Paper 83-26

Summarizing to the one-record-per-person using PROC SUMMARY: when to use CLASS and when to use BY.

Jenny Yu, Lewin Group, San Francisco, CA

ABSTRACT

Many data management projects require a programmer to read and summarize very large administrative databases. Often these data are stored longitudinally, containing numerous assessments or observations per individual. For instance, much of the data used in health care studies (insurance claims data, clinical trials, and disease registries) fall into this category. It is often necessary to create a one-record-per-person summary from these data. There are various ways to accomplish this task using SAS – each with certain strengths and drawbacks. DATA step programming can be used, but often it can be easier, faster, and more efficient to use PROC SUMMARY.

This paper shows how to solve some common summarization tasks using PROC SUMMARY. The author offers guidelines for when it is more efficient to use BY processing and when it is more efficient to use the CLASS statement.

INTRODUCTION

In longitudinal databases (e.g., claims data, disease registry data, etc.), usually there are multiple records per person, that, in sum, contain the information needed to create the desired unit of analysis (e.g., counts, events, etc.). Therefore, it is common to summarize information to the person level in order to perform the required analyses. One quick way to do this is to use PROC SUMMARY, and use the person identifier in the CLASS statement. However, when classification becomes more complex – e.g., numbers that need to be reported by gender, race, year of claim – processing all of these variables in the CLASS statement can become burdensome on CPU processing.

When you have a very large data set where these issues come into play, there are alternate ways to address this problem. This paper will discuss using the person identifier in the BY statement. A typical example using claims data follows, and three valid methods to accomplishing the same goal are presented.

SAS V6.12 was used for all examples in this paper.

TYPICAL CLAIMS DATA EXAMPLE PROBLEM:

How many claims did each patient have for drug A or drug B in 1999 and 2000?

This example is based on a sample data set of pharmacy claims with over 1 million observations for drugs A and B, over the time period from July 1998 to March 2000. In general, the way to answer this question is to do the following: create a one-record-per-patient level data set that has at least the following variables:

Patient ID

- The number of claims for drug A in 1999
- The number of claims for drug A in 2000
- The number of claims for drug B in 1999
- The number of claims for drug B in 2000

This paper will discuss three ways to create a patient-level data set with the summary variables listed above. All three methods are valid, each with its own strengths and weaknesses. They are:

1. Using the DATA STEP.
2. Running multiple PROC SUMMARY's using a WHERE clause, within a macro (macro is optional), with the person identifier in the CLASS statement.
3. Running one PROC SUMMARY with the person identifier in the BY statement.

METHOD 1) USING DATA STEP PROGRAMMING:

This method uses the DATA STEP to count the number of records for each patient, (with one record representing a single claim):

```
data temp03;
  set temp01;
  by px_id filldate;

  retain yr_1a yr_2a yr_1b yr_2b;

  if first.px_id then do;
    yr_1a=0;
    yr_2a=0;
    yr_1b=0;
    yr_2b=0;
  end;

if year(filldate)=1999 and drugtype='A'
  then yr_1a = yr_1a + 1;
else if year(filldate)=2000 and drugtype='A'
  then yr_2a = yr_2a + 1;
else if year(filldate)=1999 and drugtype='B'
  then yr_1b = yr_1b + 1;
else if year(filldate)=2000 and drugtype='B'
  then yr_2b = yr_2b + 1;

  format drugtype $drugfmt.;
  if last.px_id then output;
run;
```

NOTE: The data set WORK.TEMP03 has 37671 observations and 35 variables.
NOTE: The DATA statement used 1 minute 27.52 seconds.

```
title2 'Method #1:  Data TEMP03';
proc print data=temp03 (obs=10);
  var px_id yr_1a yr_2a yr_1b yr_2b;
  run;
```

(see Appendix for output)

In this DATA STEP, the RETAIN statement creates the new summary variables (YR_1A, YR_2A, YR_1B, YR_2B), and they are initialized to 0 for the first record of each patient (using the 'FIRST.' technique, as the data set has already been previously sorted by PX_ID). As SAS goes through each record, these summary variables are incremented by one if the conditions are true. When

the last record of each patient is reached, this last record is output to TEMP03, which will contain one record per patient with the total counts for drug A or B in 1999 and 2000.

