Paper 94

Efficiency Techniques: SQL vs. Retain Variables

Danielle Gao, Economic Policy Institute, Washington, D.C.

ABSTRACT

The SAS® DATA step allows you to perform calculations across observations by using retain variables. But as more variables are retained, coding can become cumbersome. By taking the advantage of the PROC SQL's ability to perform calculations on groups of observations, you can significantly reduce the length of your program and avoid an extra pass through the data.

SQL vs. RETAIN

Table 1 contains a portion of 1997 March Current Population Survey (CPS) survey data. This dataset has been sorted in ascending order by household ID, family ID within the household, and person ID within the family. Other than these identifiers, the extract includes family income, age and annual hours of work in the labor market. The constructed variable "kid" identifies anyone less than 18 year old.

Income analysts often are interested in "family work hours," i.e., the number of hours of work summed across the family. In this case, we are also interested in a separate analysis for families with children. Thus, the goal of the program is to sum work hours across the family, identify families with children, and place these values on the record of each family member.

Using Retain

In order to count the total number of children in the family and total hours worked by all family members, we use RETAIN variables to create two new variables: numkids and tothrs.

/* create two temporary "sum" variables: cc and dd in dataset one*/

```
retain cc dd ;
if first.fampos then
do;
  cc=0;
  dd=0;
end;

/* count total kids, and total hours*/

  cc=cc+hr;
  dd=dd+kid;

/* store total hours, total kids on
last record of family.  Details in
Table 3a */

if last.fampos then
  do;
   numkids=dd;
   tothrs=cc;
  end;

/* descending sort so that the last
family member with total kids and
total hours shows on the first record
*/

proc sort;
 by hhid fampos descending perid;
```

The necessity of this last sort is a function of our need to have the variables "numkids" and "tothrs" on the record of each family member. This is accomplished in the next data step.

```
data two(drop=ee ff);
  set one;
  by hhid fampos descending perid;

/* put numkids and tothrs to each
record, details in table 3b */

retain ee ff ;
if first.fampos then
  do;
    ee=numkids;
    ff=tothrs;
  end;

numkids=ee;
tothrs=ff;
```

Accomplishing the same task with PROC SQL

The SQL procedure supports most of the functions available in the DATA step for data creation and manipulation. For example, the summary function is available to summarize data for the entire table or for group of data in the table. You can select groups of data to be processed by using a HAVING clause. The ORDER BY clause is also available when your output should be sorted. The advantage of SQL is that the value returned from summary function is automatically assigned to each observation across the group of data, obviating the need for the extra sort and data step in the above example.

```
proc sql;
create table sq as
select hhid, fampos, perid, _faminc,
       age, _famknd, round(wgt) as
       wgt, sum(kid) as numkids,
sum(hr) as tothrs
from one
group by hhid,fampos
order by hhid,fampos,perid;

/* Note: sum(kid) as numkids, sum(hr)
as tothrs process the summation number
of kids and hours across the family*/
```

Table 3a shows the intermediate result from traditional programming. Table 2 and Table 3b show the same analytical dataset created from PROC SQL and traditional one. The same result is obtained from SQL with less code and, more importantly, in 1/5 the total time.

CONCLUSION

The method presented in this paper provides an efficient means for performing calculations across groups in a data set using PROC SQL as opposed to the more traditional data step method. By taking full advantage of the power of SQL, programmers can both simplify their code and save computer-running time.

REFERENCE

SAS Institute Inc. (1992), SQL Processing with the SAS® System: Course Notes, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1996), SAS® Guide to the SQL Procedure, Usage and Reference, Version 6, First edition, Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

I would like to thank Dr. Jared Bernsten of Economic Policy Institute for his encouragement, comments and edits.

CONTACT INFORMATION

Danielle Gao,
Economic Policy Institute
1600 L Street NW, Suite 1200,
Washington DC 20036
dgao@epinet.org

Appendix

Table 1 Portion of CPS March Survey Data

HHID	FAM POS	PER ID	_FAM INC	AGE	HR	KID
1	1	41	28254	72	0	0
1	1	42	28254	70	0	0
2	1	41	29601	39	2080	0
2	1	42	29601	67	0	0
3	1	41	116839	46	0	0
3	1	42	116839	44	0	0
3	1	43	116839	31	0	0
3	1	44	116839	25	2496	0
3	1	45	116839	22	360	0
3	1	46	116839	3	0	1
4	1	41	61000	40	2080	0
4	1	42	61000	40	2080	0
4	1	43	61000	15	0	1
4	1	44	61000	11	0	1

Table 2 Analytical Dataset from PROC SQL

HHID	FAM POS	PER ID	_FAM INC	AGE	NKIDS	TOT HRS
3	1	41	116839	46	1	2856
3	1	42	116839	44	1	2856
3	1	43	116839	31	1	2856
3	1	44	116839	25	1	2856
3	1	45	116839	22	1	2856
3	1	46	116839	3	1	2856
4	1	41	61000	40	2	4160
4	1	42	61000	40	2	4160
4	1	43	61000	15	2	4160
4	1	44	61000	11	2	4160
6	1	41	37419	32	2	4576
6	1	42	37419	30	2	4576
6	1	43	37419	11	2	4576
6	1	44	37419	5	2	4576
15	1	41	72030	50	1	3120
15	1	42	72030	34	1	3120
15	1	43	72030	7	1	3120

Table 3a. Intermediate Dataset from Traditional Programming

HHID	FAM POS	PER ID	AGE	HR	KID	NUM KIDS	TOT HRS
1	1	41	72	0	0		.
1	1	42	70	0	0	0	0
2	1	41	39	2080	0		.
2	1	42	67	0	0	0	2080
3	1	41	46	0	0		.
3	1	42	44	0	0		.
3	1	43	31	0	0		.
3	1	44	25	2496	0		.
3	1	45	22	360	0		.
3	1	46	3	0	1	1	2856
4	1	41	40	2080	0		.
4	1	42	40	2080	0		.
4	1	43	15	0	1		.
4	1	44	11	0	1	2	4160
6	1	41	32	2080	0		.
6	1	42	30	2496	0		.
6	1	43	11	0	1		.
6	1	44	5	0	1	2	4576

Table 3b. Final Analytical Dataset from Traditional Programming

HHID	FAM POS	PER ID	AGE	HR	KID	NUM KIDS	TOT HRS
3	1	41	46	0	0	1	2856
3	1	42	44	0	0	1	2856
3	1	43	31	0	0	1	2856
3	1	44	25	2496	0	1	2856
3	1	45	22	360	0	1	2856
3	1	46	3	0	1	1	2856
4	1	41	40	2080	0	2	4160
4	1	42	40	2080	0	2	4160
4	1	43	15	0	1	2	4160
4	1	44	11	0	1	2	4160
6	1	41	32	2080	0	2	4576
6	1	42	30	2496	0	2	4576
6	1	43	11	0	1	2	4576
6	1	44	5	0	1	2	4576
15	1	41	50	3120	0	1	3120
15	1	42	34	0	0	1	3120
15	1	43	7	0	1	1	3120