PhUSE 2014

Paper CC03

Overview of SAS proc format and code for combining formats catalogs

Douglas Staddon, Cmed Ltd , Horsham, UK

ABSTRACT

Using formats and managing them in SAS can be quite complicated and give strange errors leading to confusion. This paper aims to give an overview of formats and how they can be managed in a consistent and robust way when producing derived Datasets, Listings, Tables and Figures (DLTFs).

Introduction

SAS ® Proc format has many facets that can be used to create formats to enhance outputs. Formats can be available for the raw database and also for derived/analysis datasets so these need to be managed to ensure they are consistent, don't contain duplicates and are not too large. So first I will look at the basic syntax, potential problems with formats and then using this move on to how catalogs can be combined.

Proc Format

Defining and using a basic numeric and character format

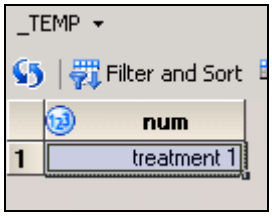
```
proc format ;

value trt
1='treatment 1'
2='treatment 2'
;
```

So format TRT is defined with start value on the left with the label on the right hand side. The format is stored in a catalog. The catalog can be in the temporary WORK library or in a permanent library.

The format is usually associated with a variable in a datastep. This can be using a format statement as below, or attrib.

```
data _temp;
  num=1;
  format num trt.;
run;
```



Character formats are defined in a similar way but have a leading $ sign.

```
proc format ;
value $c_txt
  'A'='Alpha'
;
```

Defining a format based on another format

An existing format can be used on the right hand side in square brackets. This can help keep formats consistent and any changes can be made once rather than in many places.

```
value tot_trt
  1-2 =[trt20.]
  99  ="Total"
  other="Amend tot_trt"
;
```

This format uses the TRT format defined earlier with a length of 20 and adds another option 99 as a total treatment. OTHER is useful for error checking. In this example it might highlight patients who have not been randomized and so TRT is missing.

Defining a format with range of values

Instead of discrete values a range can be used on the left hand side. LOW and HIGH are recognized by SAS as a low negative number and high positive number respectively.

```
** integer ranges **;
value rnge
 low-100 =[4.]
 150-high=[8.]
 other='Amend rnge'
;
```

Using real numbers for ranges can be done. Rounding and numeric precision problems within computer systems can be a problem in some cases; fuzz option may help with this but you need to be careful.

```
value real_rnge
 0 - <0.05  ='<0.05'
 0.05 - <1  =[5.2]
 1-high     =[9.]
 other='Amend real_rnge'
;
```

SAS checks if ranges overlap and writes an error message in the log.

```
value rng_olap
1-10 ='One to ten'
5-15 ='five to fifteen'
;
ERROR: These two ranges overlap: 1-10 and 5-15 (fuzz=1E-12).
```

Missing numeric codes

Example of numeric codes below.

```
value num
   . ='Missing'
   .x='Not done'
   .y='Not received'
   .z='Not collected'
   low-high=[8.]
   ;
```

These look useful but can be more trouble than they are worth. The main problem is making sure other code runs as expected for missing values. In the code below, testing for a missing value only works for dot rather than the coded missing values, unless “less than .z” is used.

```
data temp2;
  do i=.,.x,.y,.z,1;
    format i num.;
    if i=.      then flag ='1';  else flag ='';
    if i le .z then flag2='1';  else flag2='';
    output;
  end;
run;
```

	i	flag	flag2
1	Missing	1	1
2	Not done		1
3	Not received		1
4	Not collected		1
5	1		

Removing a format

Sometimes a format gets in the way and code is simpler if the format is removed.
This can be done by using “format <variable>;” and then the default values are used. This is especially useful in a PROC TRANSPOSE since the formatted value is used for naming the columns. If the transpose is a treatment variable the text can be very long and removing the format can keep the code generic for other studies.

```
** defining some data**;
data ae;
  subj=1;trt=1;aetxt='SOC1';output;
  subj=2;trt=2;aetxt='SOC2';output;
  subj=3;trt=1;aetxt='SOC1';output;

  **assigning format *;
  format trt trt.;
run;

** summarising data per treatment;
proc summary data =ae nway;
  class aetxt trt;
  output out=ae_sum ;
run;
```

```
** transposing by treatment**;
proc transpose data=ae_sum out=tr_ae_sum prefix=trt;
  by aetxt;
  var _freq_;
  id trt;
  format trt;
  ** removes format trt. so columns are named trt1 and trt2;
run;
```

