Paper 026-2010

The MEANS/SUMMARY Procedure: Doing More

Arthur L. Carpenter
California Occidental Consultants, Anchorage, AK

ABSTRACT

The MEANS/SUMMARY procedure is a workhorse for most data analysts. It is used to create tables of summary statistics as well as complex summary data sets. The user has a great many options which can be used to customize what the procedure is to produce. Unfortunately most analysts rely on only a few of the simpler *basic* ways of setting up the PROC step, never realizing that a number of less commonly used options and statements exist that can greatly simplify the procedure code, the analysis steps, and the resulting output.

This tutorial introduces a number of important and useful options and statements that can provide the analyst with much needed tools. Some of these tools are new, others have application beyond MEANS/SUMMARY, all have a practical utility. With this practical knowledge, you can greatly enhance the usability of the procedure and then you too will be doing more with MEANS/SUMMARY.

KEY WORDS

OUTPUT, MEANS, SUMMARY, AUTONAME, _TYPE_, WAYS, LEVELS, MAXID, GROUPID, preloaded formats

INTRODUCTION

PROC MEANS is one of SAS®'s original procedures, and it's initial mandate was to create printed tables of summary statistics. Later PROC SUMMARY was introduced to create summary data sets. Although these two procedures grew up on the opposite side of the tracks, over time both has evolved so that under the current version of SAS they actually both use the same software behind the scenes.

These two procedures completely share capabilities. In fact neither can do anything that the other cannot do. Only some of the defaults are different (as they reflect the procedures' original roots).

For the analyst faced with creating statistical summaries, the MEANS/SUMMARY procedure is indispensable. While it is fairly simple to generate a straightforward statistical summary, these procedures allow a complex list of options and statements that give the analyst a great deal of control.

Because of the similarity of these two procedures, examples will tend to show one or the other but not both. When I use MEANS or SUMMARY, I tend to select the procedure based on it primary objective of the step (SUMMARY for a summary data set and MEANS for a printed table). Even that 'rule', however is rather lax as MEANS has the further advantage of only having 5 letters in the procedure name.

BASIC STATEMENTS

The MEANS/SUMMARY procedure is so powerful that just a few simple statements and options can produce fairly complex and useful summary tables.

Using the CLASS Statement

The CLASS statement can be used to create subgroups. Unlike the BY statement the data do not have to be sorted prior to its use. Like in most other procedures that utilize the CLASS statement, there can be one or more classification variables.

In a Summary Data Set

When creating a summary data set, one can get not only the classification variable interaction statistics, but the main factor statistics as well. This can be very helpful to the statistician.

```
title1 'CLASS and a Summary Data Set';
proc summary data=sashelp.class(where=(age in(12,13,14)));
class age sex;
var height;
output out=clsummry n=ht_n mean=ht_mean std=ht_sd;
run
```

A PROC PRINT of the data set CLSUMMRY shows:

CLASS and a Summary Data Set

Obs	Age	Sex	_TYPE_	_FREQ_	ht_n	ht_mean	ht_sd
1	.		0	12	12	61.7583	3.97868
2	.	F	1	6	6	60.8333	3.90470
3	.	M	1	6	6	62.6833	4.18637
4	12		2	5	5	59.4400	3.29742
5	13		2	3	3	61.4333	4.49592
6	14		2	4	4	64.9000	2.80119
7	12	F	3	2	2	58.0500	2.47487
8	12	M	3	3	3	60.3667	3.93234
9	13	F	3	2	2	60.9000	6.22254
10	13	M	3	1	1	62.5000	.
11	14	F	3	2	2	63.5500	1.06066
12	14	M	3	2	2	66.2500	3.88909

Two additional variables have been added to the summary data set; _TYPE_ (which is described below in more detail), and _FREQ_ (which counts observations). Although not apparent in this example, _FREQ_ counts all observations, while the N statistic only counts observations with non-missing values.

