

Generating Matched Case Data Using PROC SQL

Imelda C. Go, Lexington County School District One, Lexington, SC

ABSTRACT

Some statistical methods compare data matched on certain variables. For example, pairs of subjects with the same gender are to be compared. Whenever data from males/females are to be matched with data from other males/females, a many-to-many match is involved. SAS users are discouraged from using the DATA step to perform many-to-many matches. PROC SQL can adequately handle such types of matches. The paper is an introduction on how to use PROC SQL for the said purpose whether data are to be matched on one or more variables/criteria.

DATA STEP

Consider two data sets, group1 and group2. In the examples, SS stands for scaled score.

group1 data set

GRADE	NAME	SS	LUNCH	SEX
4	Garbo, Greta	434	R	F
3	Davis, Betty	380	F	F
3	Monroe, Marilyn	324	F	F
5	Gabor, Eva	567	R	F
2	Taylor, Liz	245	N	F
6	Farrow, Mia	655	N	F

group2 data set

GRADE	NAME	SS	LUNCH
3	Midler, Bette	354	R
3	Gabor, Zsazsa	381	R
6	Seymour, Jane	656	F
4	Field, Sally	434	F
5	Loren, Sophia	577	F
2	Ryan, Meg	234	N

A DATA step can be used to merge group1 and group2 by grade. When data are merged this way, the two data sets need to be sorted by grade (or the BY-variables). When contributing data sets have variables with the same name, the variables need to be renamed in order to prevent the values of one data set from overwriting the values of the other data set during the merge.

The DATA step can handle one-to-one, one-to-many, and many-to-one matches but not many-to-many matches. For true many-to-many matches, the result should be a cross product. For example, if two records from one data set match two records from another data set, the merged results should have $2 \times 2 = 4$ records.

The results shown next are problematic because of the many-to-many situation and because the variables with the same name were not renamed or dropped from either data set.

```
proc sort data=group1; by grade;
proc sort data=group2; by grade;
```

```
data test;
  merge group1 group2;
  by grade;
```

```
proc print;
```

GRADE	NAME	SS	LUNCH	SEX
2	Ryan, Meg	234	N	F
3	Midler, Bette	354	R	F
3	Gabor, Zsazsa	381	R	F
4	Field, Sally	434	F	F
5	Loren, Sophia	577	F	F
6	Seymour, Jane	656	F	F

PROC SQL

Many-to-Many Matches

PROC SQL can correctly handle many-to-many matches. Use it to create a data set that involves all possible matches. PROC SQL can also perform, under one procedure, what could be done using a combination of DATA step statements, and the PRINT, SORT, and SUMMARY procedures. The procedure is powerful, but only a few of its features are mentioned in this paper.

- The SELECT statement allows users to specify columns for the query, create column aliases (name/rename variables), and compute arithmetic expressions.
- The FROM statement specifies the sources of the variables listed in the SELECT statement.
- The WHERE statement specifies subsetting criteria.
- The ORDER BY specifies the sort order.
- Tables or data sets do not need to be sorted to use PROC SQL. The RUN statement does not need to be used with PROC SQL.

The example below matches data sets group1 and group2 by grade using PROC SQL. Cross products resulted for many-to-many match situations.

```
proc sql;
  select group1.grade, group1.name as name1,
         group2.name as name2, group1.ss as ss1,
         group2.ss as ss2, sex
  from group1, group2
  where group1.grade=group2.grade;
```

GRADE	NAME1	NAME2	SS1	SS2	SEX
4	Garbo, Greta	Field, Sally	434	434	F
3	Davis, Betty	Midler, Bette	380	354	F
3	Davis, Betty	Gabor, Zsazsa	380	381	F
3	Monroe, Marilyn	Midler, Bette	324	354	F
3	Monroe, Marilyn	Gabor, Zsazsa	324	381	F
5	Gabor, Eva	Loren, Sophia	567	577	F
2	Taylor, Liz	Ryan, Meg	245	234	F
6	Farrow, Mia	Seymour, Jane	655	656	F

POSTERS

Names repeat in the name1 and name2 fields. In producing the final matched case sample, care must be given to make sure that there are no unwanted repetitions in the matches. This can be achieved with more SAS programming statements but will not be shown in this paper.

Syntax Review

The PROC SQL example above is repeated below. Some syntax notes are provided.

```
proc sql;
  select group1.grade, group1.name as name1,
         group2.name as name2, group1.ss as ss1,
         group2.ss as ss2, sex
  from group1, group2
  where group1.grade=group2.grade;
```

- Variables in the SELECT statement are separated by commas and can be specified using one-level or two-level names. Two-level names are of the form: `<data set name>.<variable name>` and can be used any time. One-level names are acceptable when there are no ambiguous references. `Group1.grade` refers to the grade variable from the group1 data set. `Sex` uses a one-level name because it is in group1 but not in group2.
- Variables can be renamed in the SELECT statement. `Group1.name` is renamed as `name1`.
- The FROM statement lists source data sets separated by commas. Data sets `group1` and `group2` contain the variables listed in the SELECT statement.
- The WHERE statement can contain subsetting criteria. The condition `group1.grade=group2.grade` restricts the results to data *joined* or matched with equal grade values. The results will not contain records that differ in grade values.
- Grade occurs as a variable in both data sets. The following PROC SQL example produces the error: "Ambiguous reference, column GRADE is in more than one table."

```
proc sql;
  select grade, group1.name, group2.name
  from group1, group2
  where group1.grade=group2.grade;
```

