Paper SA-03

Collapsing Levels of Predictor Variables for Logistic Regression and Weight of Evidence Coding

Bruce Lund, Marketing Associates, Detroit, MI and Wilmington, DE
Steven Raimi, Marketing Associates, Detroit, MI and Wilmington, DE

ABSTRACT

In binary logistic regression it is routine to collapse the levels of a predictor variable to achieve parsimony while maintaining predictive power. This paper gives a method and SAS® macro for collapsing nominal or numeric predictors. If numeric, the order can be maintained during collapsing. At each step the method maximizes log likelihood among all choices for collapsing. Stopping guidelines are provided. SAS code is given to convert the collapsed levels of the predictor to weight-of-evidence coding. This methodology of collapsing levels applies to the nominal multinomial target as well. All data processing was performed using Base SAS®, SAS Stat®, and JMP®.

INTRODUCTION

The paper gives an algorithm called “Collapse Levels” and SAS code for collapsing the levels of a predictor variable for multinomial logistic regression.[1] A typical purpose for collapsing of levels is to prepare the predictor for use as a CLASS variable in PROC LOGISTIC. Alternatively, the collapsed predictor may be recoded with weight-of-evidence weights.

We consider a nominal or numeric [2] predictor X with K levels and a nominal-scaled target variable Y with L+1 levels (which are coded 0 through L).

The Collapse Levels algorithm iteratively collapses levels of X according one of two modes:

(1) Only adjacent pairs are collapsed. Adjacency is determined by the ordering of the predictor levels (character or numeric). The “missing” level, if present, is not allowed to collapse with another level.
(2) Any pairs can be collapsed. The “missing” level, if present, may collapse with any other level.

The algorithm is optimal in the sense that it selects a pair of levels to be collapsed at each iteration so that the collapsed variable has maximum log likelihood as the predictor of the target Y among all possible eligible pairs.

It turns out that maximizing log likelihood is equivalent to maximizing U(Y|X), the uncertainty coefficient of Y given X.[3]

The algorithm does not automatically “stop” as determined by a stopping criterion. We believe the stopping point requires the judgment of the modeler. We will give two statistics to guide in making the stopping decision:

(1) The percent-change in U(Y|X) from iteration to iteration. U(Y|X) is the uncertainty coefficient of Y given X as computed, for example, by PROC FREQ.

(2) The “x-statistic” which equals the “model concordance” [4] for a binary logistic model. In the multinomial case, the x-statistic is the natural extension of “model concordance”. The x-statistic measures the power of a predictor X to discriminate among levels of the target.

[1] This algorithm was presented for the binary case for nominal-scaled predictors by Raimi and Lund (2011).
[2] In this paper any discussion of “numeric” predictors also applies to a predictor with ordered values.
[3] U(Y|X) is computed by PROC FREQ. U(Y|X) gives the proportional reduction in entropy of Y from knowing the value X.
U(Y|X) = {H(Y) – H(Y|X)}/H(Y) where H(Y) = ∑ p(i)*log(p(i)) for i indexing across the levels of Y and for p(i) giving the probability that Y = y_i. See the book by H Theil (1972) for a full discussion of H(Y) and H(Y|X).
[4] This is the “c” for PROC LOGISTIC; CLASS X; MODEL Y = X;

In the case of a binary target variable we provide SAS code which gives a simple way to create weight-of-evidence coding for the collapsed predictor. This code is applied to the selected stopping point of the collapsing process.

COLLAPSE LEVELS ALGORITHM

The algorithm involves the following steps:

Iteration Step 1
Compute U(Y|X), the uncertainty coefficient of "Y given X". This is iteration step "1".

Iteration Step 2
For each eligible pair (i, j) of levels of X, compute $U(Y|X_{i,j})$. Here, $X_{i,j}$ denotes X with levels i, j being collapsed. Find the pair (i, j) that gives the maximum $U(Y|X_{i,j})$ and collapse these two levels to form a new predictor X(2). This completes iteration step "2". (Eligible Pairs are determined by user selection of collapsing mode and the treatment of "missing".)

Iteration Step r > 2
For each eligible pair (i, j) of either remaining levels or already collapsed levels from X(r-1), compute $U(Y|X_{i,j})$. Find the pair (i, j) that gives the maximum $U(Y|X_{i,j})$ and collapse to form a new predictor X(r). This completes iteration step "r".

After each iteration, consider the stopping guidelines:

Stopping guidelines: Define U_r to be the uncertainty for the optimal collapse at iteration r. The stopping decision may be based on the percentage change in U_r between iterations:

$$PC_r = (U_{r-1} - U_r) / U_{r-1}$$

Another stopping rule is based on the "x-statistic" which is discussed below. Stopping guidelines are further discussed in a later section.