If you only want the statistics for the highest order interaction, you can use the NWAY option on the PROC statement.

```
proc summary data=sashelp.class(where=(age in(12,13,14)))
                nway;
```

Understanding _TYPE_

The _TYPE_ variable in the output data set helps us track the level of summarization, and can be used to distinguish the sets of statistics. Notice in the previous example that _TYPE_ changes for each level of summarization.

TYPE = 0	Summarize across all classification variables
TYPE = 1	Summarize as if the right most classification variable (SEX) was the only one
TYPE = 2	Summarize as if the next to the right most classification variable (AGE) was the only one
TYPE = 3	Interaction of the two classification variables.

In the following example there are three CLASS variables and _TYPE_ ranges from 0 to 7.

```
title1 'Understanding _TYPE_';
proc summary data=advrpt.demog(where=(race in('1','4')
                                & 12 le edu le 15
                                & symp in('01','02','03')));
class race edu symp;
var ht;
output out=stats mean= meanHT;
run;
```

Understanding _TYPE_

Obs	RACE	EDU	SYMP	_TYPE_	_FREQ_	mean HT
1		.		0	8	66.25
2		.	01	1	2	64.00
3		.	02	1	4	66.50
4		.	03	1	2	68.00
5		12		2	4	67.50
6		14		2	2	64.00
7		15		2	2	66.00
8		12	02	3	2	67.00
9		12	03	3	2	68.00
10		14	01	3	2	64.00
11		15	02	3	2	66.00
12	1	.		4	6	67.00
13	4	.		4	2	64.00
14	1	.	02	5	4	66.50
15	1	.	03	5	2	68.00
16	4	.	01	5	2	64.00
17	1	12		6	4	67.50
18	1	15		6	2	66.00
19	4	14		6	2	64.00
20	1	12	02	7	2	67.00
21	1	12	03	7	2	68.00
22	1	15	02	7	2	66.00
23	4	14	01	7	2	64.00

When calculating the value of _TYPE_, assign a zero (0) when summarizing over a CLASS variable and assign a one (1) when summarizing for the CLASS variable. In the table below the zeros and ones associated with the class variables form a binary value. This binary value can be converted to decimal to obtain _TYPE_.

	CLASS VARIABLES				
Observations	**RACE**	**EDU**	**SYMP**	**Binary Value**	**_TYPE_**
1	0	0	0	000	0
2 - 4	0	0	1	001	1
5 - 7	0	1	0	010	2
8 - 11	0	1	1	011	3
12 - 13	1	0	0	100	4
14 - 16	1	0	1	101	5
17 - 19	1	1	0	110	6
20 - 23	1	1	1	111	7
	$2^2=4$	$2^1=2$	$2^0=1$		

A binary value of 110 = $1*2^2 + 1*2^1 + 0*2^0 = 1*4 + 1*2 + 0*1 = 6 =$ _TYPE_

Some SAS programmers find converting binary values to decimal values a bit tedious. Fortunately the developers at SAS Institute have provided us with alternatives.

CREATING SUMMARY DATA SUBSETS

Once you have started to create summary data sets with MEANS/SUMMARY, you will soon discover how very useful they can be. Of course you will often find that you do not need all the information contained in the summary data set and that you need to create a data subset. As with most things in SAS there are multiple ways to do this. We have already seen the use of the NWAY option to subset for only the highest order interaction. This is fine but not very flexible. Let's look at some techniques that are a bit more useful.

Using the WAYS and LEVELS Options

The _TYPE_ variable is only one of several ways to identify levels of summarizations in the summary data set. The WAYS and LEVELS options on the OUTPUT statement provide additional discrimination capabilities. These options add the variables _LEVEL_ and _WAY_ to the summary data table.

```
title1 'Using LEVELS and WAYS Options';
proc summary data=advrpt.demog;
class race edu;
var ht;
output out=stats
       mean= meanHT
       /levels ways;
run;
```

LEVELS option

Adds the variable _LEVEL_ to the OUT= data table. This numeric variable counts the observations within _TYPE_. This means that when FIRST._TYPE_ is true _LEVEL_ will equal 1.