With format:

	aetxt	_NAME_	trttreatment 1	trttreatment 2
1	SOC1	_FREQ_	2	.
2	SOC2	_FREQ_	.	1

Without format:

	aetxt	_NAME_	trt1	trt2
1	SOC1	_FREQ_	2	.
2	SOC2	_FREQ_	.	1

Proc format options

Saving format catalog to a library

When formats are defined the format is saved to the WORK library. So the SAS code

```
proc format ;run;
```

is equivalent to:

```
proc format lib     =work;run;
proc format library=work;run;
```

So formats can be stored in other defined libraries.

```
proc format library=raw     ;run;
proc format library=derived;run;
```

Saving format catalog as a dataset – CNTLOUT=<dataset>

This can be done using CNTLOUT option and is useful for many tasks like combining catalogs as described later and exporting data between platforms. Using only datasets in an export for transferring data between platforms is simpler than importing datasets and catalogs and makes the process more robust.

```
libname raw "&ProjectPath./data/raw" ;
proc format lib=raw cntlout=rawfmts;
  value anum 1='one';
run;
```

This code creates a work dataset RAWFMTS with the variables below with FMTNAME, START, LABEL and TYPE being the important ones.

#	Variable	Type	Len	Label
1	FMTNAME	Char	32	Format name
2	START	Char	16	Starting value for format
3	END	Char	16	Ending value for format
4	LABEL	Char	3	Format value label
5	MIN	Num	3	Minimum length
6	MAX	Num	3	Maximum length
7	DEFAULT	Num	3	Default length
8	LENGTH	Num	3	Format length
9	FUZZ	Num	8	Fuzz value
10	PREFIX	Char	2	Prefix characters
11	MULT	Num	8	Multiplier

	Variable	Type	Len	Label
12	FILL	Char	1	Fill character
13	NOEDIT	Num	3	Is picture string noedit?
14	TYPE	Char	1	Type of format
15	SEXCL	Char	1	Start exclusion
16	EEXCL	Char	1	End exclusion
17	HLO	Char	13	Additional information
18	DECSEP	Char	1	Decimal separator
19	DIG3SEP	Char	1	Three-digit separator
20	DATATYPE	Char	8	Date/time/datetime?
21	LANGUAGE	Char	8	Language for date strings

The length of label and other variables will change depending on the data but the variable names remain the same.

Obs	FMTNAME	START	END	LABEL	TYPE
1	ANUM	1	1	one	N

Importing a dataset to create a FORMAT catalog – CNTLIN=<dataset>

Similarly this dataset from above can create a format. This may be useful when format names and values are listed in a specification document and so the text can be copied, then imported to a dataset and a catalog created from that.

Only a few of the variables are actually required – SAS uses default values as required.

```
data rawfmts2;
```

```
  FMTNAME='anum';start=2;label='two';output;
run;

proc format lib=raw cntlin=rawfmts2;run;
```

This code will produce the format with the one option.

If both datasets RAWFMTS and RAWFMTS2 are joined together to create the format then the variable lengths and types need to be consistent. In the rawfmts2 dataset above for example START is numeric and SAS can cope with this but it would usually be a character variable when many formats are created at the same time.

So the dataset should really define realistic lengths and types and be checked for truncation.

```
** combining two datasets **;
data rawfmts2;
  FMTNAME='anum';start='2';end=start;label='two';output;
run;

data total;
  length fmtname $32 start end $16 label $200;
  set rawfmts rawfmts2;
run;

proc format lib=raw cntlin=total;run;
```

Printing formats –FMTLIB

Printing formats is easy just by using the FMTLIB option.

```
proc format lib=raw fmtlib;run;
```

Giving this output below:

```
-------------------------------------------------------------------------------
|     FORMAT NAME: ANUM     LENGTH:   3   NUMBER OF VALUES:   1      |
|  MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH:   3  FUZZ: STD      |
|-----------------------------------------------------------------------|
|START         |END           |LABEL  (VER. V7|V8   17SEP2014:17:38:20)|
|--------------+--------------+---------------------------------------|
|          2|            2|two                                |
-------------------------------------------------------------------------------
```

However if the format has a long label the output looks truncated so PROC REPORT can be used to print the dataset version of the format catalog as below.

