

Automatically Documenting a SAS Data Set: Creating a Codebook with DOCSPGM

Steve James
Centers for Disease Control and Prevention
Atlanta, GA

Introduction

While documentation does not normally rank very high on a programmer's list of fun things to do, it is, nevertheless, a necessary component in any project. Anything that can simplify this task would be a welcome relief. One common task in documentation is to produce a codebook. A codebook provides a complete description of all of the data items in a data set or project. Fortunately, much of that information is already available within the SAS Data Set and its associated format library. What's needed is a convenient way to gather that information and present it in a useable format. DOCSPGM was written to do just that. While the output of this program may not suit all of a user's need for codebook, it provides a good foundation of one on which more can be added.

How it Works

DOCSPGM merges the information from a SAS Data Set with the information from a SAS Format Library to produce a codebook. Output from a PROC CONTENTS (Variable Name, Length, Type, Format and Label) is merged with the output of a PROC FORMAT (Format Name, and Labels). The result is a description of the variables with a list of all the possible values as defined by its format. An example of the output can be found in Figure 1.

How to Use It

To use DOCSPGM, the user must only identify a few items at the beginning of the program before running it. The user must identify the location of the SAS Data Library, the SAS Format Library, and the SAS Data Set. Optionally, the user must decide if the output should contain the unformatted values of a variable. For example, if the formatted value of a variable SEX is 'MALE', the program can optionally add the unformatted value so that it would display '(1)MALE'. Setting the value of ADDVALUE to 'YES' will include the unformatted value while setting it to 'NO' will not include the unformatted value.

Another optional parameter controls how many values of a variable will be displayed. If, for instance, you have the variable STATE, which contains all 50 state names, you may not want all of them to print in your codebook. Setting the value of the parameter MAXLABEL will limit the number of values that you will see. If there are more values to print than MAXLABEL will allow, the label '... and more...' will be displayed. Setting the value of MAXLABEL affects the output of all variables, not just selected ones. A value of '0' means all values will print.

For example, the following code uses a data set called SESUG.SD2 in C:\SASPD, and a format library stored in the same directory. The unformatted value will be added to the

formatted values of the output, and all possible values of a variable will be printed. A complete example of the code can be found in Appendix A.

```
%let library = C:\SASPD ;  
%let datalib = C:\SASPD ;  
%let member = SESUG ;  
%let addvalue = yes ;  
%let maxlabel = 0 ;
```

Conclusion

While documentation will always be a task most of us despise or won't do, there are tools available that can help the process. DOCSPGM is one of those tools. As stated earlier, DOCSPGM is not the total answer, but will provide the framework from which a more comprehensive product can be produced.

Author Contact:

Steve James
Centers for Disease Control and Prevention
4770 Buford Highway
Atlanta, GA 30341-3724
(770) 488-4269

spj1@cipcod1.em.cdc.gov

SESUG STUDY Data Set Variables	
SAS Library Name	C:\SASPD
SAS Data Set Name	SESUG
SAS Data Engine	V611
Number of Variables	5
Number of Observations	30
Date of Report	August 2, 1996

Figure 1 Sample Output

SESUG STUDY Data Set Variables						
Variable Name	Type	Length	Label	Format	Possible Values	
CASES	Num	8	Number of Cases of Study Condition	COMMA		
POP	Num	8	Population	** None **		
RACE	Char	1	Race of Study Participant	** None **		
SEX	Char	1	Sex of Study Participant	\$SEX	(1)Male	(2)Fem
YEAR	Char	4	Year of Study	\$		

Figure 2 Sample Output continued

```

proc format library=library ;
value $sex '1' = 'Male' '2'='Female' ;
value $race '1' = 'White' '2'='Black' '3'='Asian/PI'
           '4' = 'Native American' '5'='Other' ;

run;

```

CONTENTS PROCEDURE

Data Set Name:	DATA.SESUG	Observations:	30
Member Type:	DATA	Variables:	5
Engine:	V611	Indexes:	0
Created:	11:53 Friday, August 2, 1996	Observation Length:	22
Last Modified:	11:53 Friday, August 2, 1996	Deleted Observations:	0
Protection:		Compressed:	NO
Data Set Type:		Sorted:	NO
Label:			

-----Engine/Host Dependent Information-----

Data Set Page Size:	8192
Number of Data Set Pages:	1
File Format:	607
First Data Page:	1
Max Obs per Page:	369
Obs in First Data Page:	30

-----Alphabetic List of Variables and Attributes-----

#	Variable	Type	Len	Pos	Format	Label
4	CASES	Num	8	6	COMMA5.	Number of Cases of Study
5	POP	Num	8	14		Population
2	RACE	Char	1	4		Race of Study Participant
3	SEX	Char	1	5	\$SEX.	Sex of Study Participant
1	YEAR	Char	4	0	\$4.	Year of Study

