MWSUG 2018 - Paper SP-048

Using Multilabel Formats with PROC SUMMARY to Generate Report Data with Overlapping Time Segments

John Schmitz, Luminare Data, Omaha, NE

ABSTRACT

SAS® introduced the multi-label format (MLF) in Version 8. Yet, few users are familiar with the MLF or its unique capabilities. MLFs are used for data summarization where the same observation may be classified into 2 or more levels, simultaneously. This paper shows how multi-label formats can be used to generate time segments with overlapping periods. Core steps include creating the multi-label format definition, applying MLFs to CLASS variables within PROC SUMMARY, and properly understanding the results.

INTRODUCTION

Multilabel formats can provide a powerful tool for quickly summarizing data. These formats offer a unique method to summarize records that may legitimately be assigned to multiple categories (or labels) at the same time. Although it may not be legitimate to categorize a transaction as occurring in 2017 and 2018, it is certainly possible for the transaction to occur in 2018 and in October.

Although the focus here is upon date formats, multilabel formats (MLFs) can be applied anywhere that transactions may be assignment to non-mutually exclusive categories. For instance, MLFs can help track sales activities with organizations with multiple locations. Here the MLF could track activity by location, as well as tracking all activity associated with the parent organization by accumulating all transactions across the parent's many individual sites.

This paper presents the fundamentals for creating and using MLFs. Specific sections include:

- Defining MLFs
- Application of MLFs
- SUMMARY Procedure Overview
- Building MLFs for date-based summaries
- Reviewing the Summary results

DEFINING MLFS

A MLF or multilabel format is a user-defined custom format. By default, the FORMAT procedure restricts entries so that overlapping and redundant date cannot be created. This check helps avoid potentially ambiguous results. Consider a simple grade assignment format:

```
proc format;
    value $ GradeFmt
          'Paul'    = 'A'
          'Mary'    = 'A'
          'Pete'    = 'B'
          'Henry'   = 'C'
          'Jessica' = 'B'
          'Pete'    = 'A';
run;
```

Applying the data as a format creates an issue, what grade does Pete get? Line3 says, Pete gets a B while line 6 gives Pete an A. When these entries are process with the FORMAT procedure, an ERROR is generated:

```
ERROR: This range is repeated, or values overlap: Pete-Pete.
```

The FORMAT procedure allows for an option of MULTILABEL which essentially turns off the check for overlapping data. Hence, using the MULTILABEL option, these data will generate a format. The multilabel option is invoked by including the keyword (multilabel) in parentheses the format name:

```
value $ GradeFmt (multilabel)
```

Hence a multilabel format is a format that does not uniquely assign a label for a given input value. In this case, Pete receives both A and B.

APPLICATION OF MLFS

Even though the multilabel option allows the creation of the GRADE format in the previous example, it still does not answer what grade Pete gets. That is because grades in this simple example are mutually exclusive.

Now what if the format is for the pets that these kids have at home. The format may look like this:

```
proc format;
    value $ PetFmt (multilabel)
           'Paul'     = 'Cat'
           'Mary'     = 'Dog'
           'Pete'     = 'Dog'
           'Henry'    = 'Hamster'
           'Jessica'  = 'Cat'
           'Pete'     = 'Turtle';
run;
```

In this case, Pete has two pets, a dog and a turtle. There are five children total with 2 Cats, 2 Dogs, 1 Hamster, and 1 Turtle. A traditional data join approach would report 6 children with 6 pets. Deduping could produce 5 children and 5 pets. MLFs provide a more convenient method to manage this data challenge, allowing a SUMMARY procedure to accurately report the number of each pet, as well as the correct number of children.

The use of MLFs are restricted to specific procedures within SAS since they have minimal relevance outside of a summarization process. SAS supports MLFs in only select procedures, including:

- SUMMARY / MEANS[1]
- TABULATE
- REPORT

The MLF format is only used with a CLASS statement for SUMMARY and TABULATE and in a DEFINE statement for REPORT. The focus here is on its use with SUMMARY procedure. Those interested in TABULATE and REPORT can review SAS documentation for more specific details relative to these procedures.

