Paper SA-03 - 2011

Before Logistic Modeling
- A Toolkit for Identifying and Transforming Relevant Predictors

Steven Raimi, Marketing Associates, LLC, Detroit, Michigan, USA
Bruce Lund, Marketing Associates, LLC, Detroit, Michigan, USA

ABSTRACT

This paper discusses the situation where a modeler must fit a binary logistic model and has available hundreds of potential predictors. Such situations sometimes occur when third-party data sets are added to in-house transactional data for direct marketing applications. We present an efficient methodology and a tool kit (using SAS® macros) that identifies a subset of predictors for further study (transformations, correlations, etc.), and eventual usage in fitting a model.

KEY WORDS

c-statistic, Logistic, Model Preparation, Filter Predictors, Concordance, Non-Monotonic, Efficiency, Nominals

INTRODUCTION

The paper will discuss two steps in the process of identifying and preparing predictor variables for use in a binary logistic regression model.

1. Identifying which numeric variables have sufficient predictive power to merit further study. From among these variables, finding which have a non-monotonic relationship to the target variable.
2. Determining how to collapse the levels of a nominal predictor variable without losing predictive power.

The c-statistic and an "x-statistic" (to be defined) are used for numeric variables. A SAS macro called XC_STAT is provided to efficiently compute c-statistics and x-statistics for any number of variables.

For character variables, a SAS macro called COLLAPSE_NOMINAL is provided which collapses the levels of a nominal predictor while minimizing the loss of log likelihood relative to the target variable.

Related articles include Manahan (2006), Sarma (2007), Hermansen (2008), Fang (2011).

C-STATISTIC

Consider a "predictor" X with interval-scaled (or at least ordinal) values, and a binary "target" variable Y with values 0 and 1. Let M give the count of pairs of observations where one observation has Y = 0 and the other has Y = 1. The count of pairs having a higher value of X when Y = 1 than the value of X when Y = 0 will be denoted by "C" (for "concordant"). The total of pairs where the values of X are equal is denoted by "T" (for ties).

The formula for the c-statistic is: (C + **.**5 * T) / M

If there are no ties (i.e. T = 0), the c-statistic between X and Y gives the probability that a randomly chosen pair from the set of M pairs has a higher value of X when Y = 1 and a lower value of X when Y = 0.

If the c-statistic for X and Y is "c1", then reversing the coding of Y gives a new c-statistic "c2". But c1 = 1 – c2, providing equivalent information about X and Y. It is assumed in this paper that Y is coded so that c-statistics are **.**50 or greater.

Computing the c-statistic from a frequency table

The c-statistic can easily be computed from a frequency table. Consider the table of X and Y values below where $f_j(i)$ gives the frequency for the (i, j) cell where i = 1 ... k and j = 0, 1.

	Y	
X	Y = 0	Y = 1
X_1	$f_0(1)$	$f_1(1)$
X_2	$f_0(2)$	$f_1(2)$
...	...	...
X_k	$f_0(k)$	$f_1(k)$
SUM	$\sum f_0(i)$	$\sum f_1(i)$

Table 1 - Frequency Table (Formulae)

M gives the count of all pairs of observations where Y equals 0 for one member and Y equals 1 for the other member of the pair. A formula is given below.

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

Then let $A = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} f_0(i)*f_1(j)$ and let $B = .5 * \sum_{i=1}^{k} f_0(i)*f_1(i)$.

Then c-statistic = **[A + B] / M** ....[1] [2]

Example #1:

	Y	
X	Y = 0	Y = 1
1	2	0
2	0	2
3	2	0
SUM	4	2

Table 2 - Frequency Table for c-statistic calculation

The c-statistic = [(4 + 0 + 0) + **.**5*(0 + 0 + 0)] / (4 * 2) = **.**5

Other Computational Methods

PROC FREQ or PROC NPAR1WAY, followed by simple DATA step computations, can provide the c-statistic for a single predictor. See SAS code provided by Xie (2009). Hermansen (2008) provides SAS SQL code to compute the c-statistic.

The SAS macro XC_STAT (discussed later in the paper) efficiently computes c-statistics for any number of predictor variables. The computations utilize the formula [A + B] / M.

C-statistic is also a measure of fit of a Binary Logistic Regression. SAS PROC LOGISTIC provides a statistic denoted in the SAS output as "**c**" which, in this paper, will also be called the **concordance** of the model. This concordance is the same as the c-statistic of the logistic regression probability $\hat{p}$ and the target Y. It provides one of the commonly used tools to measure the fit of a binary logistic regression model.[3] For a logistic model with single predictor X, the concordance equals the c-statistic of Y and X.

The c-statistic measures the strength of the "monotonic increasing" association of X with Y. Let r and s be values of X with r < s. Within the c-statistic formula the following terms appear:

[1] The c-statistic is unaffected (apart from random chance) by sub-sampling of Y=0 records. This is seen from this formulation of c-statistic ... a sample weight "w" for the $f_0(i)$ will cancel out from numerator and denominator.
[2] A supplement to this paper providing mathematical derivation of various formulas is available from the authors.
[3] Thompson (2009) discusses six measures to rank the importance of predictors in a logistic regression model, one of them being the predictor's c-statistic versus the model's predicted probability.

$$f_0(r) * f_1(s) / M \quad \ldots (1)$$

A high frequency of $f_0(r)$ and a high frequency of $f_1(s)$ make (1) large, thereby increasing the c-statistic. If this relationship of r and s (where $r < s$) to $f_0(r)$ and $f_1(s)$ is consistently true across the values of X, then the c-statistic will be large. In this sense, the c-statistic measures a monotonic association of X with Y.[4] A monotonic decreasing association occurs when c-statistic < .50.

In Example #1, the c-statistic value of ".5" says there is no monotonic relationship between X and Y. But there is clearly a relationship between X and Y ... if X is known, then Y can be predicted perfectly. In the next section the "x-statistic" is discussed. This statistic can be used as an extension of the c-statistic to measure the strength of non-monotonic relationships between X and Y.

C-STATISTIC COMPARED TO "X-STATISTIC"

X-statistic

The term "x-statistic" will be used as a short-hand expression for the "concordance of a predictor X when it is treated as a class-variable in the logistic regression of Y against X".

It is not necessary to run PROC LOGISTIC to compute the x-statistic. The x-statistic can be computed from a frequency table according to the formula below.

