Paper 45-2012

Programming With CLASS: Keeping Your Options Open

Arthur L. Carpenter
California Occidental Consultants

ABSTRACT

Many SAS® procedures utilize classification variables when they are processing the data. These variables control how the procedure forms groupings, summarizations, and analysis elements. For statistics procedures they are often used in the formation of the statistical model that is being analyzed. Classification variables may be explicitly specified with a CLASS statement, or they may be specified implicitly from their usage in the procedure.

Because classification variables have such a heavy influence on the outcome of so many procedures, it is essential that the analyst have a good understanding of how classification variables are applied. Certainly there are a number of options (system and procedural) that affect how classification variables behave. While you may be aware of some of these options, a great many are new, and some of these new options and techniques are especially powerful. You *really* need to be open to learning how to program with CLASS.

KEY WORDS

CLASS statement, classification variables, MISSING, DESCENDING, preloaded formats, ORDER, MLF

INTRODUCTION

A number of analysis and reporting procedures use classification variables as a part of their processing. Classification variables allow the procedure to segment the data without it having been sorted. Within procedures there are two general ways of implementing classification variables. Most procedures that utilize classification variables declare them explicitly through the use of the CLASS statement. In a few other procedures, such as FREQ and REPORT, the classification variable(s) are implied by their usage.

When the CLASS statement is used it can be specified as a single statement or it can be broken up into a series of CLASS statements. The order of the CLASS statements determines the overall order of the classification variables. The two code boxes to the right are equivalent in that they declare the same classification variables and in the same order.

```
class race sex edu;
```

```
class race sex;
class edu;
```

CLASS STATEMENT OPTIONS

The CLASS statement now accepts options. Since a single class statement can be replaced by a series of CLASS statements, we have additional ability to control the application of CLASS statement options to specific classification variables.

```
class race sex;
class edu / missing;
```

One or more options are specified on a CLASS statement by preceding the option with a slash. While it is not necessary to have multiple CLASS statements just to apply CLASS statement options, multiple CLASS statements allow you to apply these options differentially. For instance when you use the MISSING option on the PROC statement, it is applied to *all* of the classification variables. By using multiple CLASS statements along with the MISSING option on the CLASS statement, you can choose which classification variables are to utilize the MISSING option.

CLASS statement options include:

- ASCENDING / DESCENDING
  Analogous to the DESCENDING option in PROC SORT and other procedures, these options allow you to reverse the order of the displayed values.
- GROUPINTERNAL and EXCLUSIVE
  You can use these two options to control how formats associated with CLASS variables are to be used when forming groups.
- MISSING
  Observations with missing levels of the classification variables are normally excluded from the analysis. This option allows missing values to represent valid levels of the classification variable.
- MLF
  Multilabel formats allow overlapping formatted levels of the classification variable.
- ORDER=
  This option allows you to control the order of the classification variable's levels.
- PRELOADFMT and EXCLUSIVE
  When formats are preloaded they can be used to establish data filters when forming groups.

The following example performs a simple SUMMARY step and generates the data set STATS. In this step the two classification variables (RACE and EDU) are used to summarize the data for the two analysis variables (HT and WT).

Examination of the partial listing below shows that only 75 (of the potential 77) observations were used in the calculation of the summary statistics. At this point it is not clear why two of the observations were excluded from the analysis.

```
title1 'Single Class Statement';
proc summary data=advrpt.demog;
   class race edu;
   var ht wt;
   output out=stats
          mean= htmean wtmean
          stderr=htse wtse
          ;
   run;
proc print data=stats;
   run;
```

```
Single Class Statement

Obs   race   edu    _TYPE_   _FREQ_    htmean    wtmean     htse      wtse

  1           .        0       75     67.6000   161.200   0.40670    3.9272
  2          10        1       11     71.3636   194.091   0.96552    5.7532
  3          12        1       19     67.0526   168.105   0.65102    6.3628
  4          13        1        4     70.0000   197.000   1.15470   10.3923
  5          14        1       10     64.2000   108.400   0.13333    1.4236
  6          15        1        7     65.2857   155.571   0.86504   11.1160
  7          16        1       10     70.4000   165.200   0.54160    6.1946
  8          17        1       10     65.2000   145.200   0.74237    7.9342
  9          18        1        4     69.0000   174.000   2.30940   15.5885
 10    1      .        2       42     68.4762   176.143   0.58756    4.0053
 11    2      .        2       17     67.6471   162.000   0.76668    8.1633
                  .... portions of the output are not shown ....
```

MISSING and DESCENDING Options

Throughout SAS, when classification variables are missing, their associated observation is excluded from the analysis. This is true for procedures with explicit CLASS statements, such as PROC MEANS and PROC GLM, as well as for those with implicit classification variables, such as PROC FREQ and PROC REPORT. Sometimes this is the behavior that you want; however, often it is important that these observations not be removed. The MISSING option allows missing values to be valid levels of the classification variable.

