PharmaSUG 2014 - Paper CC43

Give me *EVERYTHING*! A macro to combine the CONTENTS procedure output and formats.

Lynn Mullins, PPD, Cincinnati, Ohio

ABSTRACT

The PROC CONTENTS output displays SAS® data set information such as variable name, type, length, informats, and format names. The values and codes for the formats are not included in this output; therefore, a separate printout of the program that creates the formats or a printout of the format catalog via DATA step programming needs to be used in conjunction with the PROC CONTENTS output. This paper will describe a SAS® macro that combines PROC CONTENTS output with the format catalog data to create a single metadata dictionary file.

INTRODUCTION

PROC CONTENTS is a SAS® procedure which displays the contents of a SAS data set. The data set metadata displayed includes information such as the data set name, engine, number of observations, number of variables, etc. The metadata for the variables include the name, length, type, number, and format name, if applicable. When a format catalog is associated with variables in a data set, the values and codes for these format are **not** included in the PROC CONTENTS output. The format values and codes need to be displayed using another process. Not only does another process have to be performed, the format values and codes have to be matched to the format names, and there are most likely multiple variables using the same format name. This can take a lot of time to do manually.

Just like Pit Bull and Ne-Yo's song, a statistician that I was working on a project with told me to "***GIVE HER EVERYTHING***! Most project teams require the use of a data dictionary or schema to describe the data sets that will be used for analysis. I usually ran a PROC CONTENTS on of all the data sets to achieve this output but I soon found out that the PROC CONTENTS output was not sufficient enough for the statistician that I was working with to give her all the information that she needed to be able to gain a full understanding of the variables in the data sets. She wanted only one document to look at with everything about the variables in the data set. I needed a way to display the output contained from a PROC CONTENTS **and** the format values and codes.

This paper will describe a SAS® macro which combines the output from the PROC CONTENTS procedure and the format catalog to create a metadata dictionary file containing all the information necessary to give the project team ***EVERYTHING*** about a data set in one call.

THE PREVIOUS PROCESS

The three step process that I was using to create a metadata dictionary was:

1. Run PROC CONTENTS on all the data sets in the database.

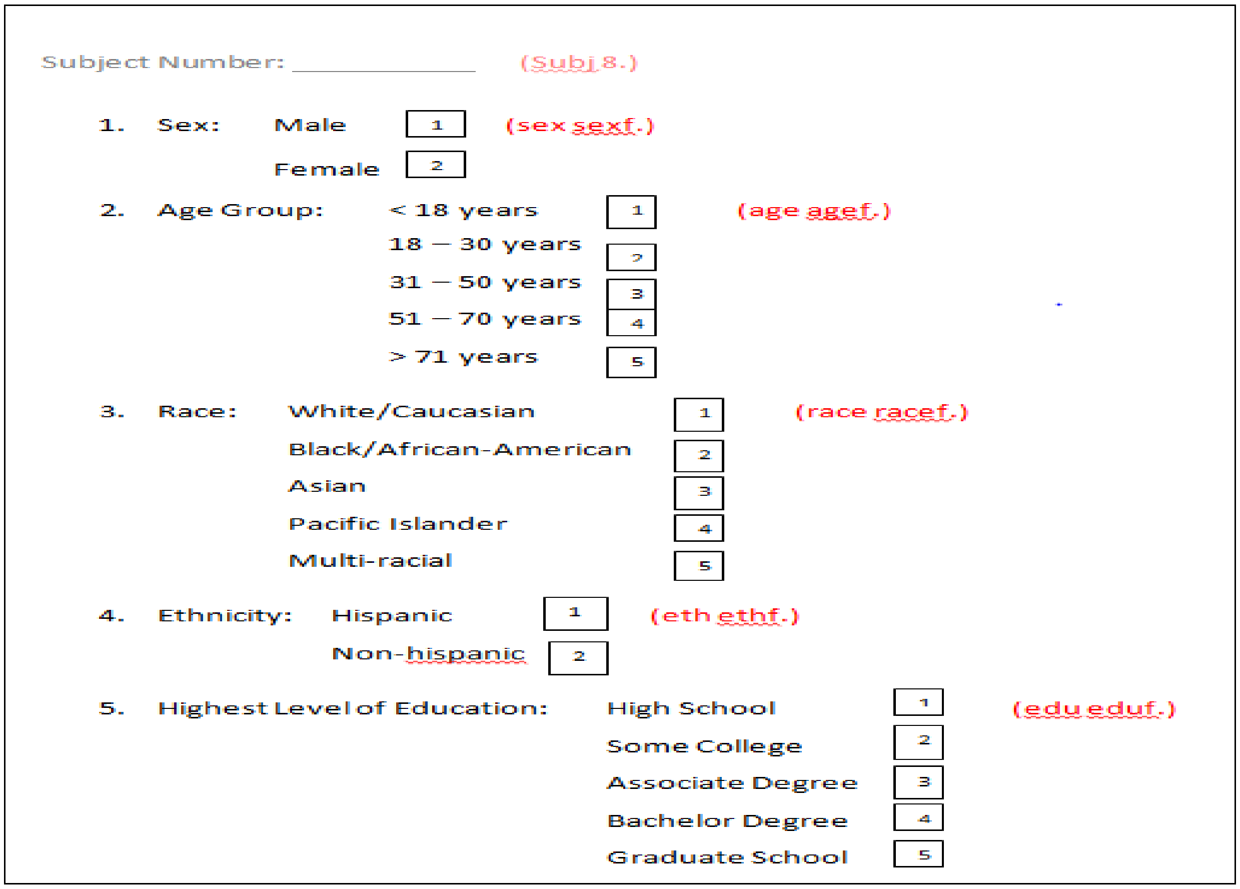
```
The SAS System

The CONTENTS Procedure

Data Set Name    DATA.DEMOG   Observations  5
Member Type      DATA         Variables     6
…
Alphabetic List of Variables and Attributes
#   Variable     Type   Len    Format       Informat     Label
3   age          Num    8      AGEGRPF.     BEST.        Age
6   edu          Num    8      EDUF.        BEST.        Education
5   eth          Num    8      ETHF.        BEST.        Ethnicity
4   race         Num    8      RACEF.       BEST.        Race
2   sex          Num    8      SEXF.        BEST.        Sex
1   subj         Num    8                   BEST.        Subject Number
```

