243-2012

My Friend the SAS® Format

Andrew H. Karp, Sierra Information Services, Sonoma, CA USA

ABSTRACT

The SAS System's FORMAT facilities provide a diverse and powerful array of tools to work with and present your data. By supplying a range of resources to control how the values of data set variables are portrayed in your SAS output (i.e., your reports and analyses), SAS Format tools offer something for all SAS users, at all experience levels, and across the wide range of ways that the SAS System is used to report, manage, analyze and display data. Applying these capabilities to your SAS data sets, as well as the reports/analyses you generate from your data, enhance the visual appeal of your output, reduce ambiguity about the meaning of terms/definitions of values of variables, reduce coding and (depending on the size of the data set) processing time, and generally make your life easier as a SAS Software user. This paper starts with the basics and then discusses many ways you can take advantage of the many ways SAS Formats can be applied to your data, reports and analyses.

INTRODUCTION

A Format contains instructions the SAS System uses to "externally represent" the values of variables (i.e., data values stored in the columns) in a SAS data set. The term "external representation" refers to how the value is portrayed in your SAS output (e.g. a report or table) versus the how the values are stored "internally," or "inside" the data set. You can apply a wide range of formats that are supplied with your version of SAS System Software. These are often called "SAS-supplied" or "built-in" Formats. You can also create custom Formats by using PROC FORMAT, a BASE SAS Software Procedure, and which is discussed later on in this paper. Formats are stored in Format Catalogs. By default, the Formats you create are stored in a temporary Catalog in the WORK Library, or you can save them in Permanent Formats Catalogs, which obviates the need to re-create them in every SAS Session in which you want to apply them. Permanent Formats Catalogs can be shared among multiple users, which provide additional features when groups of SAS users in an organization need a common set of customized formats to apply to their work. In addition to creating custom Formats (and Informats, see below), PROC FORMAT contains other tools that assist in the creation and/or management of entries in your Format Libraries.

GETTING STARTED WITH SAS FORMATS

A Format is used to change how the values of variables are displayed/portrayed in your SAS output. A more formal term for this process is "altering the external representation of the values of variables in a SAS data set." What this means is that a Format contains the rules that SAS will follow to display the variable's values in your output *without changing the values of the variables themselves*.

To fix ideas, let's consider what happens when a date value needs to be displayed in a report or table generated by a SAS procedure. For example, SAS Date Value 16537 is the number of days which have elapsed between January 1, 1960, and April 11, 2005. There are many good reasons why the SAS System stores dates as integers representing the number of days before or after the "reference date" of January 1, 1960. (For more information about how the SAS System works with date and time values, see, for example, my paper *Working with SAS Date and Time Functions,* presented at SUGI 23 and SUGI 24, and which is available via the SUGI On-Line Proceedings web site at http://support.sas.com/usergroups/sugi/proceedings/index.html. An updated version of the paper is available from my company website, www.SierraInformation.com, by clicking on the "Past Presentations" link on the home page.) But, it's impossible for a "mere human" to divine what to what month, day, year, day of week, or calendar quarter the numeric value 16537 is associated. That's where SAS Formats can help you. For a number of reasons, we need the "internal value" of a date variable for April 11, 2005 (or any other date) to be stored as the number of days before/after January 1, 1960, but we then also need a way to easily display (or portray) that value in SAS-generated output so that the users of your output know what that value *means.*

As we will discuss in more detail below, the SAS System includes a large number of "built-in" formats that handle common tasks such as altering the representation of date, time or datetime variables. Each SAS-supplied format is documented in the SAS Language Dictionary documentation, available in hardcopy from the SAS Publications Division and in the Online Documentation for all releases of SAS 9 Software. Table 1 shows how different SAS-supplied formats would display the SAS date value of 16537 in your output.

.

Table 1: Using SAS-Supplied Formats with SAS Date Value 16537

Formatted Value	*Format Applied*
16537	**No Format Applied**
04/11/05	**MMDDYY8.**
04/11/2005	**MMDDYY10.**
11/04/2005	**DDMMYY10.**
11APR05	**DATE7.**
11APR2005	**DATE9.**
APR2005	**MONYY7.**
2	**QTR.**
2005	**YEAR**
05Q2	**YYQ4.**
2005Q2	**YYQ6.**
April 11, 2005	**WORDDATE.**
2	**WEEKDAY.**
Monday	**DOWNAME.**
Monday, April 11, 2005	**WEEKDATE.**
11 April 2005	**WORDDATX.**

Table 1 demonstrates two important concepts with working with SAS Formats. First, while the *value* of the variable *remains the same,* how we "see" it in our output is governed by the choice of Format. So, when we *associate a Format to a variable,* we are NOT changing the "internal" value of the variable (that is, how it is stored in the data set), but we ARE changing how it *appears in the output*. Second, we may often have a choice among several different formats that could be appropriate for a single column variable in our data set.

Table 1 also shows an important fact about Formats in the SAS System. The period (or "dot") in the Format distinguishes it from a Variable Name. When SAS "sees" the "dot," it "knows" to associate the instructions in the given Format to the Variable. Since SAS Variable Names can only contain letters, numbers and the underscore symbol, the period symbol is how SAS detects the difference between a Variable and a Format.

