Paper 163-29

PROC DOC and Beyond: Is PROC CONTENTS Enough?

Louise Hadden, Abt Associates Inc., Cambridge, MA

ABSTRACT

This paper will present a macro driven solution for the complete documentation of SAS® data files, beyond PROC CONTENTS. This documentation process uses both SAS procedures and Microsoft Office products to produce a detailed document that provides information on variables and variable values in a database. The resulting document (and process) is invaluable for both documentation purposes and quality assurance. The example shown is for a highly complex survey database with thousands of variables and user-written formats. While specific to survey data, these techniques are also useful for the documentation of other types of SAS databases.

INTRODUCTION

SAS programmers are frequently called upon to analyze survey data, or data collected via hard copy surveys, CATI (computer assisted telephone interviews), or web-based surveys. While some large scale national surveys such as the national Census are very effectively documented, it is often the case that survey data bases are things of mystery. Once the survey data has been transferred from the data collection organization or organism, it then falls to the SAS programmer to make sense of the survey data. SAS provides a convenient set of tools with which the SAS programmer can evaluate, clean, and thoroughly document any survey database. The consequences of failing to appropriately investigate and document survey data can be disastrous in terms of inaccurate analysis and reporting.

START AT THE BEGINNING

The truth is that there are no easy answers to the documentation quandary. It takes hard, painstaking work! The payoff is great, though, in terms of clean, accurate, well-documented survey databases.

Many survey databases are transferred to programmers complete with pre-written input statements, label statements and user-defined formats. In most cases, there will be separate "flat" data files, input statements customized for various statistical packages, variable label statements, and formats or value labels. This is vastly preferable to receiving your survey data as a pre-constructed SAS database, as it allows you to thoroughly investigate the data file(s). For example, errors in a user-defined format may not be obvious from a frequency run on pre-formatted variables. Similarly, variable labels generated from a CATI (Computer Assisted Telephone Interview) survey system may not be adequate or clear for documentation purposes.

If the input statement, label statement and user-defined formats have NOT been provided, it is the SAS programmer's job to do so. This should be done with care, consulting all versions of the survey instrument and any file layouts provided. The better the input program, the better the documentation (and not so incidentally, the cleaner the data!) Input programs, whether provided or self-written, should be adequately commented and documented themselves. Whether creating a new version of existing file(s) or creating a SAS data set from external data, the SAS System provides many ways to include information about data sets and variables. The use of the label data set option allows the SAS programmer to specify useful information about the data set(s) such as the program(s) used to create or modify the data set(s). The religious use of label and format statements provides additional information to SAS programmers. Last but not least, commenting code consistently is the single best way to document processes. As well as ensuring readability and reproducible results, commenting code adds an extra measure of thought toward checking logic in complex operations.

The first step in documenting a survey database that is not adequately and exquisitely documented (such as nationally recognized surveys as the Census or NHANES III) when the various input, label and format statements HAVE been provided is to review your input statement and user-defined formats against a hard copy of the survey instrument. This may be a CATI questionnaire or screen shots of a web based survey, as well as an actual paper document. In cases where both a paper version and computerized version exist, it is important to review all the versions as discrepancies may exist and affect the cleanliness and accuracy of the data elements. Vendor-provided statements should be incorporated into a well-documented program as described above.

