Reuse, Don't Reinvent: Extending Model Selection Using Recursive Macro  
Anca M Tilea, University of Michigan, Ann Arbor, MI  
Philip L Francis III, Eastern Michigan University, Ypsilanti, MI

ABSTRACT
The existing SAS® macro: %MODEL_SELECT - used recursively to select the “best” model based on the R² selection method - is performing great for specific data and specific variables. The steps of the macro %MODEL_SELECT are as follows: perform a BEST SUBSETS selection based on R² to get the list of candidate models, calculate the estimates and associated p-values, eliminate the covariates that are never significant, and repeat. This macro does not allow for multilevel variables to be considered for model selection, it does not allow for hierarchical modeling, and it only considers one model selection method: R². This paper aims to enhance existing SAS® methods, specifically, by allowing various model selection methods (e.g., adjusted-R², Mallows's Cp, etc.) and the inclusion of categorical variables with multiple levels. The added capabilities will allow the macro to still be user friendly, yet be more robust. Several PROCEDURES, CALL SYMPUT, SCAN, DO-LOOP, IF-THEN-ELSE statements and functions, and various Output Delivery System (ODS) statements are used in expanding the %MODEL_SELECT macro. This paper is intended for the intermediate SAS® user, with intermediate statistical skills and SAS® 9.1 or above.

INTRODUCTION
The existing SAS® macro: %MODEL_SELECT - used recursively to select the “best” model based on the R² selection method - is performing great for specific data and specific variables. For example, if the goal is to find predictors of a continuous outcome (thus using PROC REG), the existing macro performs very well and provides comprehensive information on various candidate models. However, if the outcome is defined as binary categorical – thus requiring PROC LOGISTIC, the existing macro, as is, will not be a good tool. The selection process for PROC LOGISTIC would entail a similar algorithm to the one for PROC REG, and given that the existing macro was designed for eventual reuse and/or extension, we set out to extend the %MODEL_SELECT macro and make it more robust. Most of the code was adapted quite easily to meet the specifications of PROC LOGISTIC, and eventually, more procedures.

In the design of the outer wrapper, several factors had to be considered. The first factor was the outcome type: numeric versus character. If the outcome is numeric, PROC REG is called; if the outcome is categorical, PROC LOGISTIC is called. The second factor was the type of the covariates: if the list provided by the user contains categorical covariates, then for each categorical variable with n levels this enhanced macro will automatically create n-1 dummy variables; if the list provided by the user contains only continuous covariates, then the list provided is used as-is in the model.

SAS® CODE
The additional code logic that extends the %REGRESSION_MODEL_SELECT macro is described.
The first step of the macro is to assess the outcome type. The type of a given variable is easily extracted from the SAS® table that PROC CONTENTS could output. However, there is one problem: there exists the possibility that the variable name that the user has entered (the outcome) does not syntactically match the name of the variable as it is found (i.e., spelled) in the data set (i.e., typographical difference). For example, below is an excerpt from a PROC CONTENTS data output.

If the user enters outcome = LAB_8, everything will work as planned, but if the user enters outcome = lab_8, there will be no way to match a value in the NAME column with a value in the TYPE column.

Thus, instead of placing a restriction on the user to carefully enter the outcome as it really is in the data set, we developed a small macro, %ISOLATE_OUTCOME that “matches” (PROC regardless of the letter case) what the user entered with how it is spelled in the data set. Below the details of the %ISOLATE_MACRO follow:

```sas
%MACRO ISOLATE_OUTCOME();
%LET REAL_VAR = ;
PROC SQL;
   SELECT NAME INTO: TABLE_VARS SEPARATED BY " "
   FROM DICTIONARY.COLUMNS
   WHERE LIBNAME = "%UPCASE(%SCAN(&DSNAME.,1, "."))" &
   MEMNAME = "%UPCASE(%SCAN(&DSNAME.,-1, "."))";
QUIT;
%LET NUM_OF_VARS = %EVAL(%SYSFUNC(COUNTW(&TABLE_VARS.))); DATA TEMP;
   LENGTH ONE_VAR $200;
   %DO I = 1 %TO &NUM_OF_VARS.;
      ONE_VAR = "%SCAN(&TABLE_VARS., &I., ")";
      OUTPUT;
   %END;
%RUN;
%DATA _NULL_;
   SET TEMP;
   IF PRXMATCH("/^&OUTCOME.[s]*$/i", ONE_VAR) THEN CALL SYMPUT("REAL_VAR", ONE_VAR);
%MEND ISOLATE_OUTCOME;
```