- To include all variables from a table, use SELECT * instead of a variable list after the SELECT key word.

```
proc sql;
  select *
  from group1
  where grade=3;
```

GRADE	NAME	SS	LUNCH	SEX
3	Davis, Betty	380	F	F
3	Monroe, Marilyn	324	F	F

More Matching Criteria #1

Building on the first PROC SQL example, suppose that the matches are to be restricted to those where the scaled scores differ by no more than two scaled score points. An additional criteria is added to the WHERE statement to produce the desired result. The `0<=abs(group1.ss-group2.ss)<=2` condition checks to see if the absolute difference of the scaled scores is not a missing value and is at most two. (For the contrived data, the results below are very convenient because there are no repetitions of students. There are only distinct names from group1 and group2.)

```
proc sql;
  select group1.grade, group1.name as name1,
         group2.name as name2, group1.ss as ss1,
         group2.ss as ss2, sex
  from group1, group2
  where group1.grade=group2.grade and
         0<=abs(group1.ss-group2.ss)<=2;
```

GRADE	NAME1	NAME2	SS1	SS2	SEX
4	Garbo, Greta	Field, Sally	434	434	F
3	Davis, Betty	Gabor, Zsazsa	380	381	F
6	Farrow, Mia	Seymour, Jane	655	656	F

Creating a Table or an Output Data Set

The next example uses the CREATE TABLE statement to store the results of the query into a *table*. An ORDER BY statement was also added to produce the results in increasing `ss2` values. The log shows the following message after the SQL statement, "NOTE: Table WORK.POOL created, with 3 rows and 6 columns."

```
proc sql;
  create table pool as
  select group1.grade, group1.name as name1,
         group2.name as name2, group1.ss as ss1,
         group2.ss as ss2, sex
  from group1, group2
  where group1.grade=group2.grade and
         abs(group1.ss-group2.ss)<=2
  order by ss2;
```

```
proc print data=pool;
```

OBS	GRADE	NAME1	NAME2	SS1	SS2	SEX
1	3	Davis, Betty	Gabor, Zsazsa	380	381	F
2	4	Garbo, Greta	Field, Sally	434	434	F
3	6	Farrow, Mia	Seymour, Jane	655	656	F

The above example also shows that when PROC PRINT is used for the `pool` table, the output shows the query results.

More Matching Criteria #2

Suppose the lunch status needs to be part of the matching criteria. The condition can be added to the WHERE statement. The lunch variables were used in the WHERE statement even if they were not among the variables in the SELECT statement below. The lunch variable also does not appear in the results.

```
proc sql;
  select group1.grade, group1.name as name1,
         group2.name as name2, group1.ss as ss1,
         group2.ss as ss2, sex
  from group1, group2
  where group1.grade=group2.grade and
         group1.lunch=group2.lunch;
```

GRADE	NAME1	NAME2	SS1	SS2	SEX
2	Taylor, Liz	Ryan, Meg	245	234	F

To include the lunch variable in the output, add the lunch variable to the variables listed in the SELECT statement.

```
proc sql;
  select group1.grade, group1.name as name1,
         group2.name as name2, group1.ss as ss1,
         group2.ss as ss2, sex, group1.lunch
  from group1, group2
  where group1.grade=group2.grade and
         group1.lunch=group2.lunch;
```

GRADE	NAME1	NAME2	SS1	SS2	SEX	LUNCH
2	Taylor, Liz	Ryan, Meg	245	234	F	N

Arithmetic Expression

Suppose the difference of the ss variables needs to be added to the query. Add the necessary code to the SELECT statement. The difference of group1.ss-group2.ss is included in the SELECT statement below. The difference is given in the last column without a column heading.

```
proc sql;
  select group1.grade, group1.name as name1,
         group2.name as name2, group1.ss as ss1,
         group2.ss as ss2, sex,
         group1.ss-group2.ss
  from group1, group2
  where group1.grade=group2.grade and
         group1.lunch=group2.lunch;
```

GRADE	NAME1	NAME2	SS1	SS2	SEX	
2	Taylor, Liz	Ryan, Meg	245	234	F	11

Column Alias

A name can be given to the difference. For example, adding as ssdiff immediately after group1.ss-group2.ss in the previous SQL statement will name the difference as ssdiff.

```
proc sql;
  select group1.grade, group1.name as name1,
         group2.name as name2, group1.ss as ss1,
         group2.ss as ss2, sex,
         group1.ss-group2.ss as ssdiff
  from group1, group2
  where group1.grade=group2.grade and
         group1.lunch=group2.lunch;
```

GRADE	NAME1	NAME2	SS1	SS2	SEX	SSDIFF
2	Taylor, Liz	Ryan, Meg	245	234	F	11

A Final Note

It is also possible to have several statements under the same procedure declaration.

```
proc sql;
  create ...;
proc sql;
  select ...;
proc sql;
  select ...;
```

can be replaced by

```
proc sql;
  create ...;
  select ...;
  select ...;
```

REFERENCES

SAS Institute Inc., *SAS® Language Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999. 1256 pp.
 SAS Institute Inc., *SAS OnLineDoc®, Version 8*, Cary, NC: SAS Institute Inc., 1999.

TRADEMARK NOTICE

SAS is a registered trademark or trademark of the SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Imelda C. Go (icgo@juno.com)
 Lexington County School District One
 Lexington, SC