U(Y|X) is related to Log Likelihood. Let LL(X) be the log likelihood for the Logistic Model Y = X where X is a class variable. Let LL(intercept) be the log likelihood for the intercept-only model. Then:

$$U(Y|X) = [\, LL(\text{Intercept}) - LL(X) \,] / LL(\text{Intercept})$$

Consequently, the algorithm maximizes LL(X), the log likelihood of Y versus X, across all possible collapsed pairs at a given iteration. Equivalently, the likelihood ratio chi-square = (-2) * (LL(Intercept) - LL(X)) is maximized.

Of course, log likelihood is the same quantity that is maximized when fitting PROC LOGISTIC.

The Collapse Levels algorithm is implemented in our SAS macro named **%COLLAPSE_LEVELS** and is discussed in a later section.

X-STATISTIC

We consider a predictor X with K levels and a nominal target Y with L+1 levels where Y = 0 to L. Let $f_r(i)$ give the count of observations where $X = X_i$ and Y = r.

Table 1: Generic Frequency Table

	Y			
X	Y = 0	Y = 1	**…**	Y = L
X_1	$f_0(1)$	$f_1(1)$	**…**	$f_L(1)$
X_2	$f_0(2)$	$f_1(2)$	**…**	$f_L(2)$
…	…	…		
X_K	$f_0(K)$	$f_1(K)$	**…**	$f_L(K)$
SUM	$\sum f_0(i)$	$\sum f_1(i)$	**…**	$\sum f_L(i)$

Definition of x-Statistic - Binary Case (Y has 2 levels): For the binary case, the definition of x-statistic is:

$$\textbf{x-statistic} = 0.5 * [Z / M + 1]$$

where:

$M = \sum f_0(i) * \sum f_1(i)$

$Z = \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \text{Abs} (f_0(i)*f_1(j) - f_0(j)*f_1(i))$ [5] ("Abs" denotes absolute value)

M gives the count of all "informative pairs" of observations.[6]

x-Statistic and Logistic Model Concordance: The x-statistic also equals "**c**", the model concordance for the model: PROC LOGISTIC; CLASS X; MODEL Y = X; [7]

Definition of x-Statistic - Multinomial Case: The model concordance is not defined for a multinomial unordered target. We give a natural extension of the x-statistic to this case.

Let x(r,s) be the x-statistic between X and the restriction of Y to levels r and s.

Let $M(r,s) = \sum f_r(i) * \sum f_s(i)$ and define $M = \sum_{r=0}^{L-1} \sum_{s=r+1}^{L} M(r,s)$

The multinomial extension of x-statistic is given by

$$\textbf{x-statistic} = [\sum_{r=0}^{L-1} \sum_{s=r+1}^{L} M(r,s) * x(r,s)] / M \quad \text{.... Equation 1}$$

This definition is equivalent to this alternative formulation:

$$\textbf{x-statistic} = 0.5 * [Z / M + 1]$$

where Z is given by the multiple summation:

$$Z = \sum_{r=0}^{L-1} \sum_{s=r+1}^{L} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \text{Abs} (f_r(i)*f_s(j) - f_r(j)*f_s(i)) \quad \text{.... Equation 2}$$

("Abs" denotes absolute value)

The x-statistic ranges between 0.5 and 1.0. As levels of the nominal predictor X are collapsed, the associated x-statistics will decrease. A "skeleton" macro for computing the x-statistic is provided at the end of the paper.

An Interpretation of the x-Statistic:

The x-statistic for X and Y equals the model concordance for the model PROC LOGISTIC; CLASS X; MODEL Y = X; The use of CLASS X creates a model with a dummy variable (including the intercept) for each level of X and so CLASS X produces the greatest possible model concordance of any model involving only X.[8]

From equation (1) the multinomial x-statistic is seen to be a weighted sum of binary x-statistics x(r,s). Each of these binary x-statistics equals its binary model concordance where X is a CLASS variable. The specific weights in equation (1) are determined by equation (2) which gives the natural extension of the x-statistic beyond the binary case. So, in this generalized sense, the x-statistic measures the power of X to discriminate between the values of Y.

---

[5] Z measures the departure of the distribution of Y across X from being constant. Assuming $f_1(i) > 0$ for all i we can write:
$\text{Abs} (f_0(i)*f_1(j) - f_0(j)*f_1(i)) = f_1(i)*f_1(j) * \text{Abs} (f_0(i)/f_1(i) - f_0(j)/f_1(j))$. Then Z = 0 if $f_0(i)/f_1(i)$ is constant across i.

[6] Pairs where Y = 0 for one observation and Y = 1 for the other.

[7] We omit a proof of the statement but the steps of a proof would involve:

- Assume without loss of generality that c-statistic of X and Y $\geq$ 0.5
- If $P(Y=1|X=x_i)$ is non-decreasing, then x-statistic of X and Y equals c-statistic of X and Y
- The concordance "**c**" of PROC LOGISTIC; MODEL Y=X; equals c-statistic of X and Y
- The X_i can be re-ordered so that $P(Y=1|X=x_i)$ is non-decreasing. So, x-stat = c-stat for this re-ordering of the X levels.
- But ordering does not matter in the definition of the x-statistic or for model concordance when X is a CLASS variable.