ASSOCIATING FORMATS TO VARIABLES

Format are "associated," or "assigned" to variables either in a Data Step or a Procedure Step. When you include a FORMAT Statement in a Data Step, SAS adds the format association information you specified in the Data Set's Descriptor Portion. Supplying a format association in a Procedure Step means that the given format will be used only in the output generated by that PROC. If one format association is in a Data Set Descriptor Portion and a *different* format association is supplied in a Procedure Step using that Data Set, then the format association in the Procedure Step "wins," and is used only during the execution of that PROC. As we briefly saw above, and will discuss in further detail below, using different format associations for the same variable provides a convenient and time-saving way to display the values of the variable in different configurations while avoiding otherwise unnecessary Data Step programming.

WORKING WITH SAS-SUPPLIED FORMATS

In many situations, a SAS-Supplied Format may be sufficient for your reporting/presentation needs. Documentation for the SAS-Supplied (or "internal") Formats is found in the SAS Language Dictionary manual. If you can't find a

SAS-supplied Format appropriate for your data, you can create your own Formats using PROC FORMAT (see below).

Formats with width.decimal (W.D) options

Many SAS Supplied Formats contain options to control the overall width of the format and how many decimal places will be displayed. Each of these "W.D" Formats has a default width (see each SAS Supplied Format's document for specific details) that is used if you don't specify them yourself. For example, consider the numeric value 123456789.908659. Tables 2A and 2B show how the external representation of that value changes when we: 1) apply either the COMMA or the DOLLAR Format; and, 2) change the width and number of decimal places SAS should use to portray the value in our output. Here they are:

Table 2A: Using the DollarW.D Format
SAS Suplied Formats with a W.D Specification

Displayed Value	*Format Association*
123456789.907659000	No DOLLARw.d Format
$123,456,789.907659000	DOLLAR30.9
$123,456,789.907659	DOLLAR30.6
$123,456,789.90766	DOLLAR30.5
$123,456,789.908	DOLLAR30.3
$123,456,789.9	DOLLAR30.1
$123,456,790	DOLLAR30.
$123,456,790	DOLLAR12.
$123,456,790	DOLLAR15.
$123,456,789.9	DOLLAR16.1

Table 2B: Using the COMMAw.d Format
SAS-Supplied Formats with W.D Specifications

Diplayed Value	*COMMAw.d Format Used*
123456789.907659000	No COMMAw.d Format
123,456,789.907659000	COMMA30.9
123,456,789.907659	COMMA30.6
123,456,789.90766	COMMA30.5
123,456,789.908	COMMA30.3
123,456,789.9	COMMA30.1
123,456,790	COMMA30.
123,456,790	COMMA11.
123,456,790	COMMA15.
123,456,789.90	COMMA16.1

Specifying the correct overall Format width and appropriate number of decimal places is essential to having SAS present your data the way you want them to appear. Notice that when decimal specifications are supplied that are smaller than the internal decimal value, the Format "rounds up" the output to the nearest specified decimal place.

It is important to supply Formats that support "W.D" options with an overall width value large enough to accommodate the largest value of the variable to which it is applied. It's also important to make sure the decimal value you supply to this class of SAS-Supplied Formats be wide enough to display all significant digits to the right of the decimal point. Otherwise, your output may not display values of variables correctly (or, at the very least, the way you want them to)! Let's take a look at both how this problem can arise and how SAS lets you know that it's having a problem with the "W.D" format you've specified.

Let's look at a made-up data set with just four observations. To fix ideas, there are two variables in the made-up data set with identical values, which helps us inspect the difference between "internal" values of the variable and how they are displayed by the PERCENTw.d Format. Figure 1 shows the PROC PRINT code used to generate a report and Table 3 shows the report itself. Notice that the FORMAT Statement in the PROC PRINT task associates the PERCENT5.1 format to variable FMT_VALUE. Looking at the PROC PRINT output, you can see that using this format results in the loss of potentially significant information to the reader. The "width.decimal" specification of 5.1 instructs SAS to display the values of FMT_VALUE in five bytes, with one position to the right of the decimal place. That's simply "not enough" to display all digits as the variable is stored in the data set. Besides looking at the PROC PRINT-generated output, there's another way we can find out if we've incorrectly specified width and/or decimal places when associating the format to the variable and that's by carefully reading our SASLOG. If you see this message in the SASLOG: **NOTE: At least one W.D format was too small for the number to be printed. The decimal may be shifted by the BEST format.** then it's time to think about adjusting your format width specifications.

```
proc print data=percentages label noobs split = '*';
   format pct_value  8.7 fmt_value percent5.1;
label fmt_value =
'Formatted Value Displayed*Using PERCENT Format'
pct_value =
'Internal Value*as Stored in Data Set';
title1 'My Friend the SAS(r) Format';
title2 'Using the Percent Format';
run;
```

Figure 1: PROC PRINT Code to Generate Output Report

My Friend the SAS(r) Format
Using the Percent Format

group	Internal Value as Stored in Data Set	Formatted Value Displayed Using PERCENT Format
1	.4003400	40%
2	.8840700	88%
3	.0789000	8%
4	.0000063	.0%

Table 3: PROC PRINT Output

For more information about this message, and the conditions which cause it, see Kuligowski (2003). As always, care should be taken to inspect the SASLOG at each phase of your project or program. Ignoring messages supplied as NOTES (as opposed to WARNINGs and ERRORs) in the SASLOG should *always* be avoided. We can correct the problems shown in Figure 1 and Table 3 by specifying both a wider format width and more decimal places to the right of the decimal point. Using the PERCENT11.5 Format, for example, gives us enough space to correctly portray the values of all digits of the variable. Table 4 gives us the "correct" results when the PERCENT11.5 Format is applied.

