Paper 055-2012

Ethnicity and Race: When Your Output Isn't What You Expected

Philamer M. Atienza, Alcon Laboratories, Inc., Fort Worth, Texas, U.S.A.

ABSTRACT

In SAS®, when a classification variable is used to group observations with the same values and a formatted value is used for grouping data, unexpected results may come out of the procedure. If there is more than one unformatted value used for several distinct categorizations but with the same format label, SAS uses the unformatted lowest value to create the output.

Understanding the behavior of SAS when storing the unformatted values will help avoid potential mistakes in using formats and nested classification variables. This paper examines two scenarios when a variable for both ethnicity and race is used in PROC TABULATE to create an output data set: (1) with, and (2) without the use of a format.

CREATING AN ETHNICITY AND RACE TABLE

Suppose a demographic table showing the frequency counts for ethnicity and race is needed where the race classification is nested within ethnicity as shown in Table 1.

	Total		Group 1		Group 2	
	N	%	N	%	N	%
Hispanic						
White	xx	xx.x	xx	xx.x	xx	xx.x
Black or African American	xx	xx.x	xx	xx.x	xx	xx.x
Asian	xx	xx.x	xx	xx.x	xx	xx.x
American Indian	xx	xx.x	xx	xx.x	xx	xx.x
Other	xx	xx.x	xx	xx.x	xx	xx.x
Not Hispanic						
White	xx	xx.x	xx	xx.x	xx	xx.x
Black or African American	xx	xx.x	xx	xx.x	xx	xx.x
Asian	xx	xx.x	xx	xx.x	xx	xx.x
Native Hawaiian	xx	xx.x	xx	xx.x	xx	xx.x
American Indian	xx	xx.x	xx	xx.x	xx	xx.x
Other	xx	xx.x	xx	xx.x	xx	xx.x
Multi-Racial	xx	xx.x	xx	xx.x	xx	xx.x

Table 1. Sample Ethnicity and Race Table

In this case, any categorization of race can be either "Hispanic" or "Not Hispanic."

We can create a new variable, say ETHNICITY_AND_RACE, and assign different coded values for each race-ethnicity combination (see codelist in Table 2). All we need to do next is use ETHNICITY_AND_RACE with a corresponding format in our SAS procedure to create the table. We can use PROC TABULATE with a CLASS statement for ETHNICITY_AND_RACE and its format can be defined as follows:

```
proc format;
   value combofmt (notsorted)
      10000 = '^\i   Hispanic^\i0'
      10001 = 'White'
      10002 = 'Black or African American'
      10003 = 'Asian'
      10011 = 'American Indian'
      10099 = 'Other'
      20000 = '^\i   Not Hispanic^\i0'
      20001 = 'White'
      20002 = 'Black or African American'
      20003 = 'Asian'
      20010 = 'Native Hawaiian'
      20011 = 'American Indian'
      20099 = 'Other'
      20100 = 'Multi-Racial';
run;
```

Here, "^" is defined as the ODS escape character.

Value	Label	Description
10000	Hispanic	
10001	White	Hispanic White
10002	Black or African American	Hispanic Black or African American
10003	Asian	Hispanic Asian
10011	American Indian	Hispanic American Indian
10099	Other	Other Hispanic
20000	Not Hispanic	
20001	White	Not Hispanic White
20002	Black or African American	Not Hispanic Black or African American
20003	Asian	Not Hispanic Asian
20010	Native Hawaiian	Not Hispanic Native Hawaiian
20011	American Indian	Not Hispanic American Indian
20099	Other	Other Not Hispanic
20100	Multi-Racial	Multi-Racial Not Hispanic

Table 2. ETHNICITY_AND_RACE Codelist

But, because the same format label is defined for race between the two ethnicity groups (e.g., "White" under "Hispanic" vs. "White" under "Not Hispanic"), SAS will lump together the values with the same format regardless of the underlying decode. The CLASS statement will be trumped by the format, and for each unique label bucket, the lowest value in the data will be used to create the output.

ILLUSTRATION

To illustrate, the sample data for ethnicity and race consists of the following variables:

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
1	ID	Num	8			ID
3	Ethnicity	Num	8	ETHNICF.	2.	Ethnicity
5	ethnicity_and_race	Num	8	COMBOFMT.		Ethnicity and Race
4	race	Num	8	RACEF.		Race
2	treat	Num	8	TREATF.		Treatment Group

Output 1. Ethnicity and Race Sample Data Attributes from PROC CONTENTS

Note: PROC FORMAT with the FMTLIB option can be used to examine the formats assigned to each variable.

```
proc format fmtlib;
   select ethnicf racef treatf;
run;
```

The expected frequencies for ethnicity and race in the sample data are:

	Hispanic	Not Hispanic
White	70	269
Black or African American	3	54
Asian	1	30
Native Hawaiian		6
American Indian	11	1
Other	3	1
Multi-Racial		2

Table 3. Expected Frequencies for Ethnicity and Race Sample Data

To generate the table as shown in Table 1, the code could be written as:

```
proc tabulate data=sasgf2012 out=tabout1 classdata=classdat missing;
   class ethnicity_and_race treat;
   table (ethnicity_and_race=" "),
         ((all="Total" treat=" ") * (n="N" pctn<ethnicity_and_race>="%"));
run;
```

