Paper SAS538-2017

Fast Implementation of State Transition Models

Shannon D. Clark, PhD, SAS Institute Inc., Cary, NC

ABSTRACT

Implementation of state transition models for loan-level portfolio evaluation was an arduous task until now. Several features have been added to the SAS® High-Performance Risk engine that greatly enhance the ability of users to implement and execute these complex, loan-level models. These new features include model methods, model groups, and transition matrix functions. These features eliminate unnecessary and redundant calculations; enable the user to seamlessly interconnect systems of models; and automatically handle the bulk of the process logic in model implementation that users would otherwise need to code themselves. These added features reduce both the time and effort needed to set up model implementation processes, as well as significantly reduce model run time. This paper describes these new features in detail. In addition, we show how these powerful models can be easily implemented by using SAS® Model Implementation Platform with SAS® 9.4. This implementation can help many financial institutions take a huge leap forward in their modeling capabilities.

INTRODUCTION

This paper introduces three new features available in SAS Model Implementation Platform and SAS High-Performance Risk. These are model groups, model methods, and transition matrix functions. These new features accelerate the implementation and execution of complex state transition models, modularize the code so that it becomes more manageable, and separates the process logic from the model logic. These new features allow the user to focus their attention on their models instead of on the machinery needed to implement such models. This paper presumes a basic understanding of state-transition modeling techniques.

MODEL METHODS

Model methods provide a new way of declaring your models. These methods allow any model to be completely contained inside a single method that can be called explicitly or implicitly if the model is inside of a model transition matrix. Similar to a function, these methods allow for very compartmentalized code. The resulting code base is cleaner, more manageable, and easier to understand. In addition, this allows us to further break down a model into component parts. This breakdown enables significant gains in efficiency due to removing unnecessary calculations.

Here is a sample model method:

```
method example kind=model;
   block STATIC_INIT;
      static_example1=0.03+0.25*origination_fico+…;
   endblock;
   block AGE_INDEXED;
      do _HORIZON_=1 to _MAXHORIZON_;
         _ai_example1[_HORIZON_]=0.35*(LoanAge+_HORIZON_)+…;
      end;
   endblock;
   block MAIN;
      _xbeta_ = static_example1 + _ai_example1[simulationHorizon] +
0.45*var1+…;
     _model_.result = _xbeta_;
   endblock;

endmethod;
```

There are three distinct blocks in the above method. These blocks enable the segregation of component variables in the model into one of three types, which govern when the variables are calculated:

- The STATIC_INIT block is reserved for variables where their value is known at the beginning of an analysis and is never going to change. For example, a variable such as *origination_fico* is common in retail credit risk models. For a single model, this variable's contribution is unchanging. That is, its value is independent of the horizon, the scenario, or the simulation.
- The AGE_INDEX block is reserved for variables where their value is going to change in a way that is known at the beginning of an analysis. Age of the loan is a very common example of this type of variable. As of the snapshot date, each loan will have an age. In the first month of the analysis, the loan will be one month older. Even though this variable is changing, the user knows its value in every forecasted horizon.
- The MAIN block is reserved for variables where there is no knowledge of their future value. These variables must be calculated on the fly.

When these models are calculated, all the variables in the STATIC_INIT block are collapsed into a single value. The variables in the AGE_INDEX block are collapsed into an array of values that are available through an index. The default index is months since snapshot date, but any index is available to the user. The variables in the MAIN block are not executed until the model is called. The initial calculation from the MAIN block is then added to the result from the STATIC_INIT block and the appropriate indexed value from the AGE_INDEX vector. The output of this final calculation can then be transformed according to the model's link function {logit, cLogLog, and so on}.

With this model structure, the underlying engines can make efficient calls to each of the models needed in the analysis. In the case of state transition models, there are likely numerous models that need to be evaluated. This makes it imperative that these calls are as efficient as possible, especially when using Monte Carlo methods.

MODEL GROUPS

After the models are defined in the system, there needs to be a way to group them so that they will work in concert with one another. Transition matrices have always existed in SAS High-Performance Risk, but there might be other models needed that are not explicitly inside of the transition matrix. Model Groups are a way to combine several objects together and to define how they will work with each other. These objects include models, transition matrices, parameter matrices, cash-flow data sets, and the logic that governs all of these individual pieces.