Now, let's take a look at ways to create customized formats with PROC FORMAT.

My Friend the SAS(r) Format
Using the Percent Format

group	Internal Value as Stored in Data Set	Formatted Value Displayed Using PERCENT11.5 Format
1	.4003400	40.03400%
2	.8840700	88.40700%
3	.0789000	7.89000%
4	.0000063	0.00063%

Table 4: Using the correct "W.D" specifications

PROC FORMAT: AN INTRODUCTION

PROC FORMAT supplies the tools you use to create your own formats and to manage Catalogs containing formats. This BASE SAS Procedure creates VALUE or PICTURE formats for both character and numeric variables. The format labels are either written by the user in a PROC FORMAT step or supplied in an Input Control Data Set, which is discussed in detail below. In addition, PROC FORMAT can write the contents of an existing format to an output data set, which is called an Output Control Data Set. Finally, the Procedure can display information about all or some entries in a Formats Catalog in your Output Window. We will explore each of these capabilities in the following section of the paper. (PROC FORMAT can also create INFORMATs, which are beyond the scope of this paper. Please see the PROC FORMAT documentation for details about how to create customized INFORMATs.)

CREATING YOUR OWN FORMATS: Essential Concepts

Commands supplied in PROC FORMAT "steps" create either numeric or character formats. You must know in advance of writing a format whether the variable to which you want to associate it in a subsequent Data or Procedure Step is stored as either character or numeric. *A character format may not be associated to a numeric variable, nor can a numeric variable be associated to a character format.*

SAS stores formats in a Formats Catalog. By default, your formats are stored in a temporary Formats Catalog in your WORK Library. At the end of your SAS Session, the temporary Formats Catalog is deleted along with the other entries (e.g., data sets, graphics catalogs and Document Itemstores [new in SAS 9, see my paper, "A Peek at PROC DOCUMENT," available for download from my website] in the WORK Library. Optionally, you can create a "Permanent SAS Formats Catalog" in a Permanent SAS Data Library.

Using PROC FORMAT, you can write the formal label information yourself within the procedure step, or use the values of variables in an Input Control Data Set (see below) as the values of format labels.

Starting in SAS 9, Format names can be up to 32 characters long. The names of Formats to be associated to character variables MUST start with the dollar sign, which "counts towards" the 32 character maximum length. Names for the formats you create CANNOT have the same name as a SAS-Supplied Format (e.g., you cannot create a format called PERCENT or COMMA) and may not end in a number.

Let's take a look at some simple examples. The PROC FORMAT task in Table 5 creates both a character and a numeric format, both of which are stored in the WORK.FORMATS Catalog. The Catalog is automatically created the first time you execute PROC FORMAT in a SAS Session, and you can add more Formats to it during the session. (Re-running a PROC FORMAT task with the same Format name in the VALUE Statement results in having the "old" Format replaced with the "new" Format.)

```
254 proc format;
255   value animalf 1 = 'Zebra'
256                 2 = 'Aaardvark'
257                 3 = 'Quoll';
258   value $zoof   'S' = 'San Diego'
259                 'A' = 'Anchorage'
260                 'H' = 'Houston';
261   run;
```

Table 5: Using PROC FORMAT

Figure 2 shows the icon representing the Formats Catalog when using SAS in the Windows™ Operating System. Although you can click on this icon and see the names of the Formats stored in the Catalog, a better way to learn about the Format is to specify the FMTLIB option in the PROC FORMAT task. This option directs PROC FORMAT to display information about each Format in the Catalog in the Output Window. Additionally, specifying both the PAGE and FMTLIB options directs PROC FORMAT to start each Format's information on a separate page in the Output Window. Table 6 and Figure 3 present a code sample and output resulting from using the FMTLIB Option in PROC FORMAT.

Table 6: PROC FORMAT with the FMTLIB Option

```
265 proc format fmtlib;
266 title1 'My Friend the SAS(r) Format';
267 title2 'Looking at the Formats Catalog';
268 run;
```

