Document and Enhance Your SAS® Code, Data Sets, and Catalogs with SAS Functions, Macros, and SAS Metadata

Louise S. Hadden

Abt Associates Inc.

Louise Hadden has been using and loving SAS since the days of punch cards and computers the size of a “tiny house.” She spends most of her time in support of health policy analytics at Abt Associates Inc. and loves a good SAS reporting challenge. She is an ardent life long learner and reads voraciously, loves photography and volunteers at the MSPCA Boston Adoption Center walking, training and photographing dogs.

SAS® GLOBAL FORUM 2017

April 2 – 5 | Orlando, FL

Document and Enhance Your SAS® Code, Data Sets, and Catalogs with SAS Functions, Macros, and SAS Metadata

USERS PROGRAM

Paper 1314-2017

Document and Enhance Your SAS® Code, Data Sets, and Catalogs with SAS Functions, Macros, and SAS Metadata

Louise Hadden and Roberta Glass, Abt Associates Inc.

Use SAS System Macros

Information at your fingertips: when a program was last run; who ran it; what variables were created; whether the data set is sorted or indexed ; whether a program was run interactively or in batch mode and more! Some examples:

- path and program name (batch mode):
  `%SYSFUNC(GETOPTION (SYSIN));`
- path and program name (interactive):
  `%SYSGET(SAS_EXECFILEPATH);`
- date the program began: `&SASDATE`
- time the program began: `&SYSTIME`
- programmer who submitted the job: `&SYSUSERID`
- batch / interactive: `&SYSPROCESSMODE`
- condition code: `&SYSCC`

Ways to use:

- Link output and logs to program creating them
- Create variable indicating program creating a data set (or other useful information)
- Label your data sets

Tip: Use PROC DATASETS to put a label on your data sets after the fact

Introduction

By using SAS metadata in conjunction with automated documentation, you can find out when a program was last run, who ran it, what variables were created, whether the data set is sorted or indexed, and more.

You can use your metadata to write portions of your programs, and to generate codebooks.

We will give you a whirlwind tour of tools, tips and techniques to enhance your SAS programming toolkit!

There are many more SAS system macros and many other uses of %SYSFUNC – please check the SAS documentation, papers by Kathy Fraeman or Troy Martin Hughes' book "SAS Data Analytic Development: Dimensions of Software Quality" to get some other ideas.

Stored Header Macro

A simple macro can insure that every program log has a header section which contains essential information. First, save a compiled version of the macro to a macro catalog.

```
*****;
** save a compiled header macro;
*****;
OPTIONS MSTORED SASMSTORE=MYSTORE;
LIBNAME MYSTORE "C:\Sample\COMPILED_MACROS";

%macro hdr/ STORE SOURCE DES="Program Header";
      %put 0
*************************************************************************
********;
      %put 0          ** Project: Sample Project;
      %put 0          ** Program: %sysfunc(getoption(sysin));
      %put 0          ** Run by:  &sysuserid;
      %put 0          ** Run Date/time:  &sysdate - &systime ;
      %put 0
*************************************************************************
********;
%mend;
```

Insert the macro code in every program you or your team writes

```
OPTIONS MSTORED SASMSTORE=MYSTORE;
LIBNAME MYSTORE "C:\Sample\COMPILED_MACROS";
%hdr;
```

Your log will show the program header automatically.

```
0          ************************************************************
0          ** Project: Sample Project
0          ** Program:
0          ** Run by:  GlassR
0          ** Run Date/time:  27JUL15 - 15:09
0          ************************************************************
```

TIP: you can store a description of your macro when creating it using the DES= option.

Autodoc Macro: Automatically Create and Modify a Processing Log

Save an empty Microsoft Excel® file log before using the macro the first time. Begin creating a macro, specify where to store it, and describe it.

```
%MACRO autodoc(docfile) / STORE SOURCE
DES="Creates program log";
```

You can place code at the top of your macro which will prompt users to enter information before executing the rest of the code. What is entered is collected as macro variables.

