PharmaSUG 2012 – CC30

A Mean Way to Count, Enumerating the Values of Multiple Variables Using Formats with the Means Procedure

Rod Norman, PharmaNet/i3 (an InVentiv Health Company) San Diego, CA

ABSTRACT

Perhaps the most common task of the clinical SAS programmer is to count things. Things like adverse events, laboratory value grade shifts, demographics, etc. and to do this over a wide variety of scenarios. To accomplish these assignments, programmers should consider using features of the MEANS procedure in conjunction with the MULTILABEL option of the VALUE statement within the FORMAT procedure. By invoking PROC MEANS on datasets that included only variables listed in CLASS statements and not using a VAR statement, efficiently produced output will contain only counts for all combinations of the CLASS variables. By using the PRELOADFMT and MLF keyword options on CLASS listed variables, a variety of counting scenarios can be applied to the input data. A TYPES statement identifies just the combinations of variables that are desired in the dataset created using OUTPUT OUT=. This routine is unburdened with macro variables and offers transparent code that can be easily modified should specifications change, counting scenarios increased or other variables added.

INTRODUCTION

There are times when a clinical SAS® programmer has the opportunity to write code using the advance methodologies of SAS/STAT software. A much more common task, however, is counting. Counts can be subject incidences or event incidences on a variety of demographic variables, adverse events (AEs), laboratory grades, etc... Frequently, study leads will replicate an existing program or commission the creation of a program macro to handle counting. Such macros grow in length and complexity as the study evolves and new protocols are added. In addition, new specifications or additional outputs from statisticians can require careful code modifications and complex branching within these macros.

The purpose of this paper is to present the basics of counting using features and options of PROC MEANS. I argue that code used in this approach is straightforward, adaptable to many situations and easily modified to handle additional requested scenarios. When used with the MULTILABEL options of PROC FORMAT, additional functionally is achieved. The purpose of such code is readily apparent and potentially appreciated by programmers who inherit it in an ongoing study.

COUNTING WITH THE MEAN PROCEDURE AND FORMATS

First consider a scenario where subjects are to be counted in a dataset of adverse events. The first step would be to count the number of subjects by first creating a dataset where each subject and his associated treatment group are in a single obs.

One method to create this dataset (subjects) from an adverse event data set (ae1):

```
proc SQL;
   create table subjects as select distinct TrtmtCode, SubjectID, age  from ae1
   ;quit;
```

BASIC COUNT WITH PROC MEANS

To count subjects within treatment groups invoke PROC MEANS. By restricting the input data set to character variables (also numeric variables if listed in a CLASS statement) and using no VAR statement, the OUPUT data set produced will contain just counts of obs for variables listed in the CLASS statement. No default descriptive statistics will be produced.

```
proc means  data=subjects (keep=TrtmtCode SubjectID) noprint;
    class TrtmtCode  ;
    output  out=ctsubj ;
     run;
```

COUNTS WITH FORMATS APPLIED IN PROC MEANS

To extend this method to a count of subjects within treatment groups by age categories, a FORMAT can be applied before invoking PROC MEANS.

```
proc format ;
   value  trtf 1='Placebo' 2='Drug';
   value  agef low-50='less than 50'  51-60='50 to 60'  61-high='above 60';

proc means  data=subjects (keep=TrtmtCode SubjectID age) noprint;
   format TrtmtCode trtf.  age agef.;
   class TrtmtCode age ;
   output  out=ctsubj; run;
```

As constructed, PROC MEANS produces the data set ctsubj as printed below in Output 1.

TrtmtCode	AGE	_TYPE_	_FREQ_
.	.	0	150
.	less than 50	1	30
.	50 to 60	1	67
.	above 60	1	53
Placebo	.	2	63
Drug	.	2	87
Placebo	less than 50	3	13
Placebo	50 to 60	3	30
Placebo	above 60	3	20
Drug	less than 50	3	17
Drug	50 to 60	3	37
Drug	above 60	3	33

Output 1. Basic OUTPUT data from PROC MEANS using formats

This data set contains the variable _freq_ whose values are counts for all CLASS variables as well as combinations of CLASS variables. These combinations are designated by the value of the _type_ variable. The counts for age categories are achieved by use of the range value format *agef*. applied to numeric age. The printed values of TrtmtCode and Age are not stored in the data set but are format representations. The actual values can be seen by removing the formats in the ctsubj data set as shown in the following code and Output 2. It is important to remember this fact if post PROC processing is required on the data.

```
proc means  data=subjects (keep=TrtmtCode SubjectID age) noprint;
```