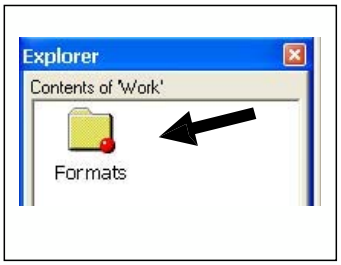
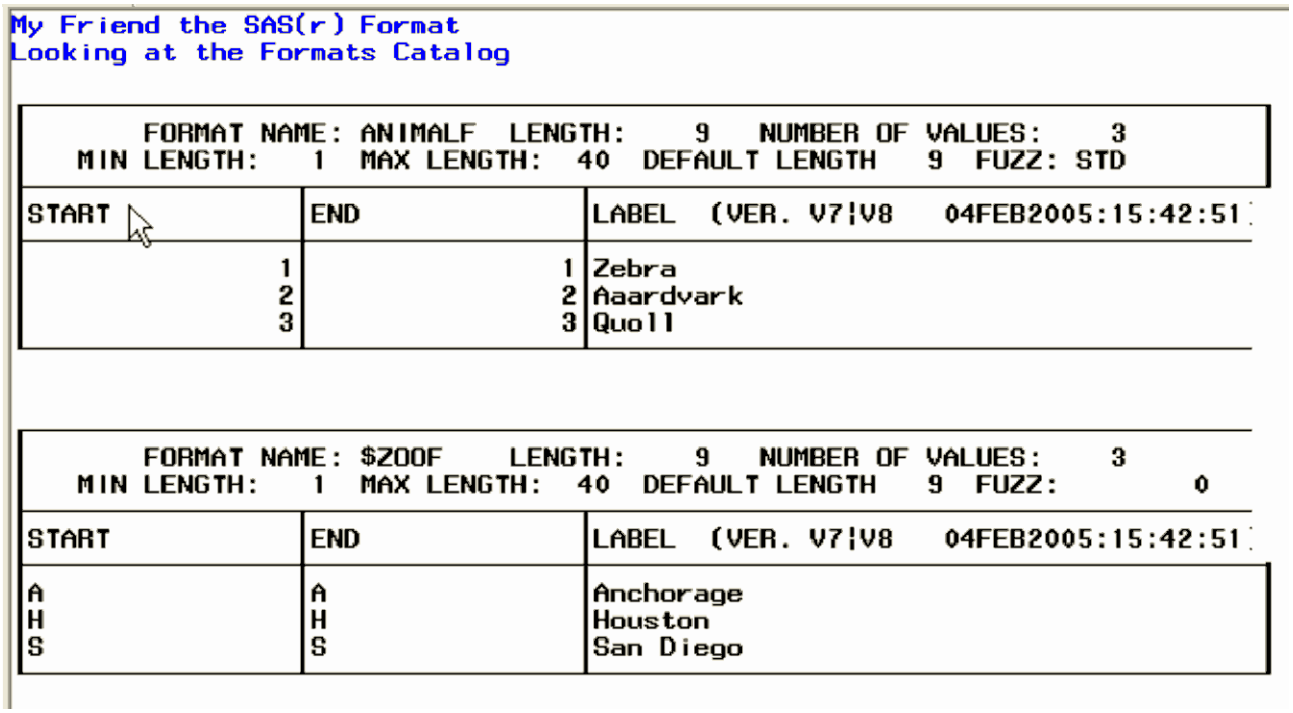


Figure 2: Formats Catalog Icon



My Friend the SAS(r) Format
Looking at the Formats Catalog

FORMAT NAME: ANIMALF LENGTH: 9 NUMBER OF VALUES: 3
MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 9 FUZZ: STD

START	END	LABEL (VER. V7|V8 04FEB2005:15:42:51
1	1	Zebra
2	2	Aaardvark
3	3	Quoll

FORMAT NAME: $ZOOF LENGTH: 9 NUMBER OF VALUES: 3
MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 9 FUZZ: 0

START	END	LABEL (VER. V7|V8 04FEB2005:15:42:51
A	A	Anchorage
H	H	Houston
S	S	San Diego

Figure 3: PROC FORMAT Output when FMTLIB Option is Specified

Permanent vs. Temporary Formats Catalogs

A Permanent Formats Catalog can be created in any library to which you first establish a Library Reference (LIBREF). This is often a good way to avoid having to re-create your formats at the start of each SAS Session, as well as an excellent method by which to make a common set of format labels available to groups of SAS users who all work with the same data sets. A health insurance company, for example, may have a standard set of thousands of medical procedure codes, the values of which need formats when they are displayed using SAS reporting or analysis tools. A shared Permanent Formats Catalog avoids the need for each SAS user at the insurance company from having to write/maintain individual Formats Catalogs. Instead, one shared Permanent Formats Catalog can be made available to all users, and, since it is being shared by multiple users, only that one common library requires periodic revisions/updates.

Creating a Permanent Formats Catalog is easy. First, assign the Library Reference (using either the LIBNAME Statement or the New Library Window) to where on your computer you want to store the Formats Catalog. Then, add the LIBRARY=*libref* option to your PROC FORMAT Task. Table 7 shows how it's done.

Now comes the tricky part. By default, when you specify a Format association, SAS will *only* look for it in the library of SAS-supplied Formats and in a Formats Catalog in the Work Library, if any. If you have formats in a Permanent Formats Catalog, you must give SAS explicit "directions" as to what Library the Formats Catalog is to be found. To do this, use the FMTSEARCH SAS System Option to instruct SAS to include one or more additional Libraries in its search for the Formats you specify. Table 8 shows both the FMTSEARCH and the NOFMTERR SAS System Options. The NOFMTERR Option instructs SAS to use the internal values of a variable if it cannot find the Format that has been associated to it. NOFMTERR is therefore helpful when someone gives you a data set, but "forgets" to give you either the associated Formats Catalog or the PROC FORMAT code you need to create it yourself.

```
265 libname SUGI 'C:\';
266 proc format library = SUGI;
267 value animalf   1 = 'Zebra'
268                 2 = 'Aaardvark'
269                 3 = 'Quoll';
270   value $zoof   'S' = 'San Diego'
271                 'A' = 'Anchorage'
272                 'H' = 'Houston';
273   run;
```