SUMMARY PROCEDURE OVERVIEW

In the SUMMARY procedure, the CLASS statement is used to define categorical variables for the desired data operation. Additional variables may be included on the CLASS statement when multiple categorical

[1] SUMMARY and MEANS procedures share identical features but with different defaults for the PRINT/NOPRINT option and different behavior when no VAR statement is included. This paper will refer to the SUMMARY procedure, but all comments apply to either SUMMARY or MEANS.

items are required. Formats can be applied to categorical variables as needed and the categorical values are derived from the formatted value. The options MISSING and NWAY, along with the TYPES statement influence which combinations of CLASS variables are returned by PROC SUMMARY. When MLFs are included in a SUMMARY procedure, the MLF formatted variables must be included on a separate CLASS statement than non-MLF items and include a statement option MLF.

As a demonstration of PROC SUMMARY, a dataset (KIDS) can be created with 3 fields: NAME, PET, and CNT. NAME will contain the child's name, PET will initial contain the child's name but will be converted to pet type after applying the MLF format below, and CNT is a simple column of 1 for an analysis variable. This dataset can be used with PROC SUMMARY as:

```
proc summary data=kids;
    class name;
    class pet / mlf;
    format pet $PetFmt.;
    var cnt;
    output out=test sum=;
run;
```

The resulting output shows 5 children and six pets and reports group counts as expected (see Table 1).

Table 1. SUMMARY Results using the PET MLF.

NAME	PET	CNT
		5
	Cat	2
	Dog	2
	Hamster	1
	Turtle	1
Henry		1
Jessica		1
Mary		1
Paul		1
Pete		1
Henry	Hamster	1
Jessica	Cat	1
Mary	Dog	1
Paul	Cat	1
Pete	Dog	1
Pete	Turtle	1

In Table 1, the first line shown is the total for all groups within the data and shows 5 for the 5 input records on the dataset. The next 4 rows show counts by type of Pet. There is a total of 6 pets shown, since Pete has 2 pets assigned by the format. The next section shows counts by child's name. There are 1 each for the 5 children included. Even though the process correctly added Pete's second pet, it also correctly reports that there is only 1 record for Pete. The Final section shows all Child/Pet combinations. The process again reflects Pete twice, one with each Pet. This is accurate that 6 unique child/pet combinations exist, even though the input data only contains 5 total records.

BUILDING MLFS FOR DATE-BASED SUMMARIES

When constructing DATE-based reporting, it is commonly desired to assign records to multiple time periods. Constructing the appropriate MLF requires assigning a starting and ending value for each time period. In some cases, these may be represented by continuous blocks such as 01SEP2018 to 30SEP2018 as 2018M9. Other cases may require multiple entries to define the appropriate grouping such as 02SEP2018, 09SEP2018, 16SEP2018, 23SEP2018, 30SEP2018 all belonging to 'SUNDAYS'. Both variations can be supported in the same format definition using MLFs. However, entering these combinations into a PROC FORMAT may become tedious.

An alternative is to construct the format definition using a DATA step and basic DO loop constructs. This can greatly simplify the format creation process. Once the DATA step is completed, the FORMAT procedure option CNTLIN can be invoked to generate the MLF from the DATA step definition.

The DATA step requires 5 fields to define date ranges for use with the CNTLIN option:

- START: beginning input data range (starting date)
- END: ending input data range (ending date)
- LABEL: value assigned to range as a CHAR field
- FMTNAME: value to assign as the format name
- HLO: optional values field for HIGH / LOW / OTHER / and in this case MULTILABEL

A basic dataset definition may look like:

```
data DateMLF;
    format
          start
          end          datetime16.
          label        $30.
          fmtname      $20.
          hlo          $3.;
    keep start -- hlo;

    retain
          FmtName      'DateMLF'
          HLO    'M';
```

By convention, it may be helpful to create format names that end in MLF so they can be readily distinguished from single-label formats. Also by convention, consider matching the DATA table name with the format name. Neither are required by SAS, but these can be convenient when complex jobs are created.

The code block above accomplishes 3 tasks. First the FORMAT statement defines the fields, field types and lengths as needed for the CNTLIN function. Second, the KEEP statement is used so that any remaining fields created within the DATA step will be dropped. Third, the RETAIN statement is used to set FMTNAME to the desired name (DateMLF) and assign 'M' to HLO. These values are assigned to every record created in the DATA step unless modified by later code.

