Paper 174-2007

SAS® Macros for Automated Model Selection when the Response Variable is Nominal: Needs and Solutions Involving PROC CATMOD[1]

Bonnie E. Kegan, U.S. Census Bureau, Washington D.C.
Todd R. Williams, Social Security Administration, Baltimore, MD

ABSTRACT

Currently, SAS does not provide the capability to fit logistic regression models using automated model selection methods, such as forward selection and backwards elimination, when the response variable is nominal. PROC LOGISTIC can provide automated model selection, but is only capable of fitting models with binary or ordinal response variables. This paper details a group of macros that are written to perform automated forward model selection using PROC CATMOD. It is intended for SAS users with a solid background in SAS Macros and the model fitting capabilities found in SAS/STAT®. These derived macros will, for a given response variable, 1) estimate parameters by fitting logistic regression models using the forward selection procedure, 2) save resulting statistics from each of the model fittings using the Output Delivery System (ODS), and use the results to calculate the Akaike Information Criterions (AIC) which determine the best fitting model. Both categorical and continuous explanatory variables can be used in this procedure; however the values of the continuous variables must be recoded to categorical variables based on their quintiles.

KEY WORDS: Logistic Regression, Forward Selection, PROC CATMOD

BACKGROUND

When selecting from a large group of variables, automated model selection (i.e. forward selection or backwards elimination) is a useful tool for the analyst trying to choose a set of explanatory variables that have the strongest correlation with the response variable. Automated model selection is currently available in SAS PROC REG and PROC LOGISTIC. PROC REG is used for fitting single or multiple regression models where the response variable is continuous. When the response variable is a binary or ordinal variable PROC LOGISTIC can be used to fit a logistic regression model. However in order to fit a model for a nominal categorical variable with more than two levels PROC CATMOD should be used to fit a generalized logit model. Unlike PROC REG and PROC LOGISTIC, PROC CATMOD does not have automated model selection available.

The need for an automated model selection procedure when fitting logistic regression models using PROC CATMOD arises due to research involving the application of Flexible Matching Imputation (FMI) to the Manufactured Housing Survey (MHS) (Kegan, Williams 2002). FMI (Williams, 2001) combines hot-deck imputation with model-based methodology. This method identifies and gives a ranking to the matching variables that should be used for any given missing variable or combination of missing variables. By fitting logistic regression models to the data in which these variables are actually observed, a set of matching variables is determined for a particular combination of missing variables. In FMI, missing continuous variables are modeled simultaneously using multivariate linear regression; whereas, missing categorical variables are modeled individually using polytomous logit models. The rankings of the matching variables are determined by fitting the models using a forward stepwise procedure in which the first explanatory variable kept is the most important, the second variable kept is the 2nd most important, and so forth. During the matching step, if no match can be found based on all the matching variables, the lowest ranked variable is dropped and a new attempt at matching is made using only the remaining variables. Dropping of variables continues until a match is found. The FMI procedure is meant to be an automated procedure, so once a group of possible matching variables is selected, the procedure will find the best set of matching variables and perform the hot-deck imputation without your intervention. In order to program the categorical modeling portion of the method, an automated model selection procedure for PROC CATMOD has to be developed. This paper describes the development of this procedure using SAS macros and provides an example of its application to the MHS.

[1] This report is released to inform interested parties of ongoing research and to encourage discussion of work in progress. The views expressed on methodological issues are those of the author(s) and not necessarily those of the U.S. Census Bureau or the Social Security Administration.

