map[subseg], ', Subsegment: ', subseg, sep='' ) )
      lines( 1:num.yrs, d$ult.lr.pred[d$seg == seg.map[subseg] & d$subseg == subseg], col='blue' )
    }
    par( mfrow=c( 1, 1 ) )


    d.plot <- rbind( data.frame( Year=d$yr, seg=d$seg, subseg=d$subseg, Segment=paste( d$seg, d$subseg ), LR=d$ult.lr, Line='Actual' )
      , data.frame( Year=d$yr, seg=d$seg, subseg=d$subseg, Segment=paste( d$seg, d$subseg ), LR=d$ult.lr.pred, Line='Predicted' ) )

    # each segment together
    seg <- 1
    ggplot( d.plot[d.plot$seg == seg,], aes( Year, LR ) ) + geom_line( aes( color=Segment, linetype=Line ) )

    # all together
    ggplot( d.plot, aes( Year, LR ) ) + geom_line( aes( color=Segment, linetype=Line ) )
```

10. REFERENCES

[1] Carlin, B. 1992. State Space Modeling of Non-Standard Actuarial Time Series. Insurance: Mathematics and Economics. October 1992. Volume 11, Issue 3. pp 209-222.
[2] De Jong, P. 2005. State Space Models in Actuarial Science. Macquarie University Actuarial Studies, Research Paper. No. 2005/02. July 2005.
[3] De Jong, P. and Zehnwirth, B. 1983. Claims Reserving, State-Space Models and the Kalman Filter. Journal of the Institute of Actuaries, 110, pp 157-181.

[4] Dempster, A. P., Laird N. M., and Rubin D. B. 1977. Journal of the Royal Statistical Society. Series B (Methodological). Vol. 39, No. 1 (1977), pp. 1-38
[5] Evans, Jonathan P. and Frank Schmid. 2007. Forecasting Workers Compensation Severities and Frequency Using the Kalman Filter. Casualty Actuarial Society Forum, 2007: Winter, pp 43–66
[6] Frees, E. and Gee, L. 2016. Rating Endorsements Using Generalized Linear Models. Variance 10:1, 2016, pp 51-74
[7] Friedman, J., Hastie, T., and Tibshirani, R. 2009. Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1):1, 2010
[8] Gelman, A. 2008. Scaling regression inputs by dividing by two standard deviations. Stat. Med. 27: 2865–2873.
[9] Geweke, J. 1977. The Dynamic Factor Analysis of Economic Time Series. Latent Variables in Socio-Economic Models. Amsterdam, North Holland.
[10] Hastie, T., Tibshirani, R., and Friedman, J. 2009. Elements of Statistical Learning, Volume 2, New York: Springer, 2009
[11] Hastie, T. and Qian, J. 2014. Glmnet Vignette. http://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html
[12] Kim C. and Nelson C. 1999. State-Space Models with Regime Switching. MIT Press, 1999. pp. 29-30, 36-37
[13] Korn, U. 2016. An Extension to the Cape Cod Method with Credibility Weighting Smoothing. Casualty Actuarial Society E-Forum, 2016: Summer, pp 5–27
[14] Taylor, G. and McGuire, G. 2007. Adaptive Reserving using Bayesian revision for the Exponential Dispersion Family. Centre for Actuarial Studies, Department of Economics, University of Melbourne.
[15] Williams, B., Hansen, G., Baraban, A., and Santoni, A. 2015. A Preactical Approach to Variable Selection – A Comparison of Various Techniques. Casualty Actuarial Society E-Forum, 2015: Summer, pp 4-40
[16] Wüthrich, M.V. and Merz, M. 2008. Stochastic Claims Reserving Methods in Insurance. Wiley.
[17] Zehnwirth B. 1996. Kalman Filters with Applications to Loss Reserving. Research Paper Number 35, Centre for Actuarial Studies, The University of Melbourne.
[18] Zou, H. and Hastie, T. 2005. Regularization and Variable Selection via the Elastic Net. J. Royal. Stat. Soc. B 2005;67(2), pp 301-320

Biography of the Author

Uri Korn is a Director of Predictive Modeling for AIG, Client Risk Solutions where he uses advanced analytics to assist clients in reducing their losses. Prior to that, he was an AVP & Actuary at Axis Capital serving as the Research and Development support for all commercial lines of insurance. He has published papers on non-aggregated loss development techniques, time series, and several papers on practical approaches to credibility. Recently, he was awarded the 2017 Ratemaking Prize for the best call paper. He graduated from the University of Pennsylvania in 2003 with a BSE in Computer Science in Engineering and is a Fellow of the Casualty Actuarial Society.