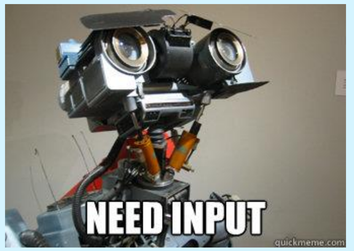
```
%window info
#4 @5 'Please enter the purpose of
this program:'
#5 @5 purpose 50 attr=underline
display=yes auto=no color = blue . . .
%display info;
%put &purpose ;
```

The macro will generate a line for each program run, using information from program prompts and system macros.

```
DATA userline;
. . .
      run_by="&sysuserid";
      run_date="&sysdate";
      run_time="&systime";
      purpose="&purpose";
```

Import the old spreadsheet, create a backup, then append the new line onto the original dataset.

```
PROC IMPORT DBMS = excel OUT = backupdocA DATAFILE = "&docfile.";
RUN;
DATA backupdoc;
      LENGTH &lengths.;
      SET backupdocA;
RUN;
PROC APPEND DATA=userline BASE=backupdoc FORCE;
RUN;
```



Create a permanent updated file, then export that information and update the existing program log spreadsheet. The file can easily be opened, additional notes added, and re-saved.

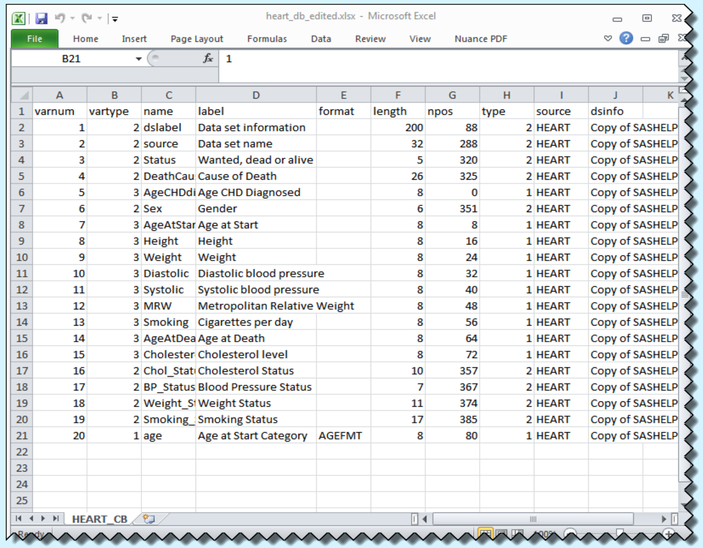
```
DATA doc.backupdoc;
   SET backupdoc;
RUN;
PROC EXPORT data= doc.backupdoc OUTFILE = "&docfile." DBMS=EXCEL LABEL REPLACE;
      SHEET="program log";
RUN;

%MEND;
```

	A	B	C	D	E	F	G	H	I
1	program_name	run_date	run_time	run_by	purpose	sas_version	system	input_files	output_files
2	docpaper_1	30JUL15	09:10	GlassR	Save a macro that creates headers for programs.	9.4	X64_ES08R2	none	macro %hdr
3	docpaper_2	30JUL15	09:12	GlassR	Add a format for treatment variable to format library.	9.4	X64_ES08R2	none	format %trtmt
4	docpaper_3	30JUL15	09:14	GlassR	Save a test dataset for SAS GLOBAL paper.	9.4	X64_ES08R2	none	dd.test
5									

Codebook Generation: Self-writing Programs

Steps for the example discussed in the paper:

Create a modified copy of SASHELP.HEART, and produce an Excel spreadsheet by exporting PROC CONTENTS output.