METHODS: DATA PREPARATION

Prior to using PROC CATMOD to fit the logistic regression models there are some necessary data preparations that must be completed. The first step separates the survey data records that need imputation from the data records that have complete information. In order to fit the models properly, the data records must contain variables that are completely reported (non-missing). The second step is to recode the continuous variables into classification variables using each variable's quintile distribution. For example, **length** is a continuous variable and the 20th, 40th, 60th, 80th, and 100th percentiles can be calculated from the data (denoted as **l_p20**, **l_p40**, etc). Then each record is assigned a value one through five (representing the quintile level) for a new variable **lgrp** as follows:

```
If length <= l_p20 then lgrp=1;
Else if length <= l_p40 then lgrp=2;
Else if length <= l_p60 then lgrp=3;
Else if length <= l_p80 then lgrp=4;
Else if length > l_p80 then lgrp=5;
```

After the continuous variables have been recoded the data need to be tabulated in an *n*-way contingency table. All possible combinations of the explanatory variables need to be included. PROC FREQ will produce this table for the data if you use the TABLES statement with the options SPARSE and OUT=. In the TABLES statement list all possible explanatory variables separated by "*" (see code below).

```
proc freq data=imp.resid2;
   tables aircond*setup* site* secured* purchase* titled*
          bedrooms* sections* dlregn* wgrp* lgrp* prgrp
          / sparse out=imp.combos noprint;
run;
```

The code reads in the survey data file (*imp.resid2*) and produces a new data file (*imp.combos*) that contains all of the variables listed in the TABLES statement and two new variables **count** and **percent**, which are the cell counts and percentages from the *n*-way contingency table.

The next step in the data preparation is to handle cells with zero cell counts. Since the SPARSE option produces all possible combinations of the levels of the variables specified in the TABLES statement (even combinations that do not occur in the data) the analyst must identify zero cell counts as either structural or sampling zeros. Structural zeros occur when a specific combination of levels of the survey variables is impossible and will never occur. All other zero counts are sampling zeros, i.e. a specific combination of levels of survey variables is possible but did not occur in this particular data. Since PROC CATMOD treats all zero counts as if they were structural zeros, it is necessary to add an arbitrary small value, in this case 0.00001, to all zero cell counts that represent sampling zeros. This allows the cells to be treated as sampling zeros instead of structural zeros. As an example, the variables **setup** and **secured** are related so that if **setup** equals one **secured** has to be zero; otherwise, if **setup** equals two, three, or four, **secured** can have the values one thru three. The table below shows the cells that represent the structural zeros.

	SECURED			
SETUP	**0**	**1**	**2**	**3**
1		0	0	0
2	0			
3	0			
4	0			

The code below shows how this edit of zero cell counts is performed. The input data file is the output data file from the PROC FREQ step displayed above.

```
data imp.strzero;
    set imp.combos;
    if count=0 then do;
        select;
            when (setup in (2,3,4) and secured in (0)) count=0;
            when (setup in (1) and secured in (1,2,3)) count=0;
            when (purchase in (2,3) and prgrp in (2,3,4,5)) count=0;
            otherwise count=1e-5;
        end;
    end;
run;
```

The final data preparation step concerns the response variable in the fitted model. In the following code, PROC FREQ creates a data file for each potential response variable. The data contains the frequency of occurrences for each level (value) of the response variable (option OUTPCT). The macro **frqcalc** reformats the data files and appends them into a single new data file "*imp.frqallcnts*". This new data file is the input for the macro **adjdata**.

```
proc freq data=imp.resid2 noprint ;
 tables aircond /out=frqac outpct ;
 tables setup /out=frqsup outpct;
 tables site /out=frqsite outpct;
 tables secured/out=frqsec outpct;
 tables purchase/out=frqpur outpct;
 tables titled/out=frqtit outpct;
 tables bedrooms /out=frqbed outpct;
run;

%macro frqcalc(dset,var1);
 data frqcnt (drop=&var1 count);
     length name $9 ;
     set &dset;
     name="&var1";
     levels=&var1;
     percent=percent/100;
 run;

 proc append base=imp.frqallcnts data=frqcnt force;
 run;
%mend frqcalc;

%frqcalc(frqac,aircond);
%frqcalc(frqsup,setup);
%frqcalc(frqsite,site);
%frqcalc(frqsec,secured);
%frqcalc(frqpur,purchase);
%frqcalc(frqtit,titled);
%frqcalc(frqbed,bedrooms);
```

The macro **adjdata**'s purpose is to adjust the tabulations in each contingency cell based on the frequency of the levels of the response variable to be modeled by PROC CATMOD. All sampling zero cells will be adjusted by multiplying 0.00001 by the proportion of the level of the response variable specific to that cell. All non-zero cell counts will be adjusted by adding an amount equal to 0.00001 times the proportion for the specific level of the response variable for that cell. All structural zeros will remain zero.