Printing formats with long labels – PROC REPORT

This format has a label that is too long for the default FMTLIB report.

```
proc format  cntlout=long_fmt fmtlib;
  value long_fmt
    1='Some long text that is truncated with FMTLIB'
    2='_________1_________2_________3_________4_________5_________6'
    3='12345678901234567890123456789012345678901234567890123456789'
    ;
run;
```

```
--------------------------------------------------------------------------------
|     FORMAT NAME: LONG_FMT LENGTH:   60   NUMBER OF VALUES:   3       |
|  MIN LENGTH:   1  MAX LENGTH:  60  DEFAULT LENGTH:  60  FUZZ: STD      |
|------------------------------------------------------------------------|
|START         |END           |LABEL  (VER. V7|V8   17SEP2014:18:10:44)|
|--------------+--------------+----------------------------------------|
|           1|           1|Some long text that is truncated with FM|
|           2|           2|_________1_________2_________3_________4|
|           3|           3|12345678901234567890123456789012345678901|
--------------------------------------------------------------------------------
```

So the format can be saved to a dataset and used with Proc Report.

```
proc report data=long_fmt nowd;
  column type fmtname start end label;

  define type    / display width=4        'Type';
  define fmtname / display width=12 flow 'Format name '    ;
  define start   / display width=10 flow 'Start'           ;
  define end     / display width=10 flow 'End  '           ;
  define label   / display width=50 flow 'Format label'    ;
run;
```

Type	Format name	Start	End	Format label
N	LONG_FMT	1	1	Some long text that is truncated with FMTLIB
N	LONG_FMT	2	2	_____1_____2_____3_____4_____5_____6
N	LONG_FMT	3	3	12345678901234567890123456789012345678901234567890

Avoiding Duplicate formats - NOREPLACE

Sometimes a format may exist in two different catalogs and so to avoid any inconsistencies it's useful to only keep one version in a combined catalog.

This can be done with the NOREPLACE option that wil take the first definition and give an error for the second as below.

```
**clear work formats catalog;
proc datasets lib=work memtype=catalog kill nolist;run;

** define same format twice but with NOREPLACE option;
** get message in the log;
proc format lib=work noreplace cntlout=dataset;
value fmt 1='One';
value fmt 1='ONE';
run;
```

So the log window highlights the duplicated format.

```
20         proc format lib=work noreplace cntlout=dataset;
```

```
21          value fmt 1='One';
NOTE: Format FMT has been output.
22          value fmt 1='ONE';
ERROR: The format FMT is already in the library, but the NOREPLACE
option was specified.
NOTE: Format FMT could not be written.
23          run;
```

SAS options related to formats – NOFMTERR

On a similar note it can be confusing if formats are created but SAS does not know where to look for them, or in which order they should be used.

```
**define location of formats catalogs;
libname raw     "&ProjectPath./data/raw";
libname derived "&ProjectPath./data/derived";
** clear out catalogs for clarity;
proc datasets lib=raw     memtype=catalog kill nolist;run;
proc datasets lib=derived memtype=catalog kill nolist;run;
proc datasets lib=work    memtype=catalog kill nolist;run;
** define same format in two different catalogs
   - but with different values **;
proc format lib=raw    ; value misc 1='ABC';run;
proc format lib=derived; value misc 1='DEF';run;

** switch on fmterr - on or off depends how SAS is setup**;
options fmterr;
** SAS looks in work.formats by default **;
** can use NOFMTERR to not give an error *;
data test;
  num=1;
  format num misc.;
run;

22          data test;
23            num=1;
24            format num misc.;
                       _____
                       48
ERROR 48-59: The format MISC was not found or could not be loaded.

25          run;

** look at current option values **;
proc options  option=fmterr option=fmtsearch ;run;

LOG window:
 FMTERR            Treat missing format or informat as an error
 FMTSEARCH=( APFMTLIB WORK LIBRARY )
                   List of catalogs to search for formats and informats
```

Using option NOFMTERR there is just a NOTE:

```
NOTE 484-185: Format MISC was not found or could not be loaded.
```

