Paper RF10-2015

A Macro for Systematic Treatment of Special Values in Weight of Evidence Variable Transformation

Chaoxian Cai, Automated Financial Systems, Exton, PA

ABSTRACT

Weight of evidence (WOE) recoding is a commonly applied technique in credit scoring model development to transform continuous predictor variables. Many predictor variables often come with some special values which are set beyond their normal ranges to annotate particular business meanings. PROC RANK and PROC UNIVARIATE are used to rank and group an input data into WOE bins for predictor variables. Special values are usually being placed at the last bucket after binning procedures. There is a possibility of losing valuable information and predictive power from these special values if we do not separate them into discrete bins. This paper presents a macro that can systematically separate every special value from their normal values and compute the associated WOE for each special value. The macro can handle hundreds of predictor variables in large scale. We can use this macro program to explore special values effectively and discover their potential predictive powers during the logistic model development process.

INTRODUCTION

Logistic regression is widely used in predictive modeling to model the causal relationships between dependent (or response) variables and independent (or predictor) variables. For example, in credit scoring, logistic regression is used to model the probability of default of an account from a number of consumer behavioral characteristic variables. In mathematical term, the logistic regression equation is

$$\log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_{n-1} x_{n-1} + \beta_n x_n ,$$

where p is the probability of an event and x_0, x_1, x_{n-1} and x_n are predictor variables. Very often, for numerical continuous variables, we need to do some variable transformations in order to improve their predictive powers in a logistic model. There are many approaches to transform the predictor variables, such as logarithm and power transformations (Steven & Lund, 2011). In credit scoring, it is common to use weight of evidence (WOE) variable transformation, in which we group an input continuous variable into discrete bins in either ascending or descending order and compute the corresponding WOE value for each bin. In the regression step, the WOE values are simply used to replace the original input values of the predictor variables that have undergone WOE transformations. There are multifold advantages in using WOE transformation: The WOE transformation can address non linearity of predictor variables, smooth the roughness of input data, and improve the overall prediction power. Although WOE transformation is mainly for continuous values, categorical values and missing values can also be treated by WOE transformation. For example, if a predictor variable has numeric data that include normal values (continuous), special values (discrete), and missing values, the WOE transformation provides a unified treatment on all numeric data regardless of whether the data is continuous, discrete, and/or missing. Further, in an early model development stage, WOE measures can be used for variable selection. We can exclude those variables from the logistic model that show little prediction powers.

Special values are values beyond the normal range but carry annotated business information. These information may be very powerful in predicting the target event. There is a possibility of losing valuable information and predictive power from these special values if we do not separate them into different groups. Therefore, during the model development life cycle, it would be beneficial if we can harness this data information. As an alternative approach, we can create a dummy variable for each special value we will encounter in an input data set, and investigate its potential predictive power. However, the shortcomings of this approach are obvious: It not only introduces too many independent variables to the model but also add too much burden to model developers, particularly at the time of early model development stage, when there are dozens or even hundred of variables that needs to be explored.

WOE TRANSFORMATION

The WOE transformation looks at bivariate relationship between the response variable and the predicator variable for each predictor at one time. The process iterates all input variables of interest. For each predictor, its input values are first separated into discrete bins in order, and then, for each bin, the proportion of accounts being good (*pct_good*) and the proportion of accounts being bad (*pct_bad*) are calculated. A WOE value measures the odds of being good

over being bad using the following formula (Siddiq, 2006, p.81):

$$WOE = \log(\frac{pct\_good}{pct\_bad}) \times 100,$$

where *pct_good* is the percentage of good accounts and *pct_bad* is the percentage of bad accounts. Therefore, the WOE value gauges the power of each predictor variable in separating good and bad accounts.