The remaining effort is to create observations with START, END and LABEL values as desired for the format. In the example here, the input transactions have DATETIME values, so START and END values will need to be DATETIME. Following the example above, the PROC SUMMARY will use a CLASS statement that includes the DATETIME variable on the MLF Class statement and a FORMAT statement to apply the MLF format to the field.

This example will create 3 distinct time groupings:

- Current plus Past 6 Quarters, by Quarter
- Current plus Past 6 Months, by Month
- Past 13 weeks (91 days), by Day of Week

These date groups definitions will be generated by DO loops. Each group has a macro variable assigned: Nqtrs, Nmons, and Ndow, respectively to define the starting date range for each group. The DO loop executes in reverse increment so that the loop processes from start to end of the date range desired. The LABEL value includes a group identifier (QTR:, MON: or DOW:). The SUMMARY results will return in sorted order by the LABEL value, so this identifier keeps all results for the group together within the results. Since the input values are DATETIME, the loop uses the DHMS (days, hours, minutes, seconds) function to convert results to DATETIME values. The INTNX function is used to simplify calculation of START and END date values. In most applications, the macrovar &CURR_DATE would be the current system date but could also be set to an effective date for the reports being generated when it is desirable to backdate the reporting process.

The QUARTER group loop can be coded as:

```
do qtr = &Nqtrs. to 0 by -1;
      start = dhms(intnx('qtr',&curr_date.,-qtr,'b'),0,0,0);
      end   = dhms(intnx('qtr',&curr_date.,-qtr+1,'b'),0,0,0)-1 ;
      label = catt('QTR:', put(datepart(start),yyq6.));
      output;
end;
```

The CODE loops from &Nqtrs, to 0, decreasing by 1 (quarter) on each iteration. By including 0, the process includes not only the past quarters, but also adds the current quarter to date data. For 2018Q1, the start and end would be ‘01JAN2018:00:00:00’ and ‘31MAR2018:23:59:59’. The LABEL would resolve to ‘QTR:2018Q1’. These values are output to the dataset, along with values for each quarter in the loop.

A similar loop is constructed for MONTH group, replacing the quarter-based terms with equivalent MONTH-based values:

```
do mon = &Nmons. to 0 by -1;
      start = dhms(intnx('month',&curr_date.,-mon,'b'),0,0,0);
      end   = dhms(intnx('month',&curr_date.,-mon+1,'b'),0,0,0)-1 ;
      label = catt('MON:', put(datepart(start),yymm7.));
      output;
end;
```

The DOW function requires a similar but different loop structure. Since the records are not continuous, the process loops over each day and assigns a label. Labels are assigned using the DOWNAME3. Format which generates the first 3 characters of the Day of Week name. This loop iterates over days and assigns a label to each day within the range. All Sundays are assembled as a reporting unit, but not within the loop structure, but rather because they all receive identical labels. That loop appears as:

```
do dow = &Ndow. to 0 by -1;
      start = dhms(intnx('day',&curr_date.,-dow,'b'),0,0,0);
      end   = dhms(intnx('day',&curr_date.,-dow+1,'b'),0,0,0)-1 ;
      label = catt('DOW:', put(datepart(start),downame3.));
      output;
end;
```

Finally, a catch-all entry is included. Any date included in the summary that does not otherwise receive a formatted value will enter the SUMMARY table unformatted, and just not aggregated with other records. To eliminate this concern, the OTHER function is used and assigned by SAS to any entry that is yet to receive one or more labels. To work as desired, this code must execute after all other assignments have been completed.

```
start = .;
end = .;
hlo = 'Mo';
label = 'Other';
output;
```

The HLO value now sets two flags, MULTILABEL and OTHER. Date values that are not captured by any of the other date ranges will be assigned the label ‘OTHER’. OTHER will be used to remove any unwanted data items below within the SUMMARY procedure.

Complete code showing the full construction of these format, summary and related items is included as an appendix to the paper.

REVIEWING THE SUMMARY RESULTS

The date format from above can be used in conjunction with the TIMEDATA file available in SASHELP (SASHELP.TIMEDATA). The dataset contains 40,330 records and two variables, DATETIME and VOLUME. PROC SUMMARY can be used to find total VOLUME for each of the time periods defined above.   Transactions in table occur between July 1997 and Oct 2000. Given the age of the transaction history, the format creation logic is run with a CURR_DATE set to ‘22OCT2000’, the last transaction date in the file.