First, let $Z = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k}$ ABS $(f_0(i)*f_1(j) - f_0(j)*f_1(i))$. (ABS = absolute value)

Then x-statistic = $.5 * (Z / M + 1)$ [5]

The x-statistic ranges between .5 and 1.0. For the example #1, x-statistic = 1.0

Relationship between c-statistic and x-statistic

Let $P_i = f_1(i) / (f_0(i) + f_1(i))$ ... for i = 1 ... k where k is the number of levels of X. (We continue to assume that Y is coded so that c-statistic $\geq$ .50)

If $P_1 \leq P_2 \leq \ldots \leq P_k$ then x-statistic = c-statistic
Otherwise, x-statistic > c-statistic ...[6]

The x-statistic can be used as an extension of the c-statistic to measure the strength of non-monotonic relationships between X and Y. A measure which compares c-statistic and x-statistic is:

$$\text{Adjusted Lift} = (\text{x-statistic} - \text{c-statistic}) / (\text{c-statistic} / (k - 1))$$

Since a large number of levels of X can inflate the x-statistic, the Adjusted Lift computes the Lift per degree of freedom. The Adjusted Lift is generally comparable across predictors.

Additionally, the values of X may be binned into groups before computing the c-statistic and x-statistic. This is required when X is "continuous" (having perhaps 100's of distinct values). See the discussion of binning in the section on the XC_STAT macro.

---

[4] Any monotone transform of X has the same c-statistic value.
[5] The x-statistic is unaffected (apart from random chance) by sub-sampling of Y=0 records. This is seen from this formulation of x-statistic ... a sample weight "w" for the $f_0(i)$ will cancel out from numerator and denominator.
[6] Using this result, if the values of X are re-labeled so that P_i are monotone or are sorted so that P_i ascend, then:

```
        proc freq data = test order = data;  tables y*x / measures; output out=_measures measures;
        data _null_; set _measures;  c_stat =  _SMDCR_  / 2 + .5;  c_stat = max(c_stat, 1 - c_stat);
```

will provide the x-statistic (= c_stat)

XC_STAT MACROS

In situations where third-party data sets are added to in-house transactional data for direct marketing applications, there may be hundreds of potential predictors. In particular, the third-party data may include many unproductive variables. The speedy identification and elimination of these variables allows for a re-focus of effort on transforming the variables that are kept for further study. Next, the comparisons of c-statistic and x-statistic support the decision making on which variables to consider for non-monotonic transformations - including, in the extreme case, a "class variable" transformation (treating the predictor as a class variable). To assist in this process the XC_STAT macros can be used.

The XC_STAT SAS macros compute the c-statistic, x-statistic and their adjusted lift. XC_STAT also computes the Information Value and Spearman Rank Correlation for X and Y. The processing steps and syntax of XC_STAT are discussed in detail at the end of the paper. There are two versions of XC_STAT:

XC_STAT_WITHOUT_RANKS

This macro processes "discrete-valued" variables. The macro imposes the limit that a variable must have no more than 50 distinct values. The SAS code involves a PROC SUMMARY step and two DATA steps which process the PROC SUMMARY output. Data sets with any number of predictor variables can be entered as input to the macro. We have run tests with 300 predictors. Variables with more than 50 distinct values can be processed by the XC_STAT_WITH_RANKS macro.

The XC_STAT_WITHOUT_RANKS can be used with nominal variables to obtain the x-statistic. The nominal variables need to be coded as numeric. Of course, the c-statistic is not meaningful.

XC_STAT_WITH_RANKS

This macro processes "continuous-valued" variables. A continuous variable may have hundreds of distinct values – examples include: distances, time, percentages, dollars, etc. The SAS code involves a PROC RANK, a PROC SUMMARY and one DATA step which processes the PROC SUMMARY output.

The user enters a parameter to select the "Groups" value for PROC RANK. Useful values for Groups might be 10 to 30. This step assigns the variable's values to discrete bins. PROC RANK does not necessarily provide optimal binning of a predictor, but at the exploratory stage this approach is convenient and should be effective.

If a continuous variable has a high frequency of a single value, then a selection of Groups = "G" may provide fewer than G groups for that variable, because PROC RANK will not assign a set of points with a common value to more than one group.

DISCUSSION AND GUIDELINES FOR USING C-STATISTIC AND X-STATISTIC IN VARIABLE SELECTION

When computed for ranks, the value of x-statistic will generally decrease as the number of ranks is decreased. In contrast, the value of c-statistic is not dependent on the number of groups. Guidelines cannot be given for usage of the x-statistic without qualification regarding the sample size as well as the number of distinct values of the predictors.

Typical Case

We'll consider the case where the target variable has 1,500 zero's and 1,500 one's and where each predictor variable has been grouped into 10 or fewer bins. This is typical of our experience in automotive direct marketing. (The sample sizes are usually larger.)

What value of x-statistic should a predictor have in order to be selected for further consideration?

If there are hundreds of predictors, then the x-statistic might be used to rank the variables so that 50 to 75 variables are kept for investigation. Otherwise, an absolute cut-off of x-statistic $\geq$ .55, when using 10 or fewer bins, provides a potential rule-of-thumb.

When does "adjusted lift" indicate that it is worthwhile to investigate non-monotonic transformations of X?

Here is our experience in automotive direct marketing. A good predictor X might have a c-statistic of **.**600. For a predictor with 10 levels the difference between a c-statistic at .600 and an x-statistic at .606 is a modest improvement, considering that the overall logistic model might have a concordance on the order of only **.**700.

When considered as an adjusted lift: (.606 - .600) / (.600 / (10 - 1)) = .111%. So, we adopt a value of adjusted lift of .100% as a criterion to search for a non-monotone transformation. Naturally, a monotonic transformation may ultimately be selected nonetheless.[7]

Examples of non-monotonic transformations of X include:

- CLASS variable … use every level of X as a “dummy”
- Weight-of-evidence: WOE = LOG(P_i/(1-P_i))*(X=X_i)
  WOE and CLASS are equivalent:
  The logistic model probabilities created by the following two models are the same:
  (1) PROC LOGISTIC; MODEL Y = WOE; (2) PROC LOGISTIC; CLASS X; MODEL Y = X;
- A “U” shaped transform (e.g. quadratic)
- X plus a dummy variable for a particular X = X_i.
  DATA OUT; Di = (X = “Xi”);
  PROC LOGISTIC DATA=OUT; MODEL Y = X Di;

Example #2
The service behavior of consumers was tracked for six months and the total dollars "X" for customer-paid service (repairs or maintenance) at a franchised store was recorded. These consumers were tracked for another six months and coded as "Y=1" if there was customer-paid service at a store or "Y=0" otherwise. There were 53,154 consumers in the sample. For 83% of the consumers, X=0.