WAYS option

Adds the variable _WAY_ to the OUT= data table. This numeric variable equals the number of classification variables that were used to calculate each observation *e.g.* for a three way interaction _WAY_ will equal 3.

Using LEVELS and WAYS Options

Obs	RACE	EDU	_WAY_	_TYPE_	_LEVEL_	_FREQ_	meanHT
1		.	0	0	1	75	67.5200
2		10	1	1	1	11	71.3636
3		12	1	1	2	18	66.8889
4		13	1	1	3	4	70.0000
5		14	1	1	4	11	64.1818
6		15	1	1	5	7	65.2857
7		16	1	1	6	10	70.4000
8		17	1	1	7	10	65.2000
9		18	1	1	8	4	69.0000
10	1	.	1	2	1	41	68.4390
11	2	.	1	2	2	17	67.6471
12	3	.	1	2	3	9	64.8889
13	4	.	1	2	4	4	64.5000
14	5	.	1	2	5	4	66.5000
15	1	10	2	3	1	11	71.3636
16	1	12	2	3	2	15	67.0667
17	1	13	2	3	3	4	70.0000
18	1	15	2	3	4	5	64.2000
19	1	16	2	3	5	2	71.0000
20	1	17	2	3	6	2	63.0000
21	1	18	2	3	7	2	73.0000
22	2	12	2	3	8	3	66.0000
23	2	16	2	3	9	6	71.0000
24	2	17	2	3	10	8	65.7500
25	3	14	2	3	11	7	64.0000
26	3	15	2	3	12	2	68.0000
27	4	14	2	3	13	4	64.5000
28	5	16	2	3	14	2	68.0000
29	5	18	2	3	15	2	65.0000

Using the WAYS and TYPE Statements

In addition to the WAYS and LEVELS options on the OUTPUT statement there are also the WAYS and TYPE statements than can also be used to control what information is written to the summary data set. These have the further advantage of controlling what is actually calculated and can therefore also save computer resources when there are a large number of classification variables.

Controlling Summary Subsets Using WAYS

The WAYS statement can be used to specify a list of combinations of class variables, which are to be displayed. Combinations of the WAYS statement for three classification variables include the following summarizations:

`ways 0;`	across all class variables
`ways 1;`	each classification variable (no cross products)
`ways 2;`	each two way combination of the classification variables
`ways 3;`	three way combination for three classification variables this is the same as using the NWAY option when there are three classification variables.
`ways 0,3;`	lists of numbers are acceptable

When the number of classification variables becomes large the WAYS statement can utilize an incremental list.

```
ways 0 to 9 by 3;
```

In the following example, the main effect summaries (_TYPE_ = 1, 2) are not even calculated.

```
title1 'Using the WAYS Statement';
proc summary
       data=advrpt.demog;
class race edu symp;
var ht;
ways 0,2;
output out=stats
      mean= meanHT;
run;
```

Using the WAYS Statement

Obs	RACE	EDU	SYMP	_TYPE_	_FREQ_	meanHT
1		.		0	64	67.1875
2		10	04	3	6	74.0000
3		10	10	3	3	69.0000
4		12	02	3	2	67.0000

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

Controlling Summary Subsets Using TYPES

The TYPES statement can be used to select and limit the data roll up summaries. The TYPES statement eliminates much of your need to understand the automatic variable _TYPE_. The TYPES statement is used to list those combinations of the classification variable that are desired. Like the WAYS statement this also can be used to limit the number of calculations that need to be performed.

```
title1 'Using the TYPES Statement';
proc summary data=advrpt.demog;
class race edu symp;
var ht;
types edu race*symp;
output out=stats mean= meanHT;
run;
```