Figure 3 Sample Data

Appendix A - SAS Source Code

```

*-----
* This program creates a codebook for a SAS data set, describing
* each of the variables in the data set, its length, type, label,
* user-defined format, and the possible values it can have based on
* the format assigned to it. It does this by combining the output
* of a PROC CONTENTS and a PROC FORMATS and printing out the
* results using PROC REPORT.
*
* To run, simply modify the %LET statements at the two TITLE
* statements at the beginning of the program and submit. Linesize
* and Pagesize options may also be changed to suit your environment.
*-----
* DOCSPGM
*
* Written by : Steve James  spjl@cipcod1.em.cdc.gov
*
* last updated 8/1/1996
*-----

%let library = C:\SASPD ;          * Identify format library ;

%let datalib = C:\SASPD ;          * Identify sas data library ;

%let member = SESUG ;             * Identify data set name ;

%let addvalue = yes ;             * ADDVALUE=YES adds unformatted value;
                                  * to format label.  eg (1)yes vs yes ;
                                  * Upper/Lowercase is OK.
                                  ;

%let maxlabel = 0 ;               * MAXLABEL = n limits each variable ;
                                  * to n entries.  maxlabel=0 means no ;
                                  * limit.
                                  ;

* Define titles that you want to go on each page of the output ;

title 'SESUG STUDY Data Set Variables' ;
title2 ' ' ;

options pagesize=64 linesize=132 nodate ;

***** NO USER MODIFY FIELDS BELOW THIS LINE *****;

options pageno=1 ;
libname data "&datalib" ;
libname library "&library" ;

proc format ;
  value vartype 1='Num' 2='Char' ;
run;

*-----
* Now create the data set that contains the formats and their
* possible values by taking the output from a PROC FORMAT.
*-----

proc format library=library
  cntlout=formats(rename=(fmtname=format label=value type=ftype)
    keep=fmtname type label start end
    where=(ftype in ('N','C')))) ;

```

```

run ;

*-----;
* Create a data set with a list of all the variables and any      ;
* user-defined format names.                                     ;
*-----;

proc contents data=data.&member noprint
      out=vars(keep=name type format label length engine nob) ;
run ;

*-----;
* Combine the two data sets by format name. PROC SQL is because MERGE;
* will not handle the case where the same format is used for more than;
* one variable, and there is no need to strip the leading '$' from the;
* name of character formats.                                     ;
*-----;

proc sql ;
  create table document as
  select vars.name,vars.type,vars.format,vars.label,vars.length,
         vars.engine,formats.start,formats.end,formats.value,vars.nobs
  from vars left join formats
  on scan(vars.format,1,'$') = formats.format ;

proc sort data=document ;
  by name start ;
run ;

options charcode ; * uses ?/ as concatenation character instead of | ;

data document ;
  length format $10 value $25 ;
  set document ;
  by name ;
  if format = ' ' then format = '** None **' ;
  *-----;
  * If ADDVALUE = YES, add the unformatted value to      ;
  * the the formatted value.                             ;
  *-----;
  if upcase("&addvalue") = "YES" and start ne ' '
  then do ;
    if start eq end
    then value = '(' ?/?/ trim(left(start)) ?/?/ ')' ?/?/ value ;
    else value = '(' ?/?/ trim(left(start)) ?/?/ '-'
                  ?/?/ trim(left(end)) ?/?/ ')' ?/?/ value ;
    end ;

  *-----;
  * If MAXLABEL is specified, count the number of      ;
  * entries for each variable and delete those past the;
  * limit.                                              ;
  *-----;
  if &maxlabel > 0
  then do ;
    if first.name then count= 0 ;
    count+1 ;
    if count = &maxlabel and not last.name
    then value = '... and more ...' ;
    else if count > &maxlabel then delete ;
    end ;
  output document ;

```

```

run;

*-----;
* The number of observations in the data set VARS equals the number ;
* of variables in the data set being documented. Also in that ;
* data set is the number of observations. Get that information ;
* about the data set and assign it and other known data to macro ;
* variables to be used later in title statements. ;
*-----;

data _null_ ;
  set vars point=_n_ nobs = numvars ;
  call symput('numvars',left(put(numvars,8.))) ;
  call symput('numobs',left(put(nobs,8.))) ;
  call symput('date',left(put(today(),worddate18.))) ;
  call symput('engine',left(put(engine,$10.))) ;
  stop ;
run ;

data _null_ ;
  file print ;
  put // ;
  put @40 "SAS Library Name" @75 "&DATALIB" ;
  put @40 "SAS Data Set Name" @75 "&MEMBER" ;
  put @40 "SAS Data Engine" @75 "&ENGINE" ;
  put @40 "Number of Variables" @75 "&NUMVARS" ;
  put @40 "Number of Observations" @75 "&NUMOBS" ;
  put @40 "Date of Report" @75 "&DATE" ;
  stop ;
run ;

*-----;
* Now use PROC REPORT to print out the data. ;
*-----;

proc report data = document nowindows headskip missing ;
  column name type length label format start value ;
  define name / Group width=8 'Variable/Name' ;
  define type / Group width=4 'Type' format=vartype. left ;
  define length / group width=6 'Length' ;
  define label / group width=40 'Label' ;
  define format / group width=10 'Format' ;
  define start / group noprint ;
  define value / group width=25 'Possible/Values' ;
  break after name / suppress skip ;
run ;

```