XC_Stat_With_Ranks was run with Groups selected to be 30. PROC RANK created only 7 groups due to the large number of observations with X=0.

Rank of X	1	2	3	4	5	6	7
Y=1 pct	18.24	61.26	67.10	63.04	55.19	56.09	54.21
Freq pct	82.65	0.68	3.33	3.33	3.33	3.33	3.33
Mean value	$0	$14	$32	$62	$163	$423	$1,364

Table 3 - Ranks created by PROC FREQ

Using the seven ranks: c-statistic = .6532, x-statistic = .6573, adjusted lift = .102%

The rule-of-thumb for "adjusted lift of at least .100%" was met and transformations of X were considered. The large number of observations for rank 1 (X=0) suggests that a dummy variable $I_{X=0}$ be added for X=0. Also the "quadratic appearance" for the "percent Y=1" across ranks 2 to 7 suggests the potential for adding a quadratic term to X.

The modeler must decide whether to evaluate X and X^2 using raw data or using the mean-values of the 7 groups. If the means of the groups are used, then the logistic **c** for the models involving $I_{X=0}$, X, and X^2 can be compared to the c-statistic and x-statistic computed above. The bulk of the improvement arises from adding $I_{X=0}$ to X, and this model essentially reaches the logistic model **c** value for the model with X as a class variable (= .6573). The addition of the quadratic term X^2 slightly reduces the **c**.

Variables in Model	Logistic "c"
X	.6532
$I_{X=0}$, X	.6572
$I_{X=0}$, X, X^2	.6571
X (class)	.6573

Table 4 - Comparing c-statistic of transformed variable

[7] In any case, a transformation of X should be found so that the relationship of X to the Log-Odds of Y is linear.

Extension of XC_STAT to Multinomial Logistic

If the target variable Y has more than two unordered levels, then PROC LOGISTIC can fit a multinomial logistic model by specifying the GLOGIT LINK option: PROC LOGISTIC; MODEL Y = X / LINK = GLOGIT.

We have generalized XC_STAT macro as follows: A "reference" level of Y is selected and then c-statistics and x-statistics are computed for each other level of Y versus the "reference".

If the modeler plans to use the "pairwise binary logistic regression" approach of Begg and Gray (1984), then these statistics are directly relevant to variable screening.[8] If the approach is to fit a multinomial model, then it is uncertain how to summarize the c-statistics and x-statistics to add in variable screening. One reasonable approach may be to consider the level of Y which gives the maximum x-statistic and use this as the threshold screening criterion. Then consider Adjusted Lift to decide if a non-monotonic relationship is present.

COLLAPSING THE LEVELS OF A CLASS VARIABLE

In cases where a predictor X, either numeric or nominal, will be entered as a CLASS variable in PROC LOGISTIC, there may be levels of this variable X that can be collapsed without loss of predictive power.

The approach suggested here involves the following steps:

1. Compute U(Y|X), the uncertainty coefficient of "Y given X".[9] [10] This is iteration step "0".
2. For each pair (i, j) of levels $1 \leq i < j \leq k$ of X: compute $U(Y|X_{i,j})$ for levels i, j collapsed.
3. 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(1). This completes iteration step "1".
4. Consider the stopping criterion (below).
5. Iteration Step "r": Find the pair (i, j) of either remaining levels or already collapsed levels from X(r-1) that gives the maximum $U(Y|X_{i,j})$ and collapse to form a new predictor X(r).
6. Consider the stopping criterion (below).

   Stopping criterion: Define U_r = to be the optimal collapse at iteration r. The stopping criterion may be based on the percentage change in U_r between iterations:

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

We give a loose guideline of stopping at the iteration just before a percentage change of 1% or more occurs. That is, if $PC_r >= 1\%$, then stop at iteration r-1. This criterion should be compared with the changes in value of the x-statistic to see if predictive power remains.

U(Y|X) is related to Log Likelihood. Let LL(X) [11] 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:

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

This relationship is true even if Y has more than 2 levels.

Consequently, our approach maximizes the Log Likelihood of Y versus $X_{i,j}$, the collapsed predictor at each iteration. Log Likelihood is the same quantity that is maximized when fitting PROC LOGISTIC.

---

[8] See also Ames, Hackett, Lund (2008)
[9] U(Y|X) is computed by PROC FREQ.
[10] U(Y|X) gives the proportional reduction in entropy of X that results from knowing the value Y. U(Y|X) = {H(X)+H(Y)-H(X,Y)}/H(Y) where, for example, H(X) = $\sum p(i)*\log(p(i))$ for i indexing across the levels of X and for p(i) giving the probability that $X = x_i$.
[11] If X has k levels, then let $X_i = 1$ where X equals its ith level, and $X_i = 0$ otherwise for $1 \leq i \leq k$.
Let $p_i = \text{Prob}(Y=1|\ X_i=1)$. Let n_{i1} = count of records where $X_i=1$ and n_{i0} = count of records where $X_i=0$. Then LL = $\text{Log}(\prod^k_{i=1} p_i^{ni1} * (1-p_i)^{ni0})$ and $p_i = \exp(xbeta)/(1+\exp(xbeta))$ where the β_i in xbeta = $\beta_0 + \beta_1*X_1 + ... + \beta_{k-1}*X_{k-1}$ are determined by maximizing LL.

COLLAPSE_NOMINAL MACRO

The Collapse_Nominal macro takes a variable X (treated as nominal) and a binary target Y and computes U(Y|X) iteratively for the collapsed of levels of X so that each successive collapse is chosen to maximize U(Y|X). The macro is based on PROC FREQ.

Since the number of pairs of levels grows at a rate of k-squared where k is the number of levels, this technique is practical only for variables X with a moderate number of levels. The COLLAPSE_NOMINAL enforces a limit of 25 levels. COLLAPSE_NOMINAL also computes the x-statistic at each iteration.

Example #3
The 2-way table below shows "X", a demographic trait of a direct mail prospect, and "Y", the response of the prospect (1 = yes, 0 = no), to a direct mail offer.

Y vs. X

	A	B	C	M	N	P	S	T
No "0"	689	2420	99	2	20	212	8818	39
col %	80.58%	74.69%	86.09%	100.00%	83.33%	73.61%	82.50%	82.98%
Yes "1"	166	820	16	0	4	76	1870	8
col %	19.42%	25.31%	13.91%	0.00%	16.67%	26.39%	17.50%	17.02%

Table 5 - Ranks created by PROC FREQ

The levels "M", "N", and "T" should be collapsed due to small sample size. Additionally, levels "S" and "T" have very similar column percentages and appear to be candidates for collapsing.