heart_db_edited.xlsx - Microsoft Excel

File Home Insert Page Layout Formulas Data Review View Nuance PDF

B21: 1

	A	B	C	D	E	F	G	H	I	J	K
1	varnum	vartype	name	label	format	length	npos	type	source	dsinfo	
2	1	2	dslabel	Data set information		200	88	2	HEART	Copy of SASHELP	
3	2	2	source	Data set name		32	288	2	HEART	Copy of SASHELP	
4	3	2	Status	Wanted, dead or alive		5	320	2	HEART	Copy of SASHELP	
5	4	2	DeathCau	Cause of Death		26	325	2	HEART	Copy of SASHELP	
6	5	3	AgeCHDdi	Age CHD Diagnosed		8	0	1	HEART	Copy of SASHELP	
7	6	2	Sex	Gender		6	351	2	HEART	Copy of SASHELP	
8	7	3	AgeAtStar	Age at Start		8	8	1	HEART	Copy of SASHELP	
9	8	3	Height	Height		8	16	1	HEART	Copy of SASHELP	
10	9	3	Weight	Weight		8	24	1	HEART	Copy of SASHELP	
11	10	3	Diastolic	Diastolic blood pressure		8	32	1	HEART	Copy of SASHELP	
12	11	3	Systolic	Systolic blood pressure		8	40	1	HEART	Copy of SASHELP	
13	12	3	MRW	Metropolitan Relative Weight		8	48	1	HEART	Copy of SASHELP	
14	13	3	Smoking	Cigarettes per day		8	56	1	HEART	Copy of SASHELP	
15	14	3	AgeAtDea	Age at Death		8	64	1	HEART	Copy of SASHELP	
16	15	3	Cholester	Cholesterol level		8	72	1	HEART	Copy of SASHELP	
17	16	2	Chol_Stat	Cholesterol Status		10	357	2	HEART	Copy of SASHELP	
18	17	2	BP_Status	Blood Pressure Status		7	367	2	HEART	Copy of SASHELP	
19	18	2	Weight_S	Weight Status		11	374	2	HEART	Copy of SASHELP	
20	19	2	Smoking_	Smoking Status		17	385	2	HEART	Copy of SASHELP	
21	20	1	age	Age at Start Category	AGEFMT	8	80	1	HEART	Copy of SASHELP	
22											
23											
24											
25											

HEART_CB

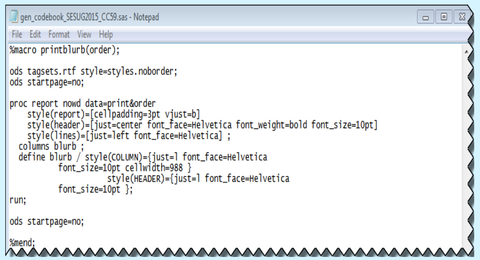
Review the documentation spreadsheet, modify (if needed), and re-import the modified spreadsheet into SAS. General concepts regarding self-writing programs:

- **SAS programs are text files**
- **SAS can create, modify and read text files**
- **SAS character functions (the fabulous feline CAT functions) make it possible to create "paragraphs"**

In our case, we write out text files using data _null_ which call different macros for different types of variables and different desired outputs (headers for variables, descriptors for variables, detail for variables.)

General concepts regarding reporting in this context:

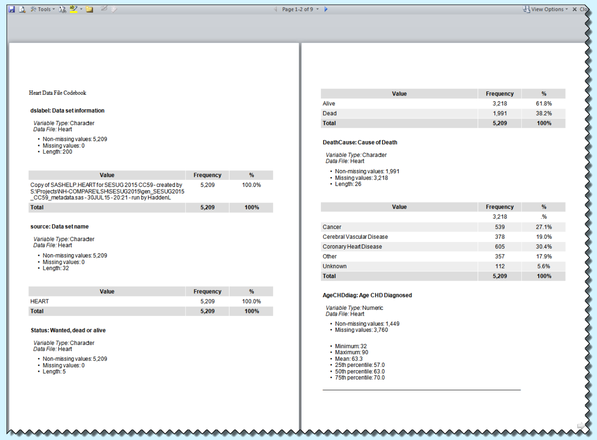
- **PROC REPORT and PROC TEMPLATE can handle SAS paragraphs (really long text fields)**
- **Create a NOBORDER style template**
- **Stack headers, descriptors and detail in a paragraph for each variable to be documented**



gen_codebook_SESUG2015_CC59.sas - Notepad

File Edit Format View Help

```
%macro printblurb(order);

ods tagsets.rtf style=styles.noborder;
ods startpage=no;

proc report nowd data=print&order
    style(report)=[cellpadding=3pt vjust=b]
    style(header)=[just=center font_face=Helvetica font_weight=bold font_size=10pt]
    style(lines)=[just=left font_face=Helvetica] ;
  columns blurb ;
  define blurb / style(COLUMN)=[just=l font_face=Helvetica
           font_size=10pt cellwidth=988 ]
                 style(HEADER)=[just=l font_face=Helvetica
           font_size=10pt ];
run;

ods startpage=no;

%mend;
```