Output 1. Output snippet from PROC CONTENTS Statement

2. Print the PROC FORMAT program.

```
proc format library = library;
  value sexf      1 = 'Male'
                  2 = 'Female';
  value agegrpf   1 = '< 18 years'
                  2 = '18 - 30 years'
                  3 = '31 - 50 years'
                  4 = '51 - 70 years'
                  5 = '> 70 years';
  value racef     1 = 'White/Caucasian'
                  2 = 'Black/African-American'
                  3 = 'Asian'
                  4 = 'Pacific Islander'
                  5 = 'Multi-racial';
  value ethf      1 = 'Hispanic'
                  2 = 'Non-hispanic';
  value eduf      1 = 'High School'
                  2 = 'Some College'
                  3 = 'Associate Degree'
                  4 = 'Bachelor Degree'
                  5 = 'Graduate School';
run;
```

3. Print the annotated Case Report Forms (aCRF).



Subject Number: ____________ (Subj 8.)

1. Sex: Male 1 (sex sexf.)
   Female 2

2. Age Group: < 18 years 1 (age agef.)
   18 – 30 years 2
   31 – 50 years 3
   51 – 70 years 4
   > 71 years 5

3. Race: White/Caucasian 1 (race racef.)
   Black/African-American 2
   Asian 3
   Pacific Islander 4
   Multi-racial 5

4. Ethnicity: Hispanic 1 (eth ethf.)
   Non-hispanic 2

5. Highest Level of Education: High School 1 (edu eduf.)
   Some College 2
   Associate Degree 3
   Bachelor Degree 4
   Graduate School 5

Output 3. Annotated Case Report Form.

This process created the information needed to interpret the required variable information but that statistician had to look at three separate outputs to get all the information. Having multiple outputs to review was very cumbersome, time-consuming, and not an efficient use of the statistician’s time, which prompted me to look for an alternative solution.

CREATING EVERYTHING WITH A ONE-STEP PROCESS

The macro creates one output containing all the variable information from the PROC CONTENTS output and the codes and values from the format catalog. The manual effort needed by the statistician was virtually eliminated. All she needed to look at was one output to gain a full understanding of all the variables in the data sets.

CREATE A DATA SET OF THE FORMATS

Prior to calling the macro, a data set of the formats needs to be created. This is done by adding “cntlout = *dsn*” to the PROC FORMAT statement and running the format program. A data set containing the format information will be created. This format data set will be merged by format name to a data set created by a PROC CONTENTS statement. An example is shown below.

```
proc format library = library;  ←——— ORIGINAL CODE
proc format cntlout = data.fmtout;  ←——— NEW CODE
```

CREATE A DATA SET OF THE FORMATS

The macro call consists of the following parameters:

```
inlib  =  input data set library name
          defaults to work

inds   =  input data set name

inflib =  format data set library name
          defaults to the value of inlib

fmtds  =  format data set name

outds  =  output data set name

outlib =  output data set library name
          defaults to the value of inlib

path   =  dictionary file output path
```

