Paper 91-28

Using a SAS® Macro to Document the Database

William C. Murphy
Howard M. Proskin & Associates, Inc., Rochester, NY

ABSTRACT

Creating the database for your client was straightforward, but now you have to document its structure by listing all of the data sets contained in the database, along with variable lists and sample contents. You could just run a PROC CONTENTS on _ALL_ members of the library to generate a list of the component data sets and the associated variable lists, followed by several PROC PRINTs to generate sample outputs. The client, however, wants the variable list interleaved with the sample printouts from each data set, along with descriptive titles. You can easily accomplish this by alternating the PROC CONTENTS with PROC PRINT, along with appropriate TITLE statements for each data set in the library. However, to do this every time you must report on a database could be an onerous programming task. But by using ODS, PROC DATASETS, and a %DO loop, you can create a macro to handle your variable and data set documentation needs for any database. Furthermore, by using a macro, you have greater ability to generate additional documentation and to enhance the appearance of your output.

INTRODUCTION

As a small statistical consulting firm in western New York, we build and analyze databases from data supplied by laboratory and clinical studies on a variety of consumer and medical products. The study data are usually segmented into a variety of SAS data sets. These data sets are functional groups of the study parameters and usually have relevant descriptive names (e.g. Demographics, AdverseEvents, VitalSigns, etc.). When we return this structured database to out clients, we include a list of the data sets, the variable content of each data set, and a sample printing of the first few observations from each data set. In fact, this documentation of the database is also used in-house as a guide in the analysis of the data.

MANUAL PROGRAMMING

To build this database documentation, we could simply run a PROC DATASETS to generate a list of the data sets contained in the project library. To obtain the contents of each data set, we could simply run a PROC CONTENTS on _ALL_ of the data sets in the project library. We would then run a PROC PRINT on each data set limiting the output to the first few observations.

This procedure is simple and direct and produces all of the documentation that is required (see Figure 1). However, we have to manually interleave the outputs from PROC CONTENTS and PROC PRINT. This manual collation will also mean that the page numbering will be off. Furthermore, if we use this program on a different study, we must change the data set names in the PROC PRINTs and adjust the number of procedures to reflect the structure of our new database.

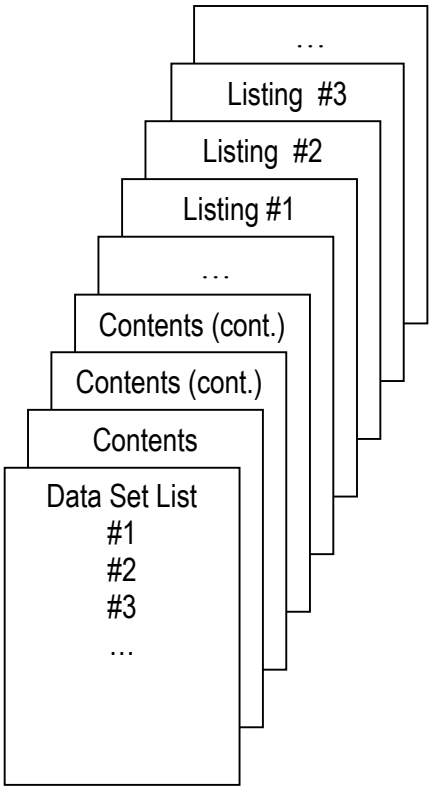

Figure 1. Documentation produced manually before collation.

MACRO SOLUTION

To avoid these problems, we could feed the list of data set names from PROC DATASETS into a data set. This could be readily accomplished using the SAS Output Delivery System (ODS). This new data set could be printed out to create our table-of-contents list of the complete database. We could also store the names of the data sets contained in this list into macro variables. Then using a %DO loop we could alternately run PROC CONTENTS and PROC PRINT on each data sets. Thus, we would have a program in which we did not have to enter the data set names and the output would be ordered without manual intervention.

DATA SET LIST

To start our new process, we must first obtain a list of the data sets in out database and save this list into a new data set:

```
ods listing close;
ods output members=DataSetList;

proc datasets mt=data library=&library;
    run;
    quit;

ods listing;
```

The first ODS command turns off the standard output. It is similar to the NOPRINT option available in a variety of SAS PROCs. The second ODS command causes the creation of an OUTPUT data set composed of the PROC DATASETS object MEMBERS. The new data set created will be called DataSetList. We then execute a PROC DATASETS, limiting the procedure to data sets (MT=DATA) contained in the libname referred to by the macro variable &LIBRARY. The final ODS statement turns the standard output back on.

