Paper 078-2010

Preventive (reporting) is better with multi-label format

Sunil Gupta, Quintiles, Inc, Thousand Oaks, CA
Deepak Asrani, Medtronic, Inc, Mounds View, MN

INTRODUCTION:

Reporting on preventive services provides invaluable insight into utilization of health dollars as well as a feedback on how best costly claims can be avoided in future. Some companies even use these utilization statistics to provide incentives to their employees to have regular screenings to prevent occurrence of sudden catastrophic claims. Among other parameters, typically, preventive services are dependent on age and gender of members with more than one preventive service being appropriate. The paper will focus on use of multi-label option with proc format to report preventive service utilization.

TYPICAL CLAIM AND MEMBERSHIP DATASETS:

When describing our input datasets for purposes of this paper, we will limit the view to only those variables that are relevant to this paper.

MEMBER

Field Name	Description	Type and Length	Description
MEMBERID	Member Identifier	$5	A unique identifier for member
Age	Member Age	3	Age > 0
Gender	Member Gender	$5	Valid Values (‘Male’,’Female’)

CLAIMS

Field Name	Description	Type and Length	Description
MEMBERID	Member Identifier	$5	A unique identifier for member
CLMNBR	Claim Number	$5	One of the many claim numbers for the member
DIAGCDE	CPT Diagnosis Code	$5	Although up to 30 diagnosis codes can be submitted with a single claim, we will limit our discussion to 1
PROCCODE	HCPCS Procedure Code	$5	Procedure Code submitted on Professional Claim

TYPICAL PREVENTIVE SERVICE REPORT

Please note that the percentages are calculated as

Number of Unique Members (from Claims) in the age range and Gender receiving the service/

Number of Unique Members (from Member) in the age group and Gender

Type of Service	Ages	Recommended Frequency	Male	Female
Adult Preventive Office Visits	19 - 64	Female every 3-5 years; Male every 5 years	33.3% (1/ 3)	50.0% (2/ 4)
Cholesterol Screening, Female	44 - 64	Every 5 years	N/A (0/ 1)	50.0% (1/ 2)
Cholesterol Screening, Male	34 - 64	Every 5 years	66.7% (2/ 3)	N/A (0/ 4)

CREATE MEMBER SUMMARY BY AGE AND GENDER FOR DENOMINATORS

We summarize the membership to get denominator counts for preventive services. If we were not using multi-label option, we would have to duplicate the membership observations for each preventive service. In this case each member would have 2 repetitions. This would cause the membership table observations to increase exponentially.

In the creation of prvntage format below, the use of keyword multilabel is mandatory to avoid error message about overlapping ranges and failure to create a format.

```
proc format;
       value prvntage (multilabel notsorted)
       19 - 64 = "AP"
       44 - 64 = "CSF"
       34 - 64 = "CSM";
       run;
```

Please note that only multilabel-enabled procedures such as PROC MEANS, PROC SUMMARY, and PROC TABULATE can use multiple labels. All other procedures and the DATA step recognize only the **primary label.**

```
proc summary data=MEMBER;
       class GENDER;
       class AGE /mlf order=data preloadfmt;
       var age;
       types GENDER*AGE;
      output out=prevmemsummary(drop=_freq_ _type_  rename=(age=prevsvc))
       n=PATCOUNT;
       format age prvntage.;
       run;
```

CREATE FLAGS FOR PREVENTIVE SERVICES TO GET A COUNT OF SERVICES FOR NUMERATOR

We need to create flags for preventive services by navigating through Claims dataset and setting flags when all the conditions which identify utilization of preventive service are satisfied. For simplicity purposes we are limiting our code to 3 preventive services, viz. Adult Preventive Office Visits, Cholesterol Screening for Females and Cholesterol Screening for Males.