The table below shows the iterations of COLLAPSE_NOMINAL. The "+" shows which levels are being collapsed at the current iteration. The "_" shows the results of collapses at prior iterations. The overall x-statistic is .54772 and is on the borderline of our suggested .55 cut-off for further consideration.

					memo:	
iteration	collapsed	U	Pct Change in U	x_statistic	deg .f	(*) LRCS X^2
1	None	0.0069826		0.54772	7	< .001
2	N+T	0.0069825	0.001%	0.54772	6	< .001
3	N_T+S	0.0069813	0.017%	0.54766	5	< .001
4	B+P	0.0069706	0.154%	0.54752	4	< .001
5	C+M	0.0069310	0.567%	0.54752	3	< .001
6	C_M+N_T_S	0.0068481	1.197%	0.54686	2	< .001
7	A+C_M_N_T_S	0.0067110	2.002%	0.54435	1	< .001

(*) LRCS X^2 = right tail X^2(df) probability for the likelihood ratio chi-square statistic.

Table 6 - Output from COLLAPSE_NOMINAL

Based on the 1% guide-line, the collapsing process will stop at iteration 5 - giving this table:

	A	B_P	C_M	N_T_S
No "0"	689	2632	101	8877
col %	80.58%	74.60%	86.32%	82.51%
Yes "1"	166	896	16	1882
col %	19.42%	25.40%	13.68%	17.49%

Table 7 – PROC FREQ output for selected collapse level

The x-statistic also indicates that collapsing should end at iteration 5. The x-statistic makes its first relatively large drop at iteration 6 ... dropping from 0.54752 at iteration 5 to 0.54686 at iteration 6.

The significance level of the Log Likelihood Chi-Square (LRCS X^2) is also shown in this example. Since LRCS X^2 decreases with sample size (for fixed cell probabilities), there is no fixed value of LRCS X^2 that provides a useful stopping-rule. For large sample sizes LRCS X^2 can be highly significant for all iterations so that significance tests are less useful.

Other methods
Two other methods are routinely and successfully used to collapse the levels of a nominal variable:

Decision tree
A tree is formed with the binary target Y and nominal variable X. The leaf nodes that are the result of the splitting define the collapsed levels. A stopping criterion must be specified.

Clustering
Another method is based on clustering of levels using SAS PROC CLUSTER and using a stopping criterion determined by the minimum chi-square statistic probability (right tail probability) of association between the target and the iterations of the collapsing predictor. [12]

The clustering method was illustrated by Manahan (2006) who provides SAS macro code. Additional code is needed to apply the chi-square probabilities. See Manahan (2006) for other references.

Extension of COLLAPSE_NOMINAL to Multinomial Logistic

The need to collapse the levels of a nominal predictor also arises in the multinomial logistic case.
The COLLAPSE_NOMINAL technique, based on maximizing U(Y|X) at each iteration, applies without change. The "skeleton" code given later in this paper can be used for the multinomial case. However, there is no analogue to the x-statistic that appears in the binary case. Decision tree and clustering methods (for the binary target) would need modification in order to apply to the multinomial case.

SYNTAX AND DISCUSSION OF XC_STAT MACROS

XC_Stat_Without_Ranks

Parameters
1. **Data set** – entered as library.membername (if library not specified, WORK is assumed)
2. **Values_Level_Limit** ($\leq$ 50) – User defined maximum number of distinct values a predictor variable may have in order to be processed. The macro overrides any value above 50 (analyze those variables using xc_stats_with_ranks).
3. **Target** – the response variable (must have numeric values 0 and 1, with no missing values). The "success" value level can be either 1 or 0 – the c-statistic is always calculated to be ≥ 0.50.
4. **Exclude_cols** – The name(s) of columns to be left out of the analysis, such as primary keys, continuous-valued columns, nominals, and columns known to be irrelevant. Multiple names must be SPACE delimited.

Usage Notes
1. This macro processes all and only numeric variables. Variables with only 1 distinct value are not processed.
2. It cannot be called from within a DATA step or PROC, and ignores any column with all nulls.
3. All internal data sets begin __X__ to avoid conflicts with user data.
4. An observation with a predictor with a missing value is ignored when calculating statistics for predictor variables.

Results
%XC_STAT_WITHOUT_RANKS (WORK.TEST, 5, Y,) using this "test" data set:

```
Data test;
   input X1 X2 X3 Y;
   datalines;
```

[12] "Collapse_Nominal" and "clustering method" collapse the levels of X in the same order.

```
1 1 6 0                         2 4 3 1
1 1 5 1                         5 4 3 0
1 1 4 1                         5 4 3 1
1 1 7 1                         5 4 4 1
2 4 8 0                         ;
```

VAR_NAME	LEVELS	C_STAT	X_STAT	IV	R_SQ	SPEAR-MAN	X_C_LIFT	X_C_LIFT_ADJ
X1	3	0.5556	0.6111	0.1831	0.0085	-0.0976	0.1000	0.0500
X2	2	0.5833	0.5833	0.1155	1.0000	-0.1581	0.0000	0.0000
X3	6	0.6667	0.9444	No calc	No calc	-0.2798	0.4167	0.0833

Table 8 - XC_STAT_WITHOUT_RANKS output

Note1: R_SQ is the linear regression R-square between the log-odds of the target (at each level of the predictor) and the value of the predictor.
Note2: For the predictor X3 there is "No calc" for IV and R_SQ because the log-odds cannot be computed for a level where "count of Y=0" or "count of Y=1" is zero.