The only categorical field in the report will be DATETIME which will be formatted using the MLF. Since there are no categorical fields that are not using MLFs, only one class statement is needed in this case.

The PROC SUMMARY code for the analysis is:

```
proc summary data=SASHELP.TIMEDATA NWAY;
    where put(datetime,dateMLF.) NE 'Other';
    class datetime / MLF;
    format datetime DateMLF.;
    var volume;
    output out=summary (drop = _freq_ ) sum=;
run;
```

Recognize that the MLF is used within the WHERE clause to drop unnecessary records. This works since records with ‘OTHER’ will be single-label entries and have no overlap with the records that receive one or more of the remaining labels. The NWAY option is used so that the output table only includes the desired DATE ranges and excludes the sum across all records.

The procedure results are output to a dataset named SUMMARY. From here, SAS programmers should be able to alter date labels, organize, assign and filter as desired. Results from the procedure are shown in Table 2. These results show breakouts for each of the desired time periods.

Validation results are show in Table 3 . These results were constructed using three distinct SUMMARY procedures, each using a pre-defined DATE format and a WHERE clause to restrict DATE ranges to match those in the previous summary. A DATE file is generated in place of the DATETIME value used before to match requirements of these pre-defined formats. Other than the change in ordering of the DOW fields and absence of the GROUP label added to each section in the DateMLF, output results are the same for each method. These results demonstrate that the MLF successfully recreated the desired date ranges while processing the SUMMARY in a single procedure call.

Table 2. SUMMARY Results by Period Using DateMLF.

DATETIME	_TYPE_	VOLUME
DOW:Fri	1	40,420
DOW:Mon	1	76,052
DOW:Sat	1	3,882
DOW:Sun	1	1,130
DOW:Thu	1	52,411
DOW:Tue	1	53,131
DOW:Wed	1	77,388
MON:2000M04	1	82,810
MON:2000M05	1	97,518
MON:2000M06	1	139,769
MON:2000M07	1	110,005
MON:2000M08	1	104,285
MON:2000M09	1	92,707
MON:2000M10	1	56,721
QTR:1999Q2	1	245,279
QTR:1999Q3	1	298,350
QTR:1999Q4	1	332,617
QTR:2000Q1	1	316,333
QTR:2000Q2	1	320,097
QTR:2000Q3	1	306,997
QTR:2000Q4	1	56,721

Table 3. Valiadation Results for SUMMARY Based on Three Individual Runs.

DATE	_TYPE_	VOLUME
Sun	1	1,130
Mon	1	76,052
Tue	1	53,131
Wed	1	77,388
Thu	1	52,411
Fri	1	40,420
Sat	1	3,882
2000M04	1	82,810
2000M05	1	97,518
2000M06	1	139,769
2000M07	1	110,005
2000M08	1	104,285
2000M09	1	92,707
2000M10	1	56,721
1999Q2	1	245,279
1999Q3	1	298,350
1999Q4	1	332,617
2000Q1	1	316,333
2000Q2	1	320,097
2000Q3	1	306,997
2000Q4	1	56,721

CONCLUSIONS

The paper demonstrates the use of MLFs to generate multiple date range summaries within a single PROC SUMMARY call. The process can provide highly flexible dates ranges. Various groups can be created and readily automated to manage varying date ranges and reporting date requirements. Validations against single pass PROC SUMMARY calls show that the two processes create the same result, despite the use of overlapping date ranges within the MLF.

Due to its high degree of flexibility, the process describe here can be readily applied to many reporting requests. It improves efficiency by reducing the number of data passes required to complete multiple date range breakouts that may be required.

Multilabel formats have many uses beyond what is described here, but the example of DATE formats provides one application that could be relevant to a large number of SAS developers.