Another component of the proposed macro is to create DUMMY variables, if categorical variables are included in the list of covariates. This could be accomplished easily via PROC FREQ followed by manipulation of data sets outputted using ODS statements. However, there is one problematic scenario: a formatted categorical variable. For example, there is a variable Age with four groups (“1”, “2”, “3”, and “4”) and the following format:

```sas
PROC FORMAT;
   VALUE $ AGE_CAT "1" = "Age <= 55"
         "2" = "Age (55, 65"
         "3" = "Age (65, 75"
         "4" = "Age > 75"
;
```

Running PROC FREQ on this variable would result in the following output, which will be a problem when trying to create DUMMY variables based on the listed categories. For example, one way to create DUMMY variables is:

```sas
... DUMMY_AGE_1 = (AGE_CATEG = "Age <= 55");
...```

However, that would not work as intended! The “real” value of AGE_CATEG is actually 1, thus we need to:

1. create a copy of the data set provided
2. remove the formats from the copy (i.e., the original data is left intact)
3. use PROC FREQ to get the categories/levels for all categorical variables
create the DUMMY variables for each categorical variable in the copy data set
update the COVARIATES list with the DUMMY variables

Below is an explanation of key SAS® code that accomplishes the aforementioned points.

```sas
%MACRO COVARIATE_LEVELS();
  PROC SQL;
  CREATE TABLE TEMP_DATA (COMPRESS = YES) AS
  SELECT * FROM &DSNAME.;

  SELECT NAME, COUNT(NAME) INTO: CHAR_VARS SEPARATED BY " ",
  :CHAR_CNT
  FROM DICTIONARY.COLUMNS
  WHERE LIBNAME = "%UPCASE(%SCAN(&DSNAME.,1, ",")")" &
  MEMNAME = "%UPCASE(%SCAN(&DSNAME.,-1, ",")")" &
  TYPE = "char";

  SELECT NAME INTO: CHAR_FORMAT_VARS SEPARATED BY " 
  FROM DICTIONARY.COLUMNS
  WHERE LIBNAME = "%UPCASE(%SCAN(&DSNAME.,1, ",")")" &
  MEMNAME = "%UPCASE(%SCAN(&DSNAME.,-1, ",")")" &
  TYPE = "char" & FORMAT NE " ";

  ALTER TABLE TEMP_DATA
  MODIFY &CHAR_FORMAT_VARS FORMAT = $75.;

  SELECT NAME INTO: TABLE_VARS SEPARATED BY " 
  FROM DICTIONARY.COLUMNS
  WHERE LIBNAME = "%UPCASE(%SCAN(&DSNAME.,1, ",")")" &
  MEMNAME = "%UPCASE(%SCAN(&DSNAME.,-1, ",")")" ;
  QUIT;

  %DO I = 1 %TO &CHAR_CNT.;
    PROC FREQ DATA = TEMP_DATA NLEVELS;
    TABLES %SCAN("&CHAR_VARS.", &I., " ");
    ODS OUTPUT NLEVELS = MY_LEVELS_&I.;
    ODS OUTPUT ONEWAYFREQS = MY_FREQ_&I.(KEEP = F_:);
    RUN;
  %END;

  PROC SQL;
  SELECT NAME INTO: FREQ_VARS SEPARATED BY "," 
  FROM DICTIONARY.COLUMNS
  WHERE LIBNAME = "WORK" & MEMNAME = "MY_FREQ_&I.";

