Paper TU05

The FORMAT procedure - more than just a VALUE statement

Lawrence Heaton-Wright, Quintiles, Bracknell, UK

ABSTRACT

The FORMAT procedure is most frequently used to define formats for variables. However, there is also extended functionality available within the format procedure which allows users to define informats, picture formats, create data sets of formats, use data sets (also known as control data sets) to create formats and to display the formats catalog information.

This paper aims to show users how informats and picture formats work, how to create and use control data sets and how to display the formats catalog.

The audience for this paper does not have to be of a high technical level but should be familiar with the workings of base SAS. The contents are not restricted to one particular operating system.

INTRODUCTION

The FORMAT procedure enables users to define formats for variables. Formats determine how raw data appears to users. For instance, a SAS date variable is displayed as 24JUN2003 using the DATE9 format. However, the actual data stored in SAS is actually the number 15880, which is the number of days since 1st January 1960.

The VALUE statement allows users to create “user-defined” formats. These are not SAS-supplied but are coded by users. They enable data to be displayed as information. For example, on a demography page from a CRF form the gender data are recorded as 1 and 2, representing Male and Female respectively. This can be enhanced by the simple addition of a format to display 1 as Male and 2 as Female.

```
DATA gender_example;
    INPUT pid gender @@;
    CARDS;
1001 1 1002 1 1003 2 1004 2 1005 1
;
RUN;

PROC FORMAT;
    VALUE genderf
           1=”Male”
           2=”Female”;
RUN;
```

pid	gender
1001	Male
1002	Male
1003	Female
1004	Female
1005	Male

Many users are not aware of the additional functionality that is built into the FORMAT procedure that allows users to call formats from within other formats, create informats, picture formats, use control data sets to create or modify formats and to display formats catalog information. There are also many options available to enhance the basic VALUE statement.

ADDITIONS TO THE VALUE STATEMENT

SPECIFYING VALUES OR RANGES

There are several methods for specifying ranges for use in formats, informats and pictures. You can for instance use a single value as in the GENDERF format previously shown. A range of values can also be used. For instance a list of values, 12-68, 'A'-'Z' can be used. If you are specifying a range of character strings then ensure that each string is enclosed in single quotes. If we used 'A-Z' the procedure would interpret this as a three-character string with A as the first character, a hyphen as the second character and Z as the third character. If the quotes are omitted then the procedure interprets these values as characters.
The LOW and HIGH keywords can be used as values in a range. The less than symbol (<) can be used to exclude values from ranges. The missing character (.) and special missing values (.A to .Z) can also be used as values/ranges. The keyword OTHER can be used to match all other values that do not fall into any of the other values or ranges.

CALLING FORMATS WITHIN OTHER FORMATS

This allows users to define their own formats but within those formats assign a SAS or an existing user-defined format to a value or range of values. A typical example is shown below:

```
DATA fwf_example;
    INPUT pid value @@;
    CARDS;
1001 10.5 1002 -11.5 1003 0 1004 . 1005 12.8
;
RUN;

PROC FORMAT;
    VALUE fwf
        0-HIGH=[BEST.]
        LOW-<0='Negative'
        .='Missing';
RUN;
```

```
 pid       value

1001            10.5
1002    Negative
1003               0
1004    Missing
1005            12.8
```

INFORMATS

Informats are used for converting raw data values. For instance users can convert a character string to a different character string, convert a character string to a number or convert a number to another number.

```
PROC FORMAT <options(s)>;
    INVALUE <$>name <(informat-option(s)>
           value-range-set(s);
RUN;
```

Typically they are used to convert character strings to numeric values. To achieve this the INPUT statement is used in a data step.

```
DATA dataset;
    SET dataset;
    newvar=INPUT(oldvar,informat_name.);
RUN;
```

A particularly common use for informats is the conversion from character date values to SAS date values using the SAS-supplied informats.

```
DATA temp;
    datechar='20030601';
    datenum=INPUT(datechar,YYMMDD8.);
    FORMAT datenum DATE9.;
RUN;
```

In the above example the YYMMDD8informat has been used to convert the character variable, DATECHAR, to a SAS date value. Note however, the DATE9 format is applied to DATENUM to ensure that the variable is displayed in a date format.