Heart Data File Codebook

dslabel: Data set information

Variable Type: Character
Data File: Heart

- Non-missing values: 5,209
- Missing values: 0
- Length: 200

Value	Frequency	%
Copy of SASHELP.HEART for SESUG 2015 CC59 - created by S:\Projects\NH-COMPARELS\HSESUG2015\gen_SESUG2015_CC59_metadata.sas - 30JUL15 - 10:21 - run by Hadden,	5,209	100.0%
Total	5,209	100%

source: Data set name

Variable Type: Character
Data File: Heart

- Non-missing values: 5,209
- Missing values: 0
- Length: 22

Value	Frequency	%
HEART	5,209	100.0%
Total	5,209	100%

Status: Wanted, dead or alive

Variable Type: Character
Data File: Heart

- Non-missing values: 5,209
- Missing values: 0
- Length: 5

Value	Frequency	%
Alive	3,218	61.8%
Dead	1,991	38.2%
Total	5,209	100%

DeathCause: Cause of Death

Variable Type: Character
Data File: Heart

- Non-missing values: 1,991
- Missing values: 3,218
- Length: 26

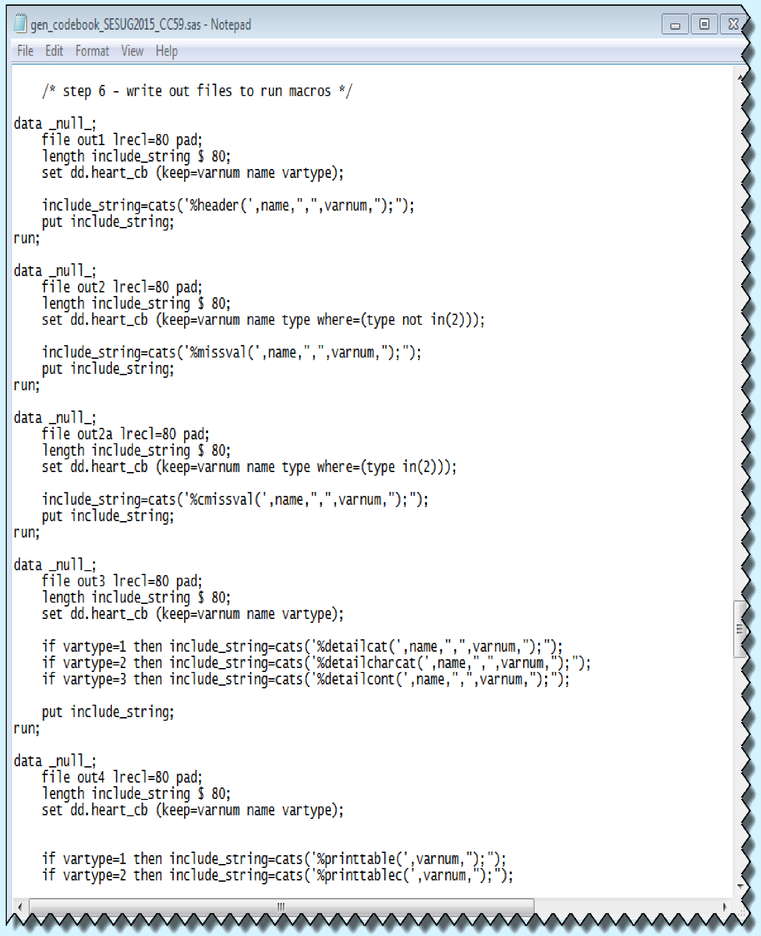
Value	Frequency	%
Cancer	539	27.1%
Cerebral Vascular Disease	378	19.0%
Coronary Heart Disease	605	30.4%
Other	357	17.9%
Unknown	112	5.6%
Total	5,209	100%

AgeCHDdiag: Age CHD Diagnosed

Variable Type: Numeric
Data File: Heart

- Non-missing values: 1,449
- Missing values: 3,760

- Minimum: 32
- Maximum: 99
- Mean: 63.3
- 25th percentile: 57.0
- 50th percentile: 63.0
- 75th percentile: 70.0