The MISSING option can be used with most procedures that have either implicit or explicit classification variables. This option can be used on a CLASS statement or for many procedures on the PROC statement. When used on the PROC statement the option applies to all the classification variables; however, when it is used on the CLASS statement it is only applied to those specific classification variables. In PROC FREQ the MISSING option is used as an option on the TABLES statement, and in PROC REPORT it can appear on the DEFINE statement.

The data table ADVRPT.DEMOG has 77 rows; however, because of missing values in one or both of the classification variables, only 75 observations have been used in the previous summary. From the LISTING above, or even by inspection of the LOG, it is unclear which classification variable has the missing values.

In the next example, the DESCENDING option is applied to RACE and the MISSING option is applied to the classification variable EDU.

```
proc summary data=advrpt.demog;
   class race/descending; ❶
   class edu/missing; ❷
   var ht wt;
   output out=stats
          mean= htmean wtmean
          stderr=htse wtse
          ;
   run;
```

❶ The groups formed by RACE are now shown in decreasing (DESCENDING) order.

❷ A missing value for the variable EDU will now be considered to be a valid level and will be included in the report. Any observation with a missing value for RACE will still be excluded.

```
Multiple Class Statements
MISSING and DESCENDING Options
```

Obs	race	edu	_TYPE_	_FREQ_	htmean	wtmean	htse	wtse
1		.	0	76	67.6053	162.237	0.40135	4.0115
2		. ❷	1	1	68.0000	240.000	.	.
3		10	1	11	71.3636	194.091	0.96552	5.7532
4		12	1	19	67.0526	168.105	0.65102	6.3628
5		13	1	4	70.0000	197.000	1.15470	10.3923
6		14	1	10	64.2000	108.400	0.13333	1.4236
7		15	1	7	65.2857	155.571	0.86504	11.1160
8		16	1	10	70.4000	165.200	0.54160	6.1946
9		17	1	10	65.2000	145.200	0.74237	7.9342
10	❶	18	1	4	69.0000	174.000	2.30940	15.5885
11	5	.	2	4	66.5000	147.000	0.86603	0.0000
12	4	.	2	4	64.5000	113.500	0.28868	0.8660
13	3	.	2	8	65.0000	112.000	0.65465	4.5826
14	2	.	2	18	67.6667	166.333	0.72310	8.8325
15	1	.	2	42	68.4762	176.143	0.58756	4.0053

. . . . portions of the table are not shown . . . .

The overall number of observations is now 76, and we can see that there is one observation with a missing value of EDU (OBS=2 in the listing ❷). Since there are 77 observations in the data set, there must be an observation with a missing value for RACE as well.

When a classification variable is both missing and formatted, even if the format maps missing values to text, the observation with the missing value is excluded unless the MISSING option is applied.

GROUPINTERNAL Option

When a classification variable is associated with a format, that format is used in the formation of the groups. In the next example, the EDULEVEL. format maps the years of education into levels of education.

```
title1 'CLASS Statement Options';
proc format;
   value edulevel ❶
      0-12 = 'High School'
      13-16= 'College'
      17-high='Post Graduate';
   run;

title2 'GROUPINTERNAL not used';
proc summary data=advrpt.demog;
   class edu; ❸
   var ht wt;
   output out=stats
          mean= MeanHT MeanWT
          ;
   format edu edulevel.; ❷
   run;
proc print data=stats;
   run;
```

❶ The EDULEVEL. format maps years of education into three ranges.

❷ In the SUMMARY step the FORMAT statement has been used to create the association between EDU and the EDULEVEL. format.

❸ The MISSING option has not been applied; consequently missing values of EDU will not be included in the summary.

```
CLASS Statement Options
GROUPINTERNAL not used

Obs          edu          _TYPE_ _FREQ_   MeanHT   MeanWT

 1                .         0     76     67.5526  160.461
 2     High School          1     30     68.6333  177.633
 3     College              1     32     67.0938  147.438
 4     Post Graduate        1     14     66.2857  153.429
```

A PROC PRINT LISTING of the resulting data table shows that the SUMMARY procedure has used the format to collapse the individual levels of EDU into the three levels of the formatted classification variable.