One thing to beware of when converting a numeric value to another numeric value is the note in the log file stating that numeric values have been converted to character values. The variables that are created are still of numeric type but this message is untidy and really should not be allowed to remain in the log file. To avoid this message the input value needs to be converted using a PUT statement and then the INPUT statement used to convert that character value.

```
PROC FORMAT;
    INVALUE num_num
        1-10=1
       11-20=2
       21-30=3;
RUN;

DATA test;
    DO i=1 TO 25;
        OUTPUT;
    END;
RUN;

DATA num_to_num;
    SET test;
    i_nowarn=INPUT(PUT(i,BEST.),num_num.);
    i_warn=INPUT(i,num_num.);
RUN;
```

In the example above the I_WARN variable has the numeric to character note but I_NOWARN does not.

```
17   DATA num_to_num;
18       SET test;
19       i_nowarn=INPUT(PUT(i,BEST.),num_num.);
20       i_warn=INPUT(i,num_num.);
21   RUN;

NOTE: Numeric values have been converted to character values at the places given by:
(Line):(Column).
      20:18
NOTE: There were 25 observations read from the data set WORK.TEST.
NOTE: The data set WORK.NUM_TO_NUM has 25 observations and 3 variables.
NOTE: DATA statement used:
      real time           0.23 seconds
      cpu time            0.00 seconds
```

A frequent use of informats is to convert character data to numeric data, often for the purposes of ordering statistics in outputs. Consider changing a variable STATISTIC (containing ‘N’ ‘MEAN’ ‘MIN’ ‘MAX’), which needs to be printed out in a report in the order N, MEAN, MIN, MAX. Clearly we could not use the normal PROC SORT for this. We have to create a numeric variable that relates to the character variable STATISTIC. One method for achieving this is to use an informat.

```
PROC FORMAT;
    INVALUE statord
        'N'=1
        'MEAN'=2
        'MIN'=3
        'MAX'=4;
RUN;

DATA char_to_num;
    LENGTH statistic $4.;
    DO statistic='N','MEAN','MIN','MAX';
        statord=INPUT(statistic,statord.);
        OUTPUT;
    END;
RUN;
```

statistic	statord
N	1
MEAN	2
MIN	3
MAX	4

Two useful options for the INVALUE statement are JUST and UPCASE. The JUST option left-justifies all the input strings prior to comparing them to the values or ranges. UPCASE converts all the input strings to uppercase characters, again prior to comparing them to the values or ranges.

```
PROC FORMAT;
    INVALUE stordju (JUST UPCASE)
        'N'=1
        'MEAN'=2
        'MIN'=3
        'MAX'=4;
RUN;

DATA char_to_num;
    LENGTH statistic $4.;
    DO statistic=' N','mean','MIN','MAX';
        statord=INPUT(statistic,statord.);
        statord_just_upcase=INPUT(statistic,stordju.);
        OUTPUT;
    END;
RUN;
```

statistic	statord	statord_ just_upcase
N	.	1
mean	.	2
MIN	3	3
MAX	4	4

PICTURE FORMATS

Picture formats can be used to present numerical values using a template, the PICTURE format. This could be used to present p-values, percentages and decimal place alignment.

```
PROC FORMAT <options(s)>;
    PICTURE <$>name <(picture-option(s)>
          value-range-set(s);
RUN;
```

PRESENTING A P-VALUE USING A PICTURE FORMAT

Here we can see that we have p-values created from a SAS procedure.

```
      pval

100.85760000
1.0000000000
0.9876544000
0.0500100000
0.0500000000
0.0499900000
0.0123440000
0.0100100000
0.0100000000
0.0099990000
0.0010010000
0.0010000000
0.0009999000
0.0004965320
 .
```

Using a picture format we can represent a p-value as "<0.001" or have flags such as "*" attached to the p-value to indicate a significant value. Creating a picture format to represent p-values involves several steps.

Firstly we need to understand some of the available terms.

value-range-set – range of values and the associated template.

value or range = '*picture*' where picture specifies a template for displaying numeric values. The picture is a sequence of characters in single quotation marks. Maximum length is 40 characters. Pictures are specified with three types of characters: digit selectors, message characters and directives.