[8] That is to say, X or any transformation of X.

%COLLAPSE_LEVELS MACRO

The %COLLAPSE_LEVELS macro takes an input variable X and a nominal target Y and computes U(Y|X) iteratively for the collapsed levels of X so that each successive collapse is chosen to maximize U(Y|X). %COLLAPSE_LEVELS is based on PROC FREQ for which U(Y|X) is one of the statistics provided by the MEASURES option.

Mode of Collapsing and Limitations: In the case of the collapse-any-pair mode, the number of pairs of levels in the first iteration is K * (K-1) / 2 where K is the number of levels of X. Due to the K-squared growth in the number of pairs, the %COLLAPSE_LEVELS sets a limit of 25 levels for X. In the case of the collapse-adjacent-pair mode, the number of pairs of levels in the first iteration is only K-1. A limit of 75 levels was set. Both of these limits can be reset in the macro by the user. [9]

If a numeric X with more than 75 levels is considered for adjacent-pair collapsing, then a PROC RANK step (or some other fine classing step) is needed to bin X into 75 or fewer bins. In the case of any-pair collapsing of a predictor with more than 25 levels (regarded as unordered), a preliminary subjective collapsing of levels is needed**.** Alternatively, a method based on PROC CLUSTER can be used. See Raimi and Lund (2011) for a discussion and references related to the method based on clustering.

Missing Values: The user may elect to include missing-values in the collapsing. For the collapse-any-pair mode the missing-value level may collapse with any other level. But for the collapse-adjacent-pair mode the missing-value level is not allowed to collapse with any other level.

Further discussion of **%COLLAPSE_LEVELS** Macro is given at the end of the paper.

SAMPLE SIZE PER LEVEL

For direct marketing applications, where there is normally no shortage of data, the minimum count per level might be set at 200. For this sample size, the distribution of Y is likely to be similar in the holdout sample or in an out-of-time sample. Ideally however, the count per level would be much higher than 200.

When collapsing a numeric predictor using adjacent-pair mode, it is often a goal to obtain a monotonic relationship between the collapsed levels and the rate of Y=1 across these levels. Here, it is particularly important to have sufficient sample size per level. This is because a level with small sample-size may, by chance, have a very high or very low rate of Y=1. In this case monotonicity may not be achieved in the collapsing process until a late iteration and meanwhile predictive power may be lost by collapsing other levels.

In the case of a nominal predictor with small sample-size levels, the collapse any-pair process is more likely to collapse a small sample size level earlier in the process. However, a preliminary subjective collapsing of levels should still be considered.

STOPPING CRITERIA

Monitoring the Change in U(Y|X) and x-Statistic during Collapsing and Stopping Point: During collapsing the modeler would monitor the changes in U(Y|X) and the decrease in the x-statistic. Negligible decreases in U(Y|X) and in the x-statistic are justified in order to gain a degree-of-freedom.

U(Y|X): We give a loose guideline of stopping at the iteration before a percentage change of 1% or more occurs. That is, if $(U_{r-1} - U_r) / U_{r-1} = PC_r >= 1\%$, then stop at iteration r-1 or before. Optionally the modeler can monitor the successive differences in PC_r. A second guideline for stopping is when these differences noticeably increase in magnitude.

[9] We think these limits reflect nearly all practical applications. Run times are still acceptable for modestly higher limits. However, in the final PROC PRINT step the COLLAPSE column may wrap since the display could require 3*(K-1) - 1 characters.

x-Statistic: A rule-of-thumb, at least for the binary case, is to consider only those iterations where the x-statistic is above 0.540. In particular, a predictor variable with an initial x-statistic below 0.550 may not merit consideration for inclusion in the model in the first place.

EXAMPLES OF %COLLAPSE_LEVELS

%COLLAPSE_LEVELS is applied to these X-Y tables using the any-pairs mode:

Table 2A: Frequency Table

	Y	
X	Y = 0	Y = 1
A	0	3
B	2	1
C	1	2
D	1	3
SUM	4	9

Table 2B: Frequency Table (Multinomial)

	Y		
X	Y = 0	Y = 1	Y = 2
A	0	2	1
B	1	1	1
C	1	2	0
D	1	1	3
SUM	3	6	5

In Table 3A and Table 3B under the column "Collapsed": "+" indicates "collapsed at this iteration". "_" indicates "already collapsed".

Discussion of Table 3A:
Consider iteration #2. The value U = 0.240115 and its related log likelihood (not shown) for C+D are the maximum among the alternatives for collapsing: A+B, A+C, A+D, B+C, B+D, or C+D.

