Paper 087-2011

Force Missing Rows and Columns with CLASSDATA in PROC TABULATE

Peter Cerussi, DMDC, Seaside, CA

ABSTRACT

When creating the same report over time, it is important that the format stays the same to ensure consistency. With the TABULATE procedure, levels of a class variable, which do not have any observations, will not appear in the report. To create consistent reports, a solution is needed that will ensure a standard report format over time. In order to account for changes that occur over time, all levels of class variables will need to be included in the standard report format. Creating a SAS data set with all levels of every class variable, along with declaring the CLASSDATA option in PROC TABULATE, will solve this problem. This paper shows how to set up the SAS data set to be used in conjunction with the CLASSDATA option, and how to avoid common errors that will cause PROC TABULATE to error out. The result is a report format that will not change over time when the data changes over time.

INTRODUCTION

The CLASSDATA option in PROC TABULATE is a powerful option that allows the programmer to create consistent reports over time. Telling SAS to use a class data set is simple, just declare CLASSDATA as one of the options in the PROC TABULATE statement. Creating a class data set that will not produce an error is where the programmer can encounter problems. However, if some simple steps are followed, errors can be avoided.

SYNTAX

The basic syntax for of a PROC TABULATE that uses the CLASSDATA option is as follows:

```
PROC TABULATE DATA={sas data set} CLASSDATA={sas data set}<options>;
CLASS variables </options>;
VAR variables </options>;
TABLE page, row, column </options>;
RUN;
```

CREATING THE CLASS DATA SET

When creating a data set to be used with the CLASSDATA option in PROC TABULATE, a few key points must be remembered:

1. All class levels of all class variables must be included
2. The class variables must have the same attributes in both the CLASS data set and the analysis data set. This can be accomplished when the analysis data set is created.

SINGLE CLASS VARIABLE

If there is only one class variable, then a simple do loop can be used to create all levels of the CLASS variable.

```
DATA CLASS;
DO SVC='A','F','N','M';
   OUTPUT;
RUN;
```

In the above example a variable is created for service, SVC, which contains the possible data values for the four military services, Army, Navy, Air Force, and Marine Corps. The OUTPUT statement tells SAS to output each value of SVC to the data set CLASS; If OUTPUT is not inside the do loop then the only value that will appear for SVC is 'M', which is the last value in the DO loop. The following table shows the SAS data set CLASS.

Obs	SVC
1	A
2	F
3	N
4	M

Table 1. CLASS Data Set With Single Class Variable

MULTIPLE CLASS VARIABLES

However, most PROC TABULATES involve multiple CLASS variables. For this example I want to create a table which shows the number of Officers and Enlisted service members in each of the four services. For CLASSDATA to function correctly, an observation is needed for all possible combinations to ensure that all columns and rows in the table have at least 1 observation. This is going to require nesting of DO loops.

```
DATA CLASS;
DO SVC='A','F','N','M';
  DO PG='E','O';
    OUTPUT;
  END;
END;
RUN;
```

This code creates the following SAS data set

Obs	SVC	PG
1	A	E
2	A	O
3	F	E
4	F	O
5	N	E
6	N	O
7	M	E
8	M	O

Table 2. CLASS Data Set With Multiple Class Variables

If there are three or more CLASS variables, just keep nesting the DO loops to include all CLASS variables. The OUTPUT statement must be in the inner most DO loop so that all possible combinations are output to the CLASS data set.

CREATING THE CLASSDATA DATA SET WITH FORMATTED VALUES

It is also possible when creating the class levels for the variables to use the formatted values. Below is the code that would have been used for a PROC FORMAT in order to see how the formatted values can be created in the CLASS data set.

```
PROC FORMAT;
    VALUE $FSVC
    'A'='ARMY'
    'N'='NAVY'
    'M'='MARINE CORPS'
    'F'='AIR FORCE';

    VALUE $FPG
    'E'='ENLISTED'
    'O'='OFFICER'
    'W'='WARRANT OFFICER';
```

Since the length of a character variable is the length of the first value encountered, unless the length has been declared before any values have been read, the value with the longest length must be listed first for all of the class variables. This also applies when the levels of the class variables are being created with the actual data values. Below is an example of what will happen when the longest character value is not stated first.

```
DATA CLASS;
DO SVC='ARMY','AIR FORCE,'NAVY','MARINE CORPS';
   OUTPUT;
END;
RUN;
```

This code produces the following SAS data set.

