Paper 153-2008

# SAS/OR®: Rigorous Constrained Optimized Binning for Credit Scoring

Ivan Oliveira, Manoj Chari, Susan Haller, SAS Institute Inc., Cary NC

## ABSTRACT

Credit scoring can be defined as a statistical modeling technique used to assign risk to credit applicants or to existing credit accounts. We present a new process that enhances the formulation and solution approach in the SAS® system during the so-called "binning" phase by exploiting SAS/OR optimization capabilities to approach the problem from a mathematically rigorous perspective.

Usually, attributes such as age, income, and so on, are segmented into grouping intervals, with the aim of creating bins for scorecards that maximize correlation with these attributes. A critical aspect of the binning process is the enforcement of various constraints, such as minimum/maximum number of bins, minimum/maximum bin widths, and maximum number of observations per bin. These requirements significantly complicate the binning process. Our approach is rigorous in the sense that global linear constraints are implemented exactly with the use of mixed-integer linear programming. Furthermore, the methodology can be extended to a fully rigorous approach (which incorporates an additional nonlinear constraint of imposed monotonicity, at the cost of computational time) within the same mathematical programming context by the addition of variables and constraints related to "weight of evidence" (WOE). The key enabling factor is the mathematical programming model structure and careful implementation of constraints. We present some results of the binning process proposed here and related scorecards, and we compare the solutions against the current state-of-the-art approach.

The methodology presented here will be available through SAS® Enterprise Miner™ for Credit Scoring in a near-future release.

## INTRODUCTION

A variable transformation called "binning"—which maps variables from a large ordered set or a continuous range into a small set of bins—is a typical first step in building scorecards for various applications (Siddiqi 2006). Analysts rely on this technique to provide a robust, interpretable solution to the problem of representing inherent nonlinearities of data in the simple predictive model of a scorecard.

A binning variable is usually called a *characteristic* in this context. Age, income, and outstanding loans are typical characteristics that influence the value of an applicant's score. For example, a 27-year-old applicant for a loan might find that her score is based on the fact that she falls in the 24–30 age group. This paper presents results of a new process that rigorously computes these ranges (bins) to optimally enhance the predictive power of the credit scoring model.

The binning process is often subject to a number of arbitrary constraints, requiring the analyst to spend significant resources carrying out the first stage of scorecard engineering. For example, in order to provide a statistically meaningful model it might be required that each bin contain *at least* a certain number of good/bad/total observations ("bad" usually denotes default on a payment of loan). To prevent any particular bin from dominating the results, it might also be required that bins have *at most* a certain number of (good/bad/total) observations. Bins might also be required to have a *minimum and/or maximum width* (the difference between the bounds of each range), and certain values must or must not be allowed as *knots*; that is, bins must not start at 73, or a bin must start at 80.

Certainly the most difficult type of constraint is the requirement that certain patterns must emerge from the aggregated scoring function. For example, the United States Equal Credit Opportunity Act (USDOJ 2006) imposes the constraint that elderly applicants must not be assigned lower score rates than applicants in younger age groups. This constraint would require score weights, which are based on a measure of risk called the "weight of evidence" (WOE), to be monotonically increasing. This type of constraint is particularly difficult because of its nonlinear relation to the variable count. Furthermore, it is best to impose any binning strategy based on some optimal representation of the data, which can be especially unclear when such difficult constraints are present.

In summary, we are interested in addressing the following types of constraints (possibly simultaneously):

- minimum and/or maximum percentage of observations per bin (good, bad, and total)
- minimum and/or maximum bin width
- minimum and/or maximum number of bins
- minimum aggregate WOE difference between bins
- monotonicity of aggregate bin WOE values (increasing, decreasing, or a combination)

One approach to creating a binning solution is to build it incrementally, starting with a fine discretization of the data (say, every age is an initial bin) and merging bins in some sequential manner. This approach has been used traditionally, and some variant of it is found in many typical credit applications. Consider the following example as an illustration of the challenges faced by software or analysts when attempting to bin a characteristic in this manner.

Suppose that at a particular processing stage a given bin does not satisfy the *minimum percentage of total observations* requirement. We can try to increase this bin's count by moving the left or right knot and thus increase the bin's range. However, this decreases the range of neighboring bins, meaning that they can now potentially violate this or another constraint, even if previously satisfied. It might be better to merge this bin with a neighbor, but it is not clear which to choose, and it is not clear how that action will impact all other bins. The merged bin might in fact be too large and violate the *maximum percentage of observations* constraint, or perhaps it violates only the *maximum percentage of "bad" observations* constraint. Merging always impacts the *total bin count* constraint, and the aggregate WOE curve might no longer be monotonic. Merging somewhere else could have been better at maintaining monotonicity. If we start with an initial collection of bins to be fixed, it might not even be clear which bin to focus on first. Since any future configuration will be dependent on the choices made at a particular phase, we can easily force ourselves into an infeasible situation even when perfectly feasible solutions exist.