To use the original data values (internal values) to form the groups, rather than the formatted values, the GROUPINTERNAL option is added to the CLASS statement.

```
class edu/groupinternal;
```

Notice that although the original values of EDU are used to form the groups, the formatted values are still displayed. In this example we could have achieved similar results by using the ORDER=INTERNAL option shown next.

```
CLASS Statement Options
Using GROUPINTERNAL

Obs          edu          _TYPE_  _FREQ_   MeanHT   MeanWT

 1                .         0      76     67.5526  160.461
 2     High School          1      11     71.3636  194.091
 3     High School          1      19     67.0526  168.105
 4     College              1       4     70.0000  197.000
 5     College              1      11     64.1818  108.091
 6     College              1       7     65.2857  155.571
 7     College              1      10     70.4000  165.200
 8     Post Graduate        1      10     65.2000  145.200
 9     Post Graduate        1       4     69.0000  174.000
```

ORDER= Option

When procedures create ordered output, often based on the classification variables, there are several different criteria that can be used to determine the order. The ORDER= option is used to establish the scheme, which establishes the ordering criteria. The ORDER= option can generally appear on the PROC statement where it applies to all the classification variables (implicit or explicit), or as an option on the CLASS statement where it can be applied to selected classification variables.

These schemes include:

- DATA order is based on the order of the incoming data
- FORMATTED values are formatted first and then ordered
- FREQ the order is based on the frequency of the class level
- INTERNAL same as UNFORMATTED or GROUPINTERNAL

The default ordering is always INTERNAL (whether or not the variable is formatted) except for PROC REPORT. In PROC REPORT, formatted variables have a default order of FORMATTED.

ORDER= FREQ

```
class edu/order=freq;
```

```
CLASS Statement Options
Using ORDER=FREQ

Obs     edu     _TYPE_     _FREQ_      MeanHT      MeanWT

 1       .        0         76       67.5526     160.461
 2      12        1         19       67.0526     168.105
 3      14        1         11       64.1818     108.091
 4      10        1         11       71.3636     194.091
 5      17        1         10       65.2000     145.200
 6      16        1         10       70.4000     165.200
 7      15        1          7       65.2857     155.571
 8      18        1          4       69.0000     174.000
 9      13        1          4       70.0000     197.000
```

Using the ORDER=FREQ option on the CLASS statement in a MEANS or SUMMARY step causes the table to be ordered according to the most common levels of education.

In this table EDU has been left unformatted. Notice that the order of the rows for EDU is based on the frequency of the level of EDU (value of _FREQ_).

ORDER= INTERNAL

This is typically the order of the variable as if it had been sorted with PROC SORT, and is usually the procedure's default. The alias of INTERNAL is UNFORMATTED. When the ORDER= option is not specified the default order for all classification variables is ORDER=INTERNAL. As a result, in this example, the symptoms will appear in alphabetical order as the variable SYMP is character.

```
title1 'Controlling Order';
title2 'ORDER=INTERNAL';
proc tabulate data=advrpt.demog;
   class SYMP/order=internal;
   class sex;
   var wt;
   table sex*wt=' '*n=' '
         ,symp
         /box='Patient Counts'
          row=float
          misstext='0';
   run;
```

Understanding ORDER= ORDER=INTERNAL

Patient Counts	symptom code							
	01	02	03	04	05	06	09	10
patient sex								
F	2	6	2	5	2	7	0	5
M	2	4	2	8	6	4	2	8

ORDER= FORMATTED

When the ORDER=FORMATTED option is used the values are first formatted and then ordered.

```
proc format;
   value $SYMPTOM
      '01'='Sleepiness'
      '02'='Coughing'
      '03'='Limping'
      '04'='Bleeding'
      '05'='Weak'
      '06'='Nausea'
      '07'='Headache'
      '08'='Cramps'
      '09'='Spasms'
      '10'='Shortness of Breath';
   run;

title2 'order=formatted';
proc means data=advrpt.demog
           n mean
           order=formatted; ❶
   class symp;
   var ht;
   format symp $symptom.; ❷
   run;
```

Understanding ORDER= order=formatted

The MEANS Procedure

Analysis Variable : ht height in inches			
symptom code	N Obs	N	Mean
Bleeding	13	13	68.6923077
Coughing	10	10	66.8000000
Limping	4	4	66.5000000
Nausea	11	11	64.0000000
Shortness of Breath	13	13	68.5384615
Sleepiness	4	4	67.5000000
Spasms	2	2	68.0000000
Weak	8	8	67.5000000

❶ When the ORDER=FORMATTED option is used on the PROC statement, rather than on the CLASS statement, it is applied to *all* classification variables (here there is only one).