Table 3A: Results of %COLLAPSE_LEVELS from Table 2A

Iteration	U	Pct Chg in U	x_statistic	Collapsed
1	0.243729		0.7917	NONE
2	0.240115	1.48%	0.7778	C+D
3	0.161267	32.84%	0.6667	B+C_D

Discussion of Table 3B:
The x-statistic gives the natural extension of model concordance to the multinomial case. Again, U and log likelihood have maximum value at each iteration among the alternatives for collapsing.

Table 3B: Results of %COLLAPSE_LEVELS from Table 2B

Iteration	U	Pct Chg in U	x_statistic	Collapsed
1	0.201098		0.7778	NONE
2	0.182881	9.06%	0.7460	B+D
3	0.104051	43.10%	0.6190	A+B_D

OTHER METHOD OF COLLAPSING NOMINAL PREDICTORS FOR THE BINARY CASE

Clustering
A method of collapsing nominal predictors (using any-pairs collapsing) is based on clustering of levels using SAS PROC CLUSTER. This method selects the pair for collapsing which maximizes the Pearson chi-square.[10] A stopping criterion is defined by selecting the iteration which produces the minimum chi-square statistic probability (right tail probability) of association between the target and the collapsed predictor.

The clustering method is illustrated by Manahan (2006) who provides SAS macro code. Manahan attributes the clustering method to M. J. Greenacre.

[10] SAS course notes "Predictive Modeling Using Logistic Regression" (2007).

Comparison of %COLLAPSED_LEVELS and Clustering
Since Clustering maximizes Pearson chi-square and %COLLAPSE_LEVELS maximizes likelihood-ratio chi-square, it is likely that the sequence of collapsing of levels would be the same for any realistic example. See Raimi and Lund (2011) for an example of the two methods applied to the same data set.

Advantages of %COLLAPSED_LEVELS: The clustering method does not apply to the "adjacent-pair" case and the clustering method does not apply to the multinomial case.

Disadvantages of %COLLAPSED_LEVELS: We recommend that the %COLLAPSED_LEVELS for "any-pair" mode be limited to 25 levels.

COARSE CLASSING METHODOLOGY FOR A NUMERIC X AND BINARY Y

Overview: Coarse classing of a numeric (ordinal) predictor variable X is the process of transforming X to discrete "class-labels" by collapsing intervals (adjacent ranges of ordered values) within the range of X.

The coarse classing mapping depends on:

- Determination of the number of intervals in the mapping
- Determination of the "cutpoints" which define lower and upper ends of the intervals

Numeric weights are then assigned to the transformed X to create a new variable for predicting Y. These weights are the weights-of-evidence.[11]

EXAMPLE OF COARSE CLASSING USING %COLLAPSE_LEVELS – ADJACENT PAIRS

To demonstrate this method we consider an example of "income_c" with 12 ordered levels and a binary target Y.

Table 4

	income_c												
Y	01	02	03	04	05	06	07	08	09	10	11	12	**Total**
0	1393	6009	5083	4519	8319	4841	2689	2090	729	292	253	294	36511
1	218	890	932	1035	2284	1593	1053	872	311	136	120	142	9586
Total	1611	6899	6015	5554	10603	6434	3742	2962	1040	428	373	436	46097
P_1 (*)	13.5%	12.9%	15.5%	18.6%	21.5%	24.8%	28.1%	29.4%	29.9%	31.8%	32.2%	32.6%	20.8%

(*) P_1 = column percentage of Y equal to 1. E.g. 13.5% = 218/1611

Step 1: In Table 5 the %COLLAPSE_LEVELS macro was applied to adjacent pairs to collapse "income_c".

We use P_1 to denote the percent of observations where Y = 1 given a value of Income_c (with the dependency on Income_c being implied).

At iteration 5 the c-statistic and x-statistic are equal. This implies that the collapsed pattern is monotonic in relationship to P_1.[12]

By iteration 8 the "Pct Chg in U" has noticeably jumped (1.97%) and the x-stat value has fallen (0.5953) versus iteration 7.

This suggests a detailed inspection of P_1 versus the collapsed income_c need only be performed for iterations 4 through 7.

[11] For a discussion see Finlay (2010) page 120, page 146.
[12] Max(c-stat, 1 - c-stat)) = x-stat if and only if P_1 is monotonic with respect to X. A proof is omitted.

Table 5: %COLLAPSE_LEVELS applied to numeric predictor (adjacent pairs)

Iteration	U	Pct Chg in U	X_STAT	C_STAT	COLLAPSED
1	0.019298	n/a	0.5980	0.5978	NONE
2	0.019297	0.00%	0.5980	0.5978	10+11
3	0.019296	0.01%	0.5979	0.5977	10_11+12
4	0.019295	0.01%	0.5979	0.5977	08+09
5	0.019285	0.05%	0.5978	0.5978	01+02
6	0.019245	0.21%	0.5975	0.5975	07+08_09
7	0.019127	0.61%	0.5971	0.5971	07_08_09+10_11_12
8	0.018751	1.97%	0.5953	0.5953	01_02+03
9	0.018346	2.16%	0.5928	0.5928	04+05
10	0.017506	4.58%	0.5890	0.5890	06+07_08_09_10_11_12
11	0.013146	24.90%	0.5646	0.5646	04_05+06_07_08_09_10_11_12

Step 2: A second macro %RECODE reads the iterations of %COLLAPSE_LEVELS and computes the P_1 values. These P_1 values are shown for iterations 4 and 5 in Table 6.

