

Exploiting SAS® Meta Data: A report writer example

Delayne Stokke, GuideOne Insurance, West Des Moines, IA

ABSTRACT

SAS® stores a lot of information about your SAS data sets. I've found that it's advantageous to assign descriptive labels to variables and to data sets. If you store your data in SAS data set format, the labels you assign may be able to serve as all of the documentation that's required. This can be especially useful if your data sets are infrequently used, or if you are supplying your data sets to a colleague or a customer for further use. You may find that it's also helpful to store format information for the variables in your data sets.

This paper will explore some of the Meta Data that SAS stores in the data set, along with some of the functions that provide access to this data. To illustrate some of the ideas, an example report writer will be used.

INTRODUCTION

As a data developer, I've used SAS extensively to create demographic estimates and projections. In most cases, the data that I create is delivered to my customers. Sometimes I've delivered SAS data sets to my customers, and at other times I've converted the data to another format, such as dBase. Regardless of the database format, I need to supply my customer with documentation that describes each variable.

If I've been thoughtful enough to include descriptive labels in my SAS data set, I could use PROC CONTENTS to create a listing of variable names and their descriptions.

In addition, I usually like to include a "snapshot report" that shows the variable values for 1 or 2 observations. PROC PRINT could be used to create this type of report.

THE PROBLEM

The PROC CONTENTS output is pretty specific to SAS, and may be confusing to some of my non-SAS customers. And since the default CONTENTS report lists the variables in alphabetic order, that usually just adds to their confusion. In addition, some of my customers have complained about the way the PROC PRINT report is formatted when it wraps a single observation onto multiple lines.

I needed to find a way to easily combine what my customers' view as the best aspects of these two procedures. From PROC CONTENTS I wanted the Data Set information, the list of variables, and the variable descriptions (in the order that they occur in the data set), and from PROC PRINT I wanted the variable values.

FINDING THE TOOLS FOR THE SOLUTION

I needed to be able to access the information in a SAS data set – not only the data, but the Meta data as well. To get access to the Meta data, I needed to learn about

some unfamiliar SAS functions. The functions that I needed to learn and use were new to the SAS System with the release of version 6.11.

One of the new functions that I found is OPEN. It's simply described in the documentation as "opens a SAS data set". This simple action provides the key to other functions that give you, the programmer, access to the Meta data.

Once a data set was opened, I learned that I could use the ATTRC function to return character attributes for a SAS data set. These character attributes include items such as the LIBREF, the data set NAME, the data set LABEL, plus several others.

```
data ds_info;
  dsid      = open("my_data","i");
  dsname    = trim(attrc(dsid,"lib"))||
              '||'
              attrc(dsid,"mem");
  dslabel   = attrc(dsid,"label");
run;
```

I also found the ATTRN function, which is similar to ATTRC, but it's used to return numeric attributes for a SAS data set. The numeric attributes include items such as the number of observations, the number of variables, the creation date, and more.

```
data ds_info;
  dsid      = open("my_data","i");
  dsobs     = attrn(dsid,"nobs");
  num       = attrn(dsid,"nvars");
  dsccrdate = dtepart(attrn(dsid,"crdte"));
run;
```

Once I had my data set open, and I knew how many variables there were in the data set, I could use several other new functions to get information about each variable, such as the name, type, label, etc.

```
data var_info;
  dsid      = open("my_data","i");
  num       = attrn(dsid,"nvars");
  do i=1 to num;
    name=varname(dsid,i);
    type=vartype(dsid,i);
    label=varlabel(dsid,i);
    format=varfmt(dsid,i);
    informat=varinfmt(dsid,i);
    length=varlen(dsid,i);
    output var_info;
  end;
run;
```

One more detail, and I can put this together to create the report I want. The opened data set will be closed automatically at the end of the DATA step. However, if you want to control both the opening and closing of any data sets you are using, you could add the statement "rc=close(dsid);" to your program.