Despite (or because of) this complexity, much of present binning for score engineering is performed or at least adjusted manually (perhaps aided by visual or heuristic tools). Yet there are strong incentives to automate tedious portions of this process. These incentives are driven by:

- Cost: many analyst-hours might be necessary if a large number of constraints are to be simultaneously satisfied. This is because the combinatorial nature of the problem defies intuitive, heuristic, or sequential approaches.
- Feasibility: it is entirely possible that a collection of constraints is so restrictive that no solution exists that would satisfy all constraints simultaneously. Mathematically, this is stated as an empty feasible set. Attempting to bin by intuitive methods does not reveal this fact until it is "felt" that all options have been exhausted (after considerable time and effort, and even then without any precise certainty). It would be best to have an *a-priori* guarantee that if a feasible solution exists, it will be found—which implies that all constraints will be correctly satisfied.
- Optimality: intuitive approaches are heuristic in nature and cannot guard against finding only suboptimal solutions. It also might not be possible to prove that a global optimal solution has already been found, leading to unnecessary work in a fruitless attempt to improve it. Rigorous consideration of optimality is critical because it has a direct impact on the predictive power of the final score.

In the approach presented here, we aim to address each of the preceding points by solving the binning problem in a computationally efficient manner while detecting infeasibility and proving optimality with mathematical certainty. The first key to the approach is to consider the problem as a (mixed-integer linear) mathematical program and to make use of the optimization capabilities of SAS/OR. The second key is to exploit the problem structure so that we can solve the resulting problem efficiently.

## BINNING BY MATHEMATICAL PROGRAMMING

The data used for binning a characteristic (or variable) $var$ can be organized into an ordered set $\mathcal{N}(var)$. A characteristic 'age,' for example, that represents ages 20 through 80, in increments of 1 year, would be organized as $\mathcal{N}(age) = \{20, 21, 22, 23, \ldots, 79, 80\}$. For every member $i \in \mathcal{N}(var)$, the number of "good" and "bad" observations can be represented as normalized counts defined as $G_i$ and $B_i$, respectively, for each element of $\mathcal{N}(var)$. Another important statistic is the total number $S_i$ of observations (both good and bad). Furthermore, a "weight of evidence" ($WOE_i$) value can be defined for every element of $\mathcal{N}(var)$:

$$WOE_i = \log \frac{G_i}{B_i} \tag{1}$$

$WOE_i$ represents a measure of risk of element $i$. A low WOE value implies a high risk, since $B_i$ is large in relation to $G_i$.

Binning is accomplished by selecting knots at particular values of $\mathcal{N}$ that imply intervals that produce a binning result $\mathcal{B}$ for each characteristic. Recalling our 'age' example, knots selected at $20$, $25$, and so on, would produce a solution $\mathcal{B}(age) = \{[20, 24], [25, 29], \dots\}$. Clearly, the difficult part of the analysis is the selection of knots to create a feasible $\mathcal{B}$ that is, in some sense, optimal.

In dealing with members $i \in \mathcal{N}$, we are looking at original data points, whereas when we deal with elements $b \in \mathcal{B}$ we are usually calculating aggregate values (in terms of total good and total bad) based on bin groupings. Given a binning configuration $\mathcal{B}$, the aggregate $WOE_b$ for bin $b$ can be calculated as

$$WOE_b = \log\left(\frac{\sum_{j\in b} G_j}{\sum_{j\in b} B_j}\right), \ \forall b \in \mathcal{B} \tag{2}$$

In our approach, we have formalized the desirable outcome into a precise mathematical program and used the tools of operations research and optimization to arrive at the exact solution as defined. The details of the formulation are proprietary, but the overall mathematical model can be stated as follows:

$$\begin{aligned}
\mathcal{B}^*(var) \ = \ & \arg\min_{\mathcal{B},z} \ \sum_{b\in\mathcal{B}} \sum_{i\in b} S_i |WOE_i - z_b| \\
\text{subject to} \quad & z_b = z_i, && \forall i \in b, \ b \in \mathcal{B}; \\
& \text{Impose monotonicity: } z_{b+1} \geq z_b, && \forall b \in \mathcal{B}; \\
& \text{Constrain bin size}_b, && \forall b \in \mathcal{B}; \\
& \text{Force knots}_k, && \forall k \in \mathcal{K}; \\
& \text{Constrain number of bins.}
\end{aligned}$$