Skeleton Code for XC_STAT_WITHOUT_RANKS
Because the complete macro code listing is very long, we've included a simplified, "skeleton" version here. This code can be copied into a SAS session and successfully run. There is no input data error-checking and only the c-statistic and x-statistic are computed.

```
%LET MAX_LEVELS_PERMITTED = 50;

%LET VAR_NUM = 3; /* NUMBER OF PREDICTOR VARIABLES */
%LET V1 = X1; /* FIRST PREDICTOR */
%LET V2 = X2; /* SECOND PREDICTOR */
%LET V3 = X3; /* THIRD PREDICTOR etc.*/

%LET DATASET = TEST;
%LET TARGET = Y;

%MACRO XC_STAT;
/**    PROC SUMMARY.  The CLASS statement specifies all predictors.
       The TYPES statement limits output to each predictor vs. Target.
       The output __X__SUMOUT data set holds distinct value combinations of each
variable*target and their frequencies. **/
PROC SUMMARY DATA = &DATASET NOPRINT;
   CLASS
   %DO I = &VAR_NUM %TO 1 %BY -1;
       &&V&I
   %END;
   &TARGET;
   TYPES (%DO I = &VAR_NUM %TO 1 %BY -1; &&V&I %END;) * &TARGET;
   OUTPUT OUT = __X__SUMOUT1; RUN;

/**    DATASTEP __X__SUMOUT2:  PROC SUMMARY sorted the values of &&V&I within
each _TYPE_ By-Group. Using RECODE the values of &&V&I are recoded to start
with 1 and successively increment by 1.  In LAST._TYPE_ the number of distinct
values for a predictor is saved in macro "_N&I" by CALL SYMPUT. **/
DATA __X__SUMOUT2;
   SET __X__SUMOUT1 END = EOF;
   BY _TYPE_;
   RETAIN RECODE MAX_RECODE NEW_VALUE OLD_VALUE 0;
   IF FIRST._TYPE_
   THEN DO;
      RECODE = 0;
      NEW_VALUE = .;
      OLD_VALUE = .;
   END;
   %DO I = &VAR_NUM %TO 1 %BY -1;
      IF &&V&I > . THEN NEW_VALUE = &&V&I; /* Exactly 1 &&V&I is non-missing */
   %END;
/* TRUE if current and prior obs. have different values for &&V&I */
   IF NEW_VALUE NE OLD_VALUE
   THEN DO;
      %DO I = &VAR_NUM %TO 1 %BY -1;
         IF &&V&I > .
         THEN DO;
            RECODE = RECODE + 1; /* RECODE has values 1 to #levels of &&V&I */
            &&V&I = RECODE; /* Assign incremented Recode to &&V&I */
         END;
      %END;
   END;
/* TRUE if current and prior obs. have same values for &&V&I */
   IF NEW_VALUE EQ OLD_VALUE
   THEN DO;
      %DO I = &VAR_NUM %TO 1 %BY -1;
         IF &&V&I > .
         THEN &&V&I = RECODE; /* Assign unchanged RECODE to &&V&I */
      %END;
   END;
   IF LAST._TYPE_
   THEN DO;
      MAX_RECODE = MAX(MAX_RECODE, RECODE);
      %DO I = &VAR_NUM %TO 1 %BY -1;
         IF &&V&I > . AND 1 < &&V&I <= &MAX_LEVELS_PERMITTED
            /* Assign final value of RECODE for &&V&I to "_N&I" */
            THEN CALL SYMPUT("_N&I", RECODE);
         ELSE IF &&V&I > . THEN CALL SYMPUT("_N&I","0");
      %END;
   END;
   OUTPUT;
   OLD_VALUE = NEW_VALUE;
RUN;

/**   DATA __X__C_STAT: Uses macro array &&V&I to populate ARRAY V{I} with
predictor variable names and uses &&_N_& to populate ARRAY VL{I} with count of
distinct levels for the variable &&V&I. Within each _TYPE_ By-Group, ARRAY
F0{I} is assigned the cell frequency for the count of Y=0 for the Ith level.
Likewise, ARRAY F1{I} for Y=1. ARRAY F0{I} and ARRAY F1{I} are used to
calculate c-statistic and x-statistic.  **/
DATA __X__C_STAT;
   SET __X__SUMOUT2 END = EOF;
   BY _TYPE_;
   LENGTH V1 - V&VAR_NUM $32;
   LENGTH VAR_NAME $32;
   ARRAY F0 {&MAX_LEVELS_PERMITTED} F0_1 - F0_&MAX_LEVELS_PERMITTED;
   ARRAY F1 {&MAX_LEVELS_PERMITTED} F1_1 - F1_&MAX_LEVELS_PERMITTED;
   ARRAY V {&VAR_NUM} $32 V1 - V&VAR_NUM;
   ARRAY VL {&VAR_NUM} VL1 - VL&VAR_NUM;
   RETAIN F0_1 - F0_&MAX_LEVELS_PERMITTED
          F1_1 - F1_&MAX_LEVELS_PERMITTED
   RETAIN VL1 - VL&VAR_NUM
          CURR_VAR_NUM 0
          V1 - V&VAR_NUM
          ;
   IF _N_ = 1
   THEN DO;
      %DO I = 1 %TO &VAR_NUM;
         V{&I} = "%CMPRES(&&V&I)" ;
```

```
            VL{&I} = &&_N&I;
         %END;
      END;
      IF FIRST._TYPE_
      THEN DO;
         DO I = 1 TO &MAX_LEVELS_PERMITTED;
            F0{I} = 0;
            F1{I} = 0;
         END;
         CURR_VAR_NUM + 1;
      END;
      %DO VAR_I = 1 %TO &VAR_NUM;
         IF &&V&VAR_I > . AND VL{&VAR_I} > 0
         THEN DO;
            IF &TARGET = 0 THEN F0{&&V&VAR_I} = _FREQ_;
             ELSE IF &TARGET = 1 THEN F1{&&V&VAR_I} = _FREQ_;
         END;
      %END;
      IF LAST._TYPE_ AND 0 < VL{CURR_VAR_NUM}
      THEN DO;
         F0_TOTAL = 0;
         F1_TOTAL = 0;
         DO I = 1 TO VL{CURR_VAR_NUM};
            F0_TOTAL = F0_TOTAL + F0{I};
            F1_TOTAL = F1_TOTAL + F1{I};
         END;
         C_PAIR  = 0;
         C_STAT  = 0;
         X_STAT  = 0;
         X_C_LIFT = 0;
         X_C_LIFT_ADJ = 0;
         DO I = 1 TO VL{CURR_VAR_NUM};
            DO J = 1 TO VL{CURR_VAR_NUM};
               IF I < J THEN C_STAT = C_STAT + F0{I}*F1{J};
               IF I < J THEN X_STAT = X_STAT + ABS(F0{I}*F1{J} - F1{I}*F0{J});
               IF I = J THEN C_STAT = C_STAT + .5 * F0{I}*F1{I};
               END; /* END of: J = 1 TO VL{CURR_VAR_NUM} */
            END; /* END of: I = 1 TO VL{CURR_VAR_NUM} */
         C_PAIR  = F0_TOTAL * F1_TOTAL;
         IF VL{CURR_VAR_NUM} > 1
         THEN DO;
            C_STAT  = MAX( C_STAT  / C_PAIR,  1 - C_STAT  / C_PAIR );
            X_STAT  = .5*(X_STAT  / C_PAIR  + 1);
            X_C_LIFT = (X_STAT - C_STAT) / C_STAT;
            X_C_LIFT_ADJ = X_C_LIFT / (VL{CURR_VAR_NUM} - 1);
         END;
         VAR_NAME = V{CURR_VAR_NUM};
         LEVELS = VL{CURR_VAR_NUM};
         OUTPUT __X__C_STAT;
      END; /* END of LAST._TYPE_ */

      IF LAST._TYPE_ AND 0 = VL{CURR_VAR_NUM}
      THEN DO;
         VAR_NAME = V{CURR_VAR_NUM};
         C_STAT  = .;
         X_STAT  = .;
         X_C_LIFT = .;
         X_C_LIFT_ADJ = .;
         LEVELS = .;
         OUTPUT __X__C_STAT;
         END;
   RUN;
   PROC PRINT DATA=__X__C_STAT;
      VAR VAR_NAME LEVELS C_STAT X_STAT X_C_LIFT_ADJ;
      FORMAT C_STAT X_STAT X_C_LIFT_ADJ 8.4;
   RUN;
   %MEND XC_STAT;
```