Table 7: Creating Permanent Formats

```
275 options fmtsearch=(SUGI) nofmterr;
276 proc freq data=sugi.zoos;
277 tables animal*zoo;
278 format animal animalf. zoo $zoof.;
279 title1 'My Friend the SAS(r) Format';
280 title2 'Using the FMTSEARCH and NOFMTERR Options';
281 run;
```

Table 8: The NOFMTERR and FMTSEARCH Options

Let's discuss these concepts in a bit more detail, since they can cause some confusion. Remember, the default "search path" in SAS for Format associations is first to the WORK Library's Formats Catalog and *then* to any additional Formats Catalogs in the Libraries for which you have supplied LIBREFs in the FMTSEARCH SAS System option. And, those Libraries are searched in the order, from left to right, you have supplied the LIBREFs to the FMTSEARCH SAS System option. Here is an example. Table 9 below is a screen capture of the SAS Program Editor in which two PROC FORMAT steps have been coded. The first creates a series of Formats and stores them in a Permanent Formats Catalog in the SUGI Library. The second creates one format and stores it in the WORK Library. Notice that a Character Format called $TWO_F is in both the SUGI and WORK Formats catalogs.

```
1  proc format LIBRARY = SUGI;
2  value satf    low-500 = '500 or below'
3                501 - 550 = '501 to 550'
4                551 - 600 = '551 to 600'
5           601 - high = '601 and higher';
6
7  value $one_f     'A' = 'Excellent'
8                   'B' = 'Very Good'
9                   'C' = 'Average'
10                  'D' = 'Below Average'
11                  'F' = 'Failed';
12 value sat1f       200 - 400 = 'Well Below Average (200-400)'
13                   401 - 450 = 'Below Average (401-540)'
14                   450 - 550 = 'Average (451-550)'
15                   551 - 600 = 'Above Average (551-600)'
16                   601 - 800 = 'Well Above Average (601-800)';
17 value $two_f    'A','B','C' = 'Passed'
18                 'D', 'F'  = 'Failed';
19 run;
```

Table 9: Creating Formats

Table 10 shows how the FMTSEARCH System is applied in a SAS Program. On line 30 of the Program Editor, OPTIONS FMTSEARCH=(SUGI) is applied. What this means is SAS will still search the WORK Library's Formats Catalog *first,* and *then* the Formats Catalog in the SUGI Library. Then, on line 41, OPTIONS FMTSEARCH=(SUGI WORK) is applied. This statement reverses the order in which the Formats Catalogs are searched. First, the SUGI Library's Format's Catalog is searched for the Format, and *then* the WORK Library's Catalog is searched.

SAS Software users need to make sure they understand both where their formats are stored, and the order in which SAS, either by default or use of the FMTSEARCH SAS System Option, searches Formats Catalogs. Otherwise, your output may not appear the way you want it to.

```
29 * default is WORK, then others;
30 options fmtsearch=(sugi);
31 ods rtf file = 'c:\formats2.rtf' bodytitle style=journal;
32 ods listing close;
33 proc freq data=sugi.grades;
34 tables final;
35 format final $two_f.;
36 title 'My Friend the SAS(r) Format';
37 title2 'Using Formats Catalogs';
38 run;
39
40 * reset the search path;
41 options fmtsearch=(sugi work);
42 proc freq data=sugi.grades;
43 tables final;
44 format final $two_f.;
45 title 'My Friend the SAS(r) Format';
46 title2 'Using Formats Catalogs';
47 title3 'Specifying Search Path Order';
48 run;
49 ods listing;
50 ods rtf close;
```

Table 10: Using the FMTSEARCH SAS SYSTEM Options

My Friend the SAS(r) Format
Using Formats Catalogs

The FREQ Procedure

Final Course Grade

FINAL	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Good	21	70.00	21	70.00
Bad	9	30.00	30	100.00

My Friend the SAS(r) Format
Using Formats Catalogs
Specifying Search Path Order

The FREQ Procedure

Final Course Grade

FINAL	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Passed	21	70.00	21	70.00
Failed	9	30.00	30	100.00

Figure 4: Selecting the "Right" Format

Creating VALUE Formats

A VALUE Format associates a LABEL to the value of a Variable. VALUE Formats can be used to assign one label to one value of a variable, or the same value label can be assigned to multiple values of the variable to which it is associated. In this way VALUE Formats are often a potent and time saving alternative to "recodes" in a Data Step, as we will see below.

Looking back to Table 9 above, we see several examples of creating VALUE Formats that are subsequently applied to variables in a data set with information about student performance. The character Format $ONE_F has one label per value of the variable, while the other Formats assign the same label to a range of values.

Let's look again the Format SAT1_F in Table 9. You'll see that the same value (450) has been assigned to two labels. Since the end of one format range "touches" the start of another format range, SAS will associate the value to the first range.

Unless you are creating MULTILABEL Formats, which are discussed below, supplying overlapping value ranges to PROC FORMAT will yield an error message in your SASLOG.

