Some Useful Techniques of Proc Format

Stan Li, Minimax Information Services, Belle Mead, NJ

ABSTRACT

SAS® Format is a very unique and powerful function. It provides system built-in standard formats and the capability of allowing users to define their own formats. These formats are often used for data input and data output. In addition, formats can also be used for SAS dataset extraction and dataset merging. It would save significant computing resources by employing format function in very large dataset sub-setting or merging. This paper will present the techniques of creating a format, listing/viewing the contents of a format, nesting formats and multi-label formats. We will also illustrate the methods of using formats for large dataset extraction and merging.

Keyword: Proc Format, Very large dataset

INTRODUCTION

There are two basic types of SAS Data objects: dataset and catalog. While a SAS dataset is used to store data or observations as we all know, a SAS catalog is used to store others, such as a macro, a graphic output, a SCL code or a user-defined format. The usages and the features of SAS Formats have been discussed in many excellent literatures [2,3,4,5]. In this presentation, we will summarize some format techniques that we have benefited from.

SAS SYSTEM FORMATS

A format is a mapping from data value(s) to a word or a string. For user's convenience, SAS System provides over 700 default formats. Those formats can be found from the SAS manual [1]. But you can also use the following sample code to generate a list of all default system formats:

```
proc sql;
  create table SysFormats as
  select distinct fmtname from dictionary.formats;
quit;

proc print data= SysFormats;
run;
```

CUSTOMIZED FORMATS

It is not hard to create a customized format, but it is very essential to note that there are some differences between a format name and a variable name:

- a format name cannot be more than 8 characters
- a format name cannot end with a numeric number
- a character format name should always start with the symbol "$"

CREATE A FORMAT WITH A LIST

In most situations, a user-defined format can be produced in an arrangement list like the following:

```
libname MyFmt "...\Formats\myformats";
Proc format lib=myfmt.StudyXYX;;
Value $gender
   'M', 'm'='Male'
   'F', 'f'='Female'
;
Value age5grp
     0-<18='<18'
   18-<45='18-45'
   65-<85='65-85'
   85-high='85+'
;
Run;
```

This procedure will create one character format $gender and one numeric format age5grp, and save these two formats into the catalog StudyXYZ in MyFmt library. Without the lib= option, the formats will be saved to SAS temporary working directory, under the catalog name FORMATS.

CONSTRUCT A NESTED FORMAT

A nested format is to define a new format based on one or more existing formats. This feature can be very handy for data reporting. The nested format below will list data in different ways, depending on the range of the output data.

```
Proc format;
Value NestFmt
       100<-high='over 100'
       Low-<0='Missing'
       0-<10=[5.2]   /* list with 2 decimals, if less than 10 */
       Other=[5.1]   /* list with only one decimal, if btwn 10 to 100 */
;
Run;
```

CREATE A FORMAT BASED ON A SAS DATASET

Frequently, we need to generate a format, basing on a SAS dataset. For example, suppose there is a SAS dataset CENTERS with two variables: Center ID and Center Name. We are required to generate a format that maps the center ID to its name. It will be very tedious, or maybe impossible, to hard-code all the center IDs and the center names. Fortunately, with some simple data manipulation on the given dataset, we can utilize the CNTLIN option in PROC FORMAT to create such a format. A sample code is listed as following:

```
*** produce a format CtrFmt with the mapping: Center_ID ==> Center Name;
Proc sort data=Centers out=center_tmp nodupkey;  *** ensure all IDs are unique;
By CenterID;
data center_Fmt;
set Center_tmp end = last;
retain FMTNAME 'CtrFmt';  *** format name;
length LABEL $30;
START=CenterID;          *** left side value;
LABEL=CenterName;       *** right side value, should be character;
output;

if last then do;
  HLO='o';             *** Other values;
  LABEL ='Unknown';    *** label for other values;
  output;
end;
run;

*** output as a format;
proc format cntlin = Center_Fmt;
run;
```

UTILIZE A MULTI-LABEL FORMAT

Usually, in a SAS format, one value can only associate with one label. However, for newer SAS versions (version 8 or newer), SAS does allow multi-label formats, however the multi-label formats can only be used in 3 procedures: PROC MEANS, PROC SUMMARY and PROC TABULATE. A multi-label format example is shown below:

```
Proc format;
Value treats (multilabel)
    0="Placebo"
    1="Treat A"
    2="Treat B"
 1, 2="Treated"    /* combined two treatments as one */
;
Run;
```

Here, treatment code 1 and treat code 2 will be assigned to "Treat A" and "Treat B" separately and also assigned to "Treated" simultaneously. With this format, we can easily generate the summary for the 3 arms, plus the statistics for two treatment groups combined:

```
Proc summary data=test noprint completetypes nway;
   Class treat / preloadfmt mlf ;
```

```
   Var measure;
   Output out=summs n=N mean=avg std=std;
   Format treat treats.;
Run;
```

The option MLF in the CLASS statement is required. It tells SAS to compute the summary for each different group defined in the given multi-label format. If this MLF option is omitted, the combined group will not be included.