The collapse of 01 and 02 in iteration 5 creates a monotone increasing pattern for P_1 across its 8 collapsed levels.

In %RECODE a PROC FORMAT with the CNTLIN option is used to transform the original input variable (income_c) to the collapsed coding from iteration 5.

For example: If the format is named INCOME_C_5_FMT, then:
PUT(“01”, INCOME_C_5_FMT.) = “01_02”
PUT(“02”, INCOME_C_5_FMT.) = “01_02”

Iteration 5 is selected for coding a WOE version of income_c as shown in step 3.

Table 6: Computing P_1 for Iterations 4 and 5 via %RECODE

Iteration 4

income_c	freq	P_1
01	1,611	13.5%
02	6,899	12.9%
03	6,015	15.5%
04	5,554	18.6%
05	10,603	21.5%
06	6,434	24.8%
07	3,742	28.1%
08_09	4,002	29.6%
10_11_12	1,237	32.2%

Iteration 5

income_c	freq	P_1
01_02	8,510	13.0%
03	6,015	15.5%
04	5,554	18.6%
05	10,603	21.5%
06	6,434	24.8%
07	3,742	28.1%
08_09	4,002	29.6%
10_11_12	1,237	32.2%

Step 3: A third macro %WOE_CODING computes the weight of evidence values for a selected iteration. Iteration 5 was selected in this example. The results are given in Table 7

In %WOE_CODING a PROC FORMAT with CNTLIN option maps the collapsed income_c from iteration 5 to income_c_woe.

Suppose the user names this format INCOME_C_WOE_FMT.
Then, for example, PUT(“01_02”, INCOME_C_WOE_FMT.) = -0.56188; (See the first row of Table 7.)

We note that the x-statistic value 0.5978 for iteration 5 equals the model concordance for the model: PROC LOGISTIC; MODEL Y = INCOME_C_WOE;

Table 7: Transformation of Iteration 5 to Log-Odds via %WOE_CODING

income_c	income_c_woe = log(odds)	Information Value (IV)	Z [13]
01_02	-0.56188	0.04897	-19.572
03	-0.35901	0.01508	-10.863
04	-0.13658	0.00216	-4.230
05	0.0447	0.00047	2.156
06	0.22581	0.00758	8.446
07	0.39978	0.01447	11.549
08_09	0.46898	0.02167	14.297
10_11_12	0.59155	0.01097	9.997
	IV =	0.12136 [14]	

Transforming Income_c to weight-of-evidence coding in the Analysis Data set
The formats created in Steps 2 and 3 can be used to create a weight-of-evidence coded transform of "income_c" on the analysis data set.[15]

COLLAPSING A NOMINAL PREDICTOR AND WEIGHT-OF-EVIDENCE CODING

%COLLAPSE_LEVELS using collapse any-pair mode and macros from steps 2 and 3 can also be used for weight-of-evidence coding of nominal predictors with a binary target. In this case the c-stat column in step 1 is not meaningful.

COMPARISON TO COARSE CLASSING BASED DECISION TREE (NUMERIC X, BINARY Y)

We consider the collapse-adjacent-pairs mode in this comparison.

Decision-Tree-Based Coarse Classing Method: The tree-based method is a "top down" approach where the entire range "R" of X is initially considered and then the first "split" is selected to divide the range into lower "R1" and upper "R2" subranges so that a statistical relationship between Y and R1 and R2 is maximized. Candidates for this statistical relationship include: entropy, Gini, information value, and chi-square statistic.

After the first split into R1 and R2 the process continues with a split of R1 into lower "R11" and upper "R12" ranges and likewise for R2.

JMP®[16] includes a decision-tree procedure called PARTITION.[17]

Coarse Classing Based on %COLLAPSE_LEVELS: This method is bottoms-up where the individual adjacent values of X are collapsed together so that at each iteration the log likelihood of the logistic regression model that fits the collapsed_X (as a CLASS variable) to Y is maximized among all choices for the adjacent collapsing.

Both the Decision-Tree Methods and the Collapse Levels Algorithm determine the cutpoints but essentially leave the decision about the number of intervals to the modeler.

DOES THE COLLAPSED LEVELS ALGORITHM LEAD TO DIFFERENT CLASSES THAN DECISION TREE?

We compared %COLLAPSE LEVELS to JMP PARTITON where we looked only at the splitting criterion determined by entropy, which is denoted by G^2 in JMP output.