What happens if you "forget" to supply a format label in a Format? Let's look at Table 11, where a Format is created that does not include value labels for some values of a variable. The Format is then associated to the variable SAT_M in the following PROC FREQ task. Figure 5 shows the output generated by PROC FREQ. The Format was used to "group" or collapse observations into rows in the output frequency table, but since no labels were supplied for the values of 463, 471 and 488, the *internal* values of these variables appear in the PROC FREQ output. This example demonstrates two important aspects of the VALUE format: first, you can use them to "group" or "bin" groups of observations in to discrete categories without using a data step to create otherwise unnecessary variables in your data set. Second, in the absence of a label, a Procedure displays/uses the internal value of the variable.

```
55 proc format library = sugi;
56   value satxf   200 - 400 = 'Well Below Average'
57                 401 - 450 = 'Below Average'
58                 510 - 550 = 'Average'
59                 551 - 600 = 'Above Average'
60                 601 - 800 = 'Well Above Average';
61 run;
62
63 ods listing close;
64 ods rtf file = 'c:\formats.rtf' style=journal bodytitle;
65 proc freq data= sugi.grades;
66 tables sat_m;
67 format sat_m satxf.;
68 title1 'My Friend the SAS(r) Format';
69 title2 'Working with VALUE Formats';
70 title3 'Using a Format to Collapse Ranges of a Numeric Variable';
71 run;
72 ods rtf close;
73 ods listing;
```

Table 11: Forgetting to Associate Labels to Values

My Friend the SAS(r) Format
Working with VALUE Formats
Using a Format to Collapse Ranges of a Numeric Variable

The FREQ Procedure

SAT Math Score				
SAT_M	Frequency	Percent	Cumulative Frequency	Cumulative Percent
463	1	3.33	1	3.33
471	1	3.33	2	6.67
488	1	3.33	3	10.00
Average	7	23.33	10	33.33
Above Average	18	60.00	28	93.33
Well Above Average	2	6.67	30	100.00

Figure 5: Forgetting to Associate Labels to Values: PROC FREQ

The PROC FORMAT Task in Table 12 shows several potentially useful options you can use when specifying ranges of values to which a Format label is to be applied. For more information about these options, please see the PROC FORMAT documentation, or Pete Lund's SUGI 26 paper, "More than Just Value: A Look in to the Depths of PROC FORMAT."

```
76 * options for values ranges;
77 * each comment within the PROC FORMAT task;
78 * refers to the range following the comment;
79 proc format;
80   value rangef
81   /* 0 to 10, inclusive */
82   0 - 10 = 'Group 1'
83   /* excluding 10, up to/including 20 */
84   10 <- 20 = 'Group 2'
85   /* 21 up to, but not including 30 */
86   21 -> 31 = 'Group 3'
87   /* all other values, EXCEPT Missing */
88   other = 'Group 4'
89   /* Missing Values */
90   . = 'Missing';
91   run;
```

Table 12: Specifying Value Ranges

Taking the time to make sure you've specified the correct value ranges in your Format Labels is obviously a critical step in writing Value Formats that do what you want them to do.

Creating Value Formats from Input Control Data Sets

The examples we've seen so far show PROC FORMAT tasks where the Value Labels have been coded/written by the user. That's just fine when we have only a few Value Labels to code. But, what happens when we have, say, a master file of part codes, procedure codes, or other values for which we want to create a Value Format? Not only is the prospect of having to write a long Value Format uninviting and error-prone, but it is unnecessary.

PROC FORMAT can create Value Formats from the values of variables supplied in an Input Control Data Set. These are just "regular" data sets you can create with a Data Step, but the variables can be read by PROC FORMAT and the observations "turned in to" Value Labels." Input Control Data Sets must have a least three character variables: 1) FMTNAME, which supplies the name of the Value Format to be created; 2) START, which gives the value of the variable to which the Format Label is to be associated; and, 3) LABEL, the Value Label itself. Other variables present in the Input Control Data Set are ignored by PROC FORMAT. Table 13 shows how an Input Control Data Set is created that "maps" one Value Label to one value of a variable.

```
 93 * build input control data set;
 94 data control1;
 95 retain fmtname 'cats_1f';
 96 length label $ 40;
 97 input start $ 1-2 label $ 3-43;
 98 datalines;
 99 1 Lynx
100 2 Puma
101 3 Canadian Lynx
102 4 Jaguar
103 5 Jaguarundi
104 6 Ocelot
105 7 Jungle Cat
106 8 Tiger
107 9 Asian Golden Cat
108 10Snow Leopard
109 11Bornean Bay Cat
110 12Snow Leopard
111 ;
112 run;
```

Table 13: Input Control Data Set Example 1

My Friend the SAS(r) Format
Creating an Input Control Data Set
Example 1

Obs	fmtname	label	start
1	cats_1f	Lynx	1
2	cats_1f	Puma	2
3	cats_1f	Canadian Lynx	3
4	cats_1f	Jaguar	4
5	cats_1f	Jaguarundi	5
6	cats_1f	Ocelot	6
7	cats_1f	Jungle Cat	7
8	cats_1f	Tiger	8
9	cats_1f	Asian Golden Cat	9
10	cats_1f	Snow Leopard	10
11	cats_1f	Bornean Bay Cat	11
12	cats_1f	Snow Leopard	12