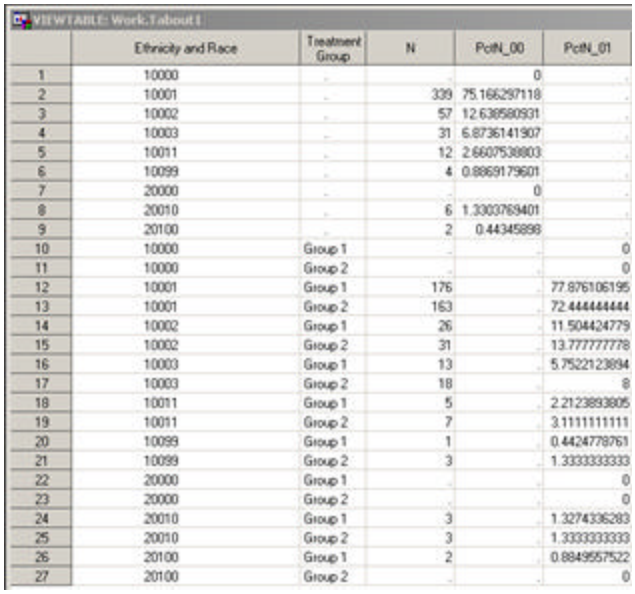
Output 2 contains the generated table and partial log.

	Total		Group 1		Group 2	
	N	%	N	%	N	%
Hispanic						
White	339	75.2	176	77.9	163	72.4
Black or African American	57	12.6	26	11.5	31	13.8
Asian	31	6.9	13	5.8	18	8.0
American Indian	12	2.7	5	2.2	7	3.1
Other	4	0.9	1	0.4	3	1.3
Not Hispanic						
Native Hawaiian	6	1.3	3	1.3	3	1.3
Multi-Racial	2	0.4	2	0.9		

```
NOTE: The data set WORK.TABOUT1 has 27 observations and 8 variables.
```

Output 2. Output and Partial Log from PROC TABULATE

We get the formatted table; however, the "Total N" column does not match the expected frequencies shown in Table 3. The output data set created from the procedure, TABOUT1, contains 27 observations including the "White" decoded value of CLASS ETHNICITY_AND_RACE equal to 10001 (but not 20001; see Display 1). Thus, "White" shows under "Hispanic" only.



VIEWTABLE: Work.Tabout1

	Ethnicity and Race	Treatment Group	N	PctN_00	PctN_01
1	10000			0	
2	10001		339	75.166297118	
3	10002		57	12.638580931	
4	10003		31	6.8736141907	
5	10011		12	2.6607538803	
6	10099		4	0.8869179601	
7	20000			0	
8	20010		6	1.3303769401	
9	20100		2	0.44345898	
10	10000	Group 1			0
11	10000	Group 2			0
12	10001	Group 1	176		77.876106195
13	10001	Group 2	163		72.444444444
14	10002	Group 1	26		11.504424779
15	10002	Group 2	31		13.777777778
16	10003	Group 1	13		5.7522123894
17	10003	Group 2	18		8
18	10011	Group 1	5		2.2123893805
19	10011	Group 2	7		3.1111111111
20	10099	Group 1	1		0.4424778761
21	10099	Group 2	3		1.3333333333
22	20000	Group 1			0
23	20000	Group 2			0
24	20010	Group 1	3		1.3274336283
25	20010	Group 2	3		1.3333333333
26	20100	Group 1	2		0.8849557522
27	20100	Group 2			0

Display 1. TABOUT1 Created by PROC TABULATE

To examine further what happened to the data, run the TABULATE code again with the additional FORMAT statement below. This will strip off the format for ETHNICITY_AND_RACE.

```
format ethnicity_and_race;
```

	Total		Group 1		Group 2	
	N	%	N	%	N	%
10000						
10001	70	15.5	37	16.4	33	14.7
10002	3	0.7	1	0.4	2	0.9
10003	1	0.2			1	0.4
10011	11	2.4	5	2.2	6	2.7
10099	3	0.7	1	0.4	2	0.9
20000						
20001	269	59.6	139	61.5	130	57.8
20002	54	12.0	25	11.1	29	12.9
20003	30	6.7	13	5.8	17	7.6
20010	6	1.3	3	1.3	3	1.3
20011	1	0.2			1	0.4
20099	1	0.2			1	0.4
20100	2	0.4	2	0.9		

```
NOTE: The data set WORK.TABOUT2 has 42 observations and 8 variables.
```

Output 3. Output and Partial Log from PROC TABULATE without Format for ETHNICITY_AND_RACE

This time, we get all the categories we want, 20001 now exists in the table, and the counts match what we expected. Note that TABOUT2, the output data set created from running the procedure, contains 42 observations. However, we want a formatted table. So, what can we do?

An easy trick that works is through the use of the same format as before, except that for the duplicate labels on race, a white space can be added as a prefix:

```
proc format;
   value combo2fmt (notsorted)
      10000 = '^\i   Hispanic^\i0'
      10001 = 'White'
      10002 = 'Black or African American'
      10003 = 'Asian'
      10011 = 'American Indian'
      10099 = 'Other'
      20000 = '^\i   Not Hispanic^\i0'
      20001 = ' White'
      20002 = ' Black or African American'
      20003 = ' Asian'
      20010 = ' Native Hawaiian'
      20011 = ' American Indian'
      20099 = ' Other'
      20100 = ' Multi-Racial';
run;
```

CONCLUSION

In conclusion, SAS programmers should keep in mind that when programming tables using formats and nested classification variables, "only the lowest unformatted value is kept with the associated bucket." (SAS Technical Support, 2007).

REFERENCE

SAS Technical Support. Copyright © 2007. "What Happened to My Data?" *SAS® Technical Paper*. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

Many thanks to the Biostatistics and Statistical Programming group of Alcon for their continued support.

CONTACT INFORMATION

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

Name: Philamer M. Atienza
Enterprise: Alcon Laboratories, Inc.
Address: 6201 South Freeway
City, State ZIP: Fort Worth, Texas 76134
Work Phone: 817-615-2307
Fax: 817-916-9380
E-mail: Philamer.Atienza@AlconLabs.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.