Paper 84-27

Using the Contents of PROC CONTENTS to Perform Multiple Operations Across a SAS® Data Library

Subrahmanyam Pilli, Luai Alzoubi, Kent Nassen
Pfizer Global Reasearch and Development, Ann Arbor, MI

ABSTRACT

Have you ever wanted to process all data sets within a library but didn't want to use multiple data step statements that add cumbersome code to programs? What would you do when new data sets are added? With custom macros you would have to get the name of the data set, add the code to process the data (remembering to type the name correctly), and then validate this "new" code. What if you could do all these steps in one simple to use and understand macro and not have to add new code, not have to remember what a data set was named or worry about how many new data sets were being added? Things like subsetting data sets or performing data manipulation on any number of data sets become as easy as submitting code. You can even include certain data sets and exclude others. The ability to do any of the above is discussed in this paper.

INTRODUCTION

How do you perform multiple operations on all datasets in a SAS library when you don't want to explicitly name every dataset? One way is to use the information provided by PROC CONTENTS to dynamically create macro variables containing the dataset names and the total number of data sets, then loop over those macro variables while performing needed data steps, procs, etc. This method is useful in many contexts, from subsetting every data set, to merging one data set into all data sets, to adding or modifying variables or labels in each data set. This article describes how to set up and use this method to perform useful data processing tasks, and provides several examples.

CONCEPTS

The general method is shown in Listing 1. This program will copy data from library "copyfrom", create a few variables, and write the resulting data sets in library "copyto" using a simple data step.

The program is dominated by a macro, named LIBDATA, so that macro statements, such as %DO loops, can be used.

%LIBDATA begins with a PROC CONTENTS which creates an output data set containing the dataset names in the library. The output data set is sorted by memname and name to prepare for removing duplicate memnames, which occur because PROC CONTENTS will create an observation for every variable found in each dataset. We are only interested in the unique memnames from the library since we are operating at a dataset level, not at a variable level within the dataset. The duplicate memnames are removed using first/last processing in the next data step. This data step may also be used to remove the memnames of data sets that are not to be processed.

A DATA _NULL_ step is used to create a series of macro variables, DS1 through DSx (where x=total number of data sets/memnames). This methodology is used to allow referencing these macro variables in a DO loop by an index running from 1 to x. _N_ is used to set the numeric suffix for the macro variable DS. A macro variable, TOTAL, is also created, which contains the total number of data sets/memnames being processed (and will become the upper limit of the DO loop). CALL SYMPUT is used here to perform both these tasks. Note that the first CALL SYMPUT creates one macro variable for each memname, while the second CALL SYMPUT executes only on the last memname and thus _N_ represents the total number of memnames/datasets.

The %DO loop is where the processing on the data sets takes place. A loop is set up, from one to the total number of data sets. This loop will create a series of data steps that process each data set in the library. Because each data set name was saved in a macro variable (DS1-DSx), we can now refer to each data set by its macro name, without explicitly naming the dataset. Two ampersands are needed as a prefix to DS, so that the macro processor will correctly resolve the dataset names. For example, during the first iteration of the DO loop, the macro processor will resolve &I to 1 and &&DS to &DS. The second iteration of the macro processor resolves the resulting &DS1 into the first memname in the library.

Listing 1

```
OPTIONS NOCENTER DATE MPRINT MLOGIC  SYMBOLGEN;

%LET NUMRX=2;  * used to create dummy treatment groups    *;

%MACRO LIBDATA;

 /* Use as many libname statements as you like */
 LIBNAME copyfrom '/original/rawdata/study999';
 LIBNAME copyto     '/test/testdata/study999';

 /* Get contents for data sets (from one of the libraries), save result
    in an output data set */
 PROC CONTENTS DATA=copyfrom._ALL_ MEMTYPE=data
                    OUT=OUT NOPRINT;
 RUN;

 /* Sort prior to selecting unique data set names */
 PROC SORT DATA=OUT;
  BY MEMNAME NAME;
 RUN;

 /* Select unique data set names, remove unneeded datasets */
 DATA A;
   SET OUT;
   BY MEMNAME NAME;
   IF MEMNAME IN ('NORMDATA','NORMLAB','NORMLAB2')
    THEN delete;  * delete the datasets
                    you do not need  *;

   /* Because each variable in a data set produces an observation in
      the output data set, we need to remove the duplicate
      MEMNAMEs. */
   IF FIRST.MEMNAME;
 RUN;

 /* Create data set names as macro  variables & get total number of
    data sets */
 DATA _NULL_;
   SET A END=LAST;
   BY MEMNAME NAME;
   /* Create a macro variable like DS1 with the value of
     MEMNAME */
   CALL SYMPUT('DS'|| LEFT(_N_),TRIM(MEMNAME));
```

```
   /* Create a macro variable for the total # of datasets */
   IF LAST THEN CALL SYMPUT('TOTAL',LEFT(_N_));
 RUN;

 /* Replace this do loop with example code from later in article. */
 %DO i=1 %TO &total;
   DATA copyto.&&DS&I;
      LENGTH ci prot ptid rxgrp 8.  arxgrp $8.;
      SET copyfrom.&&DS&I;
      ci=9999;
      prot=0001;
      rxgrp=mod(ptid,&numrx)+1;
      arxgrp=substr('AB',rxgrp,1);
   RUN;
 %END;

%MEND LIBDATA;

/* Call the macro */
%LIBDATA;
```

That outlines the general use of PROC CONTENTS to dynamically reference datasets in a library. Next we will present some examples that modify the %DO loop for other purposes.