In the current level of development, our binning capability is prototyped in the SAS/OR OPTMODEL procedure. The capability will be available as a standalone procedure and through SAS Enterprise Miner for Credit Scoring in the near future, and the detailed syntax will be documented when available. Here we focus on the fundamental capability and some preliminary results. Regarding the algorithmic approach to solving the model, we note here that the preceding problem is stated as a mixed-integer linear program (MILP) and that SAS/OR tools are applicable (the branch-and-cut algorithm is used via the SAS/OR MILP solver). Since the entire solution space is implicitly searched, these solutions come with certificates of guarantee of feasibility and optimality. See SAS Institute Inc. (2006) for more information about SAS/OR MILP algorithms.

The goals of our preceding optimization model are: (1) introduce a decision variable $z_i$ (which defines $z_b$ based on aggregation), (2) impose constraints based on this decision variable, and (3) minimize the deviation between $z$ and $WOE$ as a result of aggregation of $\mathcal{N}$ into $\mathcal{B}^*$. This last point indicates that we use $z_b$ as a surrogate for $WOE_b$ (since the only available *data* is $WOE_i$). Our model also ensures that all previously listed constraints are satisfied and that $WOE_b \to WOE_i$ as $b \to i$. So far, this model cannot ensure monotonicity of $WOE_b$ for $b \in \mathcal{B}^*$; it can only ensure monotonicity of $z_b$. We show later, however, that this might not be a severe shortcoming when $z_b$ is used as the scoring variable.

Nonetheless, it is conceivable that we require that our results impose strict monotonicity of aggregate $WOE_b$, if at all possible. This means that instead of relying on a surrogate variable for WOE, we optimize based on a variable that represents WOE exactly (if such feasible solutions exist) and impose constraints on this new variable ($w$).

$$\begin{aligned}
\mathcal{B}^*(var) \ = \ & \arg\min_{\mathcal{B},z} \ \sum_{b\in\mathcal{B}} \sum_{i\in b} S_i |WOE_i - z_b| \\
\text{subject to} \quad & w_i = z_i, && \forall i \in b, \ b \in \mathcal{B}; \\
& w_i = \log\left(\frac{\sum_{j\in b} G_j}{\sum_{j\in b} B_j}\right) && \forall i \in b, \ b \in \mathcal{B}; \\
& \text{Impose monotonicity: } w_{b+1} \geq w_b, && \forall b \in \mathcal{B}; \\
& \text{Constrain bin size}_b, && \forall b \in \mathcal{B}; \\
& \text{Force knots}_k, && \forall k \in \mathcal{K}; \\
& \text{Constrain number of bins.}
\end{aligned}$$

The introduction of $w$ and its related constraint achieves our goal of representing $WOE_b$ exactly, and thus produces the exact solution to all our requirements. This comes at a modest cost, however, since the $w$-optimal problem is more difficult to solve than the $z$-optimal problem (which can be thought of as a relaxation of the former). We have developed methodologies to solve both problems in $\mathcal{O}(1)$ second (and better) solution times for a given variable for typical scoring of the $w$-optimal problem. We have observed that the $z$-optimal problem usually solves an order of magnitude faster. We also safeguard against the possibility that no feasible solution to the exact problem might exist, and under these circumstances we find the closest possible solution.

## SAMPLE BINNING AND SCORECARD GENERATION

We present here some results of our methodology to illustrate its application to real data. The variable names have been modified for generality. Missing values must always be accounted for in binning, but they can be considered as another value in the ordered set $\mathcal{N}$. For simplicity, we assign all missing values to their own bin without any loss of generality.

We have used the current version of SAS Enterprise Miner for Credit Scoring with the ARBORETUM procedure (De Ville 2006) to produce the initial set of bins. This is the current methodology employed for credit scoring in the SAS system. These results are later compared to bins produced by our methods. SAS Enterprise Miner for Credit Scoring is used to generate final scores based on the selection of bins for all cases (SAS Institute Inc. 2005).

The data set used originated from a banking application; it consists of 5,837 observations. SAS Enterprise Miner for Credit Scoring identified six characteristics (variables) as being of interest for scoring, which we have renamed var1 through var6. Initial binning coarseness for different characteristics ranged between roughly 30 to 300 members of $\mathcal{N}$, which were aggregated into $\mathcal{B}^*$ for each methodology (current SAS Enterprise Miner for Credit Scoring / $z$-optimal / $w$-optimal).