Figure 6: PROC PRINT Output Showing Input Control Data Set's Contents

Figure 6 presents PROC PRINT Output displaying the Input Control Data Set created by the PROC FORMAT task in Table 13. Now that we have an Input Control Data Set, all we need to do to "turn it in to a Format" is to supply its name to the CNTLIN Option in PROC FORMAT, and we're in business. Table 14 shows how this is done, and Figure 7 shows the output subsequently generated by the FMTLIB Option in PROC FORMAT.

```
124 proc format library = sugi fmtlib cntlin=control1 ;
125 select cats_1f;
126 title 'My Friend the SAS(r) Format';
127 title2 'Input Control Data Sets: Example 1';
128 run;
```

Table 14: Supplying an Input Control Data Set to PROC FORMAT with the CNTLIN Option

My Friend the SAS(r) Format
Input Control Data Sets: Example 1

FORMAT NAME: CATS_1F LENGTH: 16 NUMBER OF VALUES: 12
MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 16 FUZZ: STD

START	END	LABEL (VER. V7|V8 06FEB2005:10:54:09)
1	1	Lynx
2	2	Puma
3	3	Canadian Lynx
4	4	Jaguar
5	5	Jaguarundi
6	6	Ocelot
7	7	Jungle Cat
8	8	Tiger
9	9	Asian Golden Cat
10	10	Snow Leopard
11	11	Bornean Bay Cat
12	12	Snow Leopard

Figure 7: FMTLIB Option Output Showing Format Created from Values/Variables in an Input Control Data Set (see Table 14 above).

Input Control Data Sets can also be used to associate a Format Label to a range of values. In this situation, a fourth required character variable, END, must be included in the data set.

For example, suppose we want to assocaite the label "North America" to all observations with values of 01 to 06 in a variable of interest, and "Asia" to values 07 to 12. (For illustration purposes we're just working with two ranges; most likely, your data sets will have many more value ranges to which a Format Labels are to be applied.)

By creating an Input Control Data Set that has the previously-discussed varaibles FMTNAME, LABEL, START *and* END, we supply both the beginning value (in variable START) and the ending value (in variable END) to be used when constructing the Value Labels.

Other options are available that may, depending on your requirements, may be appropriate for use when creating and applying Input Control Data Sets to create Formats. Please see the PROC FORMAT documentation chapter for additional details.

It is also possible to create a permanent or temporary SAS data set *from* an entry in a Format Catalog. For example, you may want to create a table of Format labels for documentation purposes, or transport Formats Catalogs across operating systems. Regardless of the purpose/reason that you need to do it, applying the CNTLOUT option in PROC FORMAT will create an output SAS data set from entires in your formats catalog. More details are available from the PROC FORMAT documentation.

Creating MULTILABEL Formats

As we mentioned earlier, PROC FORMAT will, by default, not allow you to create Value Formats with overlapping value ranges. In many situations, we *want* to avoid assigning the observations to multiple Format Labels, so this default may "save" us from potentially serious errors. But, there are times when we do want to create a "hierarchy table" or otherwise assign the same observation to multiple "levels" formed by Value Format. For example, we may have counties within states, and states within regions, etc.

Starting in Version 8, the MULTILABEL Option in PROC FORMAT allows you to assign the same value to overlapping value ranges. This new feature now makes it much easier to create Value Formats which can be used to portray and analyze data that fall in to hierarchies or other combinations. Before discussing the MULTILABEL Format option further, there are a few "housekeeping" items to address. First, as of SAS 9.1.3 Software (the most current production release of SAS Software as this paper is being written), only three BASE SAS Procedures (MEANS, SUMMARY and TABULATE) can be used with MULTILABEL Formats. If you create a Value Format with the MULTILABEL option and then use it with any other Procedure, that PROC will only use the "primary format labels," which are discussed below. Second, there is an important difference between SAS 8 and SAS 9 with respect to the MULILABEL Option in PROC FORMAT, which will be discussed below. Third, an understanding of the concepts of "primary" and "secondary" Format Label is *essential* to take advantage of what the MULTILABEL Format facility can do for us.

For example, we have a data set with 18 patients in it. For each observation we know that patient's first name and age (in years). We want PROC MEANS to analyze the values of patient age classified (that is, grouped by) age category. From what we have seen so far, a Value Format would be a great way to form the desired groupings or classifications. But, we want to have multiple categories within which each patient's age may fall. For example, a person whose age is 5 is to be put in both the "0 to 5" and "0 to 18" groupings. We can now do that with a MULTILABEL Format. Here is an example: first, Figure 8 shows the data set whose values we are going to analyze. Second, Figure 9 shows how the MULTILABEL Option is supplied to PROC FORMAT to build the desired Value Format. Then, Figure 10 shows how the MLF Option is supplied to the CLASS Statement in PROC MEANS to have that procedure analysis (i.e., "classify") its analysis by the formatted values of the variable age.