❷ The user defined format $SMPTOM. is applied to the classification variable SYMP.

The formatted values now determine the order of the rows for the classification variable.

ORDER= DATA

The order of the classification variables will reflect their order in the data itself. The first level detected will be written first. The data do not have to be in any particular order.

```
title2 'order=data';
proc means data=advrpt.demog
           n mean
           order=data;
   class symp;
   var ht;
   run;
```

Symptom 02 (coughing) is the first symptom in the data, followed by 10 and 06.

Understanding ORDER= order=data

The MEANS Procedure

Analysis Variable : ht height in inches			
symptom code	N Obs	N	Mean
02	10	10	66.8000000
10	13	13	68.5384615
06	11	11	64.0000000
04	13	13	68.6923077
03	4	4	66.5000000
09	2	2	68.0000000
05	8	8	67.5000000
01	4	4	67.5000000

UNIVARIATE AND THE CLASS STATEMENT

The UNIVARIATE procedure now accepts the CLASS statement, however there are both limitations and extensions that are specific to this procedure.

As is the case with a number of other summary and analysis procedures, multiple CLASS statements and CLASS statement options are supported. However, unlike other summary procedures, you can only specify up to two classification variables.

One of the CLASS statement options used specifically with UNIVARIATE is the KEYLEVEL= option. This option can be used to control plot order by specifying a *primary* or *key* value for the classification variable(s). The selected level will be displayed first.

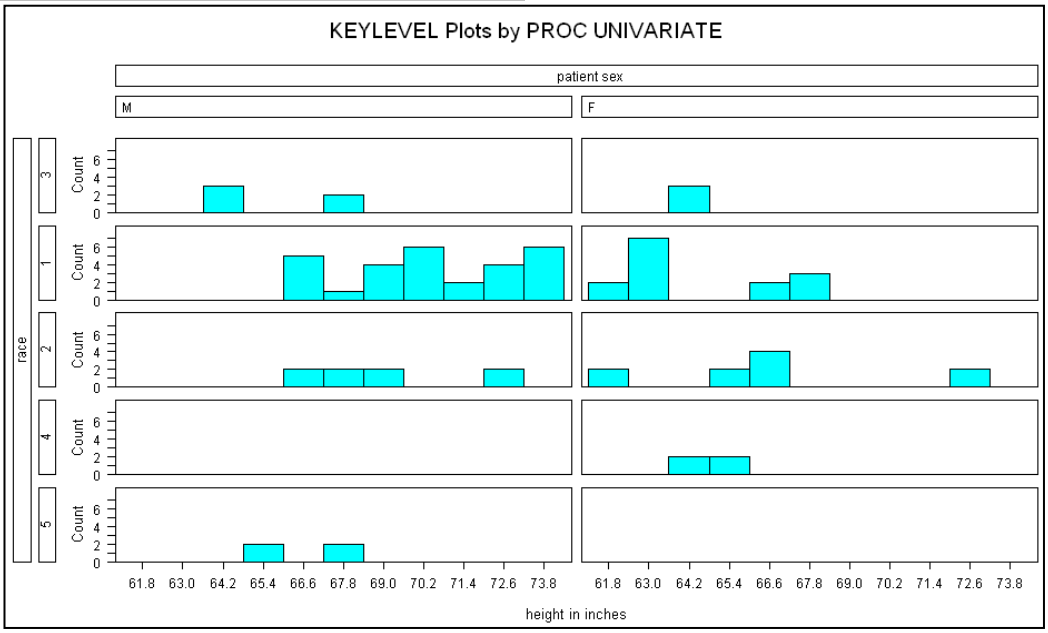
```
title1 f=arial
       'KEYLEVEL Plots by PROC UNIVARIATE';
proc univariate data=advrpt.demog;
class race sex/keylevel=('3' 'M');
var ht;
histogram /nrows=5 ncols=2
          intertile=1 cfill=cyan vscale=count
          vaxislabel='Count';
run;
quit;
```

The single CLASS statement used here could have been rewritten as two statements, one for each classification variable.

```
class race / keylevel='3';
class sex  / keylevel='M';
```

When using a CLASS statement, the printed output is also broken up into each combination of classification variables.