The constraints imposed on the problem are as follows:

```
1. Minimum WOE differentiation between bins:              0.1
2. Minimum number of bins:                                5
3. Maximum number of bins:                                10
4. Minimum count of GOOD observations per bin:            50
5. Minimum count of BAD observations per bin:             50
6. Minimum percentage of GOOD + BAD observations per bin: 5.0%
7. Monotonicity:            in the direction of general trend
```

In the current version of SAS Enterprise Miner for Credit Scoring, the ARBORETUM procedure can satisfy certain small subsets of these constraints at a time. For example, constraints 3 and 6 can be imposed (we use this option here), but others are ignored. There is also an option to impose monotonicity while relaxing other constraints. The $z$-optimal method is able to satisfy constraints 1–6 simultaneously, whereas the $w$-optimal method is able to satisfy all constraints simultaneously (if such a feasible solution exists). Furthermore, the preceding constraints are only a subset of the (previously mentioned) constraints that our methodology is capable of imposing.

### CURRENT SAS ENTERPRISE MINER FOR CREDIT SCORING BINNING

The first set of results is for binning performed by the current version of SAS Enterprise Miner for Credit Scoring, and represents traditional capabilities. Figure 1 shows the results of the binning aggregation with all six variables (var1–var3 on the top, var4–var6 on the bottom). Missing values have been collected at the end of each binning display and do not play any direct role in the binning process here (see Appendix for detailed binning results).

**Figure 1** Bins and Aggregate WOE Values—SAS Enterprise Miner for Credit Scoring Binning

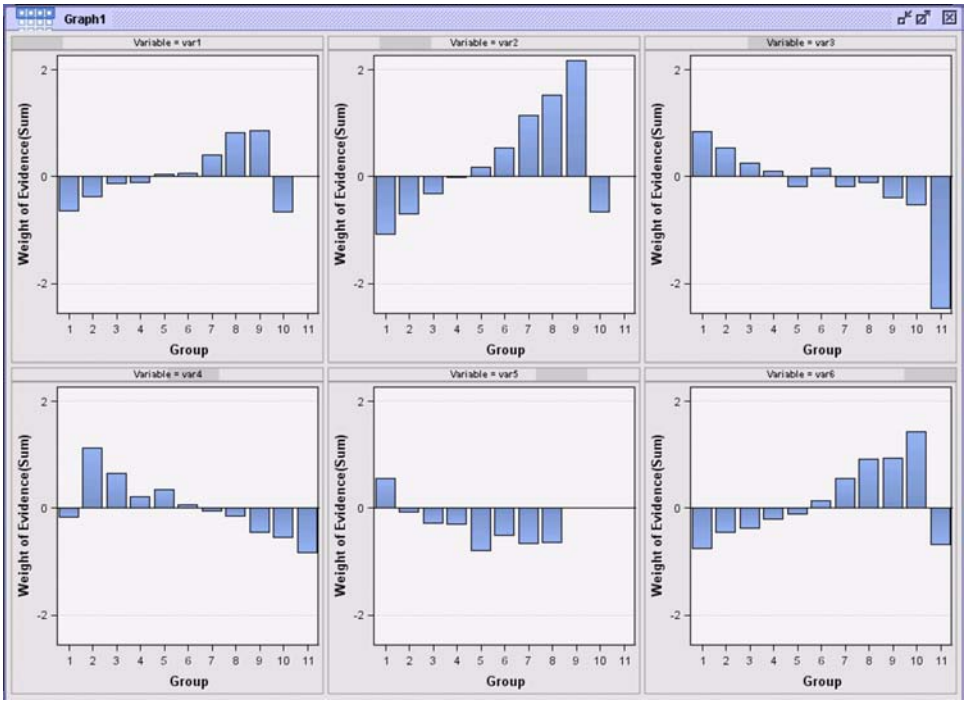
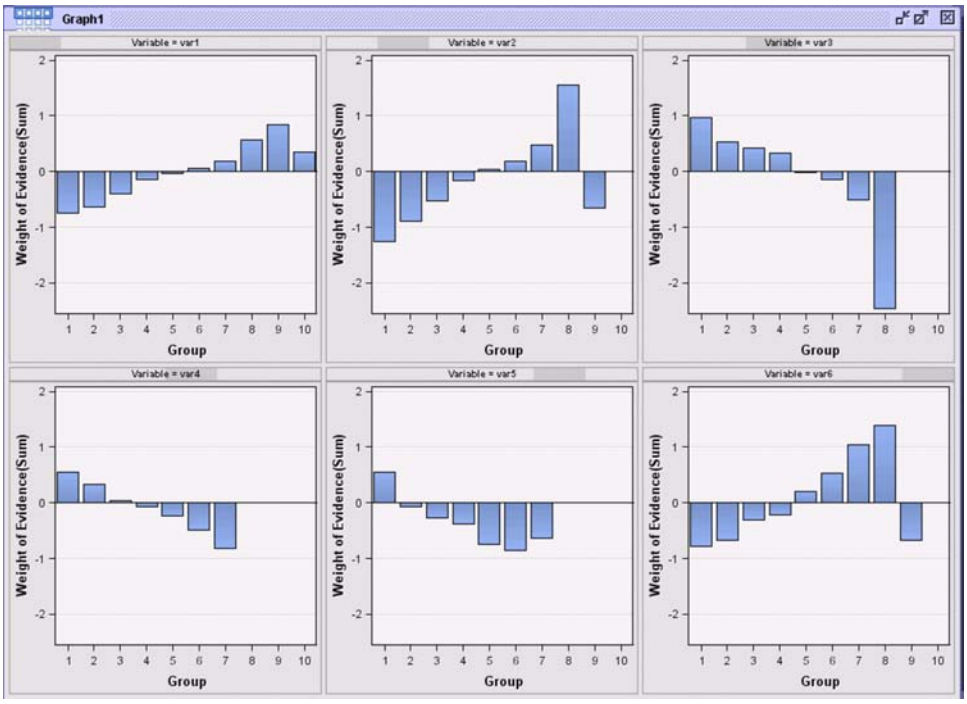