SAS options related to formats – FMTSEARCH

Using the SAS code above we can remove the Note or Error by changing where SAS looks for formats. This is done by the code below.

```
options fmtsearch=(raw derived);
title "FMTSEARCH is raw then derived:";
proc print data=test;run;
FMTSEARCH is raw then derived:
          Obs    num
           1     ABC

** order of format search is important **;
options fmtsearch=(derived raw);
title "FMTSEARCH is derived then raw";
proc print data=test;run;

FMTSEARCH is derived then raw
          Obs    num
           1     DEF
```

It also shows that the order of the catalogs is important and changes the result displayed, though the underlying value is the same. As mentioned before, having the same format in two catalogs can easily make them inconsistent so it is better to use one catalog and have only one version of a format. This is described in the next section using a combination of commands looked at previously.

Combining Formats catalogs

Points to consider

To combine catalogs I assume there is a raw and a derived library defined in a "setup.sas" program for a particular study. Setup.sas should contain study path information, SAS options for log messages etc , page size definitions and so on. It should also include or run a formats program or macro.

The formats program therefore will be run every time someone runs setup.sas and so it needs to be coded in an intelligent way that limits the number of versions and avoids locking the formats catalog for other users.

The combined catalog will contain the raw dataset formats and add the ones required for the derived datasets and TFL outputs. These typically relate to treatment groups and totals , the order of statistics, statistic labels for consistency, order of variables and parameters and visit names. All formats used should be in the formats program rather than the individual TFL programs.

The raw formats may be very large since all possible lab parameters, drug names or questionnaire options can be included over all trials. Most new formats are usually added to the derived catalog over a few days as TFLs are programmed with minimal amendments later on. Therefore the formats program and catalog only need to be updated on an adhoc basis. When a format is updated it is useful to keep an old copy of the catalog as a dataset for comparison and validation. Formats are usually validated by validation of the individual outputs but the formats program could be reviewed or even dual coded.

The raw and derived libraries are "read only" and updated when required. The derived formats catalog is stored as a dataset in derived together with any old versions. Only old copies from a particular date are kept to limit the number of versions so a comparison versus a previous version might cover a number of formats.

So looking at these requirements here is some Pseudo code:

Pseudo code for macro

%fmt (update=N (default) or Y);

- If raw formats catalog exists then create a work dataset version called rawformats
  - Otherwise create a dummy format and a work dataset version called rawformats
- Make derived library read write
- If updating formats (update=Y)
  - keep an old copy of the catalog
  - delete current derived catalog
- If derived formats catalog does not exist then
  - Import the raw formats from rawformats dataset
  - Run the code to create the derived formats
  - Check no duplicate formats with noreplace option
  - Check formats okay by printing formats to output window with fmtlib option
  - Output combined formats to a dataset
- Make derived library read only
- Macro is called with update=N by default
  - update=Y used when required

The actual SAS code is in the Appendix 1.

Using the macro

Setup.sas calls the macro with

```
%fmt(update=N);
```

When formats are added to the macro:

- Run the macro to update any version in memory.
- And call it with `%fmt(update=Y);`
- If there are any errors with the format rerun the macro and call update until it is okay.

The database and raw data may be updated with extra panels and formats so for each extract it is good idea to clear out old files so the derived formats use the latest raw formats catalog.

Example screen shot of derived library format datasets and catalogs over time.

formats.sas7bcat	07/08/2014 11:52	SAS7BCAT File	1,608 KB
formats.sas7bdat	07/08/2014 11:52	SAS Data Set	4,208 KB
old_formats_01jul2014.sas7bdat	23/06/2014 14:23	SAS Data Set	4,208 KB
old_formats_07aug2014.sas7bdat	01/07/2014 09:27	SAS Data Set	4,208 KB

Conclusion

Formats are useful for displaying data in a meaningful way, checking data is restricted to correct values and highlighting errors. However they can be confusing if stored in many catalogs, use options not relevant for a particular study protocol or are managed in multiple ways. The code given in the appendix can be used for creating a combined catalog but Standard Operating Procedures may be required for how updates are made, whether they should be qced by dual programming and the level of documentation required. Formats are an important part of a study and so defining these processes early on will help with programming the listings, tables and figures in a consistent way.