Using the TYPES Statement

Obs	RACE	EDU	SYMP	_TYPE_	_FREQ_	meanHT
1		10		2	9	72.3333
2		12		2	15	66.2667
3		13		2	2	68.0000
4		14		2	9	64.0000
5		15		2	7	65.2857
6		16		2	10	70.4000
7		17		2	10	65.2000
8		18		2	2	65.0000
9	1	.	01	5	2	71.0000
10	1	.	02	5	4	66.5000
11	1	.	03	5	2	68.0000

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

For the following CLASS statement

```
class race edu symp;
```

variations of the TYPES statement could include:

```
types ();
types race*edu edu*symp;
types race*(edu symp);
```

Using the CLASSDATA= and EXCLUSIVE Options

You can specify which combinations of levels of the classification variables are to appear in the report by creating a data set that contains the combinations of interest. These can include levels that do not exist in the data itself, but that are to none-the-less appear in the data set or report. The EXCLUSIVE option forces only those levels in the CLASSDATA= data set to appear in the report.

The following example builds the data set that is to be used with the CLASSDATA= option. It also adds a level for each classification variable that does not exist in the data.

```
title1 'Using the CLASSDATA and EXCLUSIVE Options';
data selectlevels(keep=race edu symp);
set advrpt.demog(where=(race in('1','4')
                      & 12 le edu le 15
                      & symp in('01','02','03')));
output;
* For fun add some nonexistent levels;
if _n_=1 then do;
   edu=0;
   race='0';
   symp='00';
   output;
end;
run;
proc summary
     data=advrpt.demog
      classdata=selectlevels
      exclusive;
class race edu symp;
var ht;
output out=stats mean=
meanHT;
run;
```

Using the CLASSDATA and EXCLUSIVE Options

Obs	RACE	EDU	SYMP	_TYPE_	_FREQ_	mean HT
1		.		0	8	66.25
2		.	00	1	0	.
3		.	01	1	2	64.00
4		.	02	1	4	66.50
5		.	03	1	2	68.00
6		0		2	0	.
7		12		2	4	67.50
8		14		2	2	64.00

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

The summary lines for observations 2 and 6 represent levels of the classification variables that do not appear in the data. They were generated thru a combination of the CLASSDATA= data set and the EXCLUSIVE option.

Using the COMPLETETYPES Option

All combinations of the classification variables may not exist in the data and therefore those combinations will not appear in the summary table. If all possible combinations are desired, regardless as to whether or not they exist in the data, use the COMPLETETYPES option on the PROC statement.

```
title1 'Using the COMPLETETYPES Option';
proc summary data=advrpt.demog(where=(race in('1','4')
                      & 12 le edu le 15
                      & symp in('01','02','03')))
```

```
                completetypes;
class race edu symp;
var ht;
output out=stats mean= meanHT;
run;
```

In the data there are no observations with both EDU=12 and SYMP='01', however since both levels exist somewhere in the data, the COMPLETETYPES option causes the combination to appear in the summary data set (obs=8).

Using the COMPLETETYPES Option

Obs	RACE	EDU	SYMP	_TYPE_	_FREQ_	mean HT
1		.		0	8	66.25
2		.	01	1	2	64.00
3		.	02	1	4	66.50
4		.	03	1	2	68.00
5		12		2	4	67.50
6		14		2	2	64.00
7		15		2	2	66.00
8		12	01	3	0	.
9		12	02	3	2	67.00

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

FINDING THE EXTREME VALUES

When working with data, it is not at all unusual to want to be able to identify the observations that contain the highest or lowest values of the analysis variables. These extreme values are automatically displayed in PROC UNIVARIATE output, but must be requested in MEANS/SUMMARY.

As was shown earlier the MIN and MAX statistics show the extreme value, unfortunately they do not identify the observation that contains the extreme. Fortunately there are a couple of ways to do this.