As an example:
The variable **aircond** has two levels (values):

- Level = 1 (79% of the data) where 0.00001 x 0.79 equals 0.0000079 and
- Level = 2 (21% of the data) where 0.00001 x 0.21 equals 0.0000021.

Contingency table cells with:

- Table cells containing sampling zeros where **aircond** =1, the cell count will be 0.0000079.
- Table cells containing sampling zeros where **aircond** =2, the cell count will be 0.0000021.
- Table cells containing non-zero counts where **aircond** =1, the cell count will have 0.0000079 added to it.
- Table cells containing non-zero counts where **aircond** =2, the cell count will have 0.0000021 added to it.

Our code for **adjdata** is shown below.

```
%macro adjdata(dep);
 data frqcnt (drop=name);
     set imp.frqallcnts;
     if name="&dep";
     &dep=levels;
 run;
 proc sort data=imp.strzero out=combos1(drop=percent);
     by &dep;
 run;
 data imp.combos3;
```

```
     merge combos1 frqcnt;
     by &dep;
     if count=1e-5 then count=count*percent;
     else if count ne 0 then count=count+percent*1e-5;
 run;
%mend adjdata;
```

METHODS: MODEL SETUP, FITTING AND SELECTION

Six additional macros, one macro named **models** that calls five additional macros (**mod1var-mod5var**), are written in SAS to handle the model setup, fitting, and selection. The macro **models** is responsible for directing the setup of each model to be fitted beginning with 1-variable models and ending with 5-variable models. The macro **models** also collects the Akaike Information Criterions (AIC) for all the models being fitted at each stage from the other five macros and selects the model with the minimum AIC to be the base model for the next stage. The macros **mod1var, mod2var, mod3var, mod4var,** and **mod5var** use PROC CATMOD to fit the models and calculate the AIC. The AIC is calculated by the formula:

$$\mathrm{AIC} = -2\log L + 2[(k\text{-}1) + s]$$

, where log*L* is the log of the Likelihood function, *k* is the number of levels for the response variable, and *s* is the total number of available model parameters including the intercept. Maximum likelihood analysis is the default estimation method for the default response function (generalized logits) in PROC CATMOD.

A call to **adjdata**, specifying the response variable is made by the macro **models**. Five sections comprise the remainder of **models,** one for each explanatory variable that is added to the model. In each section, a do loop is used to run all possible models at the corresponding stage. The number of cycles in the do loop is determined by the number of explanatory variables which you supply. An array, **vaname**, containing the names of all possible explanatory variables and the response variable that are fitted also has to be specified by you. The dimension of this array should be the same as the number of cycles in the do loop. In the example code below, there are 12 explanatory variables in all. The code for the first section of **models** is shown below:

```
%macro models(dep);
  %adjdata(&dep);
  %do j=1 %to 12;

 /************Run 1-var models***********************************/
   data mod1;
       array  vaname{12} $ ('aircond' 'setup' 'site' 'titled'
                     'purchase' 'secured' 'bedrooms' 'dlregn'
                       'sections' 'lgrp' 'wgrp' 'prgrp');
      k=&j*1;
       indep&j=vaname[k];

       call symput ("indep&j",indep&j);
       %put indep&j ne "&dep";
       if indep&j ^= "&dep" then
             call execute('%mod1var('|| "&j,&dep,&&indep&j" ||')');
   run;
  %end;
       proc transpose data=aicall (keep=AIC) out=aicmin ;
       run;
       data aicmin2 (keep=minaic);
             set aicmin;

             array aic{12} col1-col12;
             minaic=min(of aic[*]);
       run;
       data selectmod1;
             if _n_=1  then set aicmin2 ;
             set aicall;
             if minaic=AIC then output;
       run;
       proc datasets library=work;
             delete calcaic1-calcaic12;
       run; quit;
```

```
(END FIRST SECTION OF MODELS)


%macro mod1var(m,dep, indep1);
       proc catmod data=imp.combos3;
              weight count;
              model &dep = &indep1
                            / noprofile nodesign epsilon=0.01;
              ods output  datasummary=datasum
                          maxlikelihood=maxli
                          anova=anova1;
       run;
       quit;
       /*Get # of parameters for AIC*/
       data parms (keep=dftot);
         set anova1 (keep=model source df);
         by model;
         if source ^= "Likelihood Ratio" then dftot+df;
         if last.model=1 then output;
      run;
       /*Get value of -2logL for model*/
       data logL (keep=loglikelihood);
              set maxLi;
              by model;
              if last.model=1 then output;
       run;
       /*get number of response levels -1 = k */
       data resplevel (keep=k);
              set datasum;
              if Label2="Response Levels" then do;
                     k=nvalue2-1; output;
              end;
       run;
       /*calc total available parameters =s*/
       data plevel (keep= k s);
              merge resplevel parms;
           S=dftot;
       run;
       data calcAIC&m ;
              length model $7 indep1 $9 depvar $9;
              merge plevel logL;
              AIC=loglikelihood+2*(k+s);
              depvar="&dep";
              indep1="&indep1";
              model="model&m";
       run;
       %if &m=1 %then %do;
       data AICall;
              set calcAIC&m;
       run;   %end;
       %else %do;
              proc append base=AICall data=calcAIC&m force;
              run;
       %end;
%mend mod1var;
```