Figure 1 illustrates that binning produced by the current version of SAS Enterprise Miner for Credit Scoring has been able to satisfy the *maximum number of bins* constraint (and examination of data sets reveals adherence to a number of other constraints). However, it is immediately apparent that the WOE values do not maintain monotonicity for certain variables: var3, var4, and var5 (the last "missing values" bin does not participate in the evaluation of monotonicity). Constraints other than 3 and 6 cannot be imposed on the solution; although they might happen to be satisfied in some cases, there is no *guarantee* that they will be satisfied. An analyst working with the preceding results would need to adjust the scoring variable in order to impose monotonicity and any other violated constraints. If done manually, there will be no guidance (other than visual feedback) in determining an appropriate adjustment.

## $z$-OPTIMAL BINNING

Figure 2 shows the binning and $z$ values of the $z$-optimal problem. The variable $z$ is used as a surrogate for WOE in the scoring process, and all constraints are imposed with respect to this variable. Figure 2 indicates that monotonicity is strictly satisfied for all variables in the data set, and all other constraints are now *guaranteed* to be satisfied (and observed to be so in the generated data sets).

**Figure 2** Bins and Aggregate $z$ Values—$z$-Optimal Binning




It is important to note that Figure 2 shows only the surrogate values $z$. If we are interested in the aggregated WOE trend with this method, we must calculate $WOE_b$ after binning has been performed. That is, given $\mathcal{B}^*$ generated by the $z$-optimal solution, Equation (2) yields the WOE values of each bin. Figure 3 shows the results of such a calculation.

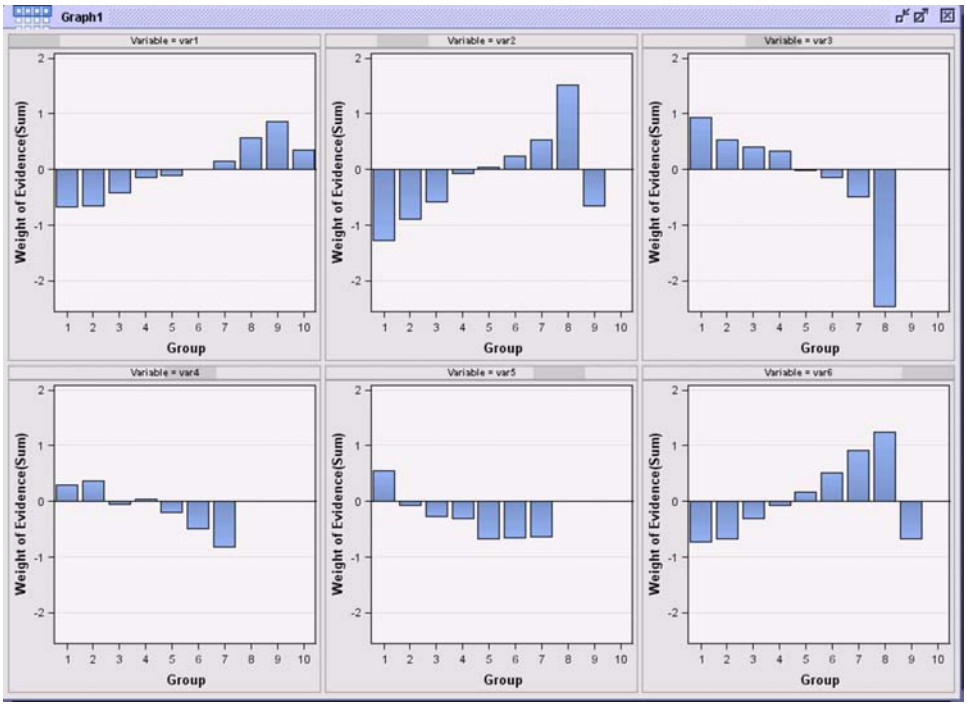
To emphasize the difference in $z$ and $WOE$, we list below these values for var4 for each bin produced as the solution of this method.