XC_Stat_With_Ranks

Parameters
1. **Data set** – entered as library.membername (if library not specified, WORK is assumed)
2. **Groups** – Number of values into which the user wants PROC RANK to group each variable's values.
3. **Target** – the response variable (must have numeric values 0 and 1, with no missing values). The "success" value level can be either 1 or 0 – the c-statistic is always calculated to be ≥ 0.50.
4. **Exclude_cols** – The name(s) of columns to be left out of the analysis, such as primary keys, discrete-valued columns, nominals, and columns known to be irrelevant. Multiple names must be SPACE delimited.

Usage Notes
1. A selection of **Groups** = "G" may provide fewer than G groups for some predictors, even if the predictor has more than "G" distinct values. The groups are determined by PROC RANK.
2. This macro processes all and only numeric columns.
3. It cannot be called from within a DATA step or PROC, and it ignores any column with all nulls.
4. All internal data sets begin __X__ to avoid conflicts with user data.
5. An observation with a predictor with a missing value is ignored when calculating statistics for predictor variables.

Results
%XC_STAT_WITH_RANKS (WORK.TEST, 4, Y,) using this "test" data set:

```
DATA TEST;
   INPUT XXX ABC DEF Y;
   DATALINES;
   11 1 1 0                        21 2 1 1
   10 1 1 0                         4 3 1 0
    8 1 1 0                         4 3 1 1
    8 1 1 1                         4 3 1 1
    8 1 1 1                         3 3 1 0
   18 2 2 1                        ;
RUN;
```

VAR_NAME	LEVELS	C_STAT	X_STAT	IV	SPEAR-MAN	X_C_LIFT	X_C_LIFT_ADJ
XXX	4	0.6000	0.6667	No calc	0.1808	0.1111	0.0370
ABC	3	0.5667	0.7000	No calc	0.1246	0.2353	0.1176
DEF	2	0.5833	0.5833	No calc	0.2887	0.0000	0.0000

Table 9 – XC_STATS_WITH_RANKS Output

SYNTAX AND DISCUSSION OF MACRO COLLAPSE_NOMINAL

Parameters
1. **Data set** – entered as library.membername (if library not specified, WORK is assumed)
2. **Input** – The name of the nominal variable to be investigated. In the full macro, this can be either numeric or character.
3. **Target** – the response variable (must have numeric values 0 and 1, with no missing values). The "success" value level can be either 1 or 0.
4. **W** – The name of the frequency variable or the value "1" if there is no frequency variable.
5. **Missing** – Y to include missing values in the frequency counts. Any other value in this parameter means that missing will be ignored.

Usage Notes

1. This macro examines a single, specified nominal variable and calculates the best way to combine value 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. The macro restricts the number of Target levels to 25 or less. Run-time for a target with 25 levels can be several minutes … depending on the computing environment.
5. Missing values (if included via parameter "Missing") are denoted by "~" in the results display.

Results

%COLLAPSE_NOMINAL(TEST, X_CHAR, Y, W,) using this "test" data set, gives result of example 3 but excluding the x-statistic.

```
DATA TEST;
   INPUT X_CHAR $ Y W;
   DATALINES;
A 0 689
A 1 166
B 0 2420
B 1 820
C 0 99
C 1 16
M 0 2
M 1 0
N 0 20
N 1 4
P 0 212
P 1 76
S 0 8818
S 1 1870
T 0 39
T 1 8
;
```

RUN_NUMBER	COLLAPSED	U(Y|X)	PRIOR U(Y|X)	%CHG IN U(Y|X)
1	NONE	0.0069826	.	.
2	N+T	0.0069825	0.0069826	0.0013594
3	N_T+S	0.0069813	0.0069825	0.0166436
4	B+P	0.0069706	0.0069813	0.1542303
5	C+M	0.0069310	0.0069706	0.5673104
6	C_M+N_T_S	0.0068481	0.0069310	1.1966065
7	A+C_M_N_T_S	0.0067110	0.0068481	2.0024513