In the above code, note the CALL EXECUTE statement that calls the first macro **mod1var**. All possible one-variable models are fit using PROC CATMOD and their AICs are calculated in **mod1var**. The PROC CATMOD is run with the model statement options of NODESIGN, NOPROFILE and EPSILON=0.01. The first two options simply suppress some of the default output. The third option changes the default convergence criterion for maximum likelihood estimation of the parameters from epsilon=1e-8 to epsilon=0.01. The number 0.01 represents the proportional change in the log likelihood. By not requiring such stringent convergence criterion, the program run time is decreased significantly with very little change to the results of the model fitting.

PROC CATMOD does not produce a value for the AIC. In order to obtain the AIC, the Output Delivery System (ODS) is used to find the values of –2log*L*, *k* and *s*. Three of the ODS tables are required: *datasummary*, *maxlikelihood*, and *ANOVA*. PROC CATMOD displays the number of response levels, *k*, in the *datasummary* table under the heading “Response Levels.” The *ANOVA* table contains the degrees of freedom which equals the number of parameters(*s*). The *maxlikelihood* table contains the value of –2log*L*. Once extracted from the three ODS tables, the values of –2log*L*, *s*, and *k* are used to determine the AIC for the model.

The AIC for each model is appended into a single data file *AICall*. Once all models have been run, the macro **models** identifies the model with the minimum AIC. The explanatory variable from this model is saved to the data file *selectmod1*, which is used by the next section of **models** that sets up all two variable models. The first explanatory variable is the variable selected from the first section of model fitting. The second variable is one of the variables being tested as an explanatory variable and will vary for each model being fitted. The interaction between the two explanatory variables is also included in the model. The model is specified in the code as shown below.

```
model &dep = &indep1 | &indep2 / noprofile nodesign epsilon=0.01.
```

The bar separating the explanatory variables in the MODEL statement produces the interaction term. The automated model process excludes fitting the response variable as an explanatory variables. The process also does not fit a model if the next explanatory variable to be tested has already been fixed in the model by an earlier stage of model fitting.

After all two variable models have been run and the minimum AIC model identified, the program continues in a similar manner to fit three variable, four variable and finally five variable models. Each subsequent modeling step consists of the previously chosen combination of explanatory variables and a single new variable chosen from the remaining variables, plus all two-variable interaction terms. The model statements are shown below.

```
model &dep = &indep1 | &indep2 | &indep3 @2
model &dep = &indep1 | &indep2 | &indep3 | &indep4 @2
model &dep = &indep1 | &indep2 | &indep3 | &indep4 | &indep5 @2
```

In each case above the **“@2**” specifies that only two-variable interaction terms should be included in each model. The final result of the macro **models** is the five variable model with the lowest AIC. The remainder of the code for **models** and **mod1var-mod5var** is found in the appendix at the end of this paper.