Using this method, the log notes show that there are 37,671 total patients in this data set of over 240,000 claims. Some have multiple claims in year 1 and year 2, and some have none (i.e., their claims were in 1998).

METHOD 2) USE 4 PROC SUMMARY'S WITH A WHERE FOR EACH DRUG AND YEAR:

This method requires much less code than method #1. The SAS macro facility is used in this example, because the same PROC SUMMARY will be run four times, with only a few arguments changing. The same result can occur by coding four separate PROC SUMMARY's manually.

Our objective is to report summary data at the patient level, and PROC SUMMARY can accomplish this goal:

```
%macro summ(drug=, fill_yr=, out=, n=);
  proc summary nway data=temp01
    (where=(drugtype="&drug" and
            year(filldate)=&fill_yr));
    class px_id ;
    id drugtype;
    var filldate;
    output out=&out n=&n;
    run;
%mend summ;

%summ(drug=A, fill_yr=1999, out=out_1a, n=yr_1a);
NOTE: The data set WORK.OUT_1A has 32642
observations and 5 variables.
NOTE: The PROCEDURE SUMMARY used 1 minute 22.96
seconds.

%summ(drug=A, fill_yr=2000, out=out_2a, n=yr_2a);
NOTE: The data set WORK.OUT_2A has 16329
observations and 5 variables.
NOTE: The PROCEDURE SUMMARY used 1 minute 17.41
seconds.

%summ(drug=B, fill_yr=1999, out=out_1b, n=yr_1b);
NOTE: The data set WORK.OUT_1B has 706 observations
and 5 variables.
NOTE: The PROCEDURE SUMMARY used 1 minute 14.48
seconds.

%summ(drug=B, fill_yr=2000, out=out_2b, n=yr_2b);
NOTE: The data set WORK.OUT_2B has 571 observations
and 5 variables.
NOTE: The PROCEDURE SUMMARY used 1 minute 14.82
seconds.
```

The output data set from each PROC SUMMARY is at the patient level because the patient identifier - "PX_ID" - is in the CLASS statement. Also, these output data sets will only include observations for the PX_ID for the year and drug specified in the WHERE clause. For example, data set OUT_1A contains 32,642 observations. Therefore, there were 32,642 patients with at least one claim for drug A in 1999.

Now the next step is to combine all of these output data sets into one single file. These data sets can be easily merged BY PX_ID, because they are already inherently sorted. Also, the non-summary variables match (PX_ID, DRUGTYPE), and the summary variables created in PROC SUMMARY do not match, so they will not be overwritten when these data sets are merged. Finally, zero's are imputed for missing values, because in this case, these missing values indicate that there were no claim records for the patient for the specified drug and year:

```
data temp04 (drop=_freq_);
  merge out_1a
 out_1b
 out_2a
 out_2b;
  by px_id;

  if yr_1a=. then yr_1a=0;
  if yr_2a=. then yr_2a=0;
  if yr_1b=. then yr_1b=0;
  if yr_2b=. then yr_2b=0;
  run;

NOTE: The data set WORK.TEMP04 has 37269
observations and 7 variables.
NOTE: The DATA statement used 1.86 seconds.

title2 "Method #2:  Data TEMP04";
proc print data=temp04 (obs=10);
  run;
(see Appendix for output)
```

METHOD 3) USE ONE PROC SUMMARY WITH A BY:

Of the three methods, this is the most efficient with respect to CPU efficiency, and requires the least amount of code. There are a couple of subtle changes to the code from method #2: instead of putting PX_ID in the CLASS statement, PX_ID goes in the BY statement. And, instead using DRUGTYPE and the flags YR_1 and YR_2 in the WHERE clause and running 4 PROC SUMMARY's, these variables are put in the CLASS statement, so that only one PROC SUMMARY is necessary:

```
proc summary nway data=temp01 (where=(drugtype in
('A','B')));
  by px_id;
  class yr_1 yr_2 drugtype;
  var filldate;
  output out=ab_claim n=n_ab;
  run;

NOTE: The data set WORK.AB_CLAIM has 64420
observations and 7 variables.
NOTE: The PROCEDURE SUMMARY used 1 minute 31.17
seconds.
```

By using YR_1, YR_2 (these are flag variables) and DRUGTYPE in the CLASS statement, there will be up to 6 observations per PX_ID (since, in this case, a patient cannot have both drug A and B). Therefore, we still need to create a patient level file. This can be easily accomplished by using the DATA STEP:

```
data temp05 (drop=n_ab _freq_);
 set ab_claim ;
 by px_id;
 retain n_1a n_2a n_1b n_2b;

 if first.px_id then do;
   n_1a=0;
   n_1b=0;
   n_2a=0;
   n_2b=0;
 end;

if yr_1=1 and drugtype='A' then n_1a=n_ab*yr_1;
```

```
if yr_2=1 and drugtype='A' then n_2a=n_ab*yr_2;
if yr_1=1 and drugtype='B' then n_1b=n_ab*yr_1;
if yr_2=1 and drugtype='B' then n_2b=n_ab*yr_2;

if last.px_id then output temp06;
 run;
```

NOTE: The data set WORK.TEMP05 has 37671 observations and 9 variables.
NOTE: The DATA statement used 1.11 seconds.

```
title "Method #3:  Data set TEMP05";
proc print data=temp06 (obs=10);
 run;
(see Appendix for output)
```

A comparison of output data sets from all three methods demonstrates that the results are the same for each patient. The only difference is which variables were saved in to the output data set. In TEMP04, in addition to the variables in the CLASS and ID statements (along with _TYPE_ and _FREQ_), the summary variables create in the PROC SUMMARY are shown in the output data set. None of the variables in the WHERE clause are output.

In TEMP05, there are more variables in the output data set, because there were more variables in CLASS statement, and BY was also used. The YR_1 and YR_2 are flags to indicate whether the patient had a claim in those years. These flags were specified in the CLASS statement.

Finally, _TYPE_ is different in TEMP04 and TEMP05, because of the number of levels created by the variables in the CLASS statement (see discussion on NWAY at the end of this paper).

COMPARING THE THREE METHODS:

The examples described in this paper all have one similar goal: to create a patient-level data set with summary variables. Each method used in this paper has its strengths and weaknesses.

Method #1: By using the DATA STEP to create the patient level data set, the code clearly states the steps taken to create the final output data set, and no knowledge of special PROC's are necessary. Also, it is easy to keep any extra variables that are part of the original data set.

However, the original variables are probably not necessary since the final unit of analysis has changed from one-record-per-claim to one-record-per-patient. Also, the data need to be pre-sorted; with very large data sets, this can potentially take a very long time (or may not be possible). Finally, once again, with very large data sets, this approach can be inefficient if you need to keep re-testing the code (in this case, 1 minute, 27.52 seconds).

Method #2: By using multiple PROC SUMMARY's with only the criteria changing, the output data sets have similar variables and are inherently sorted, thus making them easy to MERGE to create the final data set. Also, by using PX_ID in the CLASS statement, the output data sets will be at the patient level, which is ultimately what we want. Finally, if extra variables in the final data set (such as DRUGTYPE) need to appear, you can simply include these variables in the ID statement. However, be very careful that these variables are constant for each patient and do not change within each CLASS, and should also be non-missing for the last observation of the patient.
Although the code involved in accomplishing this goal is less than method #1, the processing time is greatly increased (about 5.5 minutes versus 1.5 minutes in method #1). This is because the data set needs to be processed four times (due to the variations in the WHERE clause) as opposed to only once in method #1. Plus, familiarity with macro processing is essential.

