Paper 084-31

PROC FORMAT: An Analyst's Buddy

Ben Kupronis, Centers for Disease Control and Prevention, Atlanta, GA

ABSTRACT

PROC FORMAT provides essential tools for simplified coding and program efficiency that are often overlooked by novice users. This paper presents basic formatting techniques using the VALUE statement and useful permanent formats. More advanced techniques are presented such as the utility of picture formats and conversion of databases to formats. All these tips avoid re-coding data in the DATA step and create catalogs, which can streamline workflow.

INTRODUCTION

PROC FORMAT is an often overlooked procedure for manipulating data during analysis. Sure it lacks the power of the DATA step or the utility of other procedures such as FREQ, TABULATE, or REPORT. But what it lacks in sex appeal it more than makes up in utility. This paper will describe specific ways PROC FORMAT can make an analyst's job easier as well as provide some of my favorite tricks.

THE BASICS

SAS novices think output from a procedure has to be the literal values that are stored in the dataset. However, PROC FORMAT can change the ambiguous gender of 0 and 1 into male and female.

```
data one ;
    do j=1 to 100;
       sex=round(ranuni(0));
       sex2=sex;
       output;
    end;
run;

proc format;
    value sexfmt 1="Female" 0="Male";
run;

proc tabulate data=one;
    class sex sex2;
    tables sex="Original Gender Value",sex2="Formatted Gender Value";
    format sex2 sexfmt.;
run;
```

Output:

	Formatted Gender Value	
	Male	Female
	N	N
Original Gender Value	45	.
0		
1	.	55

SAS likes the format type to match the type of variable (character or numeric). The last example used a numeric format. We could have made the format character by simply naming it $sexfmt. to match a character sex variable.

Formats are efficient. Smaller size variables take up less space in a dataset and will cause a 4 million observation dataset to run substantially faster. Take the example of ICD-9 codes (International Classification of Diseases, 9th revision) which are used in medicine to code diseases. We could store "SEPTICEMIA DUE TO OTHER GRAM-NEGATIVE ORGANISMS" or we could store 0384 or 38.4 depending on whether we like character or numeric values. The long character value had a length of 50, but the coded character value had a length of 5 which will be more efficient. It also has the benefit of being infinitely easier to code if you are writing a WHERE expression.

Another very common use for PROC FORMAT is to collapse data without having to recode the data. Systolic blood pressure is a continuous value. You could recode the data as:

```
data bp;
    set bp;
    length Interpret $21;
    if bp < 120 then Interpret="Normal";
    else if 120 <= bp < 140 then Interpret="Prehypertension";
    else if 140 <= bp < 160 then Interpret="Stage 1 Hypertension";
    else  Interpret="Stage 2 Hypertension";
run;

proc tabulate data=bp;
    class Interpret;
    tables Interpret="" all="Total",(N pctn)/box="Interpretation";
run;
```

Output:

Interpretation	N	PctN
Normal	106	21.20
Prehypertension	189	37.80
Stage 1 Hypertension	169	33.80
Stage 2 Hypertension	36	7.20
Total	500	100.00

The same exact output can be generated using a format. Note less than symbol and keywords in ranges. The less than symbol can appear on either side of the dash to exclude the value listed.

```
Proc format;
    value BPInterp low-<120="Normal"
       120-<140="Prehypertension"
       140-<160= "Stage 1 Hypertension"
       160-high="Stage 2 Hypertension"
    ;
run;

proc tabulate data=bp ;
    class bp;
    tables bp="" all="Total",(N pctn)/box="Interpretation";
    format bp BPInterp.;
run;
```

To illustrate the efficiency of PROC FORMAT, I created a dataset with half a million observations that is over a gigabyte in size. My computer took 3 min 51 sec to recode the data then another 52 seconds to tabulate. This is a grand total of 4 min 43 sec of processing time for the inefficient code. PROC FORMAT used less than a second and the PROC TABULATE using the format took 27 seconds to run. The efficient code is more than 10 times faster in

this example. Formats also keep the dataset width to a minimum which also speeds up all SAS processes, including those that do not use formats because SAS has to read through less data at every step.

Date formats are another great time saver. All the formats that have been used so far are called user defined formats. SAS also makes our life easier by providing pre-defined formats. No PROC FORMAT is required. Here are some of my favorite pre-defined formats.