APPLICATION

The code described has been used in a research project to apply the FMI procedure to impute missing data in the MHS. This survey is conducted each month by the Census Bureau to track a sample of shipments of Housing and Urban Development (HUD) inspected manufactured homes from manufacturer to dealer to buyer, until they are placed or otherwise taken off the market. If the home is placed for residential use the MHS collects detailed information, including dealer region, length, width, number of bedrooms, presence of central air-conditioning, type of location site, set-up, method for securing it in place, title and purchase information, including price if applicable. When any of this information cannot be obtained from the survey respondent, the information must be imputed so that each record is complete. Missing information is obtained by matching records with missing information to similar records (donors) that have the needed data items. Matches between records are based on the set of matching variables determined by modeling the missing variables. The fact that most of the MHS variables are categorical makes it an ideal candidate for applying this code.

The first step is to find the quintile distribution of each of the continuous variables: length, width, and price. This was done using PROC UNIVARIATE as shown below.

```
proc univariate data=imp2.resid;
  var length width price;
  output out=pctiles pctlpts=20 40 60 80 100 pctlpre= l_ w_ p_
               pctlname=p20 p40 p60 p80 p100;
run;
```

The input data set *imp2.resid* is a file containing only completely reported data. The output data file *pctiles* contains one record with each variable’s quintiles. This data file is then merged with the *imp2.resid* data file so that the quintiles appear on every data record.

Next, an *n*-way contingency table with data adjusted for sampling zeros. From this survey, we had a total of twelve variables that were available for matching (Table 1).

Table 1: MHS survey variables

AIRCOND	SETUP	SITE	TITLED
PURCHASE	SECURED	BEDROOMS	DLREGN
SECTIONS	LGRP	WGRP	PRGRP

The list of available variables must be included in the declaration of the array **vaname** in the main macro **models**. The array is declared in five different places in the macro **models**. When fitting models using a different set of data, this statement needs to be changed in each of the five locations.

```
array  vaname{12} $ ('aircond' 'setup' 'site' 'titled' 'purchase'
     'secured' 'bedrooms''dlregn' 'sections' 'lgrp' 'wgrp' 'prgrp').
```

In order to determine the best set of matching variables to use when air-conditioning (**aircond**) is missing, the macro **models** is run using the call " **%models(aircond)**." After **models** has called all the macros, there are four data files (*selectmod1- selectmod4*) which show the progression of the forward selection and a final data file, named after the response variable, containing the final selected model. The models chosen at each stage for fitting **aircond** are shown in Table 2.

Table 2: Results of Forward Selection for Aircond

Data File	Explanatory Variable Added	k-1	s	-2logL	AIC
Selectmod1	DLREGN	1	4	14925.2	14935.2
Selectmod2	PRGRP	1	20	14684.3	14726.3
Selectmod3	SETUP	1	44	14457.7	14547.7
Selectmod4	SITE	1	66	14118.1	14252.1
Aircond	TITLED	1	92	14023.4	14209.4

Now that the explanatory variables are found, the FMI procedure can determine the matching variables by rank. These explanatory variables will be used to match a donor record to a record missing the value for **aircond**. They are ranked by the order in which they are added to the final model. Looking at Table 2, **dlregn** (dealer region) is the most important matching variable and **titled** is least important. If a match cannot be made on all five variables, **titled** will be dropped and a match will be attempted again.

CONCLUSION

The concept behind the FMI procedure is to create an automated system in SAS that will find and rank matching variables that can be used to locate donors for missing data records using hot-deck imputation. The matching variables will be the most predictive explanatory variables derived from fitting regression models. FMI depends on an automated model selection method such as forward selection to determine an importance ranking based on which explanatory variables are kept in the model and the order in which they are placed. Much of the missing data is categorical with more than two possible values that do not exhibit any type of order.