Table 10 - COLLAPSE_NOMINAL output

Skeleton Code for COLLAPSE_NOMINAL

Because the complete macro code listing is very long, we've included a simplified, "skeleton" version here. This code can be copied into a SAS session and successfully run. There is limited input data error-checking and x-statistic is not computed.

```
/**    User must enter two parameters via the %COLLAPSE_NOMINAL statement.
These are the Data Set Name and the Number of Levels of the Nominal Variable.
The Numbers of Levels must be at least 3 but not more than 25.

%COLLAPSE_NOMINAL(<DATASET>, <LEVEL_N>); **/

/** Input Data set must use these variable names:
 X_CHAR for predictor ... A CHARACTER OF LENGTH 1 (MISSING ARE IGNORED)
 Y for target ... 0/1
 W for weight ... numeric (observation with non-positive weight is ignored)
**/

%MACRO MAKE_MACROVAR_ARRAYS_GLOBAL;
   %DO I = 1 %TO &NUMBER_LEVELS;
      %GLOBAL VN&I;
   %END;
   %DO I = 1 %TO %EVAL(&NUMBER_LEVELS * (&NUMBER_LEVELS - 1) / 2);
      %GLOBAL LABEL&I;
   %END;
%MEND MAKE_MACROVAR_ARRAYS_GLOBAL;
%MACRO INITIAL(DATASET);
PROC DATASETS LIB=WORK NOLIST;
   DELETE __X__:;
QUIT;
ODS SELECT CROSSTABFREQS;
ODS OUTPUT Measures = __X__MEASURES_ASSOC0;
PROC FREQ DATA = &DATASET;
   TABLES X_CHAR * Y
      / MEASURES OUT = __X__FREQOUT3(DROP = PERCENT);
WEIGHT W;
RUN;
DATA _NULL_;
   SET __X__FREQOUT3 END = EOF;
   RETAIN MAX_LENGTH_OBSERVED 0;
   MAX_LENGTH_OBSERVED = MAX(LENGTH(X_CHAR), MAX_LENGTH_OBSERVED);
  IF EOF THEN CALL SYMPUT("MAX_LENGTH_OBSERVED", MAX_LENGTH_OBSERVED);
RUN;
DATA _NULL_;
   CALL SYMPUT("NEXT_COLLAPSE", "X_CHAR");
RUN;
DATA __X__RUN_HISTORY;
   SET __X__MEASURES_ASSOC0(WHERE = (STATISTIC =: "Uncertainty Coefficient
C|R"));
   KEEP TABLE_NO COLLAPSED VALUE ASE;
   LENGTH COLLAPSED $&MAX_COL_WIDTH TABLE_NO $4;
   COLLAPSED = "NONE";
   TABLE_NO = "ALL";
RUN;
%MEND INITIAL;
%MACRO WRITE_TO_VN_MACRO_ARRAY;
   %DO II = 1 %TO &NUMBER_LEVELS;
      IF &II = COUNT_LEVELS THEN CALL SYMPUT("VN&II", &X);
   %END;
%MEND;
%MACRO FREQOUT2(DATASET, X);
PROC SORT DATA = &DATASET;
   BY &X Y;
RUN;
DATA __X__FREQOUT2(KEEP = NEXT_COLLAPSE Y COUNT)  ;
   SET &DATASET(KEEP = &X Y COUNT);
   BY &X;
   RENAME &X = NEXT_COLLAPSE; /* !!! NEEDED TO PREVENT COLUMN NAME CONFLICTS */
   RETAIN COUNT_LEVELS 0;
   IF FIRST.&X
   THEN DO;
      COUNT_LEVELS + 1;
      %WRITE_TO_VN_MACRO_ARRAY;
   END;
RUN;
%MEND FREQOUT2;
%MACRO WRITE_TO_VALUEX_DS_ARRAY;
   %DO I = 1 %TO &NUMBER_LEVELS;
      VALUEX{&I} = "&&VN&I";
   %END;
%MEND;
%MACRO WRITE_TO_LABEL_MACRO_ARRAY;
   %DO I = 1 %TO &NUMBER_LEVELS_PAIR;
      CALL SYMPUT("LABEL&I", LABEL&I);
   %END;
```

```
%MEND;
%MACRO FREQOUT3(X);
DATA __X__FREQOUT3;
   SET __X__FREQOUT2 END = EOF;
   DROP IJ_INDEX I J
   VALUE1 - VALUE&NUMBER_LEVELS
   LABEL1 - LABEL&NUMBER_LEVELS_PAIR;
   LENGTH C1 - C&NUMBER_LEVELS_PAIR
   LABEL1 - LABEL&NUMBER_LEVELS_PAIR
   VALUE1 - VALUE&NUMBER_LEVELS $ &MAX_COL_WIDTH;
   RETAIN VALUE1 - VALUE&NUMBER_LEVELS
          LABEL1 - LABEL&NUMBER_LEVELS_PAIR;
   ARRAY C{*} $ C1 - C&NUMBER_LEVELS_PAIR;
   ARRAY LABELX{*} $ LABEL1 - LABEL&NUMBER_LEVELS_PAIR;
   ARRAY VALUEX{*} $ VALUE1 - VALUE&NUMBER_LEVELS;
   IF _N_ = 1
   THEN DO;
      %WRITE_TO_VALUEX_DS_ARRAY;
   END;
   IJ_INDEX = 0;
   DO I = 1 TO &NUMBER_LEVELS - 1;
      DO J = I+1 TO &NUMBER_LEVELS;
         IJ_INDEX = IJ_INDEX + 1;
         IF (TRIM(VALUEX{I}) = TRIM(&X)) OR (TRIM(VALUEX{J}) = TRIM(&X))
         THEN C{IJ_INDEX} = '*';
      END;
   END;
   IJ_INDEX = 0;
   DO I = 1 TO &NUMBER_LEVELS - 1;
      DO J = I+1 TO &NUMBER_LEVELS;
         IJ_INDEX = IJ_INDEX + 1;
         IF C{IJ_INDEX} = '*'
         THEN DO;
            C{IJ_INDEX} = COMPRESS(VALUEX{I}||"_"||VALUEX{J});
            LABELX(IJ_INDEX) = COMPRESS(VALUEX{I})||"+"||COMPRESS(VALUEX{J});
         END;
         ELSE C{IJ_INDEX} = COMPRESS(&X);
      END;
   END;
   IF EOF
   THEN DO;
      %WRITE_TO_LABEL_MACRO_ARRAY;
   END;
RUN;
%MEND FREQOUT3;
%MACRO GENERATE_ALL_TABLE_COMBINATIONS;
   %DO I = 1 %TO &NUMBER_LEVELS_PAIR;
      TABLE C&I * Y / MEASURES;
   %END;
%MEND;
%MACRO FREQOUT3_TABLES;
   ODS SELECT NONE;
   ODS LISTING;
   ODS OUTPUT MEASURES = __X__MEASURES_ASSOC1;
   PROC FREQ DATA = __X__FREQOUT3;
     %GENERATE_ALL_TABLE_COMBINATIONS;
     WEIGHT COUNT;
     RUN;
   RUN;
%MEND FREQOUT3_TABLES;
%MACRO COLLAPSED;
DATA __X__COLLAPSED;
   LENGTH TABLE_NO $4;
   %DO I = 1 %TO &NUMBER_LEVELS_PAIR;
      COLLAPSED = "&&label&i";
      TABLE_NO = "c&i";
      OUTPUT;
   %END;
RUN;
%MEND COLLAPSED;
%MACRO REPORT;
DATA __X__UNCERTAINTY;
/* no BY variable is used in this MERGE */
   MERGE __X__MEASURES_ASSOC1(WHERE = (STATISTIC =: "Uncertainty Coefficient 
C|R"))
         __X__COLLAPSED;
   LABEL VALUE = "Uncertainty Value";
   LABEL ASE = "Asym. Std Err for Uncertainty";
RUN;
PROC SORT DATA = __X__UNCERTAINTY;
   BY DESCENDING VALUE;
RUN;
DATA _NULL_;
   SET __X__UNCERTAINTY;
   IF _N_ = 1 THEN CALL SYMPUT('BEST_COLLAPSE',TABLE);
   IF _N_ = 1 THEN CALL SYMPUT('NEXT_COLLAPSE',TABLE_NO);
RUN;
ODS SELECT ALL;
PROC FREQ DATA = __X__FREQOUT3; &BEST_COLLAPSE /* /MEASURES*/;
WEIGHT COUNT;
TITLE "BEST COLLAPSE FROM &NUMBER_LEVELS LEVELS TO %EVAL(&NUMBER_LEVELS-1)";
RUN;
ODS SELECT NONE;
PROC APPEND BASE = __X__RUN_HISTORY
            DATA = __X__UNCERTAINTY(OBS=1 DROP = CONTROL) FORCE NOWARN;
RUN;
DATA __X__RUN_HISTORY_X;
   LABEL VALUE = "U(Y|X)";
   LABEL PRIOR_VALUE = "PRIOR U(Y|X)";
   LABEL PCT_CHANGE = '%CHG IN U(Y|X)';
   SET __X__RUN_HISTORY;
   RETAIN RUN_NUMBER PRIOR_VALUE;
   IF _N_ = 1 THEN RUN_NUMBER = 0;
   IF _N_ = 1 THEN PRIOR_VALUE = .;
   RUN_NUMBER = RUN_NUMBER + 1;
   IF VALUE > 0 & PRIOR_VALUE > .
      THEN PCT_CHANGE = 100*(PRIOR_VALUE - VALUE) / PRIOR_VALUE;
   OUTPUT;
   PRIOR_VALUE = VALUE;
RUN;
ODS SELECT
%IF &NUMBER_LEVELS < 4 %THEN ALL;
%ELSE NONE;
; /* ; required to complete the ODS Statement */
PROC PRINT DATA = __X__RUN_HISTORY_X NOOBS LABEL;
VAR RUN_NUMBER COLLAPSED VALUE PRIOR_VALUE PCT_CHANGE;
   FORMAT VALUE PRIOR_VALUE PCT_CHANGE 10.7;
TITLE "HISTORY OF COLLAPSING OF LEVELS";
RUN;
%MEND REPORT;
* The 3 main macros below call the component macros;
%MACRO MANAGE_COLLAPSE_ITERATIONS;
   %DO %WHILE (&NUMBER_LEVELS >= 3);
      %COLLAPSE_NOMINAL_MAIN(__X__FREQOUT3, &NEXT_COLLAPSE, Y)
   %END;
%MEND;
```

```
%MACRO COLLAPSE_NOMINAL_MAIN(FREQDATA, X, Y);
   %LET NUMBER_LEVELS_PAIR = %EVAL((&NUMBER_LEVELS)*(&NUMBER_LEVELS - 1) /2);
   %FREQOUT2(&FREQDATA, &X);
   %FREQOUT3(NEXT_COLLAPSE);
   %FREQOUT3_TABLES;
   %COLLAPSED;
   %REPORT;
   %LET NUMBER_LEVELS=%EVAL(&NUMBER_LEVELS - 1);
   RUN;
%MEND;
%MACRO COLLAPSE_NOMINAL(DATASET, LEVEL_N);
   %LET MAX_LEVELS_PERMITTED = 25;
   %GLOBAL NUMBER_LEVELS
           NUMBER_LEVELS_PAIR
           BEST_COLLAPSE
           NEXT_COLLAPSE
           MAX_COL_WIDTH
           MAX_LENGTH_OBSERVED;
   %IF %EVAL(&LEVEL_N) GT &MAX_LEVELS_PERMITTED
   %THEN
   %DO;
      %PUT ......................... USER ENTERED LEVEL_N GREATER THAN 25;
      %PUT .... ENDING;
      %RETURN;
   %END;
   %LET NUMBER_LEVELS = &LEVEL_N; /* IS NUMBER_LEVELS Needed? – USE LEVEL_N? */
   %LET MAX_COL_WIDTH = %EVAL(&LEVEL_N * 2 - 3);
   %MAKE_MACROVAR_ARRAYS_GLOBAL;
   %INITIAL(&DATASET);
   DATA _NULL_;
      %IF %EVAL(&MAX_LENGTH_OBSERVED) GT 1
      %THEN %DO;
         %PUT ......................... LENGTH OF X_CHAR EXCEEDS 1;
         %PUT .... ENDING;
         %RETURN;
      %END;
   %MANAGE_COLLAPSE_ITERATIONS;
%MEND COLLAPSE_NOMINAL;
```