In the plot, notice that RACE level ‘3’ and SEX level ‘M’ are positioned first. Through the KEYLEVEL= option, they have been designated as the KEYLEVELs.



IMPLICIT CLASS VARIABLES

A number of procedures make use of classification variables, but do not support the CLASS statement. These procedures use implied classification variables and this means that it is not possible to assign the previously described options using the CLASS statement.

PROC FREQ does not support the CLASS statement, but that does not mean that does not mean that it does not use classification variables. In a PROC FREQ step the classification variables are implied by usage in the TABLE statement.

The variables making up the table requests (SYMP and EDU in this example) are implicit classification variables. However since there is no CLASS statement, the options discussed previously are applied differently. Some of the options that are to be applied to a classification variable are specified on the PROC statement; others are specified on the TABLE statement.

```
proc freq data=advrpt.demog
          order=formatted ❶;
   table symp / missing ❷;
   table edu;
   format symp $symptom.;
   run;
```

❶ This ORDER= option would be applied to all classification variables on all TABLE statements. The ORDER= option cannot be used on the TABLE statement.

❷ The MISSING option applies to all classification variables on this particular TABLE statement. For PROC FREQ, the MISSING option cannot be used on the PROC statement.

For procedures with implicit classification variables, you will need to experiment and read the documentation to determine how a specific option is applied.

Like in a PROC FREQ step, the CLASS statement is not used with the REPORT procedure. The classification variables are determined by their usage on the DEFINE statement. The REPORT step allows several different types of classification variables, consequently the CLASS statement is not flexible enough.

```
proc report data=demog nowd;
column region sex,ht;
define region / group order=data;
define sex    / across missing;
define ht     / analysis mean;
run;
```

In this example the variables REGION (define usage of GROUP) and SEX (define usage of ACROSS) are both implicit classification variables. Notice the use of the MISSING and ORDER= options on the DEFINE statement. It is also possible to specify these options on the PROC statement, however the use of the DEFINE statement offers a great deal more flexibility.

CLASS Variables and PROC REPORT

	patient sex	
	F	M
region	height in inches	height in inches
04	66.5	70
08	70	.
06	63	67.8
02	63.666667	64
09	65	68.142857
03	65.6	70.444444
05	63.2	70.666667
10	63	69
01	.	74

MULTILABEL FORMATS

Multilabel formats allow overlapping ranges for formatted classification variables. Usually when you create formats, format ranges with overlapping values are neither allowed nor desirable. However overlapping ranges for classification variables can have distinct advantages. Multilabel formats are created using the MULTILABEL option on the VALUE statement, and they are used with the MLF CLASS statement option.

Not all procedures can take advantage of multilabel formats the primary data summary and reporting procedures that can take advantage of this type of format include TABULATE, MEANS, SUMMARY, and REPORT. In each case the format is implemented by associating it with the classification variable and the use of the MLF option. Procedures that do not utilize overlapping format values (do not support the MLF option) will only use the primary range of the format.

Creating a Multilabel Format

In the following example we summarize across the years of education of each patient by grouping them into high school and college. At the same time we want to see the graduate school subset of those attending college. This can easily be accomplished using a multilabel format.

```
proc format;
   value edlevel (multilabel) ❶
      9-12 = 'High School'
      13-high='College'
      17-high='Graduate Studies'; ❷
   run;
```

❶ The MULTILABEL option sets up the format to be used with the MLF option on the CLASS statement.

❷ The ranges for ‘College’ and ‘Graduate Studies’ overlap (are not distinct). Without the MULTILABEL option, PROC FORMAT would generate an error and the format would not be created.

Using a Multilabel Format

The MLF option is used to associate a given multilabel format with a classification variable.

```
title1 'Multilabel Formats';
proc tabulate data=advrpt.demog;
   class edu / mlf ❸;
   class sex;
   var wt;
   table edu=' ' all,
         sex*wt*(n*f=2.
                 mean*f=5.1
                 stderr*f=6.2)
         /box=edu;
   format edu edlevel.;
   run;
```

❸ The MLF option appears on the CLASS statement associated with the formatted variable. Without this option ‘Graduate Studies’ will not be displayed as a level of EDU.