[13] See Finlay (2010) page 123.
[14] IV = 0.12136 indicates a moderate relationship between the collapsed income_c and Y. See Finlay (2010) page 123.
[15] The WOE coding is determined from the training data set. The formats are applied to the analysis data set (training plus holdout). The user must be sure that all levels of X appear in the training dataset.
[16] JMP reference manual, http://www.jmp.com/
[17] JMP PARTITION is not designed as a tool for coarse classing of a predictor variable. However, the output from "splitting" can be manually reversed to create "collapsing" for the purpose of this paper of comparison to the Collapse Levels algorithm.

The G^2 for the data set of Table 4 is G^2 = –2 * (n_0 * log(n_0 / n) + n_1 * log(n_1 / n)) = 47132.49. Here n = the number of observations, n_0 = the number with Y = 0 and n_1 = the number with Y = 1.

PARTITION: Splits are determined by finding cut-points so that G^2_{parent} – G^2_{right} – G^2_{left} = GAIN is maximized. The first split for Table 4 "Income_c" is "< 5" and ">=5" for a GAIN of 663.86.

Now we note (but don't show in this paper) that 47132.49 - 663.86 = 46468.63 is equal to -2 * LL(C5) where C5 is the dummy variable for "Income_C < 5" versus "Income_C >= 5" and LL is the log-likelihood of the model: PROC LOGISTIC; CLASS C5; MODEL Y=C5;

This illustrates the fact that the splitting process maximizes log likelihood when selecting a split.

COLLAPSE_LEVELS: Collapses are determined by collapsing two levels (including previously collapsed levels) so that uncertainly U is maximized. This criterion is the same as maximizing log likelihood when selecting a collapse.

Since both methods maximize LL(X), it might seem surprising that the sequence of collapses by %COLLAPSE_LEVELS is **not,** in general, the same as the sequence of collapses from PARTITION, which is obtained by reversing the splitting sequence. This is seen by the example of TABLE 5 as explained below.

TABLE 5 shows the final collapse of "Income_c" created two levels: 01_02_03 and 04_05_06_07_08_09_10_11_12. In contrast, the first split from PARTITION was 01_02_03_04 and 05_06_07_08_09_10_11_12.

For %COLLAPSE_LEVELS, at this final iteration the choices for collapsing were only: (a) collapse 01_02_03 with 04_05; or (b) collapse 04_05 with 06_07_08_09_10_11_12. Choice (b) was the better choice based on U(Y|X).

The reason for this difference in the sequence of collapsing arises from the stepwise nature of the two processes and the fact that %COLLAPSE_LEVELS is bottom-up and PARTITION is top-down.

Which method is better? We do not have a decisive answer. It seems that %COLLAPSE_LEVELS is better if the modeler will stop the collapsing process earlier in the sequence. This is because there is less opportunity to miss a sub-optimal choice that ultimately leads to a better outcome (higher log-likelihood) when stopping. If the collapsing needs to go deeper, then the decision-tree approach based on G^2 may be better because a small number of splits would be required to reach a stopping point and there would be less opportunity to sub-optimize.

As shown by the example from Table 5 this issue does occur in practical applications although it is much less clear that the performance of logistic models would be noticeably impacted.

Other Methods: Refaat (2011) provides software for several methods of coarse classing.

SYNTAX AND DISCUSSION OF MACRO %COLLAPSE_LEVELS

Parameters for %COLLAPSE_LEVELS

1. **Dataset** – entered as library.membername (if library not specified, WORK is assumed)
2. **Input** – Name of the predictor variable being collapsed. This can be either numeric or character. If numeric, the macro will convert the numeric variable to a character variable. Importantly, the length of the **Input** (including conversion of numeric to character) must be **one or two**. Only numerals, alphabetical characters (upper or lower case), and missing values ("." and " ") are permitted.
3. **Target** – Name of the target variable. **Target** must have integer values 0 to L (where L $\geq$ 1) and each such integer value must appear at least once in **Dataset**. There can be no missing values for the **Target**.
4. **W** – Name of the frequency variable or the value "**1**" if there is no frequency variable in **Dataset**.
5. **Missing** – **Y** to include missing values in the collapsing process. Any other value in this parameter means that missing will be ignored.
6. **Mode** – **J** for adjacent-pairs (in the natural ordering of X), **A** for any-pairs.

Usage Notes

1. This macro examines a single predictor variable and calculates the best way to combine levels. User judgment is required to determine which collapse level is best (i.e., the balance between reducing degrees of freedom vs. retaining the highest uncertainty coefficient).
2. It cannot be called from within a DATA step or PROC.
3. All internal data sets begin __X__ to avoid conflicts with user data.

