Paper RF-02-2016

Utilizing PROC CONTENTS with Macro Programming to Summarize and Tabulate Copious Amounts of Data

Kathryn Schurr, M.S., Quest Diagnostics, Hudsonville, MI
Erica Goodrich, M.S., Brigham and Women's Hospital, Boston, MA

ABSTRACT

As more complex data becomes available, the challenges and time needed to explore this data increases. Tabulating and summarizing large amounts of varying data takes a large amount of time and effort to code univariate explorations. By using PROC CONTENTS on similarly formatted fields within a data source and implementing macro programming, a user can produce numerous frequency or means tables with minimal coding effort.

INTRODUCTION

There are many scenarios where multiple datasets are set up in similar ways. Instances like these help users understand the structure of data prior to opening each dataset, as well as understanding some of the contents of the data within each dataset of a related subject. Due to the common occurrence of these similar datasets, it is beneficial to find a way to quickly summarize all the data without having to code each dataset individually. Through the use of macro programming and utilization of PROC CONTENTS, creating multiple data summaries is simplified.

BACKGROUND

In one particular example, the datasets of interest contain information regarding the presence of risk factors pertaining to infant mortality. Each of the related 17 datasets contain variables corresponding to the dataset title. For instance, the dataset titled 'FamPlan' contains information on family planning that did or did not take place for the deceased infant. A few examples of this datasets' variables range from whether or not the pregnancy was planned to the different types of birth control that was available to the family. These variables are set up in such a way that the valid responses are dichotomous. They could contain a 'P' for the risk factor being 'Present' for this infant death or 'C' for the risk factor being 'Contributory.'

The above mentioned datasets have similar structures where the variables directly correspond with the title of the dataset. These datasets also have a de-identified Case_ID number that corresponds to an individual infant and field titled 'Other' that is used for taking notes in the form of a text string. Each infant should have a presence within each dataset. Having the infant matching between datasets makes this example easier, but is not a requirement. Due to confidentiality, no actual output will be shown.

MACRO PIECES

The first step of building the macro is to create the macro name and specify the variables the macro will need in order to be called. This macro is called 'DATAPREP' because we are preparing the data for summarization and completing exploratory analysis. This macro needs information on the datasets which will be included, which will be invoked using the DATA= option, and the variables which should be excluded in analysis, which are identified using the EXCLUDE_VARS= option. The beginning of the macro looks as follows:

```
%MACRO DATAPREP(DATA=, EXCLUDE_VARS=);
     %LET Dataset_Count=%SYSFUNC(COUNTW(&DATA));
     %DO i = 1 %TO &Dataset_Count.;
```

The %LET Dataset_Count statement is creating a macro variable that will be used for counting how many datasets are listed in the macro prompt using the COUNTW function. This method is valid for SAS® 9.1 or above. For this example, 17 datasets are included so this macro statement returns the number 17. The %DO line indicates how many iterations the macro will go through. This number corresponds to the number of datasets that the user will be referencing.

The next step is to complete a PROC CONTENTS on each of the datasets without printing the results to the output window. By utilizing the %SCAN function, the macro will cycle through each of the datasets that are called when the macro is invoked so that the user doesn't have to repeat calling the individual macro for each dataset. The output of the PROC CONTENTS procedure is named with the prefix of 'Contents_' for each of the relevant datasets.

```
PROC CONTENTS DATA = %SCAN(&DATA,&i) NOPRINT OUT =
     Contents_%SCAN(&DATA,&i);
RUN;
```

Once the PROC CONTENTS output has been collected, the next step is to evaluate each of the datasets and remove variables that are not of interest. This is where knowing the structure of the data comes in handy. In some cases, the user may want to exclude certain variables which do not make sense to view as a frequency table. In the example of the infant data, the variables for free form text, 'Other', or the variable 'Fetal__F__Infant__I_' do not contain information that is pertinent to evaluating the risk factors. Because of this, these variables will be excluded from further output. A specific list of variables can be listed inside the EXCLUDE_VARS macro variable by using quotations and commas for separation since they will be resolved inside an IN statement. This prompt will work best when using the %STR() function to allow multiple comma listings inside of a macro variable. Also, any variable that does not have a length of 1 is removed. By having a longer length, the variables are more than likely a text field that is not desirable for evaluating the presence of a risk factor (P vs. C). This line could be removed for analysis of different datasets, where a longer variable name may be of interest.

```
DATA Contents_%SCAN(&DATA,&i);
     SET Contents_%SCAN(&DATA,&i);
     IF Length NE 1 THEN DELETE;
     IF Name in (&EXCLUDE_VARS) THEN DELETE;
     Label = Name;
     KEEP Name VarNum Label;
RUN;
```