Multilabel Formats

years of education	patient sex					
	F			M		
	weight in pounds			weight in pounds		
	N	Mean	StdErr	N	Mean	StdErr
College	23	142.0	7.02	23	156.6	6.96
Graduate Studies	8	142.8	9.84	6	167.7	10.64
High School	8	152.0	9.71	22	187.0	4.59
All ❹	31	144.5	5.76	45	171.4	4.75

❹ PROC TABULATE correctly counts and totals the number of persons in each education category. Notice that the count for ‘Graduate Studies’ does not contribute to the overall total.

The MLF option can also be used with CLASS variables in REPORT, MEANS, and SUMMARY procedures. In future releases of SAS it may be implemented in other procedures as well.

Calculate a Rolling Average Using a Multilabel Format

There are a number of ways to calculate a rolling or moving average within SAS. The use of multilabel formats provides a quick and easy programming solution to this often taxing problem. In this example we would like to calculate a three-visit rolling average of the patient's potassium levels. The numeric variable VISIT can take on the values of 1 - 16.

```
data control(keep=fmtname start end label hlo);
   retain fmtname 'avg' ❶
          hlo 'M'; ❷
   do start=1 to 14;
      end=start+2; ❸
      label=cats('VisitGrp', put(start,z2.)); ❹
      output Control;
   end;
   hlo='O'; ❺
   label='Unknown';
   output;
   run;

proc format cntlin=control; ❻
   run;

proc summary data=advrpt.lab_chemistry;
   by subject;
   class visit / mlf; ❼
   format visit avg.; ❽
   var potassium;
   output out=rollingAVG
          mean= Avg3Potassium;
   run;
```

```
proc format;
   value avg (multilabel)
      1 - 3 = 'VisitGrp01'
      2 - 4 = 'VisitGrp02'
      3 - 5 = 'VisitGrp03'
      4 - 6 = 'VisitGrp04'
      5 - 7 = 'VisitGrp05'
      6 - 8 = 'VisitGrp06'
      7 - 9 = 'VisitGrp07'
      8 - 10= 'VisitGrp08'
      9 - 11= 'VisitGrp09'
      10- 12= 'VisitGrp10'
      11- 13= 'VisitGrp11'
      12- 14= 'VisitGrp12'
      13- 15= 'VisitGrp13'
      14- 16= 'VisitGrp14'
      other = 'Unknown';
   run;
```

The AVG. format, which is generated using the CNTLIN= option to the left, is effectively defined as is shown in the FORMAT step above.

❶ A format control data set containing the value pairs, labels, and format name (AVG.) is created.

❷ The HLO variable is used to designate this as a multilabel format. On the VALUE statement this would be the MULTILABEL option.

❸ START is the lower bound of the moving average and END is the upper bound. In this example the width will contain up to three visits.

❹ The label is assigned a value. For the group starting with visit 4, LABEL will be VisitGrp04, which will be the average of visits 4, 5, and 6.

❺ Although not needed here, it is always a good idea to specify the 'OTHER' group.

❻ PROC FORMAT creates the format using the CONTROL data set.

❼ The overlapping ranges of a multilabel format are ignored unless the MLF option is specified.

❽ The format is associated with the classification variable.

Multilabel Formats

Obs	SUBJECT	VISIT	_TYPE_	_FREQ_	Avg3Potassium
1	200		0	14	4.32143
2	200	VisitGrp01	1	3	4.10000
3	200	VisitGrp02	1	3	4.36667
4	200	VisitGrp03	1	3	4.26667
5	200	VisitGrp04	1	4	4.40000
6	200	VisitGrp05	1	4	4.60000
7	200	VisitGrp06	1	4	4.65000
8	200	VisitGrp07	1	6	4.35000
9	200	VisitGrp08	1	5	4.32000
10	200	VisitGrp09	1	4	4.20000
11	200	VisitGrp10	1	1	4.50000
12	201		0	10	4.17000

If the data set (WORK.ROLLINGAVG) created by SUMMARY is printed we can see that the individual visits have been summarized into the various groups. This method does not control for repeat visits. Examination of the frequencies for subject 200 in some of the groups shows that this subject had a number of repeat visits.

CLASS VARIABLES AND PRELOADED FORMATS

For the TABULATE, MEANS, SUMMARY, and REPORT procedures, preloaded formats can be used to manipulate what appears in the resultant table or data set by both adding rows that are not represented in the data or by eliminating rows that do not meet the format's criteria.

The two primary options that you will need to know when using preloaded formats is PRELOADFMT and EXCLUSIVE. Their usage will depend on whether or not the procedure you are using supports the CLASS statement. There are also a number of supporting options that vary by procedure as well.