The multi-label conjunction with preloadfmt option will generate statistics for all the groups defined in the format, no matter whether is any patient in that group. Of course, if there is not any patient in a treatment group, the statistics for this group will be missing and the count N will be zero.

USE A FORMAT

LINK AND DE-LINK VARIABLES WITH FORMATS

Once a format is created and saved to a library, we can start to use it in a data step or in a procedure. If there are multiple format catalogs to be used, it is necessary to specify the format search order by a statement as following:

```
Options fmtsearch = (ProjFmt  myfmt.StudyXYX);
```

With this option, SAS will search for a format in this order: SAS system formats, Work.formats, then ProjFmt.formats and finally MyFmt.StudyXYZ. If a format cannot be found from all these catalogs, an error message will be issued.

Sometimes, we may receive a dataset with some variables that have been assigned with permanently formats, but the associated format catalog is not provided. If you still want to use such a dataset, you will need to suppress the “Formats xxx was not found” error message by the following OPTIONS statement:

```
Options nofmterr;
```

There are other ways to de-link the association between a variable and a format. One handy way is:

```
Format _all_;
```

This will remove all permanently assigned formats from the data step or procedure.

A format statement with only variable names, without any format name, will eliminate any possible format linkage for variables in the list, such as:

```
Format treat gender;
```

Any formats, if any, that link with the specified two variables Treat and Gender will be removed by this statement.

LIST/EXPORT CONTENTS OF A FORMAT

When we receive a SAS dataset from others, the first thing we need to do is to figure out the contents of the dataset. Proc Contents is a nice tool to do so. For SAS formats, we often need to find out the exact definition of formats too, especially when a set of formats are provided by an outside agency. Similarly to Proc Contents, This can be accomplished by the following techniques:

If we just need to get a list of formats in a catalog, use this:

```
proc catalog catalog=myfmt.StudyXYX;
  contents;
quit;
```

This will return all the format names, format types and created dates etc in the format catalog.

But if we want to get deep into the details of a specific format, we can utilize this sample code:

```
proc format fmtlib lib=myfmt.StudyXYX;
   select $treat center;
run;
```

This will list the definitions of the two selected formats $TREAT and CENTER. Without specify the SELECT statement, it will generate the descriptions for all the formats in the explicit catalog.

Sometimes, it may be necessary to export the specifications of formats into a SAS dataset. We can employ Proc Format with CNTLOUT option to do so:

```
proc format cntlout=outFmts lib=myfmt.StudyXYX;
  select $treat center;
run;
```

Here, the contents of the two specified formats will be extracted and saved into the SAS dataset OUTFMTS. We can then print out the format listings if we want to. It is also very easy to convert this output SAS dataset OUTFMTS back to a format catalog:

```
proc format cntlin=outFmts;
run;
```

FORMAT FOR DATA EXTRACTION

There are many very large datasets in health care industry, like GE-EMR (GE Electronic Medical Record) and Medicare/Medicaid Claim datasets. In our research, it is quite often that we are only interested in patients with certain condition, for example, patients taking hypertension medications. In this situation, we will need to extract those patients who had ever taken anti-hypertension medications for our analysis. Assume we had identified all anti-hypertension medications of our interest and a dataset with those medication ID's was generated. To subset hypertension records from the very large dataset, it is natural to think of merge technique. But this will require us to sort the huge dataset by medication ID and this would take too much computer resource. Fortunately, in such a situation, the features of FORMAT can help us avoid sorting the very large dataset.

First, we need to generate a format that maps all anti-hypertension medication IDs to '1'. We have mentioned about this skill in the previous section. The sample code can be simply like this:

```
*** produce a format HTNFmt with the mapping: Medication_ID ==> '1';
Proc sort data=HypertionID out=Med_tmp(keep=medication_ID) nodupkey;
  By Medication_ID;
Run;

data Med_Fmt;
set Med_tmp end = last;

retain FMTNAME 'HTNFmt';  *** format name: HTNFMT;
length LABEL $1;
START=Medication_ID;
LABEL='1';                *** '1' for hypertension medication;
output;

if last then do;
  HLO='o';
  LABEL ='0';           *** '0' for all other medications;
  output;
end;
run;

*** output as a format;
proc format cntlin = Med_Fmt;
run;
```

After this format has been produced, we can just make use of the WHERE statement to extract those observations of our interest easily:

```
Data Hypertension;
Set  LargeData;
 where=(put(Medication_ID, HTNFMT.)="1");

  ....
run;
```

Compared to the sort/merge method, this way will save significant computer time. Of course, there are other techniques that are suitable for this type of extraction too. One good candidate is harsh object. Users with interest on this topic can refer to these papers [5, 6].

FORMAT FOR DATA MERGING

Format not only can be used for data extraction, it can also be a nice tool to merge two dataset without sorting. This could be very efficient when we need to merge a very large dataset with a small dataset that has only a few numbers of variables. The key point for this is very similar to the idea in format/extraction.