CONTACT INFORMATION

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

John Schmitz
Luminare Data
john.schmitz@luminaredata.com
www.luminaredata.com

APPENDIX: CODE USED IN PAPER

```
** CODE TO PRODUCE MULTILABEL FORMAT AND PROC SUMMARY **;
** GENERATES CURRENT AND LAST 6 QUARTERS              **;
** ALSO ADDS CURRENT AND LAST 6 MONTHS                **;
** ALSO INCLUDES LAST 13 WEEKS BY DAY OF WEEK         **;

** SET CURRENT DATE  **;
%let curr_date = %sysfunc(date());
%put CURR_DATE:
          &curr_date (%sysfunc(putn(&curr_date,date9.)));

** BACKDATE CURR_DATE FOR STUDY DATA **;
** WOULD NOT NORMALLY BE REQUIRED.   **;
%let curr_date = %sysfunc(putn('22OCT2000'd,8.));
%put CURR_DATE:
          &curr_date (%sysfunc(putn(&curr_date,date9.)));


** SET NUMBER OF QUARTERS FOR REPORT **;
** ADD A PARAMETER FOR NUMBER OF MONTHS **;
%let NQtrs = 6;
%let NMons = 6;
%let Ndow  = 91;

** CREATE DATASET FOR FORMAT DATA  **;
** WILL USE WITH CNTLIN FOR FORMAT **;
data DateMLF;
    ** DEFINE VARIABLES NEEDED FOR CNTLIN **;
    format
          start
          end         datetime16.
          label       $30.
          fmtname     $20.
          hlo         $3. ;
    keep start -- hlo;

    ** SET AND RETAIN SELECT FIELDS **;
    retain
          FmtName     'DateMLF'
          HLO         'M';

    ** CREATE DATE RANGE AND LABELS FOR QUARTERS **;
    do qtr = &Nqtrs. to 0 by -1;
          start =
                dhms(intnx('qtr',&curr_date.,-qtr,'b'),0,0,0);
          end   =
                dhms(intnx('qtr',&curr_date.,-qtr+1,'b'),0,0,0)-1;
          label = catt('QTR:', put(datepart(start),yyq6.));
          output;
    end;

    ** CREATE DATE RANGE AND LABELS FOR MONTHS **;
    do mon = &Nmons. to 0 by -1;
          start       =
                dhms(intnx('month',&curr_date.,-mon,'b'),0,0,0);
          end   =
                dhms(intnx('month',&curr_date.,-mon+1,'b'),0,0,0)-1;
          label = catt('MON:', put(datepart(start),yymm7.));
          output;
    end;

    ** CREATE DATE RANGE AND LABELS FOR WEEKDAYS **;
    do dow = &Ndow. to 0 by -1;
          start       =
                dhms(intnx('day',&curr_date.,-dow,'b'),0,0,0);
          end   =
                dhms(intnx('day',&curr_date.,-dow+1,'b'),0,0,0)-1;
          label = catt('DOW:', put(datepart(start),downame3.));
          output;
    end;

    ** ADD OTHER CATEGORY AS CATCH-ALL (LAST) **;
    start = .;
    end = .;
    hlo = 'Mo';
    label = 'Other';
    output;
run;


** GENERATE FORMAT FROM DATASET **;
proc format cntlin=DateMLF;
run;

** SHOW CONTENTS AND DATE RANGE ON SASHELP.TIMEDATA **;
proc contents data=SASHELP.TIMEDATA ;
run;

proc sql;
    select min(datetime) format=datetime16. as min,
          max(datetime) format=datetime16. as max
    from SASHELP.TIMEDATA;
quit;


** APPLY FORMAT IN PROC SUMMARY **;
proc summary data=SASHELP.TIMEDATA NWAY;
    ** NOTE:  Can use MLF Here BUT CAUTION IS REQUIRED **;
    where put(datetime,dateMLF.) NE 'Other';
    class datetime / MLF;
    format datetime DateMLF.;
    var volume;
    output out=summary (drop = _freq_ ) sum=;
run;


data validate;
    set SASHELP.TIMEDATA;
    date = datepart(datetime);
run;


** VALIDATE PROC SUMMARY BY QTR **;
proc summary data=validate NWAY;
    where date >= '01APR1999'd;
    class date ;
    format date yyq6.;
    var volume;
    output out=summary_qtr  (drop = _freq_ ) sum=;
run;


** VALIDATE PROC SUMMARY BY QTR **;
proc summary data=validate NWAY;
    where date >= '01APR2000'd;
    class date ;
    format date yymm7.;
    var volume;
    output out=summary_mon  (drop = _freq_ ) sum=;
run;


** VALIDATE PROC SUMMARY BY QTR **;
proc summary data=validate NWAY;
    where date >= '23JUL2000'd;
    class date ;
    format date downame3.;
    var volume;
    output out=summary_dow  (drop = _freq_ ) sum=;
run;
```