Contact Information

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

Mr Douglas Staddon
Cmed Ltd
Cmed Clinical Services
Holmwood, Broadlands Business Campus
Langhurstwood Road
Horsham, West Sussex, RH12 4QP, United Kingdom
Work Phone: +44 (0)1403 755422
Fax: +44 (0)1403 755051
Email: dstaddon@cmedresearch.com
Web: www.cmedgroup.com

Appendix 1 - FMT.SAS

```
/*************************************************************************
*   Study: generic                                                       *
*   Program name:  fmt.sas                                               *
*   Programmers name:  Douglas Staddon                                   *
*   Location:  ...\programs                                              *
*   Description:  Create derived formats catalog combined with raw       *
*   Input files:  raw.formats                                            *
*   Output files:  derived.formats                                       *
*   Assumptions:  setup.sas has been run                                 *
*                 &ProjectPath is defined correctly                      *
*                 "&ProjectPath./data/derived" directory exists          *
*   Program History:                                                     *
*   ================                                                     *
*------------------------------------------------------------------------*
*   Date        |     Name         |  Reason/Comments                    *
*---------------+------------------+-------------------------------------*
*   02Sep2014   | D Staddon        | Original                            *
*---------------+------------------+-------------------------------------*
*************************************************************************
/

%macro fmt(update=N);
** export raw formats to a dataset **;
%if %sysfunc(cexist(raw.formats)) %then
%do;
proc format lib=raw cntlout=rawformats;
run;
%end;
%else %do;
proc format cntlout=rawformats;
value $yn
"Y"="Yes"
"N"="No";
run;
%end;
```

```
** library derived made read write**;
libname derived  "&ProjectPath./data/derived" ;

%if &update=Y %then
%do;
  %if %sysfunc(exist(derived.formats)) %then %do;
    proc datasets lib=derived memtype=data;
      %if %sysfunc(exist(derived.old_formats_&sysdate9.))
        %then %do;
        delete old_formats_&sysdate9.;
      %end;
    change formats=old_formats_&sysdate9.;
    run;
    quit;
  %end;

    proc datasets lib=derived memtype=catalog kill;
    run;
    quit;
%end;

%if %sysfunc(cexist(derived.formats)) %then %do;
*** do nothing - derived formats catalog already exists ;
%end;
%else %do;
***add raw formats to derived formats ***;
proc format
  /* permanant formats library */
  lib=derived
  /* specify dataset from which SAS builds formats */
  cntlin=rawformats
  /* create dataset of Biostats formats
    (including raw formats as specified in cntlin option) */
  cntlout=derived.formats
  /* prevent existing format being overwritten
       by format of the same name */
  noreplace
  /* print formats */
  fmtlib
  ;

value fstat_bi
1='n'
2='mean'
3='SD'
4='CV% mean'
5='geo-mean'
6='CV% geo-mean'
7='median'
8='minimum'
9='maximum';

value trt
```

```
1='treatment 1'
2='treatment 2'
3='Total'
;

value cohort
1='Cohort A'
2='Cohort B'
3='Total'
;


/* format for the statistics */
value stat
1   = '    n                    '
2   = '    mean                 '
3   = '    SD                   '
4   = '    minimum              '
5   = '    median               '
6   = '    maximum              '
10  = '    Mean (SD)            '
11  = '    Median               '
12  = '    Range                '
;


** format VISFOR for presenting the visits in all listings;
value vis
101-101.99  ="SCR"
102-102.99  ="BAS"
103-103.99  ="DAY2"
104-104.99  ="DAY3"
105-105.99  ="DAY4"
106-106.99  ="DAY5"
107-107.99  ="DAY6"
108-108.99  ="DAY7"
other       ="Amend vis"
;

value fstat
1 = 'n'
2 = 'mean'
3 = 'SD'
4 = 'minimum'
5 = 'median'
6 = 'maximum';

value yesno
1='Yes'
0='No';

/* excluded from populations */
value incexcp
0='Excluded'
```

```
1='Included';

run;

%end;

***Analysis datasets***;
libname derived  "&ProjectPath./data/derived" access=readonly;

%mend ;
%fmt(update=N);

** force update **;
/*
%fmt(update=Y); **do not uncomment *;
*/
```