4. Missing values of **Input** (if included via parameter **Missing**) are denoted by "~" in the results display. Otherwise, missing values are deleted before processing begins.
5. If **Mode** = **J** and **Missing** = **Y** (and if "missing" observations occur in **Dataset**), then the "Missing Level", denoted by "~" in the results display, is not permitted by the algorithm to collapse with any other level.
6. If **Mode** = **A** and **Missing** = **Y** (and if "missing" observations occur in **Dataset**), then the "Missing Level", denoted by "~" in the results display, is permitted by the algorithm to collapse with any other level.
7. The macro restricts the number of **Target** levels to 25 or less in the any-pairs mode. The limit is set at 75 for the adjacent-pairs mode where only adjacent pairs are considered for collapse.

MACROS TO SUPPORT WOE CODING FOLLOWING %COLLAPSE_LEVELS

%RECODE

The RECODE macro reads output data sets from %COLLAPSE_LEVELS, processes the rows (iterations) specified by the modeler, and, for each row, creates: (i) an output table showing P_1 (see Table 6); and (ii) a format named <**input**>_r<row>_fmt (using the predictor variable specified for %COLLAPSE_LEVELS, truncated at 18 characters if longer).

The modeler examines P_1 and FREQ to decide if sufficient sample size and a suitable pattern of P_1 have been achieved. The purpose of the format is explained in the APPLYING THE FORMATS section below.

Parameters for %RECODE

1. **Start_row** – First of the rows to be explored (see Table 5, column "Iteration").
2. **End_row** – Last of the rows to be explored (see Table 5, column "Iteration").
3. **Fmtlib** – Libname of format library (default is WORK)

Usage Notes

1. The RECODE macro must be run in the same session after the COLLAPSE_LEVELS macro has been run.
2. Unless a permanent format library is specified for **Fmtlib**, the format generated will only be available during the current SAS session.
3. This macro cannot be called from within a DATA step or PROC.

%WOE_CODING

The WOE_CODING macro reads the output from %RECODE, processes the single row specified by the modeler, and creates: (i) an output table showing the WOE values, information value, and a z-statistic; and (ii) a format named <**input**>_WOE_fmt (using the predictor variable specified for %COLLAPSE_LEVELS, truncated at 18 characters if longer). The purpose of the format is explained in the APPLYING THE FORMATS section below.

Parameters for %WOE_CODING

1. **Row** – The collapse iteration row number the modeler has selected as the best stopping point
2. **Fmtlib** – Libname of format library (default is WORK)

Usage Notes

1. The WOE_CODING macro depends on the RECODE macro output. %RECODE must be run prior to running %WOE_CODING.
2. Unless a permanent format library is specified for **Fmtlib**, the formats will only be available during the current SAS session.
3. This macro cannot be called from within a DATA step or PROC.

APPLYING THE FORMATS

The modeler can apply the formats produced by %RECODE and %WOE_CODING to create a woe-coded variable in the modeler's analytic dataset or within a production scoring platform. A sketch of this process is given for "INCOME_C":

```
%COLLAPSE_LEVELS(TRAINING,INCOME_C,Y,W,N,J);
%RECODE(4,7,WORK);
/* ASSUME ROW 5 WAS SELECTED AS THE STOPPING POINT */
%WOE_CODING(5,WORK);
DATA OUT; SET <USER DATA>;
   TEMP = PUT(INCOME_C, INCOME_C_R5_FMT.);
   INCOME_C_WOE = PUT(TEMP,INCOME_C_WOE_FMT.);
RUN;
```

How to obtain a copy of this MACRO and others mentioned in the paper:
The macros are available from the authors upon request.