It appears that only SAS PROC CATMOD can be used to fit models for predicting the values of response variables that define the above situation. However, PROC CATMOD does not presently have the capability to perform forward selection. In order to successfully create the FMI procedure in SAS, program code has to be written that will perform the forward selection method. This paper has described work done to accomplish this goal.

As SAS Institute Inc. continues to upgrade their software, there may come a time when statistical model fitting procedures such as PROC CATMOD have the capability of performing automated model selection methods like forward selection. Until then, SAS macros can be written that will perform this procedure.

REFERENCES

Williams, Todd R. (2001). "Flexible Matching Imputation: Combining Hot-Deck Imputation with Model-Based Methodology,"*2001 Proceedings of the American Statistical Association, Section on Survey Research Methods [CD-ROM]*

Kegan, Bonnie E. and Williams, Todd R. (2002). "Flexible Matching Imputation in the Manufactured Homes Survey," *2002 Proceedings of the American Statistical Association, Section on Survey Research Methods [CD-ROM].*

ACKNOWLEDGEMENTS

The authors would like to thank Lynn Weidman, U.S. Census Bureau, for his contributions to the development of the FMI methodology and his statistical expertise.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author(s):

Bonnie Kegan
U.S. Census Bureau
4700 Silver Hill Rd
Washington D.C. 20233
301-763-7639
bonnie.e.kegan@census.gov

Todd Williams
Social Security Administration
6401 Security Blvd. 4700 ME
Baltimore, MD 21235
410-594-2393
Todd.Williams@ssa.gov