Let us say our anti-hypertension medication dataset has medication ID, dose and dose unit, and we want to pass these two variables dose and dose unit to our extracted dataset. First, we will need to build two formats, one maps medication ID to dose and the other one links medication ID to dose unit, as following:

```
 *** produce two formats: Dose and Unit;
Proc sort data=HypertionID out=Med_tmp(keep=medication_ID dose unit) nodupkey;
  By Medication_ID;
Run;
data Med_Fmt;
set Med_tmp end = last;
retain FMTNAME;    *** two format names: ;
length LABEL $1;
START=Medication_ID;
LABEL=Dose; FmtName=Dose; output;
LABEL=Unit; FmtName=Unit; output;

if last then do;
  HLO='o';
  LABEL =’-‘;       *** ‘-’ for all other medications;
  FmtName”Dose; output;
  FmtName”Unit; output;
end;
run;

proc sort data=med_fmt;  *** this sort statement is a MUST;
  by FmtName;
run;
proc format cntlin = Med_Fmt;
run;

When these two formats are generated, two PUT statements can simply bring over the Dose and
Unit variables into the extracted dataset:

Data Hypertension2;
Set hypertension;
  Dose=put(medication_ID, dose.);
  Unit=put(medication_ID, unit.);
  ....
Run;
```

Please note, the PROC SORT Data=Med_Fmt statement is necessary. Without this Sort statement, the Dose and Unit formats would be still created but the mappings may be not what you want. Both format Does and Unit will map to ‘-‘.

CAUTIONS WHEN USING FORMAT

Format indeed is a very unique and flexible SAS tool. But there is one final thing we would like to bring your attention to.

Assume we have a dataset TEST like this:

Treat	Age	Gender	Outcome
A	50	M	130
B	55	F	150
B	560	F	180
....	...		

Further more, suppose TREAT variable has a permanently format $Treat as: “A”=”Treated”, “B”=”Placebo”. That is,

```
Data TEST;
.....
Format treat $treat8.;
Run;
```

Now let us run a comparison analysis like this:

```
proc mixed data=test;
  class trtcode gender;
  model m=trtcode gender age;
  ***lsmeans trtcode/cl;
  estimate 'A-B' trtcode 1 -1 /cl;
run;
```

The SAS System will happily produce the estimate and label the comparison as 'A-B' as wish, and there will not have any error or warning message. But in fact, the estimate from this analysis is for Treat B – Treat A, which is just the opposite of what is labeled 'A - B'. This is because the default order in the model is ORDERE = FORMATTED, and the format for group B="Placebo" has a higher alpha-beta order than the format for group A="Treated".

To eliminate this kind of over-looks, it would be a good habit to spell out the order option ORDER=DATA or to de-link the formats for the comparison groups in the model, like this:

```
proc mixed data=test;
  class trtcode gender;
  model m=trtcode gender age;
  lsmeans trtcode/cl;
  estimate 'A-B' trtcode 1 -1 /cl;
  FORMAT trtcode;  *** To remove the format if there is one;
run;
```

when writing the design matrix in an ESTIMATE statement, we should pay extra attention to those variables associated with formats. Otherwise, the results could be completely mis-interpreted.

CONCLUSION

SAS formats are indeed powerful tools for programmers. The formats can make our codes more flexible and more efficiency. But at the meantime, we should use them carefully in some situations as mentioned in the paper.

REFERENCES

[1] SAS Institute Inc., SAS SAS® 9.2 Language Reference Dictionary, Fourth Edition, http://support.sas.com/documentation/cdl/en/lrdict/64316/PDF/default/lrdict.pdf

[2] Andrew H. Karp, My Friend the SAS® Format, SUGI 30, http://www2.sas.com/proceedings/sugi30/253-30.pdf

[3] Tim Muir, Powerful Techniques For Data Processing Using Formats, SUGI 27, http://www2.sas.com/proceedings/sugi27/p086-27.pdf

[4] Nancy Croonen, Table Lookup: Techniques Beyond the Obvious, SUGI 27, http://www2.sas.com/proceedings/sugi27/p011-27.pdf

[5] Tatiana Nevmyrych and Jennifer Clark, Proc Format Advanced Techniques: Multi-label and Nested Formats, PharmaSUG 2007, http://www.lexjansen.com/pharmasug/2007/cc/cc10.pdf

[6] Jason Secosky and Janice Bloom, Getting Started with the DATA Step Hash Object, http://support.sas.com/rnd/base/datastep/dot/hash-getting-started.pdf

[7] Paul M. Dorfman and Koen Vyverman, Data Step Hash Objects as Programming Tools, SUGI 31, http://www2.sas.com/proceedings/sugi31/241-31.pdf

CONTACT

Your comments and questions are always valued and encouraged. Please contact the author at:

Shiqun (Stan) Li
Minimax Information Services
(908)240-8229
shiqun@gmail.com