SAS® Enterprise Miner™ provides graphical user interface and computational nodes for variable binning, rebinning, and WOE computation (Sarma, 2013; Brown, 2014). In Base SAS programming, variable binning and WOE computation are often involving PROC RANK or PROC UNIVARIATE procedures (Bilenas, 2009). PROC RANK is a Base SAS procedure designed to rank continuous variables, in which the incoming numeric data can be easily separated into a number of bins either on ascending or descending order using GROUPS= options. One shortcoming of PROC RANK is that it cannot handle weighted samples directly. PROC UNIVARIATE is also used to rank continuous variables. The procedure has a WEIGHT statement which can handle input data with sample weights. PROC UNIVARIATE produces boundaries of quartiles, deciles, or percentiles for the variables of interest. The incoming numeric data can be placed into discrete bins according to these boundaries. In this paper, we will present and implement a SAS macro program that can systematically perform WOE transformations for all predictor variables, including special values, normal values, and/or missing values.

PROGRAM FLOW CHART

The macro program flow chart is shown in Figure 1. Starting from an input data set and an input metadata, the program processes WOE transformation for each predictor variable of interest. There are three basic steps in grouping input values and computing WOE measures. The first step is to separate observations with special values, missing values, and normally ranged values into separate data sets; the second step is to rank and group the incoming data into distinct bins; the third step is to compute the bivariate statistics including WOE values. We will discuss each step and its SAS implementation in detail in the next section.

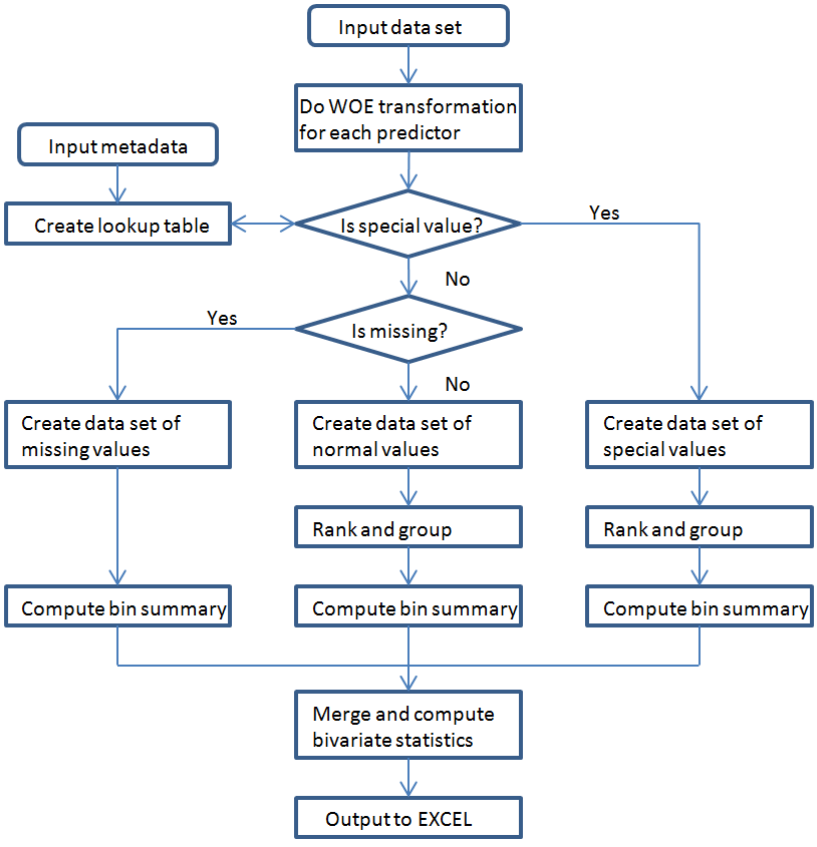

Figure 1. Program Flow Chart.

DETAILS OF IMPLEMENTATION

First, we need to separate special values from missing values and normally ranged values for an input data set. The special values of all predictor variables are documented in a metadata table, usually in an EXCEL spreadsheet. Figure 2 shows an example of such metadata table, which lists all predictor variables of interest. The normal ranges of a predictor variable are defined between its minimum and maximum values in the table, and special values are documented and used to annotate particular business meanings or situations. In the example, all special values of a predictor variable are set above its maximum value for convenience. We can use PROC IMPORT to read this spreadsheet and create a SAS data set as the input metadata table.