Using MAXID and MINID

The MAXID and MINID options in the OUTPUT statement can be used to identify the observations with the maximum and minimum values. The general form of the statement is:

MAXID(*analysis var*(*ID var*))=*PDV var*

A new variable is added to the OUTPUT data set which takes on the value of the ID variable for the maximum observation.

```
title1 'Using MAXID';
proc summary data=advrpt.demog;
class race edu;
var ht wt;
output out=stats
       mean= meanHT MeanWT
       max=maxHt maxWT
       maxid(ht(subject) wt(subject))=maxHtSubject MaxWtSubject
       ;
run;
```

Using MAXID

Obs	RACE	EDU	_TYPE_	_FREQ_	meanHT	MeanWT	max Ht	max WT	maxHt Subject	MaxWt Subject
1		.	0	75	67.5200	160.267	74	240	110	137
2		10	1	11	71.3636	194.091	74	215	110	109
3		12	1	18	66.8889	167.722	70	240	106	137
4		13	1	4	70.0000	197.000	72	215	148	117

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

The OUTPUT statement could also have been written as:

```
output out=stats
       mean= meanHT MeanWT
       max=maxHt maxWT
       maxid(ht(subject))=maxHtSubject
       maxid(wt(subject))=maxWtSubject
       ;
```

When more than one variable is needed to identify the observation with the extreme value, the MAXID supports a list. As before when specifying lists, there is a one-to-one correspondence between the two lists. In the following OUTPUT statement both the SUBJECT and SSN are used in the list of identification variables. Consequently a new variable is created for each in the summary data set.

```
output out=stats
    mean= meanHT MeanWT
    max=maxHt maxWT
    maxid(ht(subject ssn))=MaxHtSubject MaxHtSSN
    maxid(wt(subject ssn))=MaxWtSubject MaxWtSSN
    ;
```

The MAXID and MINID options allow you to only capture a single extreme. It is also possible to display a group of the extreme values using the GROUPID option.

Using the GROUPID Option

Like the MAXID and MINID options, this option allows you to capture the maximum or minimum value and associated ID variable. More importantly, however, you may select more than just the single extreme value.

```
title1 'Using GROUPID';
proc summary data=advrpt.demog;
class race edu;
var ht wt;
output out=stats
       mean= MeanHT MeanWT
       max(wt)=maxWT ❶
       idgroup(max(wt)out[2](wt subject race)=maxval ❻)
       ;          ❷     ❸       ❹          ❺
run;
```

Using GROUPID

Obs	RACE	EDU	_TYPE_	_FREQ_	MeanHT	MeanWT	max WT ❶	maxval_1 ❺	maxval_2	subject_ 1	subject_ 2	RACE_1 ❻	RACE_2
1		.	0	75	67.5200	160.267	240	240	215	137	109	2	1
2		10	1	11	71.3636	194.091	215	215	215	109	143	1	1
3		12	1	18	66.8889	167.722	240	240	185	137	119	2	1
4		13	1	4	70.0000	197.000	215	215	215	117	163	1	1
5		14	1	11	64.1818	108.091	115	115	115	131	141	4	4

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

❶ MAX statistic is superfluous in this example, and is included only for your reference.

❷ We are asking for the maximum of WT. GROUPID also is available for MIN, therefore in this example we could have also specified:

```
idgroup(min(ht)out[3](ht subject race)=minht minsub minrace)
```

❸ The top 2 values are to be shown

❹ This is a list of variables that will be shown as observation identifiers. The analysis variable is usually included. The MAX statistic has also been requested for comparison purposes ❶, however it will only provide one value and not the next highest.

❺ You can choose the prefix of the ID variable or you can let the procedure do it for you ❻. In either case, a number is appended to the variable name. In this example we can see that the second heaviest subject in the study was subject 137 with a weight of 215 pounds and a RACE of 1.

DOING MORE WITH CLASS STATEMENTS

Using Options on CLASS Statements

The CLASS statement can now accept options. These include:

ASCENDING / DESCENDING
GROUPINTERNAL
MISSING
MLF
ORDER
preloaded format options (discussed below)

Most of the following discussion applies to virtually all SAS procedures that accept the CLASS statement.

