Paper CC-09

The Power of Combining Data with the PROC SQL

Stacey Slone, University of Kentucky Markey Cancer Center

ABSTRACT

Combining two data sets which contain a common identifier with a MERGE statement in a data step is one of the basic fundamentals of data manipulation in SAS®. However, one quickly realizes that there are many situations in which a simple merge grows significantly more complicated. Real world data is usually not straightforward and often analyses require combining radically different databases. These situations require a more sophisticated type of merging. Using the SQL procedure, instead of the more traditional data step, is a powerful solution to merging data in complicated situations.

This paper will demonstrate how to combine data using PROC SQL for more complex situations.

INTRODUCTION

Merging more than one data set within a DATA step is usually a fundamental step in creating analysis data sets within SAS. However, PROC SQL can also be a powerful tool for building data sets. It can perform many of the same functions as a data step. For complex situations where merges are not straightforward or summary data is desired, PROC SQL can simplify coding into one step. This paper will focus on four situations in which using PROC SQL to merge and create data sets can greatly improve efficiency and streamline coding, including:

1. Adding summary statistics to data sets,
2. Transposing a portion of the data,
3. Range merges,
4. Combining summary statistics within a range merge.

First, the data to be used in the examples will be introduced. Secondly, a basic example of PROC SQL code to merge data will be demonstrated and compared to the analogous DATA step code.

LUNG CANCER SAMPLE DATA

To illustrate the merging of data with PROC SQL, three samples data sets from a completed randomized Phase II study in lung cancer are utilized. The study consisted of 4 cycles of chemotherapy and assessed platelet levels with at least 4 blood draws per cycle.

The first data set is the baseline data set which includes the patient identifier (PT_ID), age, gender and treatment arm (ARM). There are 33 observations in the baseline data set, BASELINE, limited to patients in the active arm. The first 5 observations are shown in Output 1. The second data set, CHEMO, contains the dates of each patient's four chemotherapy cycles along with the cycle of chemotherapy. The CHEMO data set contains the patient identifier (PT_ID), the cycle number (CYCLE) and the date of the chemotherapy treatment (D_DOSE). The third data set, PLATELETS, will be introduced in a later section.

pt_id	arm	gender	age
0145	ARM 2	FEMALE	60
0510	ARM 2	MALE	62
0591	ARM 2	MALE	64
1162	ARM 2	MALE	67
2807	ARM 2	FEMALE	65

Output 1: Sample of Baseline Data Set

pt_id	Cycle	d_dose
0145	CYCLE 1	11/07/2008
0145	CYCLE 2	11/28/2008
0145	CYCLE 3	01/09/2009
0145	CYCLE 4	02/06/2009
0479	CYCLE 1	04/02/2007

Output 2: Sample of Chemo Data Set

BASIC DATA MERGE WITH PROC SQL

The simplest code for merging two data with PROC SQL is shown below. The SELECT * tells PROC SQL to include all variables from the BASELINE and CHEMO data sets. The FROM statement specifies the data sets to be merged by the limitations given in the corresponding WHERE statement. Finally, the ORDER BY statement provides the order in which the created data set, BASIC, should be sorted.

```
proc sql;
create table basic as
select *
from baseline, chemo
where baseline.pt_id=chemo.pt_id
order by cycle, d_dose, pt_id;
```

Merged Baseline and Chemotherapy Data

pt_id	age	gender	arm	cycle	d_dose
4798	45	FEMALE	ARM 2	CYCLE 1	02/10/06
4287	59	FEMALE	ARM 2	CYCLE 1	05/12/06
3762	57	FEMALE	ARM 2	CYCLE 1	05/30/06
8960	62	MALE	ARM 2	CYCLE 1	06/28/06
3978	55	FEMALE	ARM 2	CYCLE 1	07/21/06

Output 3: Basic Example Output

The analogous code using a DATA step is shown below. Note that both data sets in the following code would need to be sorted by pt_id prior to the DATA step being executed. Also, to change the sort order of the created data set, a PROC SORT must follow the DATA step.

```
data basic;
merge baseline(in=in1) chemo(in=in2);
by pt_id;
run;

proc sort data=basic;
by cycle;
run;
```

PROC SQL has a few advantages that are apparent even with the simplest of examples.

- The data does not have to be presorted by the variables listed in the WHERE statement.
- The variables in the WHERE statement may have different names.
- The output dataset can be created with whatever sort order is specified in the ORDER BY statement.