SKELETON MACRO CODE TO COMPUTE THE X-STATISTIC

Parameters for %X_STAT

1. **Dataset** – Input data set name
2. **Input** – The name of the predictor variable to be investigated. This can be either numeric or character. No missing values.
3. **Target** – Target must have numeric values 0 to L (where L $\geq$ 1) and each such integer value must appear at least once in **Dataset**. There can be no missing values for the **Target**.
4. **Number_Levels** – Number of levels of **Input** (at least 2)
5. **Number_Target_Levels** – Number of levels of **Target** (at least 2)
6. **Count** – The name of the frequency variable (required). Only non-negative integer values.

```
%MACRO X_STAT(Dataset, Input, Target, Number_Levels, Number_Target_Levels, Count);

PROC SORT DATA = &DATASET OUT = __X__SORT;
  BY &INPUT &TARGET;
DATA __X__C_STAT;
  SET __X__SORT END = EOF;
  BY &INPUT;
  LENGTH VAR_NAME $8;
  RETAIN X_VALUE_COUNT;
  %DO R = 0 %TO &NUMBER_TARGET_LEVELS - 1;
    ARRAY F&R {&NUMBER_LEVELS} F&R._1 - F&R._%CMPRES(&NUMBER_LEVELS);
    RETAIN F&R._1 - F&R._%CMPRES(&NUMBER_LEVELS);
    RETAIN F&R._TOTAL;
  %END;

  IF _N_ = 1
  THEN DO;
    X_VALUE_COUNT = 0;
    %DO R = 0 %TO &NUMBER_TARGET_LEVELS - 1;
      DO I = 1 TO &NUMBER_LEVELS;
        F&R{I} = 0;
        F&R._TOTAL = 0;
      END;
    %END;
  END;

  IF FIRST.%CMPRES(&INPUT) THEN X_VALUE_COUNT = X_VALUE_COUNT + 1;

  %DO R = 0 %TO &NUMBER_TARGET_LEVELS - 1;
    IF &TARGET = &R THEN F&R{X_VALUE_COUNT} = F&R{X_VALUE_COUNT} + &COUNT;
  %END;

  IF EOF
  THEN DO;
    %DO R = 0 %TO &NUMBER_TARGET_LEVELS - 2;
        %DO S = &R+1 %TO &NUMBER_TARGET_LEVELS - 1;
          X_STAT&R.&S  = 0;
        %END;
    %END;
    %DO R = 0 %TO &NUMBER_TARGET_LEVELS - 1;
      DO I = 1 TO X_VALUE_COUNT;
        F&R._TOTAL = F&R._TOTAL + F&R{I};
      END;
    %END;
    %DO R = 0 %TO &NUMBER_TARGET_LEVELS - 2;
      %DO S = &R+1 %TO &NUMBER_TARGET_LEVELS - 1;
```

```
        DO I = 1 TO X_VALUE_COUNT;
          DO J = 1 TO X_VALUE_COUNT;
            IF I < J THEN X_STAT&R.&S =
                X_STAT&R.&S + ABS(F&S{I}*F&R{J} - F&R{I}*F&S{J});
          END; /* END of: J = 1 TO X_VALUE_COUNT */
        END; /* END of: I = 1 TO X_VALUE_COUNT */
      %END;
    %END;
    M = 0; /* SUM ACROSS INFORMATIVE PAIRS */
    %DO R = 0 %TO &NUMBER_TARGET_LEVELS - 2;
      %DO S = &R+1 %TO &NUMBER_TARGET_LEVELS - 1;
        C_PAIR = F&S._TOTAL * F&R._TOTAL;
        M = M + C_PAIR;
        X_STAT&R.&S  = .5*(X_STAT&R.&S  / C_PAIR  + 1);
      %END;
    %END;
    X_STAT = 0;
    %DO R = 0 %TO &NUMBER_TARGET_LEVELS - 2;
      %DO S = &R+1 %TO &NUMBER_TARGET_LEVELS - 1;
        X_STAT = X_STAT + X_STAT&R.&S * (F&S._TOTAL * F&R._TOTAL);
      %END;
    %END;
    X_STAT = X_STAT / M;
    INPUT_LEVELS = X_VALUE_COUNT;
    NUMBER_TARGET_LEVELS = &NUMBER_TARGET_LEVELS;
    VAR_NAME = "&INPUT";
    OUTPUT __X__C_STAT;
  END; /* END of EOF */
RUN;

PROC PRINT DATA=__X__C_STAT;
  VAR VAR_NAME INPUT_LEVELS NUMBER_TARGET_LEVELS X_STAT M
  %DO R = 0 %TO &NUMBER_TARGET_LEVELS - 2;
    %DO S = &R+1 %TO &NUMBER_TARGET_LEVELS - 1;
      X_STAT&R&S
    %END;
  %END;
  %DO R = 0 %TO &NUMBER_TARGET_LEVELS - 1;
    F&R._TOTAL
    %DO I = 1 %TO &NUMBER_LEVELS;
      F&R._&I
    %END;
  %END;
    ; /* ";" for end of VAR list */
  FORMAT X_STAT 8.4;
RUN;
%MEND;
```

REFERENCES

Finlay, S. (2010). *Credit Scoring, Response Modelling and Insurance Rating*, New York: Palgrave MacMillan.

Manahan, C. (2006). "Comparison of Data Preparation Methods for Use in Model Development with SAS Enterprise Miner", *Proceedings of the 31th Annual SAS Users Group International Conference*, Paper 079-31.

Raimi, S. and Lund, B. (2011). " Before Logistic Modeling - A Toolkit for Identifying and Transforming Relevant Predictors", *MWSUG 2011, Proceedings*, Midwest SAS Users Group, Inc., Paper SA-03-2011.

Refaat, M (2011). *Credit Risk Scorecards: Development and Implementation Using SAS*, Lulu.com

Theil, H. (1972). *Statistical Decomposition Analysis*, North-Holland Publishing Company.

ACKNOWLEDGMENTS

David Brotherton of Marketing Associates read several drafts of this paper and provided helpful suggestions.

CONTACT INFORMATION

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

Bruce Lund
Marketing Associates, LLC
777 Woodward Ave, Suite 500
Detroit, MI, 48226
blund@marketingassociates.com

Steven Raimi
Marketing Associates, LLC
777 Woodward Ave, Suite 500
Detroit, MI, 48226
sraimi@marketingassociates.com