TrtmtCode	AGE	_TYPE_	_FREQ_
.	.	0	150
.	39	1	30
.	51	1	67
.	61	1	53
1	.	2	63
2	.	2	87
1	39	3	13
1	51	3	30
1	61	3	20
2	39	3	17
2	51	3	37
2	61	3	33

Output 2. Results from PROC MEANS with formats removed

The value of age is the lowest age of the specific age range and the Treatment Code is the numeric value in the original data.

COUNTING WITH PROC MEANS AND MULTILABEL FORMATS

BASICS OF A MULTILABEL COUNT

To simply counting and produce, in a more direct manner, the desired output data, consider using PROC MEANS with a MULTILABEL format specifications. Here, the formats are redefined as multiple labels before executing PROC MEANS.

```
proc format ;
   value  trtf  (multilabel notsorted)  2='Drug' 1='Placebo'  1,2='Total' ;
   value  agef  (multilabel notsorted)  low-50='less than 50' 51-60='50 to 60'
                                        61-high='above 60';
```

This specification declares that values of a variable assigned this format be treated as multiple labels. The keyword NOTSORTED instructs SAS Procedures capable of using this type of format to keep values in the order specified by the VALUE statement. This is why treatment group 2 (Drug) is listed first because the specifications have this treatment as the first column of the requested table. In addition, reviewers wish a Total column for all treatment groups, and this request is accommodated by the multiple label format, which allows creation of this column by 'double' counting the treatments 1 and 2 in this grouping. PROC MEANS is constructed as below with some additional options. The coder also decides to use the MULTILABEL formatting for age for a reason detailed below.

```
proc means  data=subjects (keep=TrtmtCode SubjectID age) noprint;
   format TrtmtCode trtf.  age agef.;
   class TrtmtCode age / preloadfmt mlf order=data ;
   types TrtmtCode*age ;
   output  out=ctsubj ; run;
```

The CLASS statement is now specified with three options. PRELOADFMT instructs PROC MEANS to use formats for categories in counting. MLF is to specify that the format for trtf. and agef. are declared as MULTILABEL. ORDER=DATA instructs that the output data be arrange according to the order of vales listed in the VALUE statement. The OUTPUT data set created by OUT= is shown in Output 3.

TrtmtCode	AGE	_TYPE_	_FREQ_
Drug	less than 50	3	17
Drug	50 to 60	3	37
Drug	above 60	3	33
Placebo	less than 50	3	13
Placebo	50 to 60	3	30
Placebo	above 60	3	20
Total	less than 50	3	30
Total	50 to 60	3	67
Total	above 60	3	53

Output 3. Resulting OUTPUT dataset from PROC MEANS with MULTILABEL format applied

The values of the TrtmtCode column are arranged as desired with Drug counts followed by Placebo counts and ending by the Totals of both treatment groups. Not obvious in the printed output, the values of both TrtmtCode and Age have been converted to the character values printed, a result of using the MULTILABEL option. The code

```
proc print data=ctsubj noobs; format TrtmtCode age; run;
```

produces the same output as shown above. Thus no further post Proc processing is necessary.

EXTENSION OF MULTILABEL COUNTING TO OVERLAPPING VALUES

Now consider a request to count the most severe adverse events occurring within subjects using Common Terminology Criteria for Adverse Events (CTCAE). To perform a count by subject incidence, a derive data set is created. The following is one method to derive such a data set that contains the maximum within subject severity of an adverse event as described by the System Organ Class and Preferred Term using the Medical Dictionary for Regulatory Activities.

```
proc SQL;
   create table maxgrd as
   select distinct TrtmtCode, SubjectID, max(SeverCode) as maxCTC,
           SocTerm, PrefTerm
   from ae1
   group by  TrtmtCode, SubjectID, SocTerm, PrefTerm
 ;quit;
```

The following FORMAT statement is typical of such a count.

```
proc format;
   value  trtf  2='Drug' 1='Placebo';
   value  maxCTCf  1='Mild' 2='Moderate'  3='Severe' 4='Disabling' 5='Death' ;
```