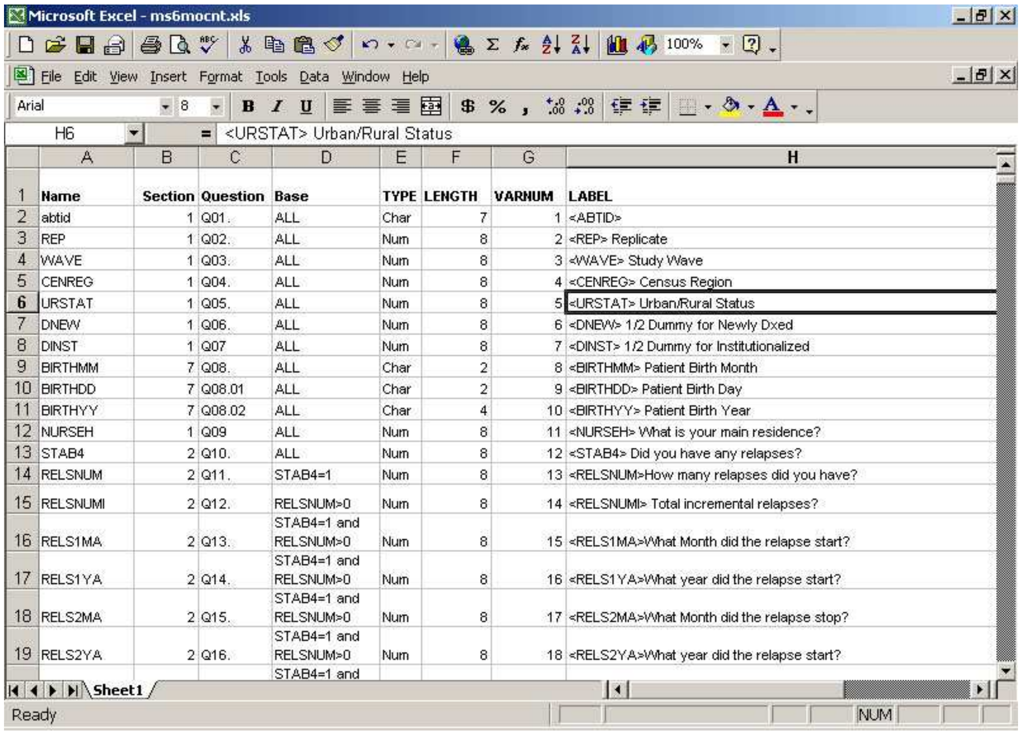
Once this review has occurred, and any discrepancies analyzed and corrected, the data file should be read in first using system options OBS=0 and ERRORABEND for testing the input statement. The log should be carefully reviewed, and then the test program should be re-run with a sample of observations to catch any remaining errors. Once this is accomplished, the file should be read in as a permanent data set WITHOUT user-defined formats and the log carefully reviewed. In addition, the file should be read in WITH user-defined formats and saved as a separate data file.

PROC CONTENTS

The ubiquitous PROC CONTENTS can do amazing things. Most SAS programmers are familiar with the procedure and use it extensively in its usual form, but fewer output the resulting data set and make use of it. Data items from the output PROC CONTENTS data set that we will be utilizing include format, label, length, name, npos, type, and varnum. The most information will be gained from the data set that is labeled, formatted and documented prior to running PROC CONTENTS. As can be seen in the example presented below (run on the formatted version of the data set), we will also add some variables to our output data set that will be saved as a permanent data set. The output data set can be stored as a SAS data set, output (printed) in HTML, output as a comma delimited (CSV) file, or output directly to Microsoft Excel via DDE or OBDC. The intent is to create an Microsoft Excel spreadsheet.

```
options ps=52 ls=135 nocenter mprint obs=max;
libname dd 'd:\yourdata';
filename odsout 'd:\yourdata';
run;

title1 'PROC DOC';
footnote "Program:  procdoc.sas - Last Run &sysdate.";
run;

proc contents data=dd.ms6mo_v1 out=dd.ms6mocnt;
run;

data dd.ms6mocnt;
        length base flags $ 50 section $ 2 question $ 5;
        set dd.ms6mocnt (keep=name format label length name npos type varnum);

        if format ne ‘’ then proctype=1;
        else proctype=2;

        label proctype=’Procedure Type (FREQ/UNIV)’
                base=’Base of Question’
                flags=’Comments’
                section=’Section of Questionnaire’
                question=’Question Number’;
run;

ods html body='w6mocont.htm' path=odsout style=styles.sasweb;

proc print data=dd.ms6mocnt;
title2 'Contents of MS Weighted Six Month Survey Data Base';
run;

ods html close;

endsas;
```

The resulting HTML file can then be read directly into MS Excel. Once saved as a workbook, the file can then be edited to incorporate some important information such as the section of the questionnaire, correct misleading labels, etc. The survey database being analyzed in the example is extremely complex, with many “multiple answer” variables and convoluted skip patterns. We added variables for question numbers and base to document this complexity. In addition, we added a procedure type flag to identify whether we wanted to run a frequency or univariate procedure on a variable.