ASCENDING / DESCENDING

Normally output (in tables or a summary data set) is placed in ascending order for each classification variable. You can change this by using the DESCENDING option on the CLASS statement.

```
title1 'Using the DESCENDING
CLASS Option';
proc summary
data=advrpt.demog;
class race/descending;
var ht wt;
output out=stats
       mean= MeanHT MeanWT
       ; run;
```

Using the DESCENDING CLASS Option

Obs	RACE	_TYPE_	_FREQ_	MeanHT	MeanWT
1		0	76	67.5526	160.461
2	5	1	4	66.5000	147.000
3	4	1	4	64.5000	113.500
4	3	1	9	64.8889	111.222
5	2	1	17	67.6471	162.000
6	1	1	42	68.4762	176.143

GROUPINTERNAL

When a classification variable is associated with a format, the format is used when forming groups.

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

title1 'Without Using the GROUPINTERNAL CLASS Option';
proc summary data=advrpt.demog;
class edu;
var ht wt;
output out=stats
       mean= MeanHT MeanWT
       ;
format edu edulevel.;
run;
```

The resulting table will show at most three levels for EDU. To use the original data values (internal values), the GROUPINTERNAL option is added to the CLASS statement.

```
class edu/groupinternal;
```

MISSING

When a classification variable takes on a missing value that observation is eliminated from the analysis. If a missing value is OK or if the analyst needs to have it included in the summary, the MISSING option can be used. Most procedures that have either an implicit or explicit CLASS statement also have a MISSING option. However when the MISSING option is used on the PROC statement it is applied to all the classification variables and this may not be acceptable. By using the MISSING option on the CLASS statement you can control which classification variables are to be handled differently.

In the following example there are three classification variables. However the MISSING option has only been applied to two of them.

```
title1 'Using the MISSING CLASS Option';
proc means data=advrpt.demog n mean std;
class race ;
class edu symp/ missing;
var ht wt;
run;
```

ORDER

When classification variables are displayed or written to a table the values are ordered according to one of several possible schemes. These include:

- data — order is based on the order of the incoming data
- formatted — values are formatted and then ordered (default when the variable is formatted)
- freq — the order is based on the frequency of the class level
- unformatted — same as INTERNAL or GROUPINTERNAL

Using the `order=freq` option on the CLASS statement causes the table to be ordered according to the most common levels of education.

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

Using the ORDER CLASS Option

The MEANS Procedure

years of education	N Obs	Variable	Label	N	Mean	Std Dev
12	19	HT	height in inches	19	66.9473684	2.7582942
		WT	weight in pounds	19	171.5263158	32.2703311
14	11	HT	height in inches	11	64.1818182	0.4045199
		WT	weight in pounds	11	108.0909091	4.3921417
10	11	HT	height in inches	11	71.3636364	3.2022719
		WT	weight in pounds	11	194.0909091	19.0811663
17	10	HT	height in inches	10	65.2000000	2.3475756
		WT	weight in pounds	10	145.2000000	25.0900600

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

Using Multiple CLASS Statements

Because CLASS statements now accept options, and because those options may not apply to all the classification variables, it is often necessary to specify multiple CLASS statements - each with its own set of options.

With or without options, when multiple CLASS statements are specified, the order of the statements themselves becomes important. The following CLASS statement

```
class race edu;
```

could be rewritten as

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

PRELOADED FORMATS

Several options and techniques are available to control which levels of classification variables are to appear in the summary. Those that were discussed earlier in this paper include the CLASSDATA and COMPLETETYPES options. Also discussed were the WAYS and TYPES statements, as well as the WAYS and LEVELS options on the OUTPUT statement.

A related set of options come under the general topic of Preloaded Formats. Variations of these options are available for most of the procedures that utilize classification variables. Like the others listed above these techniques/options are used to control the relationship of levels of classification variables that may not appear in the data and how those levels are to appear (or not appear) in the summary.

Generally speaking when a level of a classification variable is not included in the data, the associated row will not appear in the table. This behavior relative to the missing levels can be controlled through the use of *preloaded* formats.

For the MEANS/SUMMARY procedures, options used to preload formats include:

PRELOADFMT	Loads the format levels prior to execution. This option will always be present when you want to use a preloaded format.
EXCLUSIVE	Only data levels that are included in the format definition are to appear in summary table
COMPLETETYPES	All levels representing format levels are to appear in the summary

It is the interaction of these three options that gives us a wide range of possible outcomes. In each case the option PRELOADFMT will be present.

As the name of the technique implies, the control is maintained through the use of user defined formats. For the examples that follow, the format $SYMPX has been created, and it contains one level, '00', that is not in the data. In the data the values of SYMP range from '01' to '10'.