	A	B	C	D	E
1	**Variable**	**Label**	**Minimum**	**Maximum**	**SpecialValues**
2	AGE	Age	0	125	998, 999
3	TMADDR	Time at Current Address in Months	0	1000	9996, 9999
4	TMJOB	Time at Current Job in Months	0	1000	9991, 9992, 9993
5	NCARDS	Number of Active Credit Cards	1	25	998, 999
6	NLOANS	Number of Outstanding Loans	0	10	91, 92, 95
7	INCOME	Annual Income in Dollars	0	200000	999991, 999992, 999995
8	…	…	…	…	…

Figure 2. A metadata example in EXCEL spreadsheet.

The metadata data set is used as a lookup table for the separation of specials values from normal values. We set up the lookup table by declaring a hash table for this data set using the following SAS code:

```
if 0 then set &metadata (keep=Variable Maximum);
if _N_=1 then do;
    declare hash hashtab (dataset: "&metadata");
    hashtab.definekey("Variable");
    hashtab.definedata("Maximum");
    hashtab.definedone();
end;
```

OBS	AGE	TMADDR	TMJOB	NCARDS	NLOANS	INCOME	…	EVENT	WEIGHT
1	45	18	35	2	0	0	…	0	50
2	35	116	55	2	1	3200	…	1	1
3		154	120	2	1	3300	…	1	1
4		192	6	1	0	1500	…	1	1
5	30	48	108	2	2	0	…	0	50
6	25		64	2	2	0	…	1	1
7	42	9996	30	2	0	1900	…	0	50
8	35	55	9991	2	0	0	…	0	50
9	998	255	33	1	0	1700	…	1	1
10	999	18	30	2	2	3400	…	1	1
…	…	…	…	…	…	…	…	…	…

Figure 3. An input data set example.

Figure 3 shows an input data set example. Some data values such as missing and special values are included in columns AGE, TMADDR, and TMJOB for illustration purpose. The column EVENT is the target response variable. A value of 1 indicates an account in default or bad status while a value of 0 indicates an account in a non-default or good status. The sample WEIGHT reflects the event frequencies in the population. This input data set is read and checked against the lookup table to find whether there are special values that are greater than the maximum value (*&maxval*) of variable *&x*. If true, we put the incoming observation to data set *x_special*; if false, we further check whether *&x* is missing. We put all observations with missing *&x* to data set *x_missing* and all other observations to data set *x_inrange*.

```
data x_inrange x_special x_missing;
    set &data (keep=&x &y &weight);
    if &x=. then output x_missing;
```

```
    else if (&maxval ne .) and (&x > &maxval) then output x_special;
    else output x_inrange;
run;
```

Next, we rank the resulted data sets (*x_inrange, x_special, and x_missing*) by *&x* and group them into distinct bins. As mentioned earlier, there are several approaches to do variable binning. In this paper, we will handle input data sets with sample weights based on the underlying principle of PROC RANK but not a direct call of the procedure itself. As documented in the PROC RANK Procedure (SAS Institute Inc., 2015), the formula to assign group values is

$$FLOOR(\frac{rank \times k}{n+1}),$$

where *FLOOR* is the round down integer function, *rank* is the value's order rank after all values are sorted in either ascending or descending order, *k* is the values of GROUPS= option, and *n* is the number of observations of the ranking variable excluding missing values. With this formula, we can rank input data with sampling weights. For an observation with sample weight w_i, we count the sampling weight as if there were w_i duplicate observations. Therefore, the total number of observations in the population for a data set with sample weights w_i is $n=sum(w_i)$, where *sum()* is the summation function. Since the input values are sorted by *&x*, the value's order rank is from *1* to *n* sequentially with arbitrary tie breaks for duplicate values. Thus, the modified formula for assigning the group value for each observation becomes