Microsoft Excel - ms6mocnt.xls

H6 = <URSTAT> Urban/Rural Status

Name	Section	Question	Base	TYPE	LENGTH	VARNUM	LABEL
abtid	1	Q01.	ALL	Char	7	1	<ABTID>
REP	1	Q02.	ALL	Num	8	2	<REP> Replicate
WAVE	1	Q03.	ALL	Num	8	3	<WAVE> Study Wave
CENREG	1	Q04.	ALL	Num	8	4	<CENREG> Census Region
URSTAT	1	Q05.	ALL	Num	8	5	<URSTAT> Urban/Rural Status
DNEW	1	Q06.	ALL	Num	8	6	<DNEW> 1/2 Dummy for Newly Dxed
DINST	1	Q07	ALL	Num	8	7	<DINST> 1/2 Dummy for Institutionalized
BIRTHMM	7	Q08.	ALL	Char	2	8	<BIRTHMM> Patient Birth Month
BIRTHDD	7	Q08.01	ALL	Char	2	9	<BIRTHDD> Patient Birth Day
BIRTHYY	7	Q08.02	ALL	Char	4	10	<BIRTHYY> Patient Birth Year
NURSEH	1	Q09	ALL	Num	8	11	<NURSEH> What is your main residence?
STAB4	2	Q10.	ALL	Num	8	12	<STAB4> Did you have any relapses?
RELSNUM	2	Q11.	STAB4=1	Num	8	13	<RELSNUM>How many relapses did you have?
RELSNUMI	2	Q12.	RELSNUM>0	Num	8	14	<RELSNUMI> Total incremental relapses?
RELS1MA	2	Q13.	STAB4=1 and RELSNUM>0	Num	8	15	<RELS1MA>What Month did the relapse start?
RELS1YA	2	Q14.	STAB4=1 and RELSNUM>0	Num	8	16	<RELS1YA>What year did the relapse start?
RELS2MA	2	Q15.	STAB4=1 and RELSNUM>0	Num	8	17	<RELS2MA>What Month did the relapse stop?
RELS2YA	2	Q16.	STAB4=1 and RELSNUM>0	Num	8	18	<RELS2YA>What year did the relapse start?
			STAB4=1 and				