Example 1: You can use this macro to extract a subset of data from all datasets and create a test database. This is very useful when you are testing/validating your programs. A subsetting IF is used below, but other subsetting data step techniques may be used.

```
 %DO i=1 %TO &total;
   DATA test.&&DS&I;
     SET best.&&DS&I;
     IF trial=1; * Subset Condition *;
   RUN;
 %END;
```

Example 2: You can create new variables that are added to all datasets. Here we show an example where two fake treatment group variables are added to all datasets, each with 8 possible values. ARXGRP is the character version of RXGRP with 1 mapping to A, 2 to B, etc.

```
 %DO i=1 %TO &total;
   DATA test.&&DS&I;
     LENGTH rxgrp 8. arxgrp $8.;
     SET best.&&DS&I; * Create fake treatment group
          variables *;

     rxgrp=MOD(ptno,8)+1; * Eight  groups *;
     /* Convert numeric rxgrp to character variable */
     arxgrp=substr('ABCDEFGH', rxgrp,1);
 RUN;
%END;
```

Example 3: You can assign values to missing variable labels or rename variables.

```
 %DO i=1 %TO &total;
   DATA test.&&DS&I;
     SET best.&&DS&I;
     LABEL ptno='Patient ID';
     RENAME ptno=ptid;
   RUN;
 %END;
```

Example 4: You can use PROC FREQ on all data sets. Of course, other procs, such as PROC SUMMARY,PROC SORT, PROC PRINT, etc. could be used.

```
 %DO i=1 %TO &total;
   PROC FREQ DATA=best.&&DS&I;
     TABLE rxgrp;
     TITLE "Treatment group frequencies in &&DS&I";
   RUN;
 %END;
```

Example 5: You can merge variables from one data set into all data sets. In the following example, demographic information is merged with each dataset.

```
 PROC SORT DATA=best.demo
     out=demo(keep=ci prot trial ptno rxgrp age);
   BY trial ptno;
 RUN;

 %DO i=1 %TO &total;
   DATA test.&&DS&I;
     MERGE test.&&DS&I demo(in=d);
     BY trial ptno;
   RUN;
 %END;
```

ADDITIONAL USES

As you have seen, with PROC CONTENTS you can specify an output data set using the OUT= option. This option gives the user the ability to capture the valuable information seen in the printed output result of the PROC CONTENTS in a SAS data set. Each variable in each of the Data= data sets will produce one observation in the resulting data set. There are over 25 variables that are by default in the OUT= data set. Several of these variables are FORMAT, FORMATD, FORMATL, LABEL, LENGTH, MEMNAME, MEMTYPE, MODATE, NAME, NOBS, NPOS, TYPE, and VARNUM. These variables offer valuable information regarding data sets in a library as well as an easy way to check information across data sets. These variables can be used in the same manner as discussed above for MEMNAME. For example, take the variable LENGTH, which is the length of the variable NAME. We can use this field to check the length of the variable across data sets and modify it, if needed. Additionally, labels can be checked and modified using the variable LABEL, which is the label for the variable NAME. The last time the data set was modified can be determined from the variable MODATE. This may be useful in checking the last time a dataset was updated by an automated process.

CONCLUSION

The macro we have presented is a relatively simple but powerful one. It allows for more dynamic control and output of data sets. With the basic skeleton of the macro as a starting point, users can perform data set and/or procedure operations on all the data sets in a SAS data library easily without regard to the names of the files there.

REFERENCES

SAS Institute Inc., *SAS Procedures Guide*, Version 6 3rd Ed., 1990

SAS Institute Inc., *SAS Guide to Macro Processing*, Version 6, 2nd Ed., 1990

ACKNOWLEDGEMENTS

The authors would like to recognize and thank Neil Howard for her continued support, guidance and review of our ideas and final paper.

CONTACT INFORMATION

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

Subrahmanyam Pilli
Work Phone: (734) 622-3265
Email: Subra.Pilli@pfizer.com

Luai Alzoubi
Work Phone: (734) 622-2168
Email: Luai.Alzoubi@pfizer.com

Kent Nassen
Work Phone: (734) 622-2048
Email: Kent.Nassen@pfizer.com

Address:
Pfizer Global Research and Development
Ann Arbor Laboratories
2800 Plymouth Road
Ann Arbor, MI 48105

Web: www.pfizer.com