ADDING SUMMARY STATISTICS TO DATA SETS

One expedient aspect of using PROC SQL for data merges is being able to add summary statistics into to a data set in one procedure call. For example, assume we are interested in looking at the number of chemotherapy cycles each patient received over the course of the clinical trial. The goal is to create a data set that contains one observation per patient that includes demographic information and the total number of cycles per patient. Using the BASIC data set created in our simple example above, this data set can be created within one PROC SQL call as illustrated in the following code.

```
proc sql;
create table example1 as
select distinct *
from (select pt_id, age, gender, n (distinct cycle) as num_cycle
    from basic
    group by pt_id)
order by pt_id;
```

The first DISTINCT function causes the output data set to eliminate duplicate rows. The nested SELECT/FROM statement counts the number of distinct cycles and the corresponding GROUP BY statement causes the count to be on a per patient basis. This summarized data serves as the basis for the new data set EXAMPLE1. Please note that without the nested GROUP BY statement, NUM_CYCLE would equal 4, the total number of distinct cycles, for each patient.

A printout of the first 5 observations of the EXAMPLE1 data set is below:

Summary Statistics: Total Cycles of Chemotherapy

pt_id	age	gender	num_cycle
0145	60	FEMALE	4
0510	62	MALE	3
0591	64	MALE	2
1162	67	MALE	2
2807	65	FEMALE	4

Output 4: Adding Summary Statistics

TRANSPOSING A PORTION OF DATA

Assume next that we are interested in analyzing the median number of days between the initiation of cycles for the subset of patients who received all four courses of chemotherapy. To analyze the data, we must first transpose the data in order for the number of days between respective cycles to be computed. Using PROC SQL, the calculations can be completed and the data summarized within one procedure call.

```
proc sql;
create table example2 as
select a.pt_id, a.age, a.gender, b.d_dose-a.d_dose as dif1, c.d_dose-b.d_dose as
    dif2, d.d_dose-c.d_dose as dif3,
    median(calculated dif1, calculated dif2, calculated dif3) as mcl
    label="Median Cycle Length"
from (select pt_id, age, gender, d_dose from basic where cycle='CYCLE 1') a,
     (select pt_id, d_dose from basic where cycle='CYCLE 2') b,
     (select pt_id, d_dose from basic where cycle='CYCLE 3') c,
     (select pt_id, d_dose from basic where cycle='CYCLE 4') d
where (a.pt_id=b.pt_id) & (b.pt_id=c.pt_id) & (c.pt_id=d.pt_id)
order by a.pt_id;
```

Creating four subsets of the data in the FROM statement allows us to transpose the data such that each cycle date becomes a variable instead of an observation. The letter notations (a,b,c,d) after each nested SELECT/FROM statement provides aliases for each of the data subsets that can then be referenced. The variables age and gender were only kept in the first subset since only they are unique to each patient and not to each cycle. The new variables DIF1-DIF3 are the number of days between subsequent cycles. The calculation is performed in the SELECT statement and a new variable name attached. In addition, the median of these three new variables is also added to the data set in a variable called MCL. In order for PROC SQL to compute summary statistics on any new variables created within the particular procedure call, the term CALCULATED must be added in front of the newly created variable. This is demonstrated in with the MEDIAN function in the above code. Also, a label can be added to any variable with the SELECT statement as shown above with the MCL variable. Formats may be added similarly. The first five observations from the resulting data set are printed below.

Summary Statistics: Median Days Between Cycles

pt_id	age	Sex	dif1	dif2	dif3	Median Cycle Length
0145	60	FEMALE	21	42	28	28
2807	65	FEMALE	20	21	21	21
3119	63	FEMALE	21	24	18	21
3720	67	FEMALE	21	35	28	28
3978	55	FEMALE	21	21	28	21

Output 5: Transposing Data Example

RANGE MERGES

For the next two examples, an additional data set from the lung cancer trial will be introduced. The PLATELETS data set consists of all platelet values collected for each patient during the clinical trial. The variables include the patient identifier (PT_ID), the visit code (VISIT), the date of the blood draw (D_LAB) and the observed platelet value (PLT). If a patient's platelets were not in the appropriate range to receive the upcoming course of chemotherapy, then the patient's blood was redrawn a few days later and the visit labeled as "Unscheduled". These visits occurred until the patient was eligible to receive chemotherapy; hence, sometimes delaying a course by a week or more.

Five observations are printed in the output below:

Platelets Data Set (5 Obs)

pt_id	visit	d_lab	plt
0145	UNSCHEDULED	01/05/09	270
0145	DAY 1 CYCLE 4	01/26/09	94
0145	UNSCHEDULED	01/27/09	128
0145	UNSCHEDULED	01/28/09	183
0145	UNSCHEDULED	02/02/09	460

Output 6: Sample of PLATELETS Data Set

For this next example, we are interested in all platelet values measured within 1 week prior to each chemotherapy cycle. First, we need to merge the chemotherapy data, CHEMO, with the platelet data, PLATELET, to be able to compare the date of the chemotherapy (D_DOSE) with the date of each blood draw (D_LAB). Creating this data set would require difficult and tricky programming with a regular DATA step since we need to perform a "many to many" merge with no common variable on which to perform the merge. The dates of the chemotherapy and the blood draw are not the same and may not even be associated with the same cycle. Recall that if a patient's platelets were not in the proper range to receive chemotherapy, another blood draw was completed a few days later and the visit was labeled as "Unscheduled" with no associated cycle.