NEXT STEPS

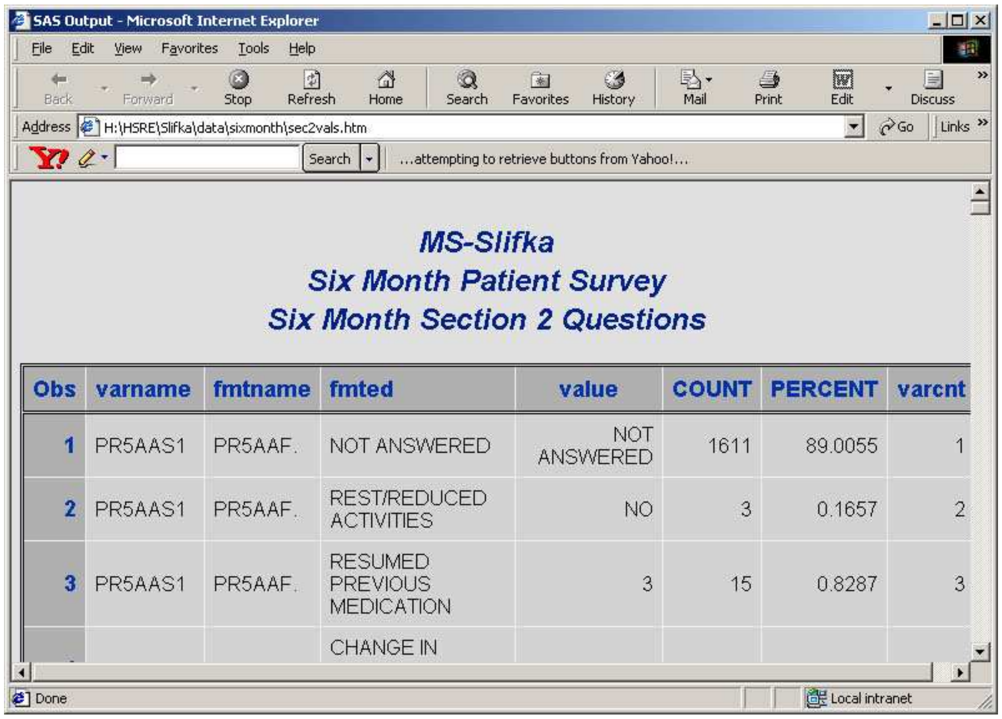
The process of editing the spreadsheet may identify some input errors, etc. Identifying the bases of the questions is also a time-consuming process (not to mention the data entry!) Once any issues are resolved and the spreadsheet is finalized, the next step is to convert the final MS Excel file back into SAS via DBMSCOPY or DDE/OBDC (Note that variable labels for the contents data set will have to be redone). This file can be used both to generate code for frequency and univariate procedures on variables and to generate the final documentation product. Due to the size and complexity of the survey database being analyzed in the example, we broke the procedural runs into sections. The intention of the procedural runs is to completely document each variable. Macros were written to perform each type of procedural run. Examples follow below.

```
%macro ffreqout(runnum,varnm,varfmt);

data anal (rename=(&varnm=value));
       set dd.ms6mo_v1_f (keep=&varnm);
run;

proc freq data=anal;
       tables value / missing noprint out=temp;
run;

data temp&runnum;
       length varname fmtname $ 8 fmted $ 40;
       set temp;
       varname="&&varnm.";
       fmted=put(value,&varfmt);
       fmtname="&&varfmt.";
run;

proc datasets library=work;
       delete anal temp;
run;

%mend ffreqout;

%macro meanout(runnum,varnm);

proc means data=anal noprint;
       var &varnm;
       output out=temp n=n nmiss=nmiss mean=mean median=median
            min=min max=max stddev=stddev;
run;
```

```
data _n(keep=varname n)
        _nmiss(keep=varname nmiss)
        _mean(keep=varname mean)
        _median(keep=varname median)
        _min(keep=varname min)
        _max(keep=varname max)
        _stddev(keep=varname stddev)
        ;
        length varname $ 8;
        set temp;
        varname="&&varnm.";

run;

data temp&runnum (keep=varname fmtname value);
        length fmtname $ 8;
        set _n(in=a)
        _nmiss(in=b)
        _mean(in=c)
        _median(in=d)
        _min(in=e)
        _max(in=f)
        _stddev(in=g)
        ;

        if a then do;
                fmtname='N';
                value=n;
                end;
        else if b then do;
                fmtname='NMISS';
                value=nmiss;
        end;
        else if c then do;
                fmtname='MEAN';
                value=mean;
        end;
        else if d then do;
                fmtname='MEDIAN';
                value=median;
        end;
        else if e then do;
                fmtname='MIN';
                value=min;
        end;
        else if f then do;
                fmtname='MAX';
                value=max;
        end;
        else if g then do;
                fmtname='STDDEV';
                value=stddev;
        end;

run;

proc datasets library=work;
        delete temp anal _n _nmiss _mean _median _min _max _stddev;
run;

%mend meanout;

%ffreqout (0001,STAB4,STAB4F.);
%meanout (0002,RELSNUM);
```

The macros were invoked for each variable in order (the contents data set can be used to generate the macro calls, but in the interest of conserving space I do not demonstrate this here.) Since we had literally thousands of variables to document, we also used proc datasets to delete temporary files, which are set together to create a permanent data set. Results of a test print on this permanent data set follow below. As can be seen, variables include variable name, the format name, formatted values, actual values, counts, percents, etc. These data items can and should be used to further review the accuracy of value labels, etc. as well as to produce a codebook.

