

Musical Macros: A Soothing Solution for a Recurrent Data Step Headache

Philip J. d'Almada, Rho, Inc., Chapel Hill, North Carolina

Abstract

The problem was to vertically concatenate an excessive number of data sets/tables, each with a varying number of sequentially named variables/columns that were to be horizontally concatenated into one variable/column in a final concatenated data set/table. The macro facility of The SAS System® was the tool of choice. Through SAS macro processing, the number of SAS statements in the source code was drastically reduced. SAS macro processing also allowed the elimination of repetitive portions of SAS code that would normally appear in the SAS log. The user was rewarded with a shortened SAS macro program that has a single rather than recurrent DATA step, the component statements of which are visually symbolic of a type of musical scale. Further, the user is mentally and visually relieved of the task of perusing a very lengthy SAS log.

Introduction

Problem

A series of reports were to be produced, each using data from a number of data sets. In one instance, one report required data from more than 20 SAS data sets. For a given report, the data sets were to be vertically concatenated. Certain text variables were also to be joined in a horizontal concatenation. The data sets were previously named and the variables of interest were already given numerically sequential names. The maximum number of these variables varied across data sets. The data sets were not given numerically sequential names.

Purpose

This dual concatenation procedure was to be repeated for every report while the names and number of the data sets for each report would vary. In this particular study, each report required its own source code. The challenge was to develop a solution to address the repetitive process of accessing and concatenating data sets and selecting and concatenating text variables, and to implement this solution for each report.

Method

Tools

The corporate standard approach to producing reports was the implementation of select procedures and data management techniques using the features of The SAS System. The source data were already stored as SAS data sets. Given these two factors,

therefore, it remained to use The SAS System to develop a solution.

Technique

Since each report was required to have its own exclusive source code, the approach was to develop a solution for one report and replicate that with minimum adjustment for implementation across all other reports. The development process comprised three steps. First, the basic code was developed to access one data set, concatenate the select text variables, and generate a new output data set. This code was replicated for the second data set for output into a second new data set, and so on. Thus, the pattern of the data step was established. The new data sets were given numerically sequential names. The second step was the development of code to concatenate the output data sets. The final step was to reduce the code by developing two macro programs. The first macro program was used to access the data, concatenate the finite number of select text variables, and generate the new output data set. This macro program was executed for each data set. The second macro program was used to concatenate the new output data sets. Finally, the report was generated with code that was external to either macro program yet specific to that report.

Implementation

The first of the two SAS macro programs that were developed is listed in Illustration 1. This macro program was defined by the required initial keyword, MACRO, and the required terminal keyword, MEND, both prefixed by the % symbol. Following the initial keyword is the name of the macro program, ODSET, that was used to call the macro program—see Illustration 2, below. After the macro program name, and enclosed in parentheses, is the string of parameters that allowed the generic form of the macro program to become report-specific. The first parameter is the macro variable that represented the input SAS data set name, ODNAM. The next parameter is TDSN, the macro variable that represented the output SAS data set name. The final four macro variables, CMSTR1-CMSTR4, represented the SAS text variable names that were to be concatenated. An earlier examination of the source data sets indicated that there were no more than four variables to concatenate. The six parameters are separated by commas. Between the initial and terminal keywords of the macro program, the repetitive SAS code was placed. It was in a

generic form where the parameters, each prefixed by an ampersand, (e.g., &ODNAM, &CMSTR3), were inserted in the body of the code at the locations where the data set name or variable name would normally appear in standard SAS code. Some SAS macro functions that occurred in the macro program resemble standard SAS code. However, the macro function keywords are preceded by the % symbol such as %UPCASE or %IF ... %THEN ...

```
%macro odset (odnam, tdsn, cmtstr1, cmtstr2,
cmtstr3, cmtstr4);
data &odnam;
  set sasorig.&odnam (keep=id1 id2 id3 &cmtstr1
&cmtstr2 &cmtstr3 &cmtstr4 );
run;
data &tdsn (drop=id1 id2 id3 &cmtstr1 &cmtstr2
&cmtstr3 &cmtstr4 );
  set &odnam;
  length id $32 dset $8 cmts $240;
  id = compress(id3)|| ' ' || put(id1, z4.)|| '-'
  || put(id2, z4.);
  dset = "%upcase(&odnam)";
  %if &cmtstr2 eq %then
    cmts = trim(&cmtstr1);
  %else %if &cmtstr3 eq %then
    cmts = trim(&cmtstr1) || ' ' || trim(&cmtstr2)
;
  %else %if &cmtstr4 eq %then
    cmts = trim(&cmtstr1) || ' ' || trim(&cmtstr2)
|| ' ' || trim(&cmtstr3) ;
  %else %if &cmtstr4 ne %then
    cmts = trim(&cmtstr1) || ' ' || trim(&cmtstr2)
|| ' ' || trim(&cmtstr3) || trim(&cmtstr4); ;
  if cmts ne '';
run;
%mend odset;
```

Illustration 1. The first macro program

Thus, each time the first macro program was called, the parameters were replaced with SAS data set and variable names as follows:

1. ODNAM, by the name of the input data set, e.g., DEMOGRAPH or MEDHX;
2. TDSN, by the name of the output data set, e.g., DS1 or DS2;
3. CMTSTR1, CMTSTR2, CMTSTR3, CMTSTR4, by the names of each text variable, or an empty field when there was no corresponding text variable in the parameter string, e.g., the series CMT1, CMT2, CMT3, CMT4 or the series CMT1, CMT2, , .