Obs	SVC
1	ARMY
2	AIR
3	NAVY
4	MARI

Table 3. Incorrect Data Set For CLASS Variable With Different Lengths

To produce the correct CLASS SAS data set the longest value will need to be listed first, in this case it is 'MARINE CORPS'.

```
DATA CLASS;
DO SVC='MARINE CORPS','ARMY','NAVY','AIR FORCE';
  OUTPUT;
END;
RUN;
```

This produces the following SAS data set.

Obs	SVC
1	MARINE CORPS
2	ARMY
3	NAVY
4	AIR FORCE

Table 4. Correct Data Set For CLASS Variable With Different Lengths

When the FORMAT statement is applied to the CLASS variables in PROC TABULATE, the values of the CLASS variables in the analysis data set will match up with the values of the CLASS data set.

COPYING ATTRIBUTES TO THE ANALYSIS DATA SET

The common variables in both data sets must have the same attributes. This includes having the same length. If you created the CLASS data set with the formatted values, it is obvious that it will not have the same length as the data values from the analysis data set. The fix for this problem is only a one line of code at the very beginning of the data step for the analysis data set.

```
DATA ANALYSIS;
IF 0 THEN SET CLASS;
```

All that is required is setting the ANALYSIS data set ANALYSIS to the CLASS data set under a false condition. Since it is false, no data is copied into ANALYSIS. However, the data set attributes of CLASS are copied over to ANALYSIS. This will work as long as the length of the values in the data set CLASS are greater than or equal to the length of the same variables in the ANALYSIS data set. If not, then assign the attributes of the ANALYSIS data set to the CLASS data set using the same technique that assigned the attributes of the CLASS data set to the ANALYSIS data set.

USING CLASSDATA IN PROC TABULATE

When the PROC TABULATE executes it creates a table with all the levels of every class variables even when and entire level of a class variable is not present. The FORMAT statement is needed when the levels created in the CLASS data set are the formatted data values and not the actual data values.

For this example another level of Pay Grade will be added in addition to Enlisted and Officer. The third level is Warrant Officer, which shows up in the data set as 'W'. The following code is used to generate the CLASS data set.

```
DATA CLASS;
DO SVC='A','N','F','M';
   DO PG='E','O','W';
      OUTPUT;
   END;
END;
```

Obs	SVC	PG
1	A	E
2	A	O
3	A	W
4	N	E
5	N	O
6	N	W
7	F	E
8	F	O
9	F	W
10	M	E
11	M	O
12	M	W

Table 5. CLASS Data Set

Below is the ANALYSIS data set, which does not contain any warrant officers.

Obs	SVC	PG
1	N	E
2	A	E
3	N	E
4	M	O
5	N	O
6	F	O
7	A	O
8	A	E
9	A	O
10	M	E
11	M	E
12	F	E

Table 6. ANALYSIS Data Set

Using the below code to generate a table counting the number of members in each service by pay grade creates a table with a column for 'WARRANT OFFICER' even though there are no warrant officers in the ANALYSIS data set.

```
PROC TABULATE DATA=ANALYSIS MISSING CLASSDATA=CLASS;
CLASS SVC PG/PRELOADFMT ORDER=DATA;
TABLE SVC, PG/MISSTEXT='0';
FORMAT SVC $FSVC. PG $FPG.;
```

MISSTEXT places '0' in any cell that has a missing value. PRELOADFMT with ORDER=DATA forces the order of the class variables in the table to have the same order that they did in the PROC FORMAT.

	PG		
	ENLISTED	OFFICER	WARRANT OFFICER
SERVICE	N	N	N
ARMY	2	2	0
NAVY	2	1	0
MARINE CORPS	2	1	0
AIR FORCE	1	1	0

Table 7. PROC TABULATE Output

CONCLUSION

The CLASSDATA options forces PROC TABULATE to include every possible level of all class variables in the table. This is necessary to produce reports with the same format when the data changes over time. To have the PROC TABULATE execute correctly, 3 rules must be remembered:

1. All levels of all class variables must be present in the CLASS data set.
2. The common variables of the CLASS data set and ANALYSIS data set must have the same attributes.
3. The longest value of a class level must be listed first for text variables or the correct length can be declared at the beginning of the CLASS data set.

REFERENCES

Bilenas, Jonas. 2007. "Making Sense of PROC TABULATE (Updated for SAS9®)." *Proceedings of the SAS Global Forum 2007*. Cary, NC: SAS Institute Inc. Paper 230-07. Available at: http://www2.sas.com/proceedings/forum2007/.

CONTACT INFORMATION

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

Name: Peter Cerussi
Enterprise: DMDC
Address: 400 Gigling Rd
City, State ZIP: Seaside, CA 93955
E-mail: Cerussi22@hotmail.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.