> Digit selectors are numeric characters (0 – 9) that define positions for numeric values. Digit selectors of 0 do not print leading zeroes. Any other digit selector will print leading zeroes. In the SAS manual 9 has been used as the non-zero digit selector and this convention has been followed throughout this paper.
>
> Message characters are non-numeric characters that print as specified in the picture.
>
> Directives are special characters that can be used to format date, time or datetime values. These are dealt with later in the paper.

There are some very useful options to both the PICTURE statement and the value/range sets. The ROUND option is applied to the PICTURE statement and the three remaining options are applied to the value/range sets.

ROUND – this option ensures that numeric variables, which contain decimal places, are rounded prior to using the template defined in the format

FILL = '*fill character*' – this option "fills" the zeroes in the formatted value which have not been "filled" by the variable value. The default FILL character is a space.
NOEDIT – this option ensures that for the value specified the picture format acts as a format (message characters).
PREFIX = '*prefix character*' – this option specifies a character to place in front of the first significant digit. Zero digit selectors must be used for the PREFIX option to have any effect.

Both the FILL and PREFIX options can be used on the same value/range set. The format places the PREFIX and then the FILL characters.

There are other options available for picture formats but these are not being discussed here as they are beyond the scope of this paper.

```
PROC FORMAT;
  PICTURE pvalpic (ROUND)
        1<-HIGH='Impossible ' (NOEDIT)
        0.05<-1='0009.999   ' (FILL='#' PREFIX='+')
     0.01<-0.05='0009.999 * ' (FILL='#')
     0.001-0.01='0009.999 **' (FILL='#')
     LOW-<0.001='  <0.001 **' (NOEDIT)
           .-.z=' Missing   ' (NOEDIT)
    OTHER=' ' (NOEDIT);

  PICTURE pval2pic
        1<-HIGH='Impossible ' (NOEDIT)
        0.05<-1='0009.999   ' (PREFIX='+')
     0.01<-0.05='0009.999 * '
     0.001-0.01='0009.999 **'
     LOW-<0.001='  <0.001 **' (NOEDIT)
           .-.z=' Missing   ' (NOEDIT)
    OTHER=' ' (NOEDIT);
RUN;
```

```
outvar         outvar2              pval

Impossible     Impossible     100.85760000
##+1.000         +1.000       1.0000000000
##+0.988         +0.987       0.9876544000
##+0.050         +0.050       0.0500100000
###0.050 *        0.050 *     0.0500000000
###0.050 *        0.049 *     0.0499900000
###0.012 *        0.012 *     0.0123440000
###0.010 *        0.010 *     0.0100100000
###0.010 **       0.010 **    0.0100000000
###0.010 **       0.009 **    0.0099990000
###0.001 **       0.001 **    0.0010010000
###0.001 **       0.001 **    0.0010000000
  <0.001 **      <0.001 **    0.0009999000
  <0.001 **      <0.001 **    0.0004965320
 Missing        Missing        .
```

As can be seen from the above example, great care must be taken to round decimal place values when using picture formats as without the ROUND option the data is truncated.

PICTURE FORMATS – DIRECTIVES

Directives are special characters that can be used to format date, time or datetime values. These are a special type of picture format character modifier which can only be used when the DATATYPE= option is specified. These directives are extremely useful when the SAS supplied date and time formats do not apply to the circumstance.

The DATATYPE= option has the possible arguments: DATE, TIME or DATETIME

The list of possible directives are:

%a	Locale's abbreviated weekday name
%A	Locale's full weekday name
%b	Locale's abbreviated month name
%B	Locale's full month name
%d	Day of the month as a decimal number (1-31), with no leading zero
%H	Hour (24-hour clock) as a decimal number (0-23), with no leading zero
%I	Hour (12-hour clock) as a decimal number (1-12), with no leading zero
%j	Day of the year as a decimal number (1-366), with no leading zero
%m	Month as a decimal number (1-12), with no leading zero
%M	Minute as a decimal number (0-59), with no leading zero
%p	Locale's equivalent of either AM or PM
%S	Second as a decimal number (0-59), with no leading zero
%U	Week number of the year (Sunday as the first day of the week) as a decimal number (0,53), with no leading zero %w Weekday as a decimal number (1= Sunday, 7)
%y	Year without century as a decimal number (0-99), with no leading zero
%Y	Year with century as a decimal number
%%	%

If a leading zero is required this can be achieved adding a zero prior to the directive. Other characters that appear in the picture appear as requested.