PUTTING THE PIECES TOGETHER

When I finally thought I knew enough to write this program, I wanted it to be as easy to use as possible. I didn't want it to be any harder than running PROC CONTENTS and PROC PRINT. What I decided to do was to create a SAS Macro so that I could create the report I wanted with a simple program like this:

```
%sample(insas=saswork.pe64fips,
        obs=2,
        title1=Sample Report,
        title2=Prepared for SAS Users Group Meeting)
```

I decided that the easiest way to proceed was to create two SAS data sets: one containing the information that I wanted about the data set itself, and another containing the information that I wanted about the variables in the data set. Using the functions that I'd learned about earlier, I put together the following program to create the data set information (ds_info) and the variable information (var_info):

```
options compress=no;
data ds_info (keep=dsname
              dsobs
              memtype
              num
              engine
              dsindex
              dscreate
              obslen
              dsmod
              dsdelete
              dslabel
              dssort)

    var_info(keep=name
              type
              label);

length name $ 8 type $ 1
        label dslabel dsname $ 40;

/* Get Data Set Info */
dsid   = open("&insas","i");
dsname = trim(attrc(dsid,"lib"))||
        '.'||attrc(dsid,"mem");
dsobs  = attrn(dsid,"nobs");
memtype = attrc(dsid,"mtype");
num    = attrn(dsid,"nvars");

...(more statements to get data set info)

/* Get Variables Info */
do i=1 to num;
    name=varname(dsid,i);
    type=vartype(dsid,i);
    label=varlabel(dsid,i);
    output var_info;
end;
rc=close(dsid);
run;
```

Note that I've specified the "compress=no" option. This is because I needed to use the "point=" data set option

to read the variable information in the report step. And since compressed SAS data sets cannot be directly accessed, it's critical to specify no compression.

You should also notice that variable named NUM represents the number of variables in &insas, and will be written to the file "ds_info". In addition, this variable is used as the ending value of the index variable "i" in the do-loop that creates the variable information data set, "var_info".

For each variable found in &insas, the data set "var_info" contains the variable's name, type, and label.

Variables and values from VAR_INFO (sample of 2 observations)

NAME	=FIPS
TYPE	=C
LABEL	=State/County FIPS Code
NAME	=P95_POP
TYPE	=N
LABEL	=July 1, 1995 Total Population

The data set "ds_info" contains all of the information about the data set that I want to include on the report.

Variables and values from DS_INFO

DSLABEL	=PE64 Age/Sex/Race/Hisp 1995-1997
DSNAME	=WORK.PE64FIPS
MEMTYPE	=DATA
ENGINE	=V612
DSCREATE	=21:01 Thursday, January 7, 1999
DSMOD	=11:55 Monday, June 7, 1999
DSSORT	=FIPS
DSOBS	=3140
NUM	=19
DSINDEX	=0
OBSLEN	=149
DSDELETE	=0

Together with the variable values found in &insas, I have three data sets that contain all of the information that I need for my report. Now, it's just a matter of coding a "data_null_" report to pull everything together. The next few code segments will highlight the key elements of the report step.

The first thing I do is to read &insas and set up two arrays: one for character variables, and one for numeric variables. These arrays will allow me to treat the two types of variables differently when I print them.

```
data _null_;
    set &insas(obs=&obs);
    array charvars _character_;
    array numvars _numeric_;
```

Next, I read in the Meta data that describes the data set:

```
if _n_ = 1 then set ds_info;
```

The next key section of code is a do-loop. This is used to step through each variable in &insas. The index variable "ci" is used to keep track of a variable's position in the "charvars" array, while the index variable "ni" is used to keep track of a variable's position in the "numvars" array. Remember that the variable "num" comes from the Meta data file "ds_info", and represents the number of variables in &insas. Notice that as each variable is processed, I use the "point=" data set option to read that variable's Meta data from "var_info".

```
ci=0;
ni=0;
do i = 1 to num;
  set var_info point=i;
  if type = "C" then do;
    ci+1;
    put @5 name $char8.
      @20 label $char40.
      @65 charvars{ci};
  end;

  else if type = "N" then do;
    ni+1;
    put @5 name $char8.
      @20 label $char40.
      @65 numvars{ni} comma15.;
  end;
end;
```

The code shown above takes care of printing the variable Meta data along with sample variable values. The "point=" option used to read the Meta data in "var_info" allows the program to cycle through all of the variables in that dataset for each observation in &insas that is printed.

Finally, the code to print the data set level Meta data is simply a formatted "put" statement, which is executed in the header section of the report.

```
put @9 "Data Set Name:" @24 dsname
  @61 "Observations:" @83 dsobs/
  @9 "Member Type:" @24 memtype
  @61 "Variables:" @83 num/
  @9 "Engine:" @24 engine
  @61 "Indexes:" @83 dsindex/
  @9 "Created:" @24 dscreate
  @61 "Obs Length:" @83 obslen/
  @9 "Last Modified:" @24 dsmod
  @61 "Deleted Obs:" @83 dsdelete/
  @9 "Sorted by:" @24 dssort/
  @9 "Data Set Label:"@24 dslabel//
;
```

I've included a complete copy of the macro as well as a sample report created by the macro in figures 1 and 2, respectively.