The %SCAN function again, cycles through the datasets and can also be used on the datasets that were created using a prefix because the end of the names are the same. Out of the PROC CONTENTS procedure, once the dataset has been restructured and restricted, the variables Name, VarNum, and Label are kept, all others are discarded. The next step is to create a GLOBAL macro variable that will be assigned a value with the subsequent SQL code.

```
%GLOBAL List_%SCAN(&DATA,&i);
```

The GLOBAL statement created an empty macro variable with the prefix of 'List_'. Again by using the %SCAN function, the dataset names that are already in use can be repurposed. The SQL code following creates a macro variable for each dataset which contains a list of distinct variables found within in each dataset. Each distinct variable name is separated by a space. This macro variable will be used for filling in the list of variable used in a PROC FREQ in future steps

```
PROC SQL NOPRINT;
     SELECT DISTINCT Name
     INTO :List_%SCAN(&DATA,&i) SEPARATED BY ' '
     FROM Contents_%SCAN(&DATA,&i)
     ORDER BY VarNum;
QUIT;
```

Now that a list of all of the variable names within each dataset has been created, another set of GLOBAL macro variables need to be created that specify the number of variables within each dataset. This is completed using the following steps. The somewhat redundant %LET statements are there to ensure there are no errors within SAS processing this macro. There have been instances where not having all the %LET statements resulted in errors. The multiple ampersands (&) allow SAS to call a macro variable within a macro variable.

```
%GLOBAL Count%SCAN(&DATA,&i);

%LET Count%SCAN(&DATA,&i) = &SQLOBS;

%LET countvar=Count%SCAN(&DATA,&i);
%LET count=&&&countvar;

%LET listvar = List_%SCAN(&DATA,&i);
%LET List = &&&listvar;
```

The last data manipulation step within the macro will be data specific. In the example infant data that is currently being used, some of the text values are either missing or erroneous for some of the variables within each dataset. The data step below allows the user to fix any of these issues and reformat the data to how they see fit. This data step codes the missing values as 'A' for Absent, as well as the other non-missing values as 'P' for Present. This data step cycles through each dataset and creates an array with the listing of variables so that all variables can be evaluated. The macro variable Count that was created above is now used to indicate the size of the array being referenced.

```
DATA %SCAN(&DATA,&i);
     SET %SCAN(&DATA,&i);
     ARRAY %SCAN(&DATA,&i) [&Count] &List;
     DO j = 1 TO &Count;
          IF %SCAN(&DATA,&i)(j) = '' THEN
               %SCAN(&DATA,&i)(j) = 'A';
          ELSE %SCAN(&DATA,&i)(j) = 'P';
     END;
     FORMAT &List $Issues.;
RUN;
```

By having the &Count and &List macro variables set up the way they are, the user can utilize them within each dataset being referenced without having to manually change each data step within the macro for each dataset being referenced. This allows the user insurmountable flexibility and the ease of one block of code versus many. The final step for this macro is creating the summary tables or exploratory analyses of interest. In this particular example, CHISQ values are evaluated for each variable within a dataset against the cause of death for the infants.

```
          PROC FREQ DATA = %SCAN(&DATA,&i);
               TABLES DeathCause*(&List) /CHISQ;
               TITLE "%SCAN(&DATA,&i) Issues";
          RUN;
```

Once the final block of code has been written for the summary tables, the original %DO loop within the macro will end and the macro will be completed with a %MEND statement. To invoke the macro simply call it with a list of all relevant datasets that the user would like to view.

```
     %END;
%MEND;
```

CALLING THE %DATAPREP MACRO

Only two pieces of information need to be addressed to use this macro. These two pieces of information are the names of the datasets of interest, and the names of the variables which are desired to be excluded. These have been addressed previously in the paper, but an example of the macro call can be seen below.

```
%DATAPREP(DATA = Social Culture Environment FamPlan FetalMed
Injuries MentalHealth Mother Payment Pediatric Preconception
Prenatal Service Substance Transitions Transport Violence,
EXCLUDE_VARS = %str('Fetal__F__Infant__I_','Other'));
```

CONCLUSION

Through the macro process described wherein, over a hundred frequency tables have been created and are ready for further exploration. This is just one way to get a quick review of a large number of variables in multiple datasets in a quick, flexible method. This paper highlights one way in which this method can be used to check for missing data. However, with a few simple changes, the bulk of this macro could be used to show frequencies of all variables of a desired method. The flexibility is dependent on the data at hand and what the end goals may be. This macro highlights how SAS can be leveraged to handle large amounts of output with minimal coding time on the user end.