APPENDIX

CODE FOR MACRO MODELS

```
%macro models(dep);
  %adjdata(&dep);
  %do j=1 %to 12;
 /**********************Run 1-var models******************************/

{SEE CODE SHOWN IN BODY OF PAPER}

/***********************Run 2-var models******************************/
%do j=1 %to 12;
    data mod2;
       array  vaname{12} $ ('aircond' 'setup' 'site' 'titled' ‘purchase'
      'secured' 'bedrooms' 'dlregn' 'sections' 'lgrp' 'wgrp' 'prgrp');
        set selectmod1;
             call symput ('indep1', indep1); /*get 1st selected var*/
             /*get # levels of 2nd variable*/
             k=&j*1;
             indep2&j=vaname[k];
             call symput ("indep2&j",indep2&j);
       if indep2&j ^= "&dep" and indep2&j ^= "&indep1"  then
       call execute('%mod2var('|| "&j,&dep,&indep1, &&indep2&j" ||')');
    run;
  %end;
       proc transpose data=aicall2 (keep=AIC) out=aicmin ;
       run;
       data aicmin2 (keep=minaic);
             set aicmin;

             array aic{12} col1-col12;
             minaic=min(of aic[*]);
       run;
       data selectmod2;
             if _n_=1  then set aicmin2 ;
             set aicall2;
             if minaic=AIC then output;
       run;
       proc datasets library=work;
             delete  /*aicall aicall2*/ calcaic1-calcaic12;
       run;
       quit;
/*******************Run 3-var models********************************/
             %do j=1 %to 12;
  data mod3;
       array  vaname{12} $ ('aircond' 'setup' 'site' 'titled' 'purchase'
       'secured' 'bedrooms' 'dlregn' 'sections' 'lgrp' 'wgrp' 'prgrp');
        set selectmod2;
       call sympput ('indep1', indep1);   /*get 1st selected variable*/
       call sympput ('indep2', indep2);   /*get 2nd selected variable*/
       k=&j*1;   /*get # levels of 3rd variable*/
       indep3&j=vaname[k];
      call sympput ("indep3&j",indep3&j);
if indep3&j ^= "&dep" and indep3&j ^= "&indep1" and
indep3&j ^= "&indep2"  then
      call execute('%mod3var('|| "&j,&dep,&indep1,&indep2,&&indep3&j"
                    ||')');
  run;
%end;
       proc transpose data=aicall3 (keep=AIC) out=aicmin ;
       run;
       data aicmin2 (keep=minaic);
              set aicmin;

              array aic{12} col1-col12;
              minaic=min(of aic[*]);
       run;
       data selectmod3;
              if _n_=1  then set aicmin2 ;
              set aicall3;
              if minaic=AIC then output;
       run;
proc datasets library=work;
       delete  /*aicall aicall2*/ calcaic1-calcaic12;
       run;
       quit;
/***********************Run 4-var models******************************/
%do j=1 %to 12;
  data mod4;
   array  vaname{12} $ ('aircond' 'setup' 'site' 'titled' 'purchase'
    'secured' 'bedrooms' 'dlregn' 'sections' 'lgrp' 'wgrp' 'prgrp');
   set selectmod3;
   call sympput ('indep1', indep1);   /*get 1st selected variable*/
   call sympput ('indep2', indep2);   /*get 2nd selected variable*/
   call sympput ('indep3', indep3);   /*get 3rd selected variable*/
   k=&j*1;   /*get # levels of 4th variable*/
   indep4&j=vaname[k];
   call sympput ("indep4&j",indep4&j);
   if indep4&j ^= "&dep" and indep4&j ^= "&indep1" and indep4&j ^=
   "&indep2" and indep4&j ^= "&indep3" then
   call execute('%mod4var('|| "&j,&dep,&indep1, &indep2, &indep3,
               &&indep4&j" ||')');
  run;
%end;
proc transpose data=aicall4 (keep=AIC) out=aicmin ;
run;

data aicmin2 (keep=minaic);
       set aicmin;
       array aic{12} col1-col12;
       minaic=min(of aic[*]);
run;
data selectmod4;
       if _n_=1  then set aicmin2 ;
       set aicall4;
       if minaic=AIC then output;
run;
proc datasets library=work;
       delete  /*aicall aicall2*/ calcaic1-calcaic12;
run;
quit;
/****************Run 5-var models*********************************/
%do j=1 %to 12;
  data mod5;
       array  vaname{12} $ ('aircond' 'setup' 'site' 'titled' 'purchase'
       'secured' 'bedrooms' 'dlregn' 'sections' 'lgrp' 'wgrp' 'prgrp');
        set selectmod4;
       call symput ('indep1', indep1);    /*get 1st selected variable*/
       call symput ('indep2', indep2);    /*get 2nd selected variable*/
       call symput ('indep3', indep3);    /*get 3rd selected variable*/
       call symput ('indep4', indep4);    /*get 4th selected variable*/

       /*get # levels of 5th variable*/
       k=&j*1;
       indep5&j=vaname[k];
       call symput ("indep5&j",indep5&j);

     if indep5&j ^= "&dep" and indep5&j ^= "&indep1" and indep5&j ^=
    "&indep2" and indep5&j ^= "&indep3" and indep5&j ^= "&indep4" then
    call execute('%mod5var('|| "&j,&dep,&indep1, &indep2, &indep3,
                &indep4, &&indep5&j" ||')');
  run;
%end;
proc transpose data=aicall5 (keep=AIC) out=aicmin ;
run;
data aicmin2 (keep=minaic);
       set aicmin;
       array aic{12} col1-col12;
              minaic=min(of aic[*]);
run;
data selectmod5;
       if _n_=1  then set aicmin2 ;
       set aicall5;
       if minaic=AIC then output;
run;
proc datasets library=work;
       change selectmod5=&dep;
       delete aicall aicall2-aicall5 calcaic1-calcaic12;
run;
quit;
%mend models;
```

CODE FOR MACROS MOD1VAR-MOD5VAR

```
%macro mod1var(m,dep, indep1);

{SEE CODE SHOWN IN BODY OF PAPER}

%mend mod1var;

%macro mod2var(m,dep, indep1, indep2);
proc catmod data=imp.combos3;
       weight count;
       model &dep = &indep1 | &indep2 /noprofile nodesign epsilon=0.01 ;
       ods output  datasummary=datasum maxlikelihood=maxli anova=anova1;
run;
quit;
       data parms (keep=dftot); /*Get # of parameters for AIC*/
              set anova1 (keep=model source df);
              by model;
              if source ^= "Likelihood Ratio" then dftot+df;
              if last.model=1 then output;
       run;
       data logL (keep=loglikelihood); /*Get value of -2logL for model*/
              set maxLi;
              by model;
              if last.model=1 then output;
       run;
       data resplevel (keep=k);  /*get number of response levels -1 = k */
              set datasum;
              if Label2="Response Levels" then do;
                     k=nvalue2-1; output;
```