Model Groups serve as a major collection of logic in the model implementation system. This grouping is necessary to enable any bank to run their portfolio, which can contain any number of products. For example, a bank's product mix could include residential fixed prime mortgages as well as HELOCs associated with those mortgages. It is highly probable that the bank has completely different models as well as completely different logic for their HELOCs as opposed to their 1st lien mortgages. Model Groups are what allows the compartmentalization of the logic and models associated with each product.

TRANSITION MATRIX METHODS

In the SAS High-Performance Risk engine, there have been some changes to the existing transition matrix methods and a new method was added that greatly assists users with executing state transition models. These items are detailed below.

TRANSMAT_ELEM

The TRANSMAT_ELEM function has been in the HPRISK procedure for some time. However, it now includes some important changes that make handling state transition probabilities considerably easier. When handling these state-transition systems, the models inside can vary greatly. Consider the following simple delinquency transition matrix.

Figure 1. Sample State Transition Matrix

	Cur	D30	D60	D90+	Prep	Default
Cur		logit			proportionalHazard	mean
D30	logit		logit		logit	logit
D60	logit	logit		logit	logit	logit
D90+	logit	logit	logit		logit	logit

The colored squares represent where individual models are needed. For a current loan, there are many possible transitions available. They can stay current, roll to a D30 state, prepay their loan, or go straight to default. This last state is included for emphasis. In general, this transition is quite rare, but it happened enough during the recent housing crisis that it can't be ignored.

The models governing each of these transitions could be of the same type, but often the types differ. In this example, the transitions are color coded with specific model types. Blue is a base transition (meaning no real model), green is a logistic model, yellow is a proportional hazards model, and gray is a model that generates the mean.

While there is nothing wrong with having this mix of different model types, extreme care must be taken when using the results of each of these models and turning all of them into a row vector of probabilities. Some of the results are oddsRatios, some of the results are probabilities, and some of the results can be either oddsRatios or probabilities depending on your preference. TRANSMAT_ELEM understands these differences and can convert all the results to a row vector of probabilities automatically, therefore eliminating a major source of very error prone code.

Here is a sample call:

```
Call TRANSMAT_ELEM(tmx_name,from_state,.,"Prob",rc);
```

The "Prob" in the above call can be replaced with "Non_Normalized" to return non-normalized results or "Thresh" to return an inverse CDF.

TRANSMAT_BALANCE

The TRANSMAT_BALANCE function is a completely new function in the HPRISK procedure. The TRANSMAT_BALANCE function allows a detailed summary of the results of all the models associated with the model group to be created. This summary is helpful for analysis and for implementing state transition models because any defect in your implementation is likely to cause a visible change in the result. This function creates a weighted roll-rate matrix of the transitions generated by the set of models in the supplied transition matrix. The most common weighting schemes are balance-weighted and count-weighted.

Here are two sample calls:

```
Call TRANSMAT_BALANCE(tmx_name,from_state,to_state,Balance);
Call TRANSMAT_BALANCE(tmx_name,from_state,to_state,1);
```

The first call would generate a balance-weighted result, whereas the second would generate a count-weighted result. The resulting *TransBalance* matrix uses values from all the horizons that are associated with your analysis and combines them together. Here is an example of what such a TransBalance matrix might look like.

Figure 2. Sample TransBalance Matrix

	Cur	D30	D60	D90+	Prep	Default
Cur	0.96	0.02			0.015	0.005
D30	0.15	0.6	0.2		0.01	0.04
D60	0.13	0.1	0.3	0.4	0.005	0.065
D90+	0.1	0.05	0.02	0.75	0.002	0.078

CONCLUSION

These new features added to SAS High-Performance Risk greatly assists anyone interested in implementing state-transition modeling frameworks. They are mixed across all stages of the development and implementation process. From the initial state of describing your individual models, to implementing the logic necessary to use those models, and finally to examining the results of the entire system. These improvements are meant to lessen the burden previously borne by the implementation team and to greatly reduce the time needed to implement such systems.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Shannon D. Clark, PhD
SAS Institute, Inc.
100 SAS Campus Drive
Cary, NC 27513
Shannon.Clark@sas.com