  SELECT &FREQ_VARS. INTO: _LEVELS_&I. SEPARATED BY "**" 
  FROM MY_FREQ_&I.;
  SELECT NLEVELS INTO: NUM_LEVELS_&I.
  FROM MY_LEVELS_&I.;
  WHERE TABLEVAR = "%SCAN(&CHAR_VARS., &I., " ");
  QUIT;

  ... DATA FINAL_DATA;
  SET TEMP_DATA;

  %LET VAR = %SCAN(&CHAR_VARS., &I., " ");
  %DO J = 1 %TO &NUM_LEVELS_&I.- 1 ;
    %LET CURRENT_LEVEL= %SCAN("_LEVELS_&I.,&J.,"**");
    %PUT &CURRENT_LEVEL .
    %LET VALUE = %SCAN( &CURRENT_LEVEL., &J., "**");
    DUMMY_&VAR._&J. = (&SCAN(&CHAR_VARS., &I., " ") = 
                     &CURRENT_LEVEL.);
    %END;
  RUN;
  %END;

  ...%END;
```

- create a copy of the data set
- select the character variables and the number (how many there are) into two macro variables CHAR_VARS and CHAR_CNT
- separately select the variables that have a format attached to them in a macro variable called CHAR_FORMAT_VARS
- modify the copied data set (from above) and remove the formats
- select all variables existing in the data set in a macro variable called TABLE_VARS for later use
- do-loop through all categorical variables
- run PROC FREQ
- save the levels information in a SAS® data set
- select the levels of each categorical variable into a macro variable (_LEVELS_&I.) separated by "**", and store the corresponding number of levels into a macro variable (_NUM_LEVELS_&I.)
- create the DUMMY variables and add to the copy data set
- using PROC SQL and a couple of data steps, update the covariates list by replacing the categorical variables with the newly created DUMMY variables
Once the outcome has been assessed and the covariates list has been scanned for categorical variables, the %REGRESSION_MODEL_SELECT macro is ready to be employed. This macro uses several logical IF-THEN-ELSE statements, as it was depicted in the flowchart.

The %REGRESSION_MODEL_SELECT macro takes five parameters:

- three are required:
  - DATA SET NAME
  - OUTCOME of interest
  - COVARIATES list
- two are optional:
  - SELECTION METHOD
  - NUM_SUBSETS

If the selection method is not provided, the macro will use the R² – for PROC REG, and the SCORE statistic for PROC LOGISTIC. The top 5 models will be included if BEST option is not provided.