For example we may wish to have a datetime variable appearing in the format DDMMMYYYY:HH:MM. The SAS DATETIMEw.d will not accommodate this particular format.
To achieve this we need to use directives within a picture format.

```
PROC FORMAT;
    PICTURE dthhmm
        LOW-HIGH='%0d%b%Y:%0H:%0M' (DATATYPE=DATETIME);
RUN;

DATA test;
    x="02dec2003:09:18"dt;
    y=x;
    FORMAT x DATETIME20. y dthhmm.;
RUN;

                  x           y

02DEC2003:09:18:00    02DEC2003:09:18
```

CONTROL DATA SETS

Control data sets are data sets that contain descriptions of formats and informats contained in the format catalog. They can either be produced from the FORMAT procedure (Output control data sets) or can be produced using a data step and loaded into a FORMAT procedure (Input control data sets).

Output control data sets are produced from the FORMAT procedure by using the CNTLOUT=*data_set_name* option. Input control data sets are produced using DATA step processing ensuring that a minimum set of required variables is used. To specify an input control data set the CNTLIN=*data_set_name* option is specified.

```
PROC FORMAT LIBRARY=library_reference CNTLOUT/CNTLIN=dataset_name;
    <SELECT/EXCLUDE> <$>name;
RUN;
```

A restricted list of the variables in a control data set is:

START – Character variable containing the start value for the range
END – Character variable containing the end value for the range
FMTNAME – Character variable containing the format/informat name
LABEL – Character variable containing the value of the label associated with the specified range (START to END)

TYPE – Character variable that indicates the type of format. Possible values are:
- N = numeric format (excluding pictures)
- C = character format
- I = numeric informat
- J = character informat
- P = picture format

Note: N is the assumed TYPE if this variable is not included in an input control data set.

SEXCL/EEXCL – Character variables indicating whether the range's start [SEXCL] or end [EEXCL] values are excluded from the range. Possible values are:
- Y = the range's start/end value is excluded
- N = the range's start/end value is not excluded

The SELECT and EXCLUDE statements are valid for use with the CNTLOUT option, however you cannot use a SELECT statement and an EXCLUDE statement in the same PROC FORMAT step. The SELECT and EXCLUDE statements allow the user to restrict which formats/informats/pictures are sent to the output control data set.

DISPLAYING FORMATS CATALOG INFORMATION

To display the contents of a format catalog the FMTLIB option should be used. This is especially useful when you need to find out what the range of values a particular format can take.

The SELECT and EXCLUDE statements are valid for use with the FMTLIB option, with the same restrictions as mentioned previously. If you specify a SELECT or EXCLUDE statement without either the FMTLIB or CNTLOUT= options the default action for PROC FORMAT is to invoke the FMTLIB option. The PAGE option prints information about each entry in the format catalog on a separate page. The PAGE option automatically invokes the FMTLIB option.

```
PROC FORMAT LIBRARY=library_reference FMTLIB;
    <SELECT/EXCLUDE> <$>name;
RUN;
```

CONCLUSION

The format procedure can be used in many different ways. It can be used to convert numeric variables to character using the VALUE statement. This is the most common usage – usually converting categories to representative information. It can be used to convert character variables to numeric using informats. It can change the way a numeric value can be displayed using picture formats. Data step processing can be used to modify PROC FORMAT information using control data sets. This also is a relatively easy method for transporting formats across versions and platforms. We can also display the information stored in the format catalog using the FMTLIB option.

Overall, it can be seen that there is more to the format procedure than just the VALUE statement. There is far more than that to this very versatile and under-used procedure.

REFERENCES

SAS® Procedures Guide, Version 8.

CONTACT INFORMATION

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

Lawrence Heaton-Wright
Quintiles Limited
Station House, Market Street
Bracknell, Berkshire, RG12 1HX
United Kingdom
Work Phone: +44 1344 708320
Fax: +44 1344 708106
Email: lawrence.heaton-wright@quintiles.com
Web: www.quintiles.com