**var4 Binning Details (z-optimal):**

| Bin Number | low | high | z | WOE | Good Count | Bad Count | Total Percent Rate |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 18 | 0.55360 | 0.29766 | 1944 | 372 | 40% |
| 2 | 20 | 40 | 0.32839 | 0.36050 | 946 | 170 | 19% |
| 3 | 42 | 48 | 0.03035 | -0.04721 | 248 | 67 | 5% |
| 4 | 50 | 60 | -0.06965 | 0.03325 | 345 | 86 | 7% |
| 5 | 62 | 74 | -0.23792 | -0.19774 | 363 | 114 | 8% |
| 6 | 76 | 98 | -0.48512 | -0.49704 | 491 | 208 | 12% |
| 7 | 100 | 100 | -0.82631 | -0.82631 | 304 | 179 | 8% |
| | | | | | | | ======= |
| | | | | | | | 100% |

Clearly, var4 and var5 violate the monotonicity constraint in terms of WOE. Nonetheless, $z$ can be considered an acceptable adjustment of the WOE prediction. It can be used in building the scorecard as an "adjusted" WOE, which is a common practice in credit scoring. The advantage of using this variable rather than manually fixing the results of Figure 1 is that it is optimal in the sense that it is a best *global* representation of the data as defined by the weighted deviation from $WOE_i$. Furthermore, we will later show that the scorecards resulting from the $z$-optimal method are strong even when compared to the $w$-optimal (exact) method.

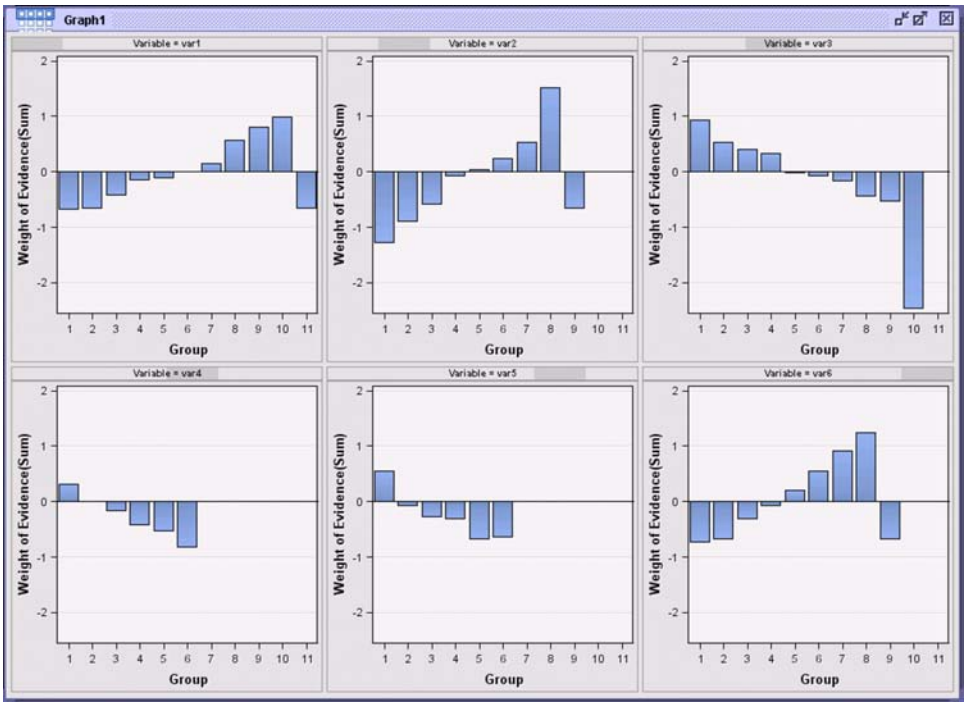
**Figure 3** Bins and Aggregate WOE Values—$z$-Optimal Binning