Repeated Code

```
              end;
       run;
       data plevel (keep= k s); /*calc total available parameters =s*/
              merge resplevel parms;
        S=dftot;
       run;
data calcAIC&m ;
       length model $7 indep1 $9 indep2 $9 depvar $9;
       merge plevel logL;
       AIC=loglikelihood+2*(k+s);
       depvar="&dep";
       indep1="&indep1";
       indep2="&indep2";
       model="model&m";
run;
%if &m=1 %then %do;
       data AICall2;
              set calcAIC&m;
       run;
%end;
%else %do;
       proc append base=AICall2 data=calcAIC&m force;
       run;
%end;
%mend mod2var;

%macro mod3var(m,dep, indep1, indep2, indep3);
PROC CATMOD data=imp.combos3;
       weight count;
       model &dep = &indep1 | &indep2 | &indep3 @2 / noprofile nodesign
                   epsilon=0.01 ;
       ods output  datasummary=datasum maxlikelihood=maxli anova=anova1;
run;
quit;

{DUPLICATE CODE AS SHOWN IN modvar2}

data calcAIC&m ;
       length model $7 indep1 $9 indep2 $9 indep3 $9 depvar $9;
       merge plevel logL;
       AIC=loglikelihood+2*(k+s);
       depvar="&dep";
       indep1="&indep1";
       indep2="&indep2";
       indep3="&indep3";
       model="model&m";
run;
%if &m=1 %then %do;
       data AICall3;
              set calcAIC&m;
       run;
%end;
%else %do;
       proc append base=AICall3 data=calcAIC&m force;
       run;
%end;
%mend mod3var;

%macro mod4var(m,dep, indep1, indep2, indep3, indep4);
PROC CATMOD data=imp.combos3;
       weight count;
       model &dep = &indep1 | &indep2 | &indep3 | &indep4 @2 / noprofile
                 nodesign epsilon=0.01;
       ods output  datasummary=datasum maxlikelihood=maxli anova=anova1;
run;
quit;
```

```
{DUPLICATE CODE AS SHOWN IN modvar2}

data calcAIC&m ;
   length model $7 indep1 $9 indep2 $9 indep3 $9 indep4 $9 depvar $9;
merge plevel logL;
      AIC=loglikelihood+2*(k+s);
       depvar="&dep";
       indep1="&indep1";
       indep2="&indep2";
       indep3="&indep3";
       indep4="&indep4";
       model="model&m";
run;
%if &m=1 %then %do;
data AICall4;
       set calcAIC&m;
run;
%end;
%else %do;
proc append base=AICall4 data=calcAIC&m force;
run;
%end;
%mend mod4var;

%macro mod5var(m,dep,indep1, indep2, indep3, indep4, indep5);
proc catmod data=imp.combos3;
       weight count;
       model &dep = &indep1 | &indep2 | &indep3 | &indep4 | &indep5 @2 /
      noprofile nodesign epsilon=0.01;
       ods output  datasummary=datasum maxlikelihood=maxli anova=anova1;
run;
quit;

{DUPLICATE CODE AS SHOWN IN modvar2}

data calcAIC&m ;
      length model $7 indep1 $9 indep2 $9 indep3 $9 indep4 $9 indep5 $9
             depvar $9;
       merge plevel logL;
       AIC=loglikelihood+2*(k+s);
       depvar="&dep";
       indep1="&indep1";
       indep2="&indep2";
       indep3="&indep3";
       indep4="&indep4";
       indep5="&indep5";
       model="model&m";
run;
%if &m=1 %then %do;
       data AICall5;
             set calcAIC&m;
       run;
%end;
%else %do;
             proc append base=AICall5 data=calcAIC&m force;
             run;
%end;
%mend mod5var;
```