$$FLOOR(\frac{rank \times k}{sum(w_i)+1}).$$

We will use this formula to group the predictor variable *&x* into different bins for the data set *x_inrange*. The grouping for the data set *x_special* is straightforward, and we just enumerate each special value and create a separate bin for every special value. For the data set *x_missing*, we create a separate single bin for all observations with missing *&x*. The SAS code for doing ranking and grouping is shown below.

```
/* create a single bin for all x missing */
data x_missing;
    set x missing;
    r_&x =.;
run;

/* separate each special value to a separate bin */
proc sort data=x_special; by &x; run;
data x_special;
    set x_special;
    by &x;
    if _N_=1 then count=&groups-1;
    if first.&x then count+1;
    r_&x=count;
    drop count;
run;

proc sql noprint;
select sum(&weight) into :EXPANDED_N
from x_inrange;
quit;

/* rank and group observations with normal x values */
proc sort data=x_inrange; by &x; run;
data x_inrange;
    set x_inrange;
    by &x;
    retain rank r_&x 0;
    rank + &weight;
    if first.&x then r_&x=int(rank*&groups/(&EXPANDED_N+1));
run;
```

After ranking and grouping, we compute the bivariate statistics for each group, including WOE values, information values, and so on. We first use PROC SQL to aggregate the summary statistics of data set *x_inrange, x_special, and*

x_missing separately and then merge the results together via two union operations. The bivariate statistics for each predictor variable are subsequently calculated from the combined summary statistics. The SAS code is listed as follows:

```
/* compute summary satistics and union */
proc sql noprint;
create table xbin_sum as
select r_&x,
       sum(&weight) as n,
       sum(&weight*&x)/sum(&weight) as xmean,
       min(&x) as xmin,
       max(&x) as xmax,
       sum(&weight*&y) as n_bad,
       (calculated n) - (calculated n_bad) as n_good,
       (calculated n_bad)/(calculated n) as ymean
from x_inrange
group by r_&x
union
select r_&x,
       sum(&weight) as n,
       sum(&weight*&x)/sum(&weight) as xmean,
       min(&x) as xmin,
       max(&x) as xmax,
       sum(&weight*&y) as n_bad,
       (calculated n) - (calculated n_bad) as n_good,
       (calculated n_bad)/(calculated n) as ymean
from x_special
group by r_&x
union
select r_&x,
       sum(&weight) as n,
       . as xmean,
       . as xmin,
       . as xmax,
       sum(&weight*&y) as n_bad,
       (calculated n) - (calculated n_bad) as n_good,
       (calculated n_bad)/(calculated n) as ymean
from x_missing;

/* compute bivariate statistics */
create table bivar_sum as
select *,
       sum(n_good) as tot_good,
       sum(n_bad) as tot_bad,
       n_good/(calculated tot_good) as pct_good,
       n_bad/(calculated tot_bad) as pct_bad,
       log(calculated pct_good/calculated pct_bad) as woe,
        (calculated pct_good - calculated pct_bad)*(calculated woe) as infoval,
       n_bad/(n_bad+n_good) as badrate,
        (calculated tot_bad)/(calculated tot_good + calculated tot_bad) as tot_badrate
from xbin_sum
where r_&x ne . or n_good ne . or n_bad ne .;
quit;
```

After the above bivariate statistics are computed, PROC EXPORT is called to output the results to an EXCEL file, one sheet for each predictor variable *&x*.

```
proc export data=bivar_sum outfile="&dir.\woe_summary.xls"
    dbms=excel replace;
    sheet="&x";
run;
```

The complete macro program is displayed in Appendix A. The steps of special value separation, variable binning, and

WOE calculation are implemented in the macro function *%xtran2woe()*. In order to compute the bivariate statistics for each predictor variable, we need to process all predictor variables in the input data set sequentially; for each predictor variable, the macro function *%xtran2woe()* is called to perform the task. We automate the entire process by implementing a main macro function *%do_xtran2woe()*. The commands to invoke *%xtran2woe()* macro function is dynamically generated and executed at the run time by CALL EXECUTE statement as shown below.

```
command='%xtran2woe' || "(data=&data, x=" || strip(name) || ", y=&y,
        groups=&groups, weight=&weight, maxval=" || strip(maxval) || ")";
call execute('%nrstr(' || strip(command) || ')');
```