## $w$-OPTIMAL BINNING

For cases where only exact adherence to the WOE monotonicity constraint is acceptable, we can solve the $w$-optimal problem (at a slight penalty in solution time). Figure 4 shows the aggregate WOE results of the $w$-optimal problem. Note that all variables completely observe the monotonicity of WOE, and these WOE values are no longer calculated *a-posteriori*, but rather are intrinsic in the optimization calculation itself.

**Figure 4** Bins and Aggregate WOE Values—$w$-Optimal Binning



To contrast the results with the $z$-optimal case, we show below the binning results, with $z$, $w$, and $WOE$ for var4. Variables $z$ and $w$ are produced as a result of the optimization, whereas $WOE$ is computed *a-posteriori* for each bin. Note that all three quantities take the same value for each bin, as will always be the case whenever a feasible solution exists. In essence, we have tied the difficult $WOE$ quantity directly to the problem that is solved, and in the process have been able to satisfy (whenever mathematically possible) all constraints, including monotonicity.

**var4 Binning Details (w-optimal):**

| Bin Number | low | high | z | w | WOE | Good Count | Bad Count | Total Percent Rate |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 40 | 0.31780 | 0.31780 | 0.31780 | 2890 | 542 | 59% |
| 2 | 42 | 60 | -0.00119 | -0.00119 | -0.00119 | 593 | 153 | 13% |
| 3 | 62 | 72 | -0.16362 | -0.16362 | -0.16362 | 313 | 95 | 7% |
| 4 | 74 | 82 | -0.41468 | -0.41468 | -0.41468 | 223 | 87 | 5% |
| 5 | 84 | 98 | -0.53554 | -0.53554 | -0.53554 | 318 | 140 | 8% |
| 6 | 100 | 100 | -0.82631 | -0.82631 | -0.82631 | 304 | 179 | 8% |
| | | | | | | | | ======= |
| | | | | | | | | 100% |

## SCORECARD COMPARISONS

Certain measures are available to compare scorecards generated by different binning choices. These are often used as rule-of-thumb values that help detect significant deviation from desired behavior. Our goal was to impose a number of constraints on the problem exactly (or as closely as possible), but of course our final scorecards must be competitive with the good predictability strength of current approaches. Therefore, we require scorecards that rely on our binning results to compare favorably in these measures with the scorecard created by the binning capability in SAS Enterprise Miner for Credit Scoring. Table 1 shows the result of scorecards produced with our methodology with respect to the "KS" (Kolmogorov-Smirnov) and "C" statistics that are often used to evaluate such scores.

**Table 1** Scorecard Comparisons

| Binning Approach | Statistic Used for Scoring | KS | C |
|---|---|---|---|
| SAS Enterprise Miner for Credit Scoring | $WOE$ (violates monotonicity) | 0.412 | 0.768 |
| $z$-optimal binning | $WOE$ (violates monotonicity) | 0.414 | 0.770 |
| $z$-optimal binning | surrogate $z$ (monotonic) | 0.409 | 0.769 |
| $w$-optimal binning | $WOE$ (monotonic) | 0.412 | 0.770 |

Since our approach to binning is so fundamentally different from the traditional approach in SAS Enterprise Miner for Credit Scoring, it is surprising (and reassuring) to note that these statistical measures are all very comparable. This gives us confidence that the methodology's predictive power is strong, even for the case of using the surrogate ($z$) values for scoring. Most importantly, the last two methodologies adhere to all constraints imposed on the problem, including monotonicity.

In some cases, monotonicity violation might be observed even in the solution of the $w$-optimal problem (this is not the case in the example considered here). In those cases, our methodology allows us to declare, with mathematical certainty, that no solution exists that would satisfy strict monotonicity of WOE (given all imposed constraints), no matter what choices are made in binning. Rather than expending time trying to find such a solution, the analyst can now either use the $z$ values as a closest surrogate even for this problem or experiment with relaxing some of the constraints for this particular variable and re-optimizing in real-time with optimal feedback (recall that our goal has been to keep run times in the $\mathcal{O}(1)$-second range per variable). In either case, the analyst is now able to elevate his/her work to higher-level decisions by exploring what-if scenarios and is no longer encumbered by trying to manually find a binning solution that satisfies all constraints.