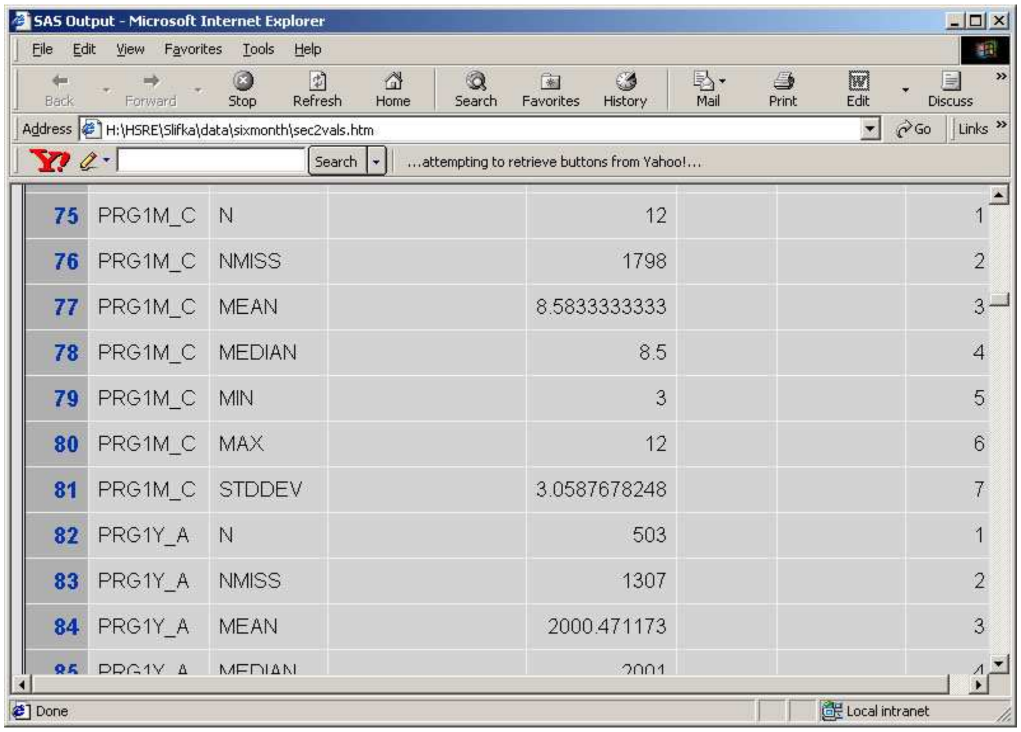
Example of %ffreqout output:


SAS Output - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address H:\HSRE\Slifka\data\sixmonth\sec2vals.htm
Go Links
Search ...attempting to retrieve buttons from Yahoo!...
Done
Local intranet

MS-Slifka
Six Month Patient Survey
Six Month Section 2 Questions

Obs	varname	fmtname	fmted	value	COUNT	PERCENT	varcnt
1	PR5AAS1	PR5AAF.	NOT ANSWERED	NOT ANSWERED	1611	89.0055	1
2	PR5AAS1	PR5AAF.	REST/REDUCED ACTIVITIES	NO	3	0.1657	2
3	PR5AAS1	PR5AAF.	RESUMED PREVIOUS MEDICATION	3	15	0.8287	3
			CHANGE IN				

Example of %meanout output:


SAS Output - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address H:\HSRE\Slifka\data\sixmonth\sec2vals.htm
Go Links
Search ...attempting to retrieve buttons from Yahoo!...
Done
Local intranet