Category	**Format**	**Description**
Character	$CHAR*w*. / $*w*.	Writes character data with a length of *w*
Date and Time	DATE*w*.	SAS date values, DATE9. = 26MAR2006
	DATETIME*w.d*	SAS datetime values, DATETIME9.= 26MAR2006 DATETIME20.1 = 26MAR2006:03:49:19.0
	DTMONYY*w*.	SAS datetime values, extracts month and year DTMONYY7. = OCT2006
	DTYYQC*w*.	SAS datetime values, extracts year and quarter DTYYQC6. =2006:1
	MMDDYY*w*.	SAS date values, MMDDYY10. = 03/26/2006
	MONYY*w*.	SAS date values, MONYY7. = MAR2006
	WEEKU*w*.	SAS date values, WEEKU5. = 06W13
	YEAR*w*.	SAS date values, YEAR4. = 2006
	YYMMDD*w*.	SAS date values, YYMMDD10. = 2006-03-26
	YYQ*w*.	SAS date values, YYQ6. = 2006Q1
Numeric	BEST*w*.	SAS chooses the best notation
	COMMA*w.d*	COMMA10.2 = 16,886.00
	SSN*w*.	SSN. = 123-45-6789
	w.d	For value 3.26, format 6.1 = 3.3
	Z*w.d*	For value 3.26, Z6.1 = 0003.3

As an example of efficiency and ease, consider a quarterly report. You could use the DATA step to code each quarter from the date and then run a PROC FREQ or just use a format in a single step with PROC FREQ.

```
data qtrdata;
    set sasuser.sale2000;
    /*a savvy programmer would use the following*/
    quarter= put(date,yyq6.);
    /*a novice programmer might use the following*/
    quarter2=compress(year(date)||"Q"||qtr(date));
run;

proc freq data=qtrdata;
    tables quarter quarter2 /nocum;
run;

/*Most efficient method*/
proc freq data=sasuser.sale2000;
    tables date /nocum;
    format date yyq6.;
run;
```

The most efficient method was more than 10 times faster again.

Is the boss fickle and want the report by month after you sweated for hours coding? Formats enable you to restructure the data by simply changing the format. The data does not have to be recoded. Simply change the format from `yyq6.` to `monyy7.` and rerun.

```
proc freq data=sasuser.sale2000;
    tables date /nocum;
    format date monyy7.;
run;
```

The formats above were applied to SAS date values. The same formats are available for datetime values. You could even report the results by week if you needed.

Output: By Quarter

Date	Frequency	Percent
2000Q1	26	16.67
2000Q2	26	16.67
2000Q3	26	16.67
2000Q4	78	50.00

Output: By Month

Date	Frequency	Percent
JAN2000	9	5.77
FEB2000	8	5.13
MAR2000	9	5.77
APR2000	9	5.77
MAY2000	9	5.77
JUN2000	8	5.13
JUL2000	9	5.77
AUG2000	9	5.77
SEP2000	8	5.13
OCT2000	9	5.77
NOV2000	9	5.77
DEC2000	60	38.46

ADVANCED TOPICS

A PICTURE format can allow you to avoid the DATA step when you calculate simple rates. The multiplier option calculates the rate and the string of zeros provides the placeholders for the number to go into. It is very important to use the ROUND option or the number will be truncated. For instance the 5689 and 10897 numbers below would be truncated to 5.6 and 10.8 instead of the appropriate 5.7 and 10.9 that you would expect.

```
Data rate;
    input incidence;
    cards;
        1000
        5689
        10897
    ;
run;

proc format;
    picture per_thou (round) low-high='0,000.0'
              ( mult=.01 );
run;

proc print data=rate label;
    var incidence;
    format incidence per_thou.;
    label incidence="Rate per 1,000";
run;
```

Output:

Obs	Rate per 1,000
1	1.0
2	5.7
3	10.9

The PICTURE format is also good for adding a suffix to the end of values such as what you would do for hard drive storage units.

```
Data hd;
    input space;
    cards;
                  89
                1000
                5689
             1606897
        12345678901
    ;
run;


proc format;
    picture sto  low-999='0,000.0' Bytes
       1000-   999999='0,000.0' KB  ( mult=.01 )
       1000000-999999999='0,000.0' MB  (
mult=.00001 )
       10000000000-high='0,000.0' GB  (
mult=.00000001 )
    ;
run;

proc print data=hd label ;
    var space;
    format space sto.;
    label space="Storage Available";
run;
```