TABLE OF CONTENTS

The first page output that we create should have a list of all of the data set contained in the database. We can readily accomplish this by printing out the data set created by the PROC DATASETS:

```
title3 "Data Sets in Library '&Library'";
proc print data=DataSetList(drop=memtype)
                label nobs;
    format file_size comma20.;
    run;
```

where we have dropped the variable MEMTYPE, since only data sets are allowed. We also format the file size parameter to inset commas and add a descriptive title at the top of the output. The first two TITLE statements are reserved for the project description, which would be assigned by the calling program.

MACRO VARIABLES

The next thing that we need to do is determine the number of data sets contained in our database:

```
data _null_;
   if 0 then set DataSetList
         nobs=nobs;
   call symput('nDataSet',nobs);
   run;
```

where we use a DATA _NULL_ step since we do not wish to create an output data set. The IF statement is always false and thus the data set DataSetList will not be read into the DATA step. However, when the DATA step contains a SET statement, the informational header of the data set is made available. From this header we obtain the number of observations (nobs) and thus the number of data sets in our database, which we store in the macro variable &nDataSet by using CALL SYMPUT.

Again using a DATA _NULL_ step, we can now read the data set names in our database and save them in macro variables:

```
data _null_;
    set DataSetList;
    call
    symput('DataSet'||left(_n_),memname);
    run;
```

where we use the CALL SYMPUT statement to store the contents of the variable MEMNAME into macro variables. These macro variables will have the name of &DataSet with a suffix supplied by the system reserved variable _n_ (i.e. &DataSet1, &DataSet2, &DataSet3…). The last data set name will be stored in the macro variable &&DataSet&nDataSet.

We should note that these two DATA _NULL_ steps could have been combined into one (Murphy, 2002). However, knowing the number of data sets in the database before the execution of the second DATA step allows us to declare the macro variables %LOCAL (*i.e.* the macro variables are only available within the macro routine).

LOOPING

Now we construct a macro %DO loop. The macro index, &i, will run from one to the total number of data sets, &nDataSet; each cycle processes one of the data sets contained in our macro variables.

Within the loop, we first produce the list of our variables:

```
    title3  "Structure of Data Set
                              &&DataSet&i";

    proc contents
            data=&library..&&DataSet&i;
        run;
```

where we use the direct output of the PROC CONTENTS procedure. The title, TITLE3 statement, will contain the name of the data set. Next we do our sample listing:

```
    title3 "Partial Listing of Data Set
                              &&DataSet&i";

    proc print
    data=&library..&&DataSet&i(obs=5) ;
        run;
```

where again we have added a descriptive TITLE statements. We have limited our output to the first five observations

These statements will produce alternate outputs, first from PROC CONTENTS and then from PROC PRINT, of all of the data sets in our database.

ETIQUETTE

To prevent conflicts with the main program and other macro routines, we have declared all of the macro variables that we use as %LOCAL. We have also chosen a name for our data set (DataSetList) that we hope will not conflict with a name used elsewhere. Finally, after the macro executes, we have cleaned up after ourselves. First, we cleared the TITLE so that it will not show up on

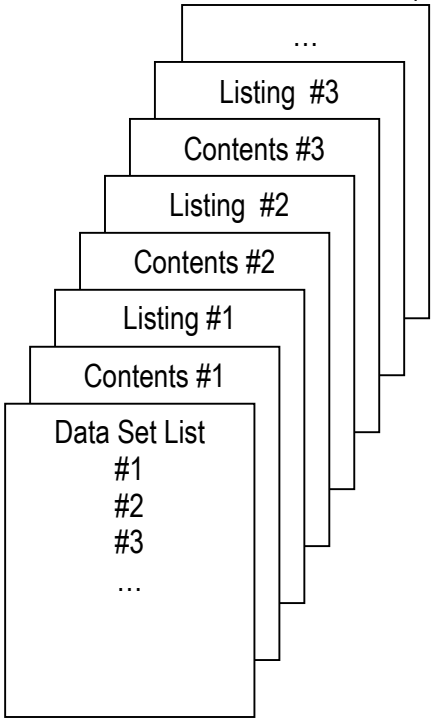

Figure 2. Documentation produced by Macro.

the next output:

```
title3;
```

Then we deleted the data set created and used by the macro program to save memory and prevent any possible future conflicts elsewhere in the program:

```
proc datasets nolist library=work;
   delete DataSetList;
   run;
```

We could now compile our macro and save it in a central library and everyone could have access to it (Murphy, 1998).