Suppose, however, a more complex accounting of adverse events is desired. Supplementing the direct counts of maximum AE grades as above, the statistician requests rows for AE counts of any grade as well as counts of 'Severe and Disabling' within subjects. To obtain these counts directly without extensive code modification, the format VALUE statement is changed as below.

```
proc format;
   value  trtf (multilabel notsorted) 1,2,3,4,5='Any'  3,4='Severe and Disabling'
                1='Mild'  2='Moderate'  3='Severe'  4='Disabling'  5='Death';
```

The code for counting using PROC MEANS is the same as before relying on the format change to instruct how the counts are made. Below, I emphasize this fact by letting the format values be macro variables.

```
%let trtfmt=trtf. ;
%let ctcfmt=maxCTCf. ;
proc means  data=maxgrd (keep=SocTerm PrefTerm maxCTC TrtmtCode)  noprint;
   format TrtmtCode &trtfmt   maxCTC &ctcfmt ;
   class  SocTerm PrefTerm ;
   class maxCTC TrtmtCode  / preloadfmt mlf order=data  ;
   types   TrtmtCode*maxCTC*SocTerm*PrefTerm;
   output out=ctgrd ;
```

Here two CLASS statements are used. The first for the AE text terms, SocTerm and PrefTerm, whose values are extensive, data driven and not effectively formatted. The second is for maxCTC and TrtmtCode to which the listed options are applied as before. The overall order of the CLASS variables determines the arrangement of the data in the OUT= dataset (ctgrd) reflecting the desire to have the data arranged by System Organ Class, Preferred Term, maximum within subject AE value and treatment. Within variables maxCTC and TrtmtCode, the ORDER=DATA instructs SAS to arrange the values according to the order listed in the VALUE statement of PROC FORMAT. An example of the output from the code follows in Output 4, restricted here to the Preferred Term of 'Tinnitus' for illustration.

SocTerm	PrefTerm	maxCTC	TrtmtCode	_TYPE_	_FREQ_
EAR AND LABYRINTH	Tinnitus	Any	Drug	15	11
EAR AND LABYRINTH	Tinnitus	Any	Placebo	15	12
EAR AND LABYRINTH	Tinnitus	Any	Total	15	23
EAR AND LABYRINTH	Tinnitus	Severe and Disabling	Drug	15	5
EAR AND LABYRINTH	Tinnitus	Severe and Disabling	Placebo	15	1
EAR AND LABYRINTH	Tinnitus	Severe and Disabling	Total	15	6
EAR AND LABYRINTH	Tinnitus	Mild	Drug	15	5
EAR AND LABYRINTH	Tinnitus	Mild	Placebo	15	5
EAR AND LABYRINTH	Tinnitus	Mild	Total	15	10
EAR AND LABYRINTH	Tinnitus	Moderate	Drug	15	1
EAR AND LABYRINTH	Tinnitus	Moderate	Placebo	15	6
EAR AND LABYRINTH	Tinnitus	Moderate	Total	15	7
EAR AND LABYRINTH	Tinnitus	Severe	Drug	15	2
EAR AND LABYRINTH	Tinnitus	Severe	Placebo	15	1
EAR AND LABYRINTH	Tinnitus	Severe	Total	15	3
EAR AND LABYRINTH	Tinnitus	Disabling	Drug	15	3
EAR AND LABYRINTH	Tinnitus	Disabling	Total	15	3

Output 4. Resulting OUTPUT data for Preferred Term Tinnitus from PROC MEANS on data set maxgrd

Through some simple post proc processing, a derived variable is created (named display) defined as the counts and percentage of total subjects within each treatment classification. (Code not shown). The data can now be quickly structured for output to the final table using the TRANSPOSE procedure as follows.

```
proc transpose  data=ctgrd out=ctgrdT ;
    by  SocTerm PrefTerm  maxCTC  notsorted;
    id TrtmtCode;
    var display;
```

The resulting data for 'Tinnitus' is shown below in Output 5.