We will create formats for Procedure codes and Diagnosis codes using proc format. This is particularly useful when we don't want to hard code logic for preventive service in code. We can also use alternate means like an external excel file or a look up table to create these formats.

```
proc format;
       value $approc
       '99385' - '99387','99395' - '99397','99401' - '99404','99411' -
       '99412','99420','99429','G0344' = 'AP';

       value $csproc
       '80061','83700' - '83701','83704','83715' - '83716','83721'='CS';

       value $apdiag
       'V700','V703','V705' - 'V706','V708' - 'V709' = 'AP';
```

The format below is a very good example of the use of **nested formats.** What we are trying to achieve in this code is to apply PERCENT8.1 format for valid input values and force the final value to N/A in case of missing or 0.

```
      value pctmod
             . = 'N/A'
             0 = 'N/A'
             OTHER = [PERCENT8.1];
      run;
```

The following code is used to split the multilabel format prvntage into 3 different formats with age range for each of the preventive services. These new formats are AGEAP, AGECSM and AGECSF.

```
proc format library=work cntlout=agefmt;
       select prvntage;
       run;

data agefmt;
       set agefmt;
       fmtname=cats("AGE",label);
       run;

proc format library=work cntlin=agefmt;
       run;
```

The reader will appreciate the ease of maintenance of code with proc format. Infact, we are shifting a bunch of if..else statements in creation of the preventive service flags to a set of few formats defined above.

Thus, the complex code below:

```
data CLAIMS;
       merge CLAIMS MEMBER;
       by MEMBERID;

       AP=0; CSF=0; CSM=0;

       If (19 <= age <= 64) and
              (('99385' <= proccode <= '99387') or
              ('99395' <= proccode <= '99397') or
              ('99401' <= proccode <= '99404') or
              proccode in ('99411','99412','99420','99429','G0344') or
              diagcde in ('V700','V703','V705','V706','V708','V709')) then
       ADULTP=1;

       if GENDER = 'Female' and (44 <= age <= 64) and
      proccode in
      ('80061','83700','83701','83704','83715','83716','83721')
       then CSF=1;

       if GENDER = 'Male' and (34 <= age <= 64) and
      proccode in
      ('80061','83700','83701','83704','83715','83716','83721')
       then CSM=1;
       run;
```

was replaced by the simple code:

```
data CLAIMS;
       merge CLAIMS MEMBER;
       by MEMBERID;
       AP=0; CSF=0; CSM=0;

       If put(age, ageap.) = "AP" and (put(proccode, $approc.) = "AP" or
      put(diagcde, $apdiag.) = "AP" )  then  AP=1;
```

```
       If GENDER = 'Female' and put(age, agecsf.) = "CSF" and
       put(proccode, $csproc.) = "CS" then CSF=1;
       If GENDER = 'Male' and put(age, agecsm.) = "CSM" and
       put(proccode, $csproc.) = "CS" then CSM=1;
       run;
```

Summarize the preventive services by member.

```
proc summary data=CLAIMS nway missing;
       var AP CSF CSM ;
       class MEMBERID GENDER;
       output out=prevcaresummary(drop=_type_ _freq_) sum=;
       run;
```

If multiple preventive services per member have occurred, then set the count per member as 1 to remove duplicates.

```
data prevcaresummary(drop=_i);
       set prevcaresummary;
       array arr(3) AP CSF CSM;
       do _i=1 to 3;
              if arr[_i] ge 1 then arr[_i]=1;
       end;
       run;
```

```
proc summary data=prevcaresummary nway missing;
       var AP CSF CSM;
       class GENDER;
       output out=prevcaresummary(drop=_type_ _freq_) sum=;
       run;

proc transpose data=prevcaresummary out=prevcaresummary(rename=(_name_=prevsvc
       col1=CLMCOUNT));
       var AP CSF CSM;
       by GENDER;
       run;
```

MERGE PREVENTIVE SERVICE SUMMARY WITH MEMBER SUMMARY

Finally, we merge preventive service summary with numerator and member summary with denominator to generate the final dataset.

```
proc sql noprint;
       create table prevcare as select a.gender,a.prevsvc,a.patcount,
        b.clmcount,(case
       when patcount>0 then compbl(put(round(CLMCOUNT/PATCOUNT,0.001),pctmod.)||
       "  ("||put(CLMCOUNT, best.)||"/"||put(PATCOUNT, best.)||")")
       else ' '
       end) as value
       from prevmemsummary a inner join prevcaresummary b on
       a.gender=b.gender and a.prevsvc=b.prevsvc order by
       prevsvc,gender desc;
       quit;
```

GENERATE THE REPORT

The following 3 value statements demonstrate creation of formats using values obtained by applying the format created in the above value statement. This will help avoid triplication of column age in the final table and at the same time will help achieve a look of 3 different columns in the final output.

```
proc format;
value $prevsvc
       "AP"  = "Adult Preventive Office Visits"
       "CSF" = "Cholesterol Screening, Female"
       "CSM" = "Cholesterol Screening, Male";
value $prvnage
       "AP"  = "19 - 64"
       "CSF" = "44 - 64"
       "CSM" = "34 - 64";
value $prvfreq
       "AP"  = "Female every 3-5 years; Male every 5 years"
       "CSF" = "Every 5 years"
       "CSM" = "Every 5 years"; run;
```

Note that the final output requires us to have 2 value columns based on Gender. Typically to achieve this we would break the dataset into 2 datasets one which has Male values column and other with Female values column. Finally, we would merge the two datasets to get the 2 value columns in a single dataset.