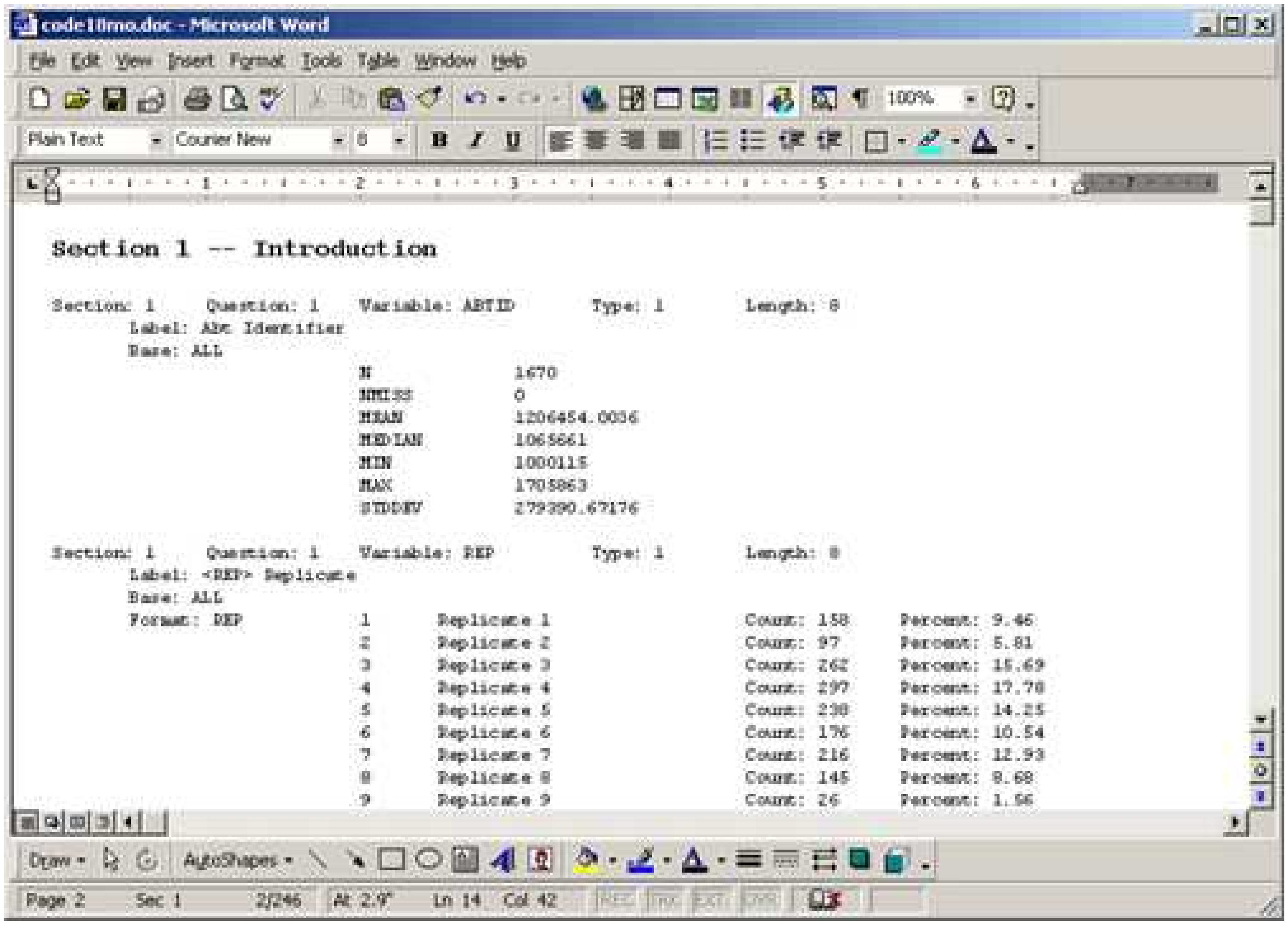
75	PRG1M_C	N		12			1
76	PRG1M_C	NMISS		1798			2
77	PRG1M_C	MEAN		8.5833333333			3
78	PRG1M_C	MEDIAN		8.5			4
79	PRG1M_C	MIN		3			5
80	PRG1M_C	MAX		12			6
81	PRG1M_C	STDDEV		3.0587678248			7
82	PRG1Y_A	N		503			1
83	PRG1Y_A	NMISS		1307			2
84	PRG1Y_A	MEAN		2000.471173			3
85	PRG1Y_A	MEDIAN		2001			4

THE FINAL PRODUCT

The combination of the output of the procedural macros and the contents procedure, as detailed below, generate a fully descriptive codebook documenting our survey database. We used data_null_ and put statements for greater flexibility, but could easily have taken advantage of the power of the output delivery system to create a RTF, PDF or HTML file that could be bookmarked by section. Examples of code to output to ODS will be available from the author during the "Meet the Presenters" session.

```
data temp1;
    length valname $ 8 sec_desc $ 40;
    set dd.ms18desc (where=(varname ne 'INPUT'));
    by section varnum varname varcnt;

    file 'code18mo.txt' lrecl=132 n=4;

     if section=0 then sec_desc='Derived';
     if section=1 then sec_desc='Introduction';
     if section=2 then sec_desc='Clinical Status';
     if section=3 then sec_desc='Logbook';
     if section=4 then sec_desc='Health Care Utilization';
     if section=5 then sec_desc='Home Visits';
     if section=6 then sec_desc='Disability Days';
     if section=7 then sec_desc='Purchases';

      format1=fmtname;
      if fmtname in('N','NMISS','MEDIAN','MIN','MAX','MEAN','STDDEV') then
         do;
         valname=fmtname;
         fmtname='';
      end;
      if valname ne '' then meanflag=1;
      else meanflag=0;
      if meanflag=0 and value=. and fmted='' then fmted='Not Applicable';

   format fmted $30.;

   if first.section then put @1 "Section " section "--"
                             @14 trim(sec_desc);

   if varcnt = 0 then
            put /
            @1  "Section: " section
                "@Question: " question
                "@Variable: " varname
                "@Type: " Type
                "@Length: " Length /
            @1  "@Label: " Label /
            @1  "@Base: " Base;

         else if varcnt eq 1 and meanflag=0 then
              put @1  "@Format: " format1
                  @25 "@" value
                      "@" fmted
                  @60 "@Count: " COUNT
                      "@Percent: " PERCENT ;
                format PERCENT 7.2;
         else if varcnt not in(0,1) and meanflag=0 then
              put @25 "@" value
                      "@" fmted
                  @60 "@Count: " COUNT
                      "@Percent: " PERCENT ;
                format PERCENT 7.2;
         else if varcnt ne 0 and meanflag=1 then
              put @25 "@" valname
                  @40 "@" value;

run;
```