Output:

Obs	Storage Available
1	89.0 Bytes
2	1.0 KB
3	5.6 KB
4	1.6 MB
5	12.3 GB

Formats also make code maintenance easier. Rather than recoding a variable 3 or 4 times to aggregate the data differently for multiple reports, the original coding can be maintained. This reduces database size and increases the efficiency of all procedures and DATA steps. Since the format is compiled once and then available for the rest of the session, all formats can be located in a single place for easier maintenance. A format at the beginning of a program

makes minor tweaking of the program possible without delving into the multiple joins and manipulation of data that often occurs during SAS programming.

If you run a PROC FORMAT like my previous ICD-9 example which has over 15,000 values every time you ran your program, you will not experience the efficiency gains that you thought you could get. The solution is to use a format catalog to pre-compile the format so it is ready every time you start SAS. You can either type in the PROC FORMAT like I have shown before or read in a database. With large formats that are updated often, the database method is the only way to go.

Generally only 3 variables (Start, Label, and Fmtname) are needed. A 4th variable (End) is required to specify ranges of values.

Variable	Description
Start	The value to be formatted or the beginning of the range
End	The end of the range or blank if there is no range
Label	The final formatted value
Fmtname	The name of the format. Preface character formats with $

This example code will turn a MS Access database into a format catalog. There are two format tables which are concatenated into a single SAS database and then they are converted to a SAS format catalog.

```
libname icd_MS access 'c:\Documentation\Coding\ICD-9\ICD9 2003AUG.MDB';
libname icd9  'c:\Documentation\Coding\ICD-9\sasfmt';

data icd9diag;
   set icd_MS."icd9 diag"N; /*2 word table referenced with a named literal*/
   rename code=start desc=label;
   fmtname='$DIAG';
run;

data icd9proc;
   set icd_MS.icd9proc;
   rename code=start desc=label;
   fmtname='$PROC';
run;

data all_codes;
   set icd9diag icd9proc;
run;

/*A format catalog is created in the ICD directory*/
proc format library=icd9 cntlin=all_codes;
run;
```

Use the FMTSEARCH option to specify where the catalog you created resides. The order of the directories is important because SAS will look for the format name you are using in each of the libraries in the order you specify. Work is the directory where all the temporary formats have been stored until now. In the example below, the format $DIAG. could be in both the icd9 directory and work directory. SAS would use the $DIAG. from the icd9 directory since it is specified first.

```
libname icd9 'c:\Documentation\Coding\ICD-9\sasfmt' ACCESS=READONLY;
options  fmtsearch=(icd9 work);

proc freq data=diseases;
    title 'Frequency of ICD-9 Disease Code Description';
    tables icd9diagnosis;
    format icd9diagnosis $DIAG.;
run;
```

If you peruse the documentation, you will find that the values in a format catalog can be output to regular SAS datasets. An easy way to look into a temporary or permanent format catalog is the following code which prints all the values out.

```
/*Print format values*/
proc format library=icd9 fmtlib;
run;
```

Output: (Abbreviated)

FORMAT NAME: $DIAG LENGTH: 50 NUMBER OF VALUES:15651
MIN LENGTH: 1 MAX LENGTH: 50 DEFAULT LENGTH 50 FUZZ: 0

START	END	LABEL (VER. V7|V8 29MAR2004:14:34:31)
001	001	CHOLERA
0010	0010	CHOLERA DUE TO VIBRIO CHOLERAE
0011	0011	CHOLERA DUE TO VIBRIO CHOLERAE EL TOR
0019	0019	CHOLERA UNSPECIFIED
002	002	TYPHOID AND PARATYPHOID FEVERS
0020	0020	TYPHOID FEVER

CONCLUSION

In short, formats can save time and grief during the coding process and during the executions of programs.

REFERENCES

SAS Institute Inc. 2004. *SAS® 9.1.3 Language Reference: Dictionary, Volumes 1 and 2.* Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

Ben Kupronis
Centers for Disease Control and Prevention
Division of Healthcare Quality Promotion
1600 Clifton Road N.E.
Mailstop A-24
Atlanta, GA 30333
E-mail: BBK3@CDC.GOV
Web: WWW.CDC.GOV