CALL THE MACRO

The call to the macro should look like:

```
%METADATA(inlib=data, inds=demog, inflib=data, fmtds=fmtout, outds=metadata,
          outlib=metalib, path=U:\PSUG)
```

THE METADATA FILE

The macro creates the metadata dictionary as a Microsoft Excel file. The variable name, length, number, label, format, informat, type, and format codes with the format values are displayed in one file! A separate document with the format values is no longer needed. Needless to say, the statistician that I was working with was ecstatic! The trick to having the format codes lined up in the column is the linefeed character '0A'x concatenated to the format code and value text but the format value column will have all the values as one line when the file is first opened. In order for the format values to line up, the column width will need to be manually adjusted and the column format needs to have wrap text set in the alignment property. A sample of the final metadata dictionary file is below:

NAME	LENGTH	VARNUM	LABEL	FORMAT	INFORMAT	VTYPE	VAL
subj	8	1	Subject Number		BEST	Num	
sex	8	2	Sex	SEXF	BEST	Num	1=Male 2=Female

NAME	LENGTH	VARNUM	LABEL	FORMAT	INFORMAT	VTYPE	VAL
age	8	3	Age	AGEGRPF	BEST	Num	1=< 18 years 2=18 - 30 years 3=31 - 50 years 4=51 - 70 years 5=> 70 years
race	8	4	Race	RACEF	BEST	Num	1=White/Caucasian 2=Black/African-American 3=Asian 4=Pacific Islander 5=Multi-racial
eth	8	5	Ethnicity	ETHF	BEST	Num	1=Hispanic 2=Non-hispanic
edu	8	6	Education	EDUF	BEST	Num	1=High School 2=Some College 3=Associate Degree 4=Bachelor Degree 5=Graduate School

Table 1. Metadata dictionary file

CONCLUSION

Everyone would agree that less is best when it comes to reviewing output. The metadata macro eliminates the need to review multiple documents for understanding variables in a data set. All the necessary information about the variables needed for a statistician to generate analyses in one document. This macro is general enough to also be used for many other applications where a metadata dictionary would be helpful in understanding ***EVERYTHING*** about a data set.

REFERENCES

SAS Institute, "TS-DOC: TS-642 – Reading EBCDIC files on ASCII Systems. Available at https://support.sas.com/techsup/technote/ts642.html.

DISCLAIMERS

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of PPD.

CONTACT INFORMATION

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

Lynn Mullins
PPD
6947 Newbridge Drive
Cincinnati, Ohio 45239
+1 (910) 558-4343 (Phone)
+1 (919) 654-9907 (Fax)
lynn.mullins@ppdi.com

ATTACHMENT 1 - The METATDATA.SAS program

```
%MACRO METADATA(inlib=work, inds=, inflib=&inlib, fmtds=, outds=, outlib=&inlib, path=);

  /***********************************************/
  /** inlib   = input data set library name  **/
  /**             defaults to work           **/
  /** inds    = input data set name          **/
  /** inflib  = format data set library name**/
  /**             defaults to the value of inlib **/
  /** fmtds   = format data set name         **/
  /** outds   = output data set name         **/
  /** outlib  = output data set library name**/
  /**             defaults to the value of inlib **/
  /** path    = dictionary file output path  **/
  /***********************************************/

   proc contents data = &inlib..&inds
                 out = &inds.cont (keep = name length label type  informat format varnum) noprint;
   run;

  data &inds.cont1 (drop = type);
    set &inds.cont;

  attrib vtype length = $4. format = $4. informat = $4. label = "Variable Type";

  select (type);
     when (1) vtype = "Num";
     when (2) vtype = "Char";
     otherwise;
  end;
  run;

  proc sort data = &inds.cont1;
     by format;
  run;

  data &inds.form (keep = fmtname val);
     set &inflib..&fmtds.;
     by fmtname;

  length val $600.;

  if first.fmtname then val = "";

  length labeln $72.;
  labeln = strip(label) || '0A'x ;

  retain val "";

  val = cats(val, start, " = ", labeln);

  if last.fmtname then output;
  run;

  data &inds.metadata;
     merge &inds.cont1 (in=a)
           &inds.form  (in=b rename = (fmtname = format));
     by format;
     if a;
  run;

  libname &outlib excel path = "&path\&outds..xlsx" scantext=yes;
```

```
proc sort data = &inds.metadata
          out = &outlib..&outds;
   by varnum;
run;

data &outlib..dictionary;
   set &outlib..&outds;
   by varnum;
run;

libname &outlib clear;

 /***************/
 /* Clean Up  */
 /***************/
 proc datasets lib=work  nolist ;
    delete &inds.:;
 run;
 quit;

%MEND METADATA;
```