- PRELOADFMT Loads the format levels prior to execution. This option will always be present when using preloaded formats.
- EXCLUSIVE Only data levels that are included in the format definition are to appear in table.

As the name implies preloaded formats key off of what is generally a user-defined format. The formats $REGX., $GENDERU., and $SYMP., which are defined here, are used in some of the examples that follow. Each contains one level that is not in the data ❶, and both $REGX. and $SYMP. exclude levels that *are* found in the data. The format $REGX. is used with the variable REGION, which ranges from '1' through '10'. The format $SYMP. is used with the variable SYMP, which ranges from '01' through '10'. The format $GENDERU. is used with the variable SEX which takes on the values of 'F' and 'M'.

```
proc format;
   value $regx
      '1'=' 1'
      '2'=' 2'
      'X'=' X' ; ❶
   value $genderu
      'M'='Male'
      'F'='Female'
      'U'='Unknown'; ❶
   value $symp
      '00'= 'Unspecified' ❶
      '01'= 'Sleepiness'
      '02'= 'Coughing'
      '03'= 'Limping';
   run;
```

Using a Preloaded Format with REPORT

Preloading formats with the EXCLUSIVE option allows only those levels that are on the format *and* in the data. In PROC REPORT these options, which are to be associated with the classification variable, are applied on the DEFINE statement.

```
title2 'Using PRELOADFMT with EXCLUSIVE';
proc report data=demog nowd;
   column region sex,(wt=n wt);
   define region / group
                   format=$regx6.
                   preloadfmt exclusive;
   define sex    / across        format=$Genderu. 'Gender';
   define n      / analysis n    format=2.0 'N';
   define wt     / analysis mean format=6
   run;
```

Using the format $REGX. along with these two options causes only regions 1 and 2 to appear in the report, as these are the only two regions that are both in the format and in the data.

```
Preloading Formats in PROC REPORT
Using PRELOADFMT with EXCLUSIVE

                    Gender
              Female        Male
  region   N  Weight   N  Weight
   1       .     .     4  195.00
   2       6  109.67   3  105.00
```

The capability to add rows to a table without first modifying the original data set can be especially useful. In this example Visit three never appears in the data, but we want it to appear in the report that is to be generated. Let's assume that we need to generate a report of mean lab chemistry values for lab visits. The report must contain the first 10 visits regardless of whether or not they appear in the data.

```
proc format; ❶
   value visits
        1='1'
        2='2'
        3='3'
        4='4'
        5='5'
        6='6'
        7='7'
        8='8'
        9='9'
        10='10';
   run;
proc report data=advrpt.lab_chemistry nowd
             completerows;❷
   column visit sodium potassium chloride;
   define visit / group
                  f=visits. preloadfmt ❸
                  'Visit' order=data;
   define sodium /analysis mean f=5.2;
   define potassium /analysis mean f=5.3;
   define chloride/analysis mean f=5.1;
   run;
```

❶A format is created which contains each of the first 10 visits.

❷ The COMPLETEROWS option, which is unique to PROC REPORT, is used to ensure that every row in the preloaded format will appear in the report.

❸ The PRELOADFMT option will always be present when using preloaded formats. Here the PRELOADFMT option is associated with the format to be preloaded by placing both on the DEFINE statement. The implied classification variable VISIT (define usage of GROUP) receives the preloaded format.

Preloading Formats in PROC REPORT
PRELOADFMT without EXCLUSIVE

Visit	sodium	potassium	chloride
1	14.01	4.206	100.6
2	14.07	4.294	101.3
3	.	.	.
4	14.05	4.231	102.8
5	14.21	4.500	101.4

A portion of the resultant report is shown here. Notice that VISIT 3 appears in the table although it is never in the data.

Using a Preloaded Format with TABULATE

When using preloaded formats with the TABULATE procedure the PRELOADFMT and EXCLUSIVE options are applied on the CLASS statement. As with the REPORT procedure these two options interact.

In each of the examples in this section the user-defined format, $SYMP. is used. This format, which is defined earlier, contains only three of the 10 possible values that can occur in the data, and one additional value that does not occur in the data.

```
title2 'Using PRINTMISS With the EXCLUSIVE option';
proc tabulate data=advrpt.demog;
   class symp /preloadfmt exclusive; ❶
   var ht wt;
   table symp,
         (ht wt)*(n*f=2. min*f=4.
                  median*f=7.1 max*f=4.)
         / printmiss; ❸
   format symp $symp.; ❷
   run;
```

❶ The PRELOADFMT and EXCLUSIVE options appear on the CLASS statement associated with the classification variable that is to receive the preloaded format.