My Friend the SAS(r) Format
Using Multilabel Formats

Obs	name	age
1	mary	10
2	fred	28
3	john	7
4	erica	29
5	tim	5
6	susan	13
7	andrew	47
8	peter	37
9	cindy	16
10	thea	21
11	joe	20
12	tilly	58
13	ruth	75
14	rick	8
15	richard	19
16	helen	26
17	alexa	10
18	heather	2

Figure 8: Patient Data Set

```
185 proc format;
186   value MLF1_FMT (multilabel)
187     0 - 12 = 'Child'
188     13 - 19 = 'Adolescent'
189     0 - 19 = 'Children & Adolescents'
190     20 - 21 = 'Young Adult'
191     low - 21  = 'Children, Adolescents & Young Adults'
192     22 - high = 'All Adults'
193     low - high = 'All Patients';
194     run;
```

Figure 9: Using the MULTILABEL Option

```
198 ods listing close;
199 ods rtf file = 'c:\MLF_ex1.rtf' bodytitle style=journal;
200 proc means data=patients mean median maxdec=2;
201 * use MLF _without_ NOTSORTED;
202 format age mlf1_fmt.;
203 class age/MLF; * <<== USE Multilabel Format Option;
204 var age;
205 title3 'Using Multilabel Formats';
206 title4 'Default: SORTED';
207 run;
208 ods rtf close;
209 ods listing;
```

Figure 10: Using the MLF Option in PROC MEANS

Now, let's take a look at the output table generated by PROC MEANS. That's shown in Figure 8, below. What PROC MEANS did is calculate the MEAN and MEDIAN of patient AGE by both the Primary and Secondary Format labels. For example, "Heather, " who is 2 years old (observation number 18 in the example data set), was included when calculating the mean and median age in both the "Child" and "Children and Adolescents" rows of the output report.

Using Multilabel Formats
Default: SORTED

The MEANS Procedure

Analysis Variable : age

age	*N Obs*	*Mean*	*Median*
Adolescent	3	16.00	16.00
All Adults	7	42.86	37.00
All Patients	18	23.94	19.50
Child	6	7.00	7.50
Children & Adolescents	9	10.00	10.00
Children, Adolescents & Young Adults	11	11.91	10.00
Young Adult	2	20.50	20.50

Figure 8: Using a MULTILABEL Format

Figure 8 shows two important things to keep in mind when using MULTILABEL Formats. First, the Primary Label is defined as the first time within your PROC FORMAT task you associate a Value Label to the value of a variable. All other associations are considered Secondary Labels. Second, PROC FORMAT, by default, stores Value Labels in sort order, regardless of the way you enter them in your code. That's why the output in Figure 8 shows the "rows" in sort (i.e., alphabetical order). While the analytics (e.g., the means and medians) are correct, the ordering of the rows is at best confusing.

Starting in SAS 9.1, both the MULTILABEL and NOTSORTED options can be specified at the same time. This allows you to create a MULTILABEL Format whose rows are in the order you need them for subsequent reporting/analysis purposes. In order to use them in, say, a PROC REPORT task , you need to add the PRELOADFMT and ORDER=DATA options, along with the previously-mentioned MLF option, to your CLASS Statement. Figure 11 shows you how to specify both the MULTILABEL and NOTSORTED Options in a PROC FORMAT task. Figure 12 displays a PROC MEANS task where MLF, ORDER=DATA and PRELOADFMT are used in the CLASS Statement. Finally, Table 9 shows the PROC MEANS-generated output when the code in Figures 11 and 12 were executed.

```
211 proc format;
212  value MLF2_FMT (multilabel notsorted)
213    0 - 12 = 'Child'
214    13 - 19 = 'Adolescent'
215    0 - 19 = 'Children & Adolescents'
216    20 - 21 = 'Young Adult'
217    low - 21  = 'Children, Adolescents & Young Adults'
218    22 - high = 'All Adults'
219    low - high = 'All Patients';
220    run;
```

Figure 11: Using the NOTSORTED and MULTILABEL Options in PROC FORMAT

The SAS System

Using Multilabel Formats
Using the NOTSORTED Option in SAS 9.1

The MEANS Procedure

Analysis Variable : age

age	*N Obs*	*Mean*	*Median*
Child	6	7.00	7.50
Adolescent	3	16.00	16.00
Children & Adolescents	9	10.00	10.00
Young Adult	2	20.50	20.50
Children, Adolescents & Young Adults	11	11.91	10.00
All Adults	7	42.86	37.00
All Patients	18	23.94	19.50

Table 9: Using the NOTSORTED Option in SAS 9.1

```
222 ods listing close;
223 ods rtf file = 'c:\MLF_ex2.rtf' bodytitle style=journal;
224 proc means data=patients mean median maxdec=2;
225 * use MLF _with_ SORTED;
226 format age mlf2_fmt.;
227 class age/MLF ORDER=DATA PRELOADFMT;
228 var age;
229 title3 'Using Multilabel Formats';
230 title4 'Using the NOTSORTED Option in SAS 9.1';
231 run;
232 ods rtf close;
233 ods listing;
```

Figure 12: Using ORDER=DATA and PRELOADFMT

CONCLUSION

SAS' Format facilities provide a range of tools with which to work with your data, of which just a few have been highlighted here. More information about these resources is available in the SAS Documentation and in prior /SAS Global Forum/SUGI proceedings, some of which have been referenced above.

ACKNOWLEDGMENTS

Thanks to Rick "Mr. Format" Langston, Manager of Core Systems Development at SAS Institute Worldwide Headquarters for both his outstanding development of the Format facilities with the SAS System and for his unstinting support of the SAS user community by his willingness to share his knowledge of SAS tools at numerous SAS user group conferences and meetings.

CONTACT INFORMATION

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

Andrew H. Karp
Sierra Information Services
19229 Sonoma Highway PMB 264
Sonoma, CA 95476 USA
+1 707 996 7380
Email: Andrew@SierraInformation.com
Web: www.SierraInformation.com