CONCLUSION

SAS has given programmers access to a lot of information about our data sets. It's up to us to utilize the available Meta Data access methods to make use of this information.

As you can see from the example macro, you may have to do some extensive "data_null_" programming to get a report formatted the way you want. However, if the report (or application) is something that gets used many times, and if it adds value by providing a better product (better

documentation, fewer customer questions, etc.), then it will be well worth the effort.

I hope that this paper will give you some ideas as to how you can use SAS Meta data to help make your projects easier to manage and document.

Although not used in my example, you might want to take a look at the functions FETCH, FETCHOBS, GETVARC, and GETVARN. These functions can be used to read observations and variable values from a SAS data set.

In addition, I would encourage you to investigate your SAS System by using online help (look under "What's new in Base SAS") and by reviewing recent SAS documentation and User Group Proceedings— you might find something new that you can use.

REFERENCES

Polzin, Jeff and O'Conner, Susan (1995) *Macro Facility Enhancements for Release 6.11*, SUGI 20 Proceedings, SAS Institute Inc., Cary, NC.

CONTACT INFORMATION

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

Delayne Stokke
 GuideOne Insurance
 1111 Ashworth Road
 West Des Moines, IA 50265-3538
 Work Phone: (515) 267-2212
 Email: dstokke@guidemail.com or
 dstokke@aol.com

```

/*-----*
Macro Name:      Sample.sas
Purpose:         Create a sample report from a SAS data set.
Date:            January 13, 1999
Programmer:      Delayne Stokke
Parameters:      (5)
                  insas=,          Input data set.  Can be a 1 or 2
                                level name.
                  obs=1,           Number of observations to be printed.
                                Default is 1.
                  title1=,         Optional first title line.
                  title2=,         Optional second title line.
                  title3=         Optional third title line.
*-----*/

%macro sample(insas=,
              obs=1,
              title1=,
              title2=,
              title3=);

options compress=no; /* Need to use POINT= option later, no compression! */

/* Create 2 SAS Datasets.  Note that there isn't a SET, MERGE, or INFILE! */
data ds_info (keep=dsname
               dsobs
               memtype
               num
               engine
               dsindex
               dscreate
               obslen
               dsmod
               dsdelete
               dslabel
               dssort)

    var_info(keep=name
             type
             label);

length name $ 8 type $ 1 label dslabel dsname $ 40;

/* Get Data Set Info */
dsid   = open("&insas","i");
dsname = trim(attrc(dsid,"lib"))||'.'||attrc(dsid,"mem");
dsobs  = attrn(dsid,"nobs");
memtype = attrc(dsid,"mtype");
num     = attrn(dsid,"nvars");
engine  = attrc(dsid,"engine");
dsindex = attrn(dsid,"isindex");
dscreate = datepart(attrn(dsid,"crdte"));
dsctime = timepart(attrn(dsid,"crdte"));
dscreate = put(dsctime,hhmm5.)||' '||left(put(dscreate,weekdate30.));
obslen  = attrn(dsid,"lrecl");
dsmdate = datepart(attrn(dsid,"modte"));
dsmdtime = timepart(attrn(dsid,"modte"));

```

```

dsmod    = put(dsmdtime,hhmm5.)||' '||left(put(dsmddate,weekdate30.));
dsdelete = attrn(dsid,"ndel");
dslabel  = attrc(dsid,"label");
dssort   = attrc(dsid,"sortedby");
if dssort = ' ' then dssort = 'Not Sorted';
output ds_info;

/* Get Variables Info */
do i=1 to num;
    name=varname(dsid,i);
    type=vartype(dsid,i);
    label=varlabel(dsid,i);
    output var_info;
end;
rc=close(dsid);
run;

/* Create the Report */
data _null_;

    set &insas(obs=&obs);          /* Read input SAS data set, limit observations */

    if _n_ = 1 then set ds_info; /* Initialization only, read Data Set Info. */
    array charvars _character_; /* Set up array of all the character variables... */
    array numvars _numeric_;    /* ... and an array of all the numeric variables. */

    /*----- B e g i n       A U D I T S -----*/
    today = date();

    length title1 title2 title3 $ 100;

    file print header=newpage notitles pagesize=53 linesize=96 n=pagesize
           line=lines;

    title1 = "&title1";
    title2 = "&title2";
    title3 = "&title3";

    put _page_;

    ci = 0;
    ni = 0;

    do i = 1 to num;                /* For each variable in the dataset, */
        set var_info point=i;      /* Read the observation in 'var_info' */

        if type = "C" then do;     /* Character variables are printed here... */
            ci+1;
            put @5 name $char8.
                @20 label $char40.
                @65 charvars{ci};
            end;

        else if type = "N" then do; /* ... and Numeric variables are printed here. */
            ni+1;
            put @5 name $char8.
                @20 label $char40.
                @65 numvars{ni} comma15.;
            end;
        end;
    end;

    return;