CONTACT INFORMATION

Your comments and questions are much appreciated. Contact the authors at:

Kathryn Schurr, M.S.
Health Information Analyst
Quest Diagnostics
Hudsonville, MI 49426
Work Phone: 616.340.0045
Work Email: Kathryn.M.Schurr@QuestDiagnostics.com

Erica Goodrich, M.S.
Biostatistician
Brigham and Women's Hospital
75 Francis Street
Boston, MA 002115
Work Phone: 617.278.0316
Work Email: EGoodrich@BWH.Harvard.edu

APPENDIX – MACRO Code

```
/**************************************************************/
/* DATAPREP Macro                                             */
/**************************************************************/
/* This macro will evaluate a list of datasets, and run
 frequency tables on all variables existing- except for
 specific exclusions specified by user or have a variable
 length not equal to 1. */
**************************************************************/
/* MACRO PROMPTS:
DATA = Add a list of datasets of interest only separating by
space.

EXCLUDE_VARS = Include a list of variables that should be
excluded from a PROC FREQ analysis by including them inside
of a %str() and using quotes, separating by a comma.
Example:
%DATAPREP(DATA= ds1 ds2 ds3 ds4,
EXCLUDE_VARS = %str(subject_ID, text_field1, textfield2");*/
/**************************************************************/
PROC FORMAT;
     VALUE $Issues  'A' = 'Absent'
                    'P' = 'Present';
RUN;

%MACRO DATAPREP(DATA=, EXCLUDE_VARS=);
     %LET Dataset_Count=%SYSFUNC(COUNTW(&DATA));
     %DO i = 1 %TO &Dataset_Count.;

     PROC CONTENTS DATA = %SCAN(&DATA,&i) NOPRINT OUT =
     Contents_%SCAN(&DATA,&i);
     RUN;


     DATA Contents_%SCAN(&DATA,&i);
          SET Contents_%SCAN(&DATA,&i);
          /* Change or remove the following line if
          desired output changes*/
          IF Length NE 1 THEN DELETE;
          IF Name in (&EXCLUDE_VARS) THEN DELETE;
          Label = Name;
          KEEP Name VarNum Label;
     RUN;

     %GLOBAL List_%SCAN(&DATA,&i);

     PROC SQL NOPRINT;
          SELECT DISTINCT Name
          INTO :List_%SCAN(&DATA,&i) SEPARATED BY ' '
          FROM Contents_%SCAN(&DATA,&i)
          ORDER BY VarNum;
     QUIT;

     %GLOBAL Count%SCAN(&DATA,&i);

     %LET Count%SCAN(&DATA,&i) = &SQLOBS;

     %LET Countvar=Count%SCAN(&DATA,&i);
     %LET Count=&&&Countvar;

     %LET Listvar = List_%SCAN(&DATA,&i);
     %LET List = &&&Listvar;

     /*Data specific changes should be made in below data step*/
     DATA %SCAN(&DATA,&i);
          SET %SCAN(&DATA,&i);
          ARRAY %SCAN(&DATA,&i) [&Count] &List;
          DO j = 1 TO &Count;
               IF %SCAN(&DATA,&i)(j) = '' or
               %SCAN(&DATA,&i)(j) = 'U'
               THEN %SCAN(&DATA,&i)(j) = 'A';
               ELSE %SCAN(&DATA,&i)(j) = 'P';
          END;
          FORMAT &List $Issues.;
     RUN;
     /* Analysis output types can be changed below */
     PROC FREQ DATA = %SCAN(&DATA,&i);
          TABLES DeathCause*(&List) /CHISQ;
          TITLE "%SCAN(&DATA,&i) Issues";
     RUN;

     %END;
%MEND;
```

```
%DATAPREP(DATA = Social Culture Environment FamPlan FetalMed
Injuries MentalHealth Mother Payment Pediatric Preconception
Prenatal Service Substance Transitions Transport Violence,
EXCLUDE_VARS = %str('Fetal__F__Infant__I_','Other'));
```