After WOE values are exported to the EXCEL file, variable rebinning is usually needed to monotonize the bivariate response of WOE measures versus the predictor variables. This can be accomplished first by coarse rebinning , followed by fine-tuned rebinning manually. In EXCEL, we may implement certain monotone algorithm and graphical drawings to facilitate variable rebinning using EXCEL VBA or other programming languages. Finally, if-conditions or input formats are created from fine-tuned WOE bins and are applied to read the input data set and convert the original values to the associated WOE values, which are used in place of its original values in the subsequent steps of logistic regression.

CONCLUSION

In this paper, we have presented a SAS macro program that implements a systematic treatment on special values, normal values, and missing values for WOE transformations of predictor variables in logistic regression. One distinguished feature in our implementation is that, for each predictor variable, we separate special values, normal values, and missing values into three disjoint data sets and perform variable binning within each data set. In the step of variable binning, we have used an ab-initio approach that leads to a concise implementation of ranking and grouping procedure that can handle input data with sampling weights. The macro can be used to process hundreds of predictor variables in large scale. We can use this macro program to explore special values effectively and discover their potential predictive powers during the logistic model development process.

REFERENCES

1. SAS Institute Inc. (2015). *Base SAS® 9.4 Procedures Guide*, *Fourth Edition.* Available at https://support.sas.com/documentation/cdl/en/proc/67916/PDF/default/proc.pdf
2. Siddiqi, N. (2006). *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring*. Hoboken, NJ: John Wiley & Sons, Inc.
3. Brown, I. L. J. (2014). *Developing Credit Risk Models Using SAS® Enterprise Miner™ and SAS/STAT®: Theory and Applications*. Cary, NC: SAS Institute Inc.
4. Sarma, K. S. (2013). *Predictive Modeling with SAS® Enterprise Miner™: Practical Solutions for Business Applications, Second Edition*. Cary, NC: SAS Institute Inc.
5. Bilenas, J. V. (2009). “Using PROC RANK and PROC UNIVARIATE to Rank or Decile Variables”, *NESUG 2009 Proceedings*, Northeast SAS Users Group, Inc.
6. Steven, R., and Lund B. (2011). “Before Logistic Modeling – A Toolkit for Identifying and Transforming Relevant Predictors”, *MWSUG 2011 Proceedings*, Midwest SAS Users Group, Inc.

CONTACT INFORMATION

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

Name: Chaoxian Cai
Enterprise: Automated Financial Systems
Address: 123 Summit Drive
City, State ZIP: Exton, PA 19341
E-mail: cai.charles.x@gmail.com