To invoke this macro, we must point to a library with a LIBNAME statement and then use the library name in the macro call:

```
libname base 'c:\SomeDirectory';
%Doclist(base);
```

The output generate by this macro is illustrated in Figure 2.

FUTURE ENHANCEMENTS

You could enhance the database documentation by capturing the output of PROC CONTENTS in the %DO loop and then custom designing the output. You could even replace the PROC PRINT with PROC REPORT. By further use of ODS, you could also send your documentation output directly to MicroSoft Word and thus use it as part of your formal project documentation. If you don't like PROC DATASETS and want to avoid, DATA _NULL_ steps and CALL SYMPUT statements, you could also replace most of our programming logic by exploiting the views in the SASHELP library with PROC SQL (Davis, 2001).

CONCLUSION

Using the SAS Output Delivery System and a simple macro program, a database composed of SAS data sets can easily be documented. Furthermore, this program can be applied to any such database and produces titled output in the desired order.

REFERENCES

Davis, M. (2001), "You Could Look It Up: An Introduction to SASHELP Dictionary Views," *Proceedings of the Twenty-Sixth Annual SAS Users Group International Conference*, 26, Paper 17-26.

Murphy, W. C. (1998), "Creating and Maintaining a Central SAS® Library for Health Care Management," *Proceedings of the Twenty-Third Annual SAS Users Group International Conference*, 23, 1128-1130.

Murphy, W. C. (2002), "Documenting the Database: A SAS® Macro to Automate the Process," *Proceedings of the 2002 Pharmaceutical Industry SAS® Users Group Conference*, 103-105.

APPENDIX

The following is a complete listing of the macro program that is discussed in this paper:

```
%macro DocList(library);

   %***** Declare Macro Variables Local
                                        *****;
   %local i nDataSet;

    %***** Get List of the Data Sets in the
       Library *****;
    ods listing close;
    ods output members=DataSetList;

    proc datasets mt=data library=&library;
       run;
       quit;

    ods listing;
```

```
title3 "Data Sets in Library
                          '&Library'";
proc print
    data=DataSetList(drop=memtype)
    label nobs;
    format file_size comma20.;
    run;

%***** Get the Total Number of Data Sets
        into a Macro Variable   *****;
data _null_;
   if 0 then set DataSetList
         nobs=nobs;
   call symput('nDataSet',nobs);
   run;

%***** Declare Data Set Macro
                Variables Local *****;
%do i=1 %to &nDataSet;
   %local DataSet&i;
   %end;

%***** Put Data Set Names *****;
%***** into Macro Variables *****;
    Variable *****;
data _null_;
    set DataSetList;
    call symput
       ('DataSet'||left(_n_),memname);
    run;

%***** Loop through Each Data Set
                                    *****;
%do i=1 %to &nDataSet;

    title3 "Structure of Data Set
                          &&DataSet&i";
    proc contents
       data=&library..&&DataSet&i;
       run;

    title3 "Partial Listing of Data Set
                          &DataSet&i";
    proc print
    data=&library..&&DataSet&i(obs=5);
       run;

    %end;

%***** Clean Up *****;
title3;
proc datasets nolist library=work;
    delete DataSetList;
    run;
    quit;

%mend;
```

ACKNOWLEDGEMENTS

The author would like to express his appreciation to Howard M. Proskin for his helpful suggestions in the creation of this paper.

TRADEMARKS

CONTACT

William C. Murphy
Howard M. Proskin & Associates, Inc.
2468 E. Henrietta Rd.
Rochester, NY 14623
Phone 585-359-2420
FAX 585-359-0465
Email wmurphy@hmproskin.com or
wcmurphy@usa.net
Web www.hmproskin.com