SocTerm	PrefTerm	maxCTC	Drug	Placebo	Total
EAR AND LABYRINTH	Tinnitus	Any	11 (13%)	12 (19%)	23 (15%)
EAR AND LABYRINTH	Tinnitus	Severe and Disabling	5 (6%)	1 (2%)	6 (4%)
EAR AND LABYRINTH	Tinnitus	Mild	5 (6%)	5 (8%)	10 (7%)
EAR AND LABYRINTH	Tinnitus	Moderate	1 (1%)	6 (10%)	7 (5%)
EAR AND LABYRINTH	Tinnitus	Severe	2 (2%)	1 (2%)	3 (2%)
EAR AND LABYRINTH	Tinnitus	Disabling	3 (3%)		3 (2%)

Output 5. Transposed data for Preferred Term Tinnitus

Note that there is a missing value for Placebo in the row 'Disabling'. There were no Placebo subjects with an AE in this category. The reader familiar with the COMPLETETYPES option that can be used in PROC MEANS would note that this keyword could be used to populate this missing value with 0 (0%) as well as generate an entire row of zeros counts for the 5='Death' category as shown in Output 6.

EAR AND LABYRINTH	Tinnitus	Death	0 (0%)	0 (0%)	0 (0%)

Output 6. Resulting data row from PROC MEANS using COMPLETETYPES keyword

This option, however, must be applied over all class variables and would result in an output dataset with zero values for every combination of SocTerm and PrefTerm, a huge number of non-desirable obs. Thus is the context of this approach, use of the COMPLETETYPES option is unwarranted and a better approach might be to zero fill any missing categories as desired.

COUNTING WITH PROC MEANS EXTENDED WITH EXCLUSIVE OPTION

Finally, consider a situation where a restricted counting is requested. Suppose a counting of AEs classified at CTCAE grade 3 or higher as well as each grade of 3 or higher is requested, and this count should be only among the subjects taking the study drug, not placebos. In this situation, PROC FORMAT would be used to create trtsdf and max3CTCf, two new formats for use in PROC MEANS.

```
proc format;
   value  trtsdf  2='Drug';
   value  max3CTCf  3,4,5='Special Concern'  3='Severe' 4='Disabling' 5='Death' ;
```

The input data set and PROC MEANS are unaltered from that used above, except that an additional option EXCLUSIVE is applied with the CLASS statement.

```
%let ctcfmt=maxgt3ctcf. ;
%let trtfmt=trtsdf. ;
proc means   data=maxgrd (keep=SocTerm PrefTerm maxCTC TrtmtCode)  noprint;
   format TrtmtCode &trtfmt   maxCTC &ctcfmt ;
   class  SocTerm PrefTerm ;
   class maxCTC TrtmtCode  / preloadfmt mlf order=data  exclusive ;
   types   TrtmtCode*maxCTC*SocTerm*PrefTerm ;
   output out=ctgrd ;
```

In this code, the EXCLUSIVE option instructs PROC MEANS to count only values listed in the VALUE statement of PROC FORMAT. The resulting dataset ctgrd now only includes the desired counts and, as in the following Output 7, is again restricted to 'Tinnitus'.

SocTerm	PrefTerm	maxCTC	TrtmtCode	_TYPE_	_FREQ_
EAR AND LABYRINTH	Tinnitus	Special Concern	Drug	15	5
EAR AND LABYRINTH	Tinnitus	Severe	Drug	15	2
EAR AND LABYRINTH	Tinnitus	Disabling	Drug	15	3

Output 7. Resulting OUTPUT data for Preferred Term Tinnitus from PROC MEANS with EXCLUSIVE keyword

CONCLUSION

Although counting is not a glamorous SAS programming activity, it is certainly one of most common purposes of writing SAS code. Not infrequently, analysts build upon existing programs to account for the continually changing and growing ways in which values of multiple variables are arranged and counted. After several rounds of this expansion, programs can become needlessly complex and heavily populated with macro variables. Such code is difficult to troubleshoot or modify if not properly documented and maintained. In contrast, this paper recommends the use of some basic SAS statements to improve program clarity. PROC MEANS when used in conjunction with MULTILABEL formats is a powerful construction for efficiently counting the values of variables and variable combinations. The coding statements are transparent in purpose and easily modified to accommodate other counting scenarios.

ACKNOWLEDGMENTS

CONTACT INFORMATION

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

Name: Rod Norman
Enterprise: PharmaNet/i3
Address: 10052 Mesa Ridge Court
City, State ZIP: San Diego, CA 92064
Work Phone: 858 431 3017
Fax: 858 597 1004
E-mail: rodney.norman@pharmanet-i3.com

Common Terminology Criteria for Adverse Events (CTCAE), National Cancer Institute, National Institutes of Health, Department of Health and Human Services, USA.