Method #3: In this method, PX_ID is used in the BY statement (instead of CLASS), with the other classification variables in the CLASS statement (instead of in the WHERE clause). These changes allow the use of only one PROC SUMMARY to output all of our required summary data.

However, the output data set is not at the patient level, so some further data manipulation is required to create a final patient level data set.

Nevertheless, compared to method #1, the processing time is approximately the same (about 1.5 minutes), but much less code is required. Compared to method #2, both processing time and code are greatly reduced (about 5.5 minutes versus 1.5 minutes).

QUICK NOTE ON NWAY:

The examples used in this paper use the NWAY option, so that the output data sets only include the combinations of nonmissing values for all of the variables in the CLASS statement – i.e., observations of the highest _TYPE_. The _TYPE_ variable is a function of the number of levels in the CLASS statement.

However, the NWAY option does not supply grand totals (overall and by each variable in the CLASS statement – i.e., each _TYPE_). To get these, do not use the NWAY option.

CONCLUSION – USING CLASS VERSUS BY:

This paper describes three valid ways to solve a common problem – summarizing data sets to the person-level. When working with very large data sets (i.e. millions of records), such as claims or transaction data, PROC SUMMARY becomes a very efficient method to solve such a problem.

By using the person identifier *alone* in the CLASS statement – you will get a one-record-per-person output data set. However, when you need to produce summaries by all possible combinations of multiple variables (e.g., year 1 * year 2 * drug A * drug B), – the combinations of these variables with the person identifier creates a large number of levels to process. In SAS 6.12, the limit on the maximum numbers of levels in the CLASS statement is 200 million, and maximum number of CLASS variables is 30. While these limits are theoretical, these estimates vary depending on the machine. Therefore, processing may become burdensome much earlier. On smaller data sets (i.e., <500,000 records), this difference in burden on CPU processing is not as much of an issue.

Therefore, as a rule of thumb, when you have a combination of variables for which you need to view summary data – use BY for the variables with a relatively large number of values (e.g., person identifier), and use CLASS for the variables with the relatively small number of values (e.g., gender, race, year of prescription claim).

CONTACT INFORMATION:

Jenny Yu
jenny88keys@yahoo.com

Appendix – SAS output

Method #1: Data TEMP03

OBS	PX_ID	YR_1A	YR_2A	YR_1B	YR_2B
1	1	10	0	0	0
2	2	0	0	0	0
3	3	0	0	0	5
4	4	5	0	0	0
5	5	25	15	0	0
6	6	0	0	20	0
7	7	25	0	0	0
8	8	50	15	0	0
9	9	10	0	0	0
10	10	0	0	50	15

Method #2: Data TEMP04

OBS	DRUGTYPE	PX_ID	_TYPE_	YR_1A	YR_2A	YR_1B	YR_2B
1	Drug A	1	1	10	0	0	0
2	Drug B	2	1	0	0	0	5
3	Drug A	3	1	5	0	0	0
4	Drug A	4	1	25	15	0	0
5	Drug B	5	1	0	0	20	0
6	Drug A	6	1	25	0	0	0
7	Drug A	7	1	50	15	0	0
8	Drug A	8	1	10	0	0	0
9	Drug B	9	1	0	0	50	15
10	Drug A	10	1	25	5	0	0

Method #3: Dataset TEMP05

OBS	PX_ID	YR_1	YR_2	DRUGTYPE	_TYPE_	N_1A	N_2A	N_1B	N_2B
1	1	1	0	Drug A	7	10	0	0	0
2	2	0	0	Drug A	7	0	0	0	0
3	3	0	1	Drug B	7	0	0	0	5
4	4	1	0	Drug A	7	5	0	0	0
5	5	1	0	Drug A	7	25	15	0	0
6	6	1	0	Drug B	7	0	0	20	0
7	7	1	0	Drug A	7	25	0	0	0
8	8	1	0	Drug A	7	50	15	0	0
9	9	1	0	Drug A	7	10	0	0	0
10	10	1	0	Drug B	7	0	0	50	15