However by using PROC SQL and adding a condition to compare the date of chemotherapy with the date of the blood draw within the WHERE statement, the data set will be limited to the visits we want to match. The code followed by a PROC PRINT of the first 10 observations is below.

```
proc sql;
create table example3 as
select chemo.pt_id, chemo.d_dose, chemo.visit as chemo, platelets.visit,
platelets.d_lab,  platelets.plt
from chemo, platelets
where (chemo.pt_id=platelets.pt_id) & (0 le chemo.d_dose-platelets.d_lab le 7)
order by pt_id, d_lab;
```

All Blood Draws within 7 Days Prior to Chemotherapy

pt_id	d_dose	chemo_cycle	lab_draw	d_lab	plt
0145	11/07/08	CYCLE 1	SCREENING	11/03/08	287
0145	11/28/08	CYCLE 2	WEEK 3 CYCLE 1	11/21/08	58
0145	11/28/08	CYCLE 2	DAY 1 CYCLE 2	11/24/08	113
0145	11/28/08	CYCLE 2	DAY 5 CYCLE 2	11/28/08	301
0145	01/09/09	CYCLE 3	UNSCHEDULED	01/05/09	270
0145	01/09/09	CYCLE 3	DAY 5 CYCLE 3	01/09/09	270
0145	02/06/09	CYCLE 4	UNSCHEDULED	02/02/09	460
0145	02/06/09	CYCLE 4	DAY 5 CYCLE 4	02/06/09	461
0479	04/02/07	CYCLE 1	SCREENING	04/02/07	250
0479	04/02/07	CYCLE 1	DAY 5 CYCLE 1	04/02/07	250

Output 7: Range Merge Example

RANGE MERGE WITH SUMMARY STATISTICS

For the final example, we want to focus on the nadir platelet value measured within the week prior to chemotherapy. We can use PROC SQL to summarize the data set we created in the previous example to give us the nadir value. The code to calculate the minimum platelet value is

```
proc sql;
create table example4 as
select distinct *
from (select pt_id, chemo_cycle, min(plt) as nadir_plt
     from example3
     group by pt_id, chemo_cycle)
order by pt_id;
quit;
```

Nadir Platelet Value within Blood Draws within 7 Days of Chemotherapy

pt_id	chemo_ cycle	nadir_ plt
0145	CYCLE 1	287
0145	CYCLE 2	58
0145	CYCLE 3	270
0145	CYCLE 4	460
0479	CYCLE 1	250
0479	CYCLE 2	175
0479	CYCLE 3	156
0479	CYCLE 4	143

Output 8: Range Merge & Summary Statistics Example

Note that the previous data set can be created directly from the raw chemotherapy and blood draw data sets by nesting subqueries to create the data. The following code results in the same data set as the previous example except it uses the raw data sets, CHEMO and PLATELETS.

```
proc sql;
select distinct pt_id, chemo_cycle, nadir_plt
from (select chemo.pt_id, chemo.d_dose, chemo.cycle as chemo_cycle, plt.visit as
     lab_draw, plt.d_lab, plt.plt, min(plt.plt) as nadir_plt
     from chemo, platelets plt
     where (chemo.pt_id=plt.pt_id) & (0 le chemo.d_dose-plt.d_lab le 7)
     group by chemo.pt_id, chemo_cycle)
order by pt_id;
```

CONCLUSION

PROC SQL is a valuable tool for the statistical SAS® programmer. There are several advantages to using PROC SQL in place of a DATA step even in the simple instances of merging data. However, the true strength in merging data with PROC SQL lies in the situations when tricky data merging is required. In particular, when summary statistics or range merges are required. The power of PROC SQL comes from being able to nest queries and perform multiple tasks within one call of the procedure.

The primary weakness to PROC SQL is that it can be trickier to merge multiple data sets at one time. However, since one can include multiple CREATE TABLE AS statements within one call of PROC SQL, it is not unmanageable. This paper only touches on the possibilities of using PROC SQL in respect to the DATA step. PROC SQL is a versatile, powerful tool to have at your disposal.

REFERENCES

SAS Institute Inc., 2004. *SAS® 9.1 SQL Procedure User's Guide*. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

The author would like to thank Dr. Susanne Arnold for sharing a portion of data from her randomized Phase II lung cancer study. Also, the suggestions and critiques from L. Todd Weiss and Dr. Emily Van Meter were greatly appreciated and strengthened the impact of the paper.

CONTACT INFORMATION

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

Name: Stacey Slone
Enterprise: University of Kentucky Markey Cancer Center
Address: 800 Rose Street, CC440
City, State ZIP: Lexington, KY 40536
Work Phone: 859-323-1723
Fax: 859-257-7715
E-mail: Stacey.Slone@uky.edu