```

Exploiting SAS Meta Data: A report writer example – Figure 1

```

newpage;
pageno+1;

put @1 today worddate18.
  @((96-length(title1)) / 2) title1
  @88 "Page" +1 pageno//
  @((96-length(title2)) / 2) title2/
  @((96-length(title3)) / 2) title3//;

/* Output Data Set Info only on Page 1 */
if pageno = 1 then

  put @9 "Data Set Name:"      @24 dsname
    @61 "Observations:"      @83 dsobs/
    @9 "Member Type:"        @24 memtype
    @61 "Variables:"          @83 num/
    @9 "Engine:"              @24 engine
    @61 "Indexes:"            @83 dsindex/
    @9 "Created:"             @24 dscreate
    @61 "Observation Length:" @83 obslen/
    @9 "Last Modified:"       @24 dsmod
    @61 "Deleted Observations:" @83 dsdelete/
    @9 "Sorted by:"           @24 dssort/
    @9 "Data Set Label:"      @24 dslabel//
  ;

  put @5 "Field Name      Field Description"
    @65 "Field Contents"/
    @5 "-----"
    @65 "-----"
  ;

return;

/*----- E n d      A U D I T S -----*/
run;

%mend sample;

```

June 27, 1999

Sample Report

Page 1

Prepared for SAS Users Group Meeting

Data Set Name:	SASWORK.PE64FIPS	Observations:	3140
Member Type:	DATA	Variables:	19
Engine:	V612	Indexes:	0
Created:	21:01 Thursday, January 7, 1999	Observation Length:	149
Last Modified:	11:55 Monday, June 7, 1999	Deleted Observations:	0
Sorted by:	FIPS		
Data Set Label:	PE64 Age/Sex/Race/Hispanic 1995-1997		

Field Name	Field Description	Field Contents
FIPS	State/County FIPS Code	01001
P95_POP	July 1, 1995 Total Population	39,154
P96_POP	July 1, 1996 Total Population	40,277
P97_POP	July 1, 1997 Total Population	41,306
P95_WH	July 1, 1995 White Population	30,737
P95_BL	July 1, 1995 Black Population	8,173
P95_AI	July 1, 1995 American Indian Population	78
P95_API	July 1, 1995 Asian/Pacific Islander Pop	166
P95_HISP	July 1, 1995 Hispanic Population	363
P96_WH	July 1, 1996 White Population	31,573
P96_BL	July 1, 1996 Black Population	8,449
P96_AI	July 1, 1996 American Indian Population	77
P96_API	July 1, 1996 Asian/Pacific Islander Pop	178
P96_HISP	July 1, 1996 Hispanic Population	399
P97_WH	July 1, 1997 White Population	32,286
P97_BL	July 1, 1997 Black Population	8,754
P97_AI	July 1, 1997 American Indian Population	81
P97_API	July 1, 1997 Asian/Pacific Islander Pop	185
P97_HISP	July 1, 1997 Hispanic Population	448

June 27, 1999

Sample Report

Page 2

Prepared for SAS Users Group Meeting

Field Name	Field Description	Field Contents
FIPS	State/County FIPS Code	01003
P95_POP	July 1, 1995 Total Population	119,966
P96_POP	July 1, 1996 Total Population	124,049
P97_POP	July 1, 1997 Total Population	128,842
P95_WH	July 1, 1995 White Population	102,929
P95_BL	July 1, 1995 Black Population	15,999
P95_AI	July 1, 1995 American Indian Population	713
P95_API	July 1, 1995 Asian/Pacific Islander Pop	325
P95_HISP	July 1, 1995 Hispanic Population	1,649
P96_WH	July 1, 1996 White Population	106,343
P96_BL	July 1, 1996 Black Population	16,654
P96_AI	July 1, 1996 American Indian Population	710
P96_API	July 1, 1996 Asian/Pacific Islander Pop	342
P96_HISP	July 1, 1996 Hispanic Population	1,824
P97_WH	July 1, 1997 White Population	110,334
P97_BL	July 1, 1997 Black Population	17,432
P97_AI	July 1, 1997 American Indian Population	729
P97_API	July 1, 1997 Asian/Pacific Islander Pop	347
P97_HISP	July 1, 1997 Hispanic Population	2,060