The program code for variable concatenation depended on the non-missingness of the respective text variables. Thus, if the second text variable specified in the parameter string were missing, then the concatenation would include only the first text variable. If the third text variable were missing, the concatenation would include the first and second text variables, and so on. To identify, within the output observation, which was the source data set

for a given concatenation, the macro function, %UPCASE, was applied to the input data set name represented by the macro variable, &ODNAM. Thus, the values of the variable DSET were the names of the input data sets and converted where necessary to upper case. To control for missingness in the first text variable, a truth statement was included at the end of the data step where the new concatenated variable was evaluated for missingness.

In order to call the macro program, the macro program name, prefixed by the % symbol, was specified with the parenthetical string of parameters for which values were supplied. That is, the actual SAS data set and variable names were specified in place of the representative macro variable names within the parentheses. Since the number of text variables fluctuated across data sets, empty spaces were appropriately preserved between commas when there were fewer than four text variables in a given data set. The stream of macro calls for the 23 input data sets is represented, in brief, below.

```
%odset (demograph, ds1, cmt1, , , );
%odset (medhx, ds2, cmt1, cmt2, cmt3, cmt4);
%odset (drugdose, ds3, cmt1, , , );
%odset (physexam, ds4, cmt1, cmt2, cmt3, );
.
.
%odset (interrupt, ds23,cmt1, cmt2, cmt3, );
```

Illustration 2. Repeated calling of the first macro program

The central feature of the second macro program was a macro %DO loop that, upon processing, listed a series of sequentially named SAS names—see Illustration 3. For every report, the total number of data sets was known and this maximum number was used in the second macro program as the maximum number of iterations of the %DO loop. The macro variable, DSNAM, represented the prefix, DS, used in naming the output data sets generated from implementing the first macro program. Appropriately calling this macro program would facilitate the concatenation of the data sets that were output from the repeated calling of the first macro.

```
%macro count (dsnam);
  %do seq = 1 %to 23 ;
    &dsnam&seq
  %end;
%mend;
```

Illustration 3. The second macro program

The effect of the %COUNT(DS) macro call as shown in the DATA step in Illustration 4, was to generate the series of SAS data set names for concatenation.

```
data comb (keep=id dset cmts);  
  set %count(ds);  
run;
```

Illustration 4. Calling the second macroprogram

Thus, once the macro processing was completed and all macro variables resolved, SAS would see the SET statement in the DATA step as

```
set ds1 ds2 ds3 ds4 ... ds23;
```

The outcome from implementing the first macro program a predetermined number of times and the second macro program once, was a single analysis data set from which the desired report was generated.

Summary

A predetermined but exceptionally large number of SAS data sets were each accessed by calling, once per data set, a SAS macro program that effected the concatenation of a number of selected text variables and generated an output SAS data set. A second macro program was called that concatenated the known number of output data sets into one final SAS analysis data set from which the final report was generated.

Conclusion

The macro facility of The SAS System was handy to overcome a DATA step headache, by repeating a relatively short piece of code, without having to replace key names—data set names and variable names—at each repetition of the code. This was done by a series of macro calls that utilized one line of code for each input data set, in which the report-specific data set names and variables names were specified. A visual inspection of the series of 23 macro calls yielded an aesthetically pleasing arrangement of lines of code with the appearance of a rhythmic movement. Regardless of the author's divergence but titular inspiration, the result of implementing SAS macro processing was the reduction in size of the SAS log since the fundamental code was written just once. What was repeated, then, in the SAS log, was not the whole DATA step but the single line of code that was the macro call for each data set.

Acknowledgements

The author appreciates the support of his employer, Rho, Inc., and his department Director, Steve Noga, who is also Conference Cochair for SESUG '03. To Heidi Markovitz, Section Cochair for Coders' Corner, much gratitude is extended for her invaluable editing recommendations. Finally, the author also deeply appreciates the support of his wife, Shelley, and children, Sammy and Abby.

SAS is a registered trademark or trademark of SAS Institute, Inc. in the USA and other countries.

® indicates USA registration.

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

Contact

Phil d'Almada
Rho, Inc.
100 Eastowne Dr.
Chapel Hill
NC 27514

Telephone: 919-929-6200

E-mail: pdalmada@rhoworld.com