## CONCLUSION

Our goal has been to present a methodology that allows analysts to focus on high-level decisions by relegating the tedious, complex task of binning to an automated, real-time solution framework. We showed results that solve optimal representations of the binning problem subject to a number of difficult constraints that are often required in credit

applications. The most problematic such constraint, the (nonlinear) monotonicity requirement, has been successfully addressed in two ways: approximately, through a surrogate to the WOE variable; and exactly, whenever such a solution exists. Since we use the tools of mathematical programming to explicitly consider the combinatorial nature of the problem, our solutions are guaranteed to be both feasible and optimal. Run times are observed to be in the range of $\mathcal{O}(1)$ second per variable. The methodology will be available through SAS Enterprise Miner for Credit Scoring in a near-future release.

## APPENDIX: BINNING TABLES

**Table 2** Binning Knots for SAS Enterprise Miner for Credit Scoring Binning (Figure 1)

| Characteristic | Knots | Number Bins | Minimum % Rate |
|---|---|---|---|
| var1 | 53, 77, 109, 129, 148, 171, 205, 261 | 9 | 9.2% |
| var2 | 634, 653, 670, 684, 701, 725, 747, 772 | 9 | 9.3% |
| var3 | 83, 90, 94, 97, 100, 101, 109, 112, 120 | 10 | 5.2% |
| var4 | 9, 24, 30, 36, 50, 66, 75, 85, 100 | 10 | 5.0% |
| var5 | 1, 2, 3, 4, 6, 9 | 7 | 5.1% |
| var6 | 900, 2370, 4400, 7279, 10573, 15772, 22491, 30996.5, 45499.5 | 10 | 9.1% |

**Table 3** Binning Knots for $z$-Optimal Binning (Figure 2 and Figure 3)

| Characteristic | Knots | Number Bins | Minimum % Rate |
|---|---|---|---|
| var1 | 25.5, 46.5, 73.5, 112.5, 127.5, 151.5, 181.5, 214.5, 283.5 | 10 | 6.0% |
| var2 | 607.5, 640.5, 661.5, 673.5, 688.5, 706.5, 721.5 | 8 | 6.1% |
| var3 | 79.5, 85.5, 91.5, 94.5, 97.5, 112.5 | 7 | 6.3% |
| var4 | 19, 41, 49, 61, 75, 99 | 7 | 5.4% |
| var5 | 0.5, 1.5, 2.5, 3.5 | 5 | 5.1% |
| var6 | 513.5, 1540.4, 6675.3, 10783.2, 15918.1, 22079.9, 34403.7 | 8 | 6.3% |

**Table 4** Binning Knots for $w$-Optimal Binning (Figure 4)

| Characteristic | Knots | Number Bins | Minimum % Rate |
|---|---|---|---|
| var1 | 25.5, 46.5, 73.5, 112.5, 127.5, 151.5, 181.5, 214.5, 304.5 | 10 | 6.0% |
| var2 | 607.5, 640.5, 661.5, 673.5, 688.5, 706.5, 721.5 | 8 | 6.1% |
| var3 | 79.5, 85.5, 91.5, 94.5, 97.5, 100.5, 112.5, 118.5 | 9 | 6.3% |
| var4 | 25, 41, 49, 61, 73, 83, 99 | 8 | 5.3% |
| var5 | 0.5, 1.5, 2.5, 3.5 | 5 | 5.6% |
| var6 | 513.5, 1540.4, 6675.3, 10783.2, 16945.1, 22079.9, 34403.7 | 8 | 6.3% |

## REFERENCES

De Ville, B. (2006), *Decision Trees for Business Intelligence and Data Mining Using SAS Enterprise Miner*, Cary, NC: SAS Publishing.

SAS Institute Inc. (2005), "Enterprise Miner 5.1 Fact Sheet," World Wide Web, http://sww.sas.com/training/prework/documents/EM_Fact_Sheet.pdf.

SAS Institute Inc. (2006), *SAS/OR 9.1.3 User's Guide: Mathematical Programming 2.1*, Cary, NC: SAS Institute Inc.

Siddiqi, N. (2006), *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring*, New Jersey: John Wiley & Sons.

USDOJ (2006), "Title VII—Equal Credit Opportunity Act," http://www.usdoj.gov/crt/housing/documents/ecoafulltext_5-1-06.htm.

## Contact Information

Ivan Oliveira, SAS/OR Research and Development
100 SAS Campus Drive
Room R5329
Cary, NC 27513
(919) 531-0097
Ivan.Oliveira@sas.com