Contact the authors for macro code for the complete XC_STAT and COLLAPSE_NOMINAL.

References

Ames, J., Hackett, C., and Lund, B. (2008). Long-Term Value Modeling in the Automobile Industry", *MWSUG 2010, Proceedings*, Midwest SAS Users Group, Inc. Paper S01-2008

Begg, C. B. and Gray, R. (1984). Calculation of polychotomous logistic regression parameters using individualized regressions, Biometrika 71, 1, pp. 11-18

Carpenter, Art, 2004, Carpenter's Complete Guide to the SAS® Macro Language 2nd Edition, Cary, NC: SAS Institute Inc., 2004

Fang, J. (2011). "Compare the Effectiveness of Different methodologies in Prediction of Dichotomous Outcome", *SAS Global Forum 2011*, Paper 343-2011.

Hermansen, S. W. (2008). "Evaluating Predictive Models: Computing and Interpreting the c Statistic", *SAS Global Forum 2008*, Paper 143-2008.

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.

Sarma, K. S. (2007), "Variable Selection and Transformation of Variables in SAS® Enterprise Miner™ 5.2", *NESUG 2007 Proceedings*, Northeast SAS Users Group, Inc.

Thompson D. (2009), "Ranking predictors in logistic regression", *MWSUG 2009, Proceedings*, Midwest SAS Users Group, Inc., Paper D10-2009.

Xie, L. (2009). "AUC calculation using Wilcoxon Rank Sum Test", *SAS Blog: SAS Programming for Data Mining Applications*, http://www.sas-programming.com/2009/10/auc-calculation-using-wilcoxon-rank-sum.html, October 23, 2009.

Contact Information

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

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

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