```
%MACRO REGRESSION_MODEL_SELECT(DSNAME = , OUTCOME = , COVARIATES = , SELECTION_METHOD = , NUM_SUBSETS = );
    ...
    %ISOLATE_OUTCOME();
    %COVARIATE_LEVELS();
    %LET DSNAMEN = WORK.FINAL_DATA;
    %IF &LEVELS. = 2 %THEN %DO;
        %INCLUDE "&PATH\MODEL_SELECT.SAS";
        %IF &NUM_SUBSETS. NE &THEN %DO;
            %MODEL_SELECT(MY_DATA = &DSNAME., outcome = &OUTCOME., covariates = &covariates, best = &NUM_SUBSETS.);
        %END;
    %ELSE %DO;
        %MODEL_SELECT(MY_DATA = &DSNAME., outcome = &OUTCOME., covariates = &covariates, best = 5);
    %END;
    %END;
    %MEND REGRESSION_MODEL_SELECT;
```

### SAMPLE OUTPUT

Sample results from the %REGRESSION_MODEL_SELECT macro are shown in Figure 2 with the following key components:

1. Original Sample Size: the original size of the data set, with missing data included
2. Complete-case N: sample size for the BEST SUBSETS models (complete-case data including all covariates of interest)
3. R-Square: Model-based R² for the BEST SUBSETS models
4. Number in Model: total number of covariates in each model
5. Variables in Model: list of the covariates used in each model
6. Variables Significant in the Model: significant covariates from each model
7. Potential N: complete-case sample size for a model using a subset of variables (≥ Complete-case N)
8. Potential R-square: Model-based R² for models based on Potential N.
The rest of the columns represent the individual estimates and p-values

<table>
<thead>
<tr>
<th>Model Index</th>
<th>Original Sample Size</th>
<th>Complete Data Sample Size</th>
<th>R-Square</th>
<th>Model significant? (Yea or No)</th>
<th>Number in Model</th>
<th>Variables in Model</th>
<th>Variables Significant in This Model</th>
<th>Potential N</th>
<th>Potential R</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2141</td>
<td>1272</td>
<td>0.0024</td>
<td>Yea</td>
<td>1</td>
<td>LAB_4</td>
<td>LAB_4</td>
<td>1713</td>
<td>0.0001</td>
<td>0.005</td>
</tr>
<tr>
<td>2</td>
<td>2141</td>
<td>1272</td>
<td>0.0047</td>
<td>Yea</td>
<td>1</td>
<td>ETHNIC_SEX, 1</td>
<td>ETHNIC_SEX, 1</td>
<td>2133</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>3</td>
<td>2141</td>
<td>1272</td>
<td>0.0005</td>
<td>No</td>
<td>0</td>
<td>1_bni</td>
<td>1_bni</td>
<td>1581</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>4</td>
<td>2141</td>
<td>1272</td>
<td>0.0003</td>
<td>Yea</td>
<td>0</td>
<td>1_regn</td>
<td>1_regn</td>
<td>2133</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>5</td>
<td>2141</td>
<td>1272</td>
<td>0.0040</td>
<td>Yea</td>
<td>0</td>
<td>2_LAB_4, bni</td>
<td>LAB_4, bni</td>
<td>1272</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>6</td>
<td>2141</td>
<td>1272</td>
<td>0.0003</td>
<td>Yea</td>
<td>0</td>
<td>2_age, sex, LAB_4</td>
<td>LAB_4</td>
<td>1713</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>7</td>
<td>2141</td>
<td>1272</td>
<td>0.0003</td>
<td>Yea</td>
<td>0</td>
<td>2_ETHNIC_SEX, sex, LAB_4</td>
<td>LAB_4</td>
<td>1713</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

Figure 2. Sample output result

LIMITATIONS

In the presence of multilevel categorical outcome, the current macro does not implement PROC GENMOD – mostly due to its lack of model selection method(s). The authors pondered the possibility of creating (and running) ALL possible models for a given set of covariates, but realized the run-time complexity and the logic still needed to be thought through. Furthermore, the current macro does not handle time-varying data (i.e., PROC MIXED) or survival models. A shell has been developed to implement a baseline PROC PHREG, and it is envisioned that it will shortly be integrated in the current macro.

CONCLUSION

The updated macro makes the task of (manual) model selection much easier and time-efficient. The authors provided a tool that will aid the analyst in the rather tedious process of selecting a model. The code in its entirety (and all the macros) is available on the [http://www-personal.umich.edu/~atilea](http://www-personal.umich.edu/~atilea) website.

REFERENCES

2. Christopher J. Bost, MDRC To FREQ, Perchance To MEANS support.sas.com/resources/papers/proceedings11/093-2011.pdf

ACKNOWLEDGMENTS

I am very grateful to MWSUG 2013 for allowing me to share my work with other SAS users. I would also like to thank very much to Philip Francis for providing valuable programmatic feedback. I would also like to thank Flannery Campbell, Deanna Chyn, and Sarah Casino for their time and positive feedback.

CONTACT INFORMATION

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

Anca M Tilea
University of Michigan
1415 Washington Heights 3645A
Ann Arbor, MI, 48104
734.763.6611
734.763.4004
atilea@med.umich.edu
http://www-personal.umich.edu/~atilea/

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.