The text file resulting from the data step described above was read into MS Word to create a fully functional codebook with minimal post-processing. A sample section is shown below.


code18mo.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Plain Text
Courier New
8
100%
Draw
AutoShapes
Page 2 Sec 1 2/246 At 2.9" Ln 14 Col 42

```
Section 1 -- Introduction

Section: 1      Question: 1     Variable: ABTID        Type: 1        Length: 8
          Label: Abt Identifier
          Base: ALL
                                N                   1670
                                NMISS               0
                                MEAN                1206454.0036
                                MEDIAN              1065661
                                MIN                 1000115
                                MAX                 1705863
                                STDDEV              279390.67176

Section: 1      Question: 1     Variable: REP          Type: 1        Length: 8
          Label: <REP> Replicate
          Base: ALL
          Format: REP           1          Replicate 1                Count: 158     Percent: 9.46
                                2          Replicate 2                Count: 97      Percent: 5.81
                                3          Replicate 3                Count: 262     Percent: 15.69
                                4          Replicate 4                Count: 297     Percent: 17.78
                                5          Replicate 5                Count: 238     Percent: 14.25
                                6          Replicate 6                Count: 176     Percent: 10.54
                                7          Replicate 7                Count: 216     Percent: 12.93
                                8          Replicate 8                Count: 145     Percent: 8.68
                                9          Replicate 9                Count: 26      Percent: 1.56
```

CONCLUSION

The SAS system provides numerous opportunities for creating self-documenting data sets. With care at the onset of a project, SAS programmers can utilize the power of the SAS system to ensure quality data and accurate documentation. The products of base SAS procedures can help identify data and programmer errors, as well documenting databases. While the example shown utilized Microsoft Office products in tandem with SAS procedures, the entire process could have been performed in SAS using the full screen editor in place of Microsoft Excel and outputting the codebook file directly from SAS using the Output Delivery System. Simple documentation is not enough to ensure quality data and analyses. SAS can show us the way.

Full code samples are available via email from the author (contact information listed below.)

REFERENCES

SAS Online Documentation (PC SAS V8.1, AIX UNIX SAS V8.2)

ACKNOWLEDGMENTS

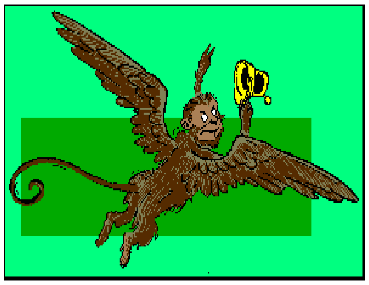
The author wishes to thank Kathy Hill, formerly of Abt Associates Inc., and Jessica Johnston of Abt Associates Inc. for their diligent work on the actualization of this complete documentation product.

CONTACT INFORMATION

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

Louise Hadden
Abt Associates Inc.
55 Wheeler St.
Cambridge, MA 02138
Work Phone: 617-349-2385
Fax: 617-349-2675

Email: louise_hadden@abtassoc.com



KEYWORDS

SAS, PROC CONTENTS, PROC FREQ, PROC UNIVARIATE, WEIGHTS, EXCEL, WORD, DOCUMENTATION, DATA QUALITY