```
proc format;
value $sympx
  '01' = 'Sleepiness'
  '02' = 'Coughing'
  '00' = 'Bad Code' ;
run;
```

PRELOADFMT with EXCLUSIVE

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 or a WHERE clause.

```
title1 'Preloading and the EXCLUSIVE Option';
proc summary data=advrpt.demog;
class symp / preloadfmt
             exclusive;
var ht;
output out=stats mean= meanHT;
format symp $sympx.;
run;
```

Only symptoms that are both on the format $SYMPX. and in the data, are included on the summary table.

Preloading and the EXCLUSIVE Option

Obs	SYMP	_TYPE_	_FREQ_	mean HT
1		0	14	67.0
2	Sleepiness	1	4	67.5
3	Coughing	1	10	66.8

PRELOADFMT with the COMPLETETYPES Option

The COMPLETETYPES option requests that all combinations of levels appear in the summary. When it is used with preloaded formats, the complete list of levels comes from the format rather than from the data itself. In this example the format %SYMPX. is again preloaded, however rather than using the EXCLUSIVE CLASS statement option, the COMPLETTYPES option appears on the PROC statement.

```
title1 'Preloading and the COMPLETETYPES Option';
proc summary data=advrpt.demog completetypes;
class symp / preloadfmt;
var ht;
output out=stats mean= meanHT;
format symp $sympx.;
run;
```

The summary now contains an observation for each SYMP in the data as well as each in the format $SYMPx.

Preloading and the COMPLETETYPES Option

Obs	SYMP	_TYPE_	_FREQ_	meanHT
1		0	65	67.2000
2	Bad Code	1	0	.
3	Sleepiness	1	4	67.5000
4	Coughing	1	10	66.8000
5	03	1	4	66.5000
6	04	1	13	68.6923

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

PRELOADFMT with the COMPLETETYPES and the EXCLUSIVE Options

When a preloaded format is used with both the COMPLETETYPES and the EXCLUSIVE options, the summary includes all levels of the format, but not necessarily all levels in the data.

```
title1 'Preloading With Both';
title2 'the COMPLETETYPES and EXCLUSIVE Options';
proc summary data=advrpt.demog
             completetypes;
class symp / preloadfmt
             exclusive;
var ht;
output out=stats
       mean= meanHT;
format symp $sympx.;
run;
```

```
Preloading With Both
the COMPLETETYPES and EXCLUSIVE Options

                                                  mean
Obs       SYMP         _TYPE_     _FREQ_         HT

 1                        0         14          67.0
 2      Bad Code          1          0            .
 3      Sleepiness        1          4          67.5
 4      Coughing          1         10          66.8
```

SUMMARY

The MEANS /SUMMARY procedure produces a wide variety of summary reports and summary data tables. It is very flexible and, while it can be quite complex, a few basic statements allow the user to create useful summaries.

As you develop a deeper knowledge of the MEANS/SUMMARY procedure, you will find that the generation of highly sophisticated summarizations is possible from within a single step.

ABOUT THE AUTHOR

Art Carpenter's publications list includes four books, and numerous papers and posters presented at SUGI, SAS Global Forum, and other user group conferences. Art has been using SAS® since 1976 and has served in various leadership positions in local, regional, national, and international user groups. He is a SAS Certified Professional™ 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

TRADEMARK INFORMATION