APPENDIX A

The following is the integrated macro program discussed in the context:

```
/*************************************************************************
** do_xtran2woe.sas
** A macro for systematic treatment of special values in weight
** of evidence variable recoding for logistic models
** 05SEP2015 - Chaoxian Cai
**
** Input parameters:
**   data: the SAS data set that stores relational data between response
**         variable and all predictor variables
**   metadata: the SAS data set that contains variable range definitions
**   y: the binary response variable (1: bad, 0: good)
**   weight: the sampling weight
**   groups: the number of initial bins
**
** Output file:
**   woefile: An EXCEL file that contains bivariate statistics
**   between the response variable and each predictor variable in separate
**   spreadsheets.
*************************************************************************/
%let dir=C:;
%let woefile=woe_summary.xls;

%macro do_xtran2woe(data=, metadata=, y=, groups=, weight=);

/* delete old woe_summary.xls if existed */
%let rc=%sysfunc(fileexist("&dir.\&woefile"));
%if &rc %then %sysexec del "&dir.\&woefile";

proc contents data=&data (keep=_numeric_) noprint out=numericvars;
run;

data numericvars;
    set numericvars;
       if upcase(name) eq upcase("&y") or upcase(name) eq upcase("&weight") then
delete;
run;

proc sort data=numericvars; by name; run;

data  null ;
    if 0 then set &metadata (keep=Variable Maximum);
       if _N_=1 then do;
           declare hash hashtab (dataset: "&metadata");
              hashtab.definekey("Variable");
              hashtab.definedata("Maximum");
              hashtab.definedone();
       end;

    set numericvars;
       if (hashtab.find(key:name)=0) then maxval=Maximum;
       else maxval=.;

       command='%xtran2woe' || "(data=&data, x=" || strip(name) || ", y=&y,
groups=&groups,
             weight=&weight, maxval=" || strip(maxval) || ")";
       put command;
       call execute('%nrstr(' || strip(command) || ')');
run;
%mend do_xtran2woe;
```

```
%macro xtran2woe(data=, x=, y=, groups=, weight=, maxval=);
data x_inrange x_special x_missing;
    set &data (keep=&x &y &weight);
       if &x=. then output x_missing;
       else if (&maxval ne .) and (&x > &maxval) then output x_special;
       else output x_inrange;
run;

/* create a single bin for all x missing */
data x_missing;
    set x_missing;
       r_&x =.;
run;

/* separate each special value to a separate bin */
proc sort data=x_special; by &x; run;
data x_special;
    set x_special;
       by &x;
       if _N_=1 then count=&groups-1;
       if first.&x then count+1;
       r_&x=count;
       drop count;
run;

proc sql noprint;
select sum(&weight) into :EXPANDED_N
from x_inrange;
quit;

/* rank and group observations with normal x values */
proc sort data=x_inrange; by &x; run;
data x_inrange;
    set x_inrange;
       by &x;
       retain rank r_&x 0;
       rank + &weight;
       if first.&x then r_&x=int(rank*&groups/(&EXPANDED_N+1));
run;

/* compute summary satistics and union */
proc sql noprint;
create table xbin_sum as
select r_&x,
       sum(&weight) as n,
          sum(&weight*&x)/sum(&weight) as xmean,
          min(&x) as xmin,
          max(&x) as xmax,
          sum(&weight*&y) as n_bad,
          (calculated n) - (calculated n_bad) as n_good,
          (calculated n_bad)/(calculated n) as ymean
from x_inrange
group by r_&x
union
select r_&x,
       sum(&weight) as n,
          sum(&weight*&x)/sum(&weight) as xmean,
          min(&x) as xmin,
          max(&x) as xmax,
          sum(&weight*&y) as n_bad,
          (calculated n) - (calculated n_bad) as n_good,
```

```
          (calculated n_bad)/(calculated n) as ymean
from x_special
group by r_&x
union
select r_&x,
       sum(&weight) as n,
          . as xmean,
          . as xmin,
          . as xmax,
          sum(&weight*&y) as n_bad,
          (calculated n) - (calculated n_bad) as n_good,
          (calculated n_bad)/(calculated n) as ymean
from x_missing;

/* compute bivariate statistics */
create table bivar_sum as
select *,
       sum(n_good) as tot_good,
          sum(n_bad) as tot_bad,
          n_good/(calculated tot_good) as pct_good,
          n_bad/(calculated tot_bad) as pct_bad,
          log(calculated pct_good/calculated pct_bad) as woe,
       (calculated pct_good - calculated pct_bad)*(calculated woe) as infoval,
          n_bad/(n_bad+n_good) as badrate,
       (calculated tot_bad)/(calculated tot_good + calculated tot_bad) as tot_badrate
from xbin_sum
where r_&x ne . or n_good ne . or n_bad ne .;
quit;

/* Export bivar_sum to excel spreadsheet for next step of
   monotonic rebinning */
proc export data=bivar_sum outfile="&dir.\&woefile"
    dbms=excel replace;
      sheet="&x";
run;

%mend xtran2woe;


/* Following is an example to make macro fuction call of %do_xtran2woe */
%let data=testindata;
%let metadata=testmtdata;
%let y=EVENT;
%let groups=20;
%let weight=WEIGHT;

%do_xtran2woe(data=&data, metadata=&metadata, y=&y, groups=&groups, weight=&weight)

/* End of do_xtran2woe.sas */
```