gen_codebook_SESUG2015_CC59.sas - Notepad

File Edit Format View Help

```
   /* step 6 - write out files to run macros */

data _null_;
   file out1 lrecl=80 pad;
   length include_string $ 80;
   set dd.heart_cb (keep=varnum name vartype);

   include_string=cats('%header(',name,",",varnum,");");
   put include_string;
run;

data _null_;
   file out2 lrecl=80 pad;
   length include_string $ 80;
   set dd.heart_cb (keep=varnum name type where=(type not in(2)));

   include_string=cats('%missval(',name,",",varnum,");");
   put include_string;
run;

data _null_;
   file out2a lrecl=80 pad;
   length include_string $ 80;
   set dd.heart_cb (keep=varnum name type where=(type in(2)));

   include_string=cats('%cmissval(',name,",",varnum,");");
   put include_string;
run;

data _null_;
   file out3 lrecl=80 pad;
   length include_string $ 80;
   set dd.heart_cb (keep=varnum name vartype);

   if vartype=1 then include_string=cats('%detailcat(',name,",",varnum,");");
   if vartype=2 then include_string=cats('%detailcharcat(',name,",",varnum,");");
   if vartype=3 then include_string=cats('%detailcont(',name,",",varnum,");");

   put include_string;
run;

data _null_;
   file out4 lrecl=80 pad;
   length include_string $ 80;
   set dd.heart_cb (keep=varnum name vartype);


   if vartype=1 then include_string=cats('%printtable(',varnum,");");
   if vartype=2 then include_string=cats('%printtablec(',varnum,");");
```

Other Uses of the Codebook Generation Program

Similarly, metadata can be accessed to create label, format, and length, etc. statements. The resulting statements can be included in other programs seamlessly.

gen_label_fmt_stmnt_SESUG2015_CC59.sas - Notepad

```
data runrun;
    length include_string $ 180;
    include_string=';';
run;


data temp1;
    length include_string $ 180;
    set dd.heart_cb;

    label=compress(label,'"');

    qlabel=cats('"',label,'"');
    include_string=catx(' ',name,'=',qlabel);
run;

data templabel (keep=include_string);
    file out1 lrecl=180 pad;
    length include_string $ 180;
    set runlabel temp1 runrun;
    put include_string;
run;


data temp2;
    length include_string $ 180;
    set dd.heart_cb (where=(format ne ''));
    qformat=cats(format,'.');
    include_string=catx(' ',name,qformat);
run;

data tempfmt (keep=include_string);
    file out2 lrecl=180 pad;
    length include_string $ 180;
    set runformat temp2 runrun;
    put include_string;
run;
```

heart_labelstm.txt - Notepad

```
*** Label Statement for heart;
LABEL
dslabel = "Data set information"
source = "Data set name"
Status = "Wanted, dead or alive"
DeathCause = "Cause of Death"
AgeCHDdiag = "Age CHD Diagnosed"
Sex = "Gender"
AgeAtStart = "Age at Start"
Height = "Height"
Weight = "Weight"
Diastolic = "Diastolic blood pressure"
Systolic = "Systolic blood pressure"
MRW = "Metropolitan Relative Weight"
Smoking = "Cigarettes per day"
AgeAtDeath = "Age at Death"
Cholesterol = "Cholesterol level"
Chol_Status = "Cholesterol Status"
BP_Status = "Blood Pressure Status"
Weight_Status = "Weight Status"
Smoking_Status = "Smoking Status"
age = "Age at Start Category"
;
```

Conclusion

Use SAS metadata, system macros and processes to:

- Automatically generate a program header
- Keep a processing log up-to-date
- Label your data sets, variables, and catalog entries
- Identify the code that created datasets, .logs, .lsts, and tables
- Generate components of your SAS programs without typing a word - and more!
- Create user-friendly documentation
- And more!

Contact us!

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

Roberta Glass
Roberta_Glass@abtassoc.com
Louise Hadden
Louise_Hadden@abtassoc.com

Scan me! Sample code available:

Documenting SAS® Catalogs

General concepts:

Save format libraries with specific, two-level names:

```
LIBNAME Fmtlib "C:\...\Fmtlib";
PROC FORMAT LIBRARY = fmtlib.test;
    VALUE $trtmt 'C' = "Control"
    'T' = "Treatment" Other = "Error";
RUN;
```

To use a saved “named” format, specify the format library in the options statement.

```
OPTIONS MSTORED FMTSEARCH =
(Fmtlib.test);
```

Descriptions for formats can be added using a PROC CATALOG MODIFY statement.

Descriptions for macros (and other catalogs other than formats) can be added using the DES option while creating the catalog entry:

```
%MACRO autodoc(docfile)/ STORE SOURCE
DES="Creates program log";
```

Similarly, you can add descriptions to macro (or graphic, or other) catalogs with a PROC CATALOG MODIFY statement.

Adding these descriptions can save you a lot of time at the end of a project (and during it).

SAS® GLOBAL FORUM 2017

April 2 – 5 | Orlando, FL