Luckily we have a way to reduce the steps by defining usage of GENDER as across in the proc report steps. However, we hit a bottleneck since we have already defined value as a character variable.

No problem, proc format to the rescue. The steps below create a format using the value column and dummy column util,

```
data prevcare;
       set prevcare;
       start=_n_;
       end=_n_;
       util = _n_;
       fmtname="prevval";
       label=value;
       run;

proc format library=work cntlin=prevcare;
       run;
```

Finally, the proc report statement produces the final report in a single step.

```
proc report data=prevcare split='*';
       column prevsvc prevsvc=prvnage prevsvc=prvfreq gender,util;
       define prevsvc/ group order=formatted format=$prevsvc. 'Type of
Service*__ ';
       define prvnage/ group order=data format=$prvnage. 'Ages*__ ';
       define prvfreq/ group order=data format=$prvfreq.
'Recommended*Frequency*__ ';
       define gender / across order=data ' ';
       define util / analysis Format=prevval. ' ';
       run;
```

The code in this paper may appear overwhelming when reading for the first time. However, a quick run through the code using the datasets creation code in the Appendix will clarify the concepts used.

CONCLUSION

By following the simple steps outlined in this paper, it is possible to generate a complex preventive report. Further the use of proc format can avoid creating extra columns in large claims datasets and make the code easier to maintain at the same time.

This paper was the first step in defining the basic preventive report. Typically customers would like a report comparing multiple periods or comparing locations for a single or multiple periods. This dimensional increase is very simple to achieve using the proc report above.

APPENDIX

```
data MEMBER;
length MEMBERID $5 AGE 3 GENDER $6;
input MEMBERID $ AGE GENDER $;
datalines;
A0001 35 Male
A0002 48 Female
A0003 12 Male
A0004 56 Male
A0005 52 Female
A0006 39 Male
A0007 40 Female
A0008 18 Male
A0009 42 Female
A0010 9 Female
;

data CLAIMS;
length MEMBERID $5 CLMNBR $5 DIAGCDE $5 PROCCODE $5;
INFILE datalines missover;
input MEMBERID $ CLMNBR $ DIAGCDE $ PROCCODE $ ;
datalines;
A0001 M0001 V700
A0001 M0002 99500 80061
A0002 F0001 91500 83721
A0003 M0012 V700
A0004 M0102 99500 80061
A0005 F1021 90000
A0005 F1021 90100 G0344
A0006 M9444 85000
A0007 F9999  80061
A0008 M0001  V700
A0009 F5678 V700 80061
A0010 F4545 V700 80061
;
```

REFERENCES

Ron Cody, *“Using Advanced Features of User-defined Formats and Informats*”, SAS Global Forum 2009
Andrew H. Karp, *“My Friend the SAS® Format”*, SUGI 30

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:
Sunil Gupta
Phone: (805)-584-6182
E-mail: Sunil@GuptaProgramming.com
Sunil is a bestselling SAS author and global corporate trainer. Currently, he is the Director of Statistical Programming at Quintiles. Most recently, he was honored to be one of the top 100 notable people in the Medical Device Industry for 2008. He has been using SAS® software for over 15 years and is a SAS Base Certified Professional. He is also the author of Quick Results with the Output Delivery System, Data Management and Reporting Made Easy with SAS Learning Edition 2.0, and Sharpening Your SAS Skills. Most recently, he is teaching his latest popular course, Best Practices in SAS Statistical Programming in Regulatory Submission.

Deepak Asrani
Phone: (763)-526-2799
E-mail: deepak.g.asrani@medtronic.com
Deepak has over 13 years of SAS programming experience in Pharmaceutical, Biotech and Healthcare industry. He is a SAS Certified Advanced Programmer.