

FIRST. AND LAST. TO THE RESCUE

Sanjiv Ramalingam, Gary Cope, Octagon Research Solutions, Inc., Wayne PA

ABSTRACT

Numerous situations may arise where data manipulations are needed and use of standard procedures may not be possible. Here we present four different clinical programming scenarios that demanded a more subtle approach to data manipulation using FIRST. and LAST. which would have otherwise not been possible using standard procedures like PROC SORT or merges.

CASE 1

Consider a dataset that has multiple entries for a single subject.

Subject	Typ1	Typ2	Typ3
A101	1	0	0
A101	0	1	1
A101	0	1	0
A102	1	0	0
A102	0	1	0

Typ1, Typ2 and Typ3 are flags, which represent the existence of a condition. The flags have a possible combination of either 0's or 1's.

If we want to then have only a single entry for each subject, which includes all possible entries. The typical output expected is:

Subject	Typ1	Typ2	Typ3
A101	1	1	1
A102	1	1	0

The above situation was encountered when a concomitant medication table was being programmed. As patients will have multiple entries for their medication start and end dates it was decided to create flags to denote the interval when they would have taken the drug. The flags Typ1, Typ2 and Typ3 denote such intervals.

Solution

```
Data ds1;  
Input subject $ typ1 typ2 typ3;  
Cards;  
A101      1      0      0  
A101      0      1      1  
A101      0      1      0  
A102      1      0      0  
A102      0      1      0  
;  
run;
```

```
/* Since the first. and last. is being used it must be ensured that the data is sorted first */  
proc sort data=ds1;  
by subject typ1 typ2 typ3;  
run;
```

/* The idea is to create separate flags typx1, typx2 and typx3 that basically do a logical OR operation on each types(typ1, typ2 and typ3). The retain statement has been used to retain the result of each OR operation (achieved by using the MAX function). */

```
data ds2;  
set ds1;  
by subject;  
retain typx1 typx2 typx3;  
if first.subject then do;  
typx1=0;  
typx2=0;  
typx3=0;  
end;  
typx1=max(typx1,typ1);  
typx2=max(typx2,typ2);  
typx3=max(typx3,typ3);  
if last.subject then output;  
drop typ1 typ2 typ3;  
run;
```

CASE 2

Consider the following two datasets ds1 and ds2 :

Dataset ds1:

Subject	Typ1	Typ2	Typ3
A101	1	0	1
A102	0	1	1
A103	1	1	1

Dataset ds2:

Subject	Typ1a	Typ2a	Typ3a
A101	0	1	1
A102	0	1	0
A103	1	0	1

The typical output expected is :

Dataset ds3:

Subject	Typx1	Typx2	Typx3
A101	0	0	1
A102	0	1	0
A103	1	0	1

A situation such as this was encountered when a concomitant medication table was being programmed. The flags Typ1, Typ2 and Typ3 in dataset ds1 represent flags for that subject if they had study drug exposure in a particular interval. The flags Typ1a, Typ2a and Typ3a in dataset ds2 represent flags for that subject if they had a concomitant medication in that interval. The total counts for each of Typ1, Typ2 and Typ3 would then represent the population (N*) who took the study drug in a particular interval. As the count (n) should represent patients who not only had concomitant medication during a particular interval (Typ1a, Typ2a and Typ3a) but also these patients should have been exposed to the study drug during that interval (Typ1, Typ2 and Typ3), it would require a logical AND operation between the two types.

Solution

Data ds1;

Input subject \$ typ1 typ2 typ3;

Cards;

A101	1	0	1
A102	0	1	1
A103	1	1	1

;

run;

```

Data ds2;
Input subject $ typ1a typ2a typ3a;
Cards;
A101          0          1          1
A102          0          1          0
A103          1          0          1
;
run;

```

```

proc sort data=ds1;
by subject typ1 typ2 typ3;
run;

```

```

proc sort data=ds2;
by subject typ1a typ2a typ3a;
run;

```

```

data dsf;
merge ds1 ds2;
by subject;
run;

```

```

data ds3;
set dsf;
by subject;
retain typx1 typx2 typx3;
if first.subject then do;
typx1=0;
typx2=0;
typx3=0;
end;
typx1=typ1*typ1a;
typx2=typ2*typ2a;
typx3=typ3*typ3a;
if last.subject then output;
keep subject typx1 typx2 typx3;
run;

```

CASE 3

Common adverse event tables frequently require that the preferred term names for a treatment code be displayed only if the percentage of count exceeds a minimum X% for at least one of the treatment codes.

Eg:

Pt_name	Trtcd	Count	Denominator	Percentage
Diabetes	101	3	245	1.2
Diabetes	102	15	250	6.0
Vomiting	101	45	245	18.4
Vomiting	102	12	250	4.8
Headache	101	4	245	1.6
Headache	102	6	250	2.4

If the minimum percentage was 4% then Diabetes and Vomiting would qualify as the percentage of either of preferred terms is greater than 4% for at least one of the treatment codes and hence values for both the treatment codes would have to be displayed but Headache would not qualify.

Solution

```
Data ds1;
```

```
Input pt_name $ trtcd $ count denominator percentage;
```

```
Cards;
```

```
Diabetes      101      3      245      1.2
Diabetes      102     15     250     6.0
Vomiting      101     45     245    18.4
Vomiting      102     12     250     4.8
Headache      101      4     245     1.6
Headache      102      6     250     2.4
```

```
;
```

```
run;
```

```
/* Create a flag, flag1 that is high (1) if the percentage value is greater than the minimum
percentage */
```

```
data ds1;
```

```
set ds1;
```

```
if percentage ge 4 then flag1=1;
```

```
run;
```

```
/* sort pt_name and flag1 by descending order so that a new flag, flag2 can be created
that takes a value of 1 if the flag1 is high for at least one of the treatment code. */
```

```
proc sort data=ds1;
```

```
by pt_name descending flag1;
```

```
run;
```

```
data ds2;
set ds1;
by pt_name;
retain flag2;
if first.pt_name then do;
flag2=0;
end;
flag2=max(flag2,flag1);
output;
run;
```

/* select records for which flag2=1 which ensures that records are selected if the percentage of preferred terms is greater than 4% for at least one of the treatment codes */

```
data ds2;
set ds2;
if flag2=1;
run;
```

CASE 4

As a clinical programmer one of the most nagging problems can be to create a listing program that can handle missing data. Particular care must be given to PROC TRANSPOSE and a common error that can be encountered is :

ID “XXX” occurs twice in the same BY group.

This kind of problem is especially common when handling lab data. One of solutions is to insert record numbers for each record and also have the record number as one of the BY variables in the transpose. An another solution is to insert record numbers for different records for each subject.

Solution

```
Data lab;
set lab;
if first.subject then
recno=1;
else recno+1;
run;
```

CONCLUSION

These are some of the very few of the many examples where the FIRST. and LAST. option has come in handy. As there are 'n' number of ways to arrive at a particular solution we have tried to present a few options where FIRST. and LAST. can be used in a robust manner.

ACKNOWLEDGEMENTS

The authors would like to thank their fellow programmers Robert Schechter and Tod Mergner for reviewing the manuscript and their suggestions.