❷ The appropriate format is assigned to the classification variable.

❸ The PRINTMISS option allows the display of missing values in a PROC TABULATE table. Without including this option, levels added by the preloaded format, which would necessarily always be missing, would not be displayed.

The PRELOADFMT and EXCLUSIVE options used together eliminate all values of SYMP that are not on the format, while including values on the format that are not in the data.

Because the PRINTMISS option ❸ has been used, the "Unspecified" row appears in the table with the appropriate values for N.

Using Preloaded Formats With TABULATE
Using PRINTMISS With the EXCLUSIVE option

	height in inches				weight in pounds			
	N	Min	Median	Max	N	Min	Median	Max
symptom code								
Unspecified	0	.	.	.	0	.	.	.
Sleepiness	4	64	67.5	71	4	115	138.5	162
Coughing	10	66	67.0	67	10	131	155.0	155
Limping	4	65	66.5	68	4	147	154.5	162

Using a Preloaded Format with MEANS and SUMMARY

As was the case with PROC TABULATE, the PRELOADFMT and EXCLUSIVE options appear on the CLASS statement, when they are used with the MEANS and SUMMARY procedures.

Preloading with the CLASS statement options PRELOADFMT and EXCLUSIVE limits the levels of the classification variable to those that are both on the format and in the data. Essentially the format acts as a filter without resorting to either a subsetting IF statement or a WHERE clause. However unlike PROC TABULATE, this combination of CLASS statement options does NOT insert a row for the formatted value of SYMP that is not in the data (SYMP='00'). To add this level using the MEANS and SUMMARY procedures, the COMPLETETYPES option must also be included.

The PROC statement option COMPLETYPES will interact with the PRELOADFMT and EXCLUSIVE options. As a result of this interaction, levels of the classification variable that are not in the data, but are on the format are now included in the summary. However, because of the use of the EXCLUSIVE option, levels not on the format are still not included in the table.

```
title2 'With EXCLUSIVE and COMPLETETYPES';
proc summary data=advrpt.demog
             completetypes;
   class symp / preloadfmt
                exclusive;
   var ht;
   . . . code not shown . . .  .
```

The 'Unspecified' level for SYMP now appears in the report even though it is not in the data (_FREQ_=0).

Using Preloaded Formats With MEANS/SUMMARY
With EXCLUSIVE and COMPLETETYPES

Obs	symp	_TYPE_	_FREQ_	meanHT
1		0	18	66.8889
2	Unspecified	1	0	.
3	Sleepiness	1	4	67.5000
4	Coughing	1	10	66.8000
5	Limping	1	4	66.5000

SUMMARY

Very often when we think of classification variables we think in terms of those variables that are specified on the CLASS statement, however this way of looking at classification variables limits what we can do with them.

Classification variables can be explicitly specified on the CLASS statement in procedures such as MEANS and TABULATE. They can also be implicitly specified through their usage as in PROC FREQ. PROC REPORT uses more than one type of classification variable and the usage is specified on the DEFINE statement associated with the classification variable.

When the CLASS statement is used, it can be broken up into multiple CLASS statement, and broken up or not, options can be specified on the CLASS statement. These options allow us to control how the classification variables are to be used. Procedures that do not support the CLASS statement, but still utilize classification variables, often implement many of these same options using different syntax.

Get to know how to take full advantage of classification variables and their associated options. Your tables and reports can generate the information that you need when you know how to program with class.

ABOUT THE AUTHOR

Art Carpenter's publications list includes five books, and numerous papers and posters presented at SUGI, SAS Global Forum, and other user group conferences. Art has been using SAS® since 1977 and has served in various leadership positions in local, regional, national, and international user groups. He is a SAS Certified Advanced Professional Programmer and through California Occidental Consultants he teaches SAS courses and provides contract SAS programming support nationwide.

AUTHOR CONTACT

Arthur L. Carpenter
California Occidental Consultants
10606 Ketch Circle
Anchorage, AK 99515

(907) 865-9167
art@caloxy.com
www.caloxy.com

REFERENCES

Many of the examples in this paper have been borrowed (with the author's permission) from the book Carpenter's Guide to Innovative SAS® Techniques by Art Carpenter (SAS Press, 2012).

TRADEMARK INFORMATION