Getting What You Want Out of a Many-to-Many Merge

Sonal Pathak, University of South Florida, Tampa, FL
Shabnam Mehra, University of South Florida, Tampa, FL

ABSTRACT

Many-to-many data merges in any system can be very tricky and challenging, especially when dates are involved. In SAS, using the *RETAIN* statement, selecting the intended record using *FIRST*. (or *LAST*.), and *INITIALIZING* new variables can help alleviate some specific problems that often arise.

When merging two datasets with multiple event dates, or records, per person in both datasets, it is difficult to ensure the subsequent many-to-many merge provides the information the coder is looking for. This can be resolved in a 3-step process. To begin with, use the *RETAIN* statement to identify and keep the new variables that will be used in the DATA step. Then, the *FIRST*. (or *LAST*.) variable can be used to select the desired record. Next, *INITIALIZE* the new variables that were identified in the *RETAIN* statement. These 3 steps will make working the many-to-many merge a little easier.

Using the DATA step in SAS, with variables that are readily available, a messy many-to-many merge can be simplified to a unique record that includes all the necessary information.

INTRODUCTION

As an analyst working at the Policy and Services Research Data Center (PSRDC) in the Department of Mental Health Law and Policy at the Louis de la Parte Florida Mental Health Institute, it is routine to work with large administrative datasets. Conducting secondary data analysis is a daily challenge. It is important for the analyst to understand not only the data, but also the goals of the study to ensure that the final datasets can answer the study questions.

An issue that arises often at the data center is the need to merge two or more administrative datasets with multiple records per person, with the final goal being to output one record per event per person. *RETAIN*, *INITIALIZE*, *FIRST*. and *LAST*. together can be used to simplify the dataset to one record per event per person.

USING SQL TO MERGE

In this challenging analysis, there were multiple records per person, with multiple dates per person. Each person had an Episode date – the date of a specific event (Table 1). Each person also had different Medicaid eligibility time periods, or spans (Table 2).

Using SQL code to merge data is a resourceful way of creating a master dataset. The issue arises when each dataset contains multiple records per person.

```
Proc sql;
Create table ONE as
Select *
From EligFile, EpisodeFile
Where EligFile.ID = EpisodeFile.ID;
;
quit;
```

The code above created a many to many merge (Table 3), so for persons who had multiple Medicaid eligibility files, and only one Episode date, it looked as if they had more than one Episode. To rectify this situation, new datasets were created, using ID and Episode Date combination as the unique identifier.

USING *RETAIN*

RETAIN is a very useful SAS option that allows the value of the specified variable to be *retained* from one data step to the next[1]. This is especially useful when working with dates that need to be ordered chronologically.

[1] The Little SAS Book – a primer. Second Edition

The issue was to tag *when* each episode took place.

There were four options:
1) Before the person was Medicaid eligible,
2) After the Medicaid eligibility span,
3) During the Medicaid Eligibility span, or
4) Between spans of Eligibility.

The following code starts the process of creating new variables and also shows how *RETAIN* keeps the variables for the next sequence of the data-step. The format statement is also used to ensure the SAS dates have user-defined formats.

```
Data two;
        RETAIN  firststart firstend
                betbeg betend When;
Set one;
By ID EpisodeDate;

Format firststart mmddyy10. firstend mmddyy10. betbeg
mmddyy10.;betend mmddyy10.;
```

USING FIRST. TO INITIALIZE NEW VARIABLES

The next part of the code *INITIALIZES* the dates to missing by using the SAS variable *FIRST*. *FIRST*. is only available if the data is set *by* the needed variables. In this case, the data was set by ID and EpisodeDate, making EpisodeDate available to use with *FIRST*.

```
If first.EpisodeDate then do;
        Firststart = .;
        Firstend = .;
        Betbeg = .;
        Betend = .;
        When = ' ';
End;
```

In the above code, *FIRST*.EpisodeDate will select the first unique Episode Date per ID, and for each unique Episode Date, ID combination, it will initialize the four *retained* date variables to missing, as well as the "When" variable (Table 4).

USING NEW VARIABLES TO TAG THE RECORDS

Now that the new variables have been created, they are ready to be used in the rest of the code. Continuing with the code, this part tags the initiation as occurring during the Medicaid eligibility period:

```
        If EpisodeDate  GE startspan_dt and
        EpisodeDate LE endspan_dt then do;
        Betbeg = startspan_dt;
        Betend = endspan_dt;
        When = 'WhenElig';
        End;
```

Startspan_dt is the beginning date of Medicaid eligibility, and Endspan_dt is the end date of Medicaid eligibility.

The next step uses the *initialized* variables to continue tagging when the initiation occurred:

```
If EpisodeDate gt endspan_dt then do;

        If firstend eq . then do;
        Firstend = endspan_dt;
        When= 'After Elig';
        End;

Else if endspan_dt gt firstend_dt then do;
        Firstend = endspan_dt;
        When = 'After Elig';
        End;
End;
```

Similarly, the rest of the Episode dates are tagged as to when they occurred in correspondence with the Medicaid Eligibility spans.

USING LAST. TO OUTPUT

The final piece of the code outputs one record per event per person:

```
If last.EpisodeDate then;
        Do;
Output;
        End;
Run;
```

This specifies the *last* record of each unique Episode date to be output (Table 5). Why do we want the last record? The last record is the only one that has all the *retaine*d information for each unique combination of ID and Episode Date. It is the only one with the complete information for each combination.

EXAMPLE DATA:

TABLE 1: Event Data, sorted by ID and EpisodeDate

ID	EpisodeDate
001	5/13/02
001	7/29/03
002	1/3/00
003	11/9/01
003	8/15/03

+

TABLE 2: Medicaid Data, sorted by ID, Eligibility span Start and End Dates

ID	Startspan_Dt	Endspan_dt
001	5/1/01	7/15/03
002	11/1/99	2/28/00
003	1/15/02	3/15/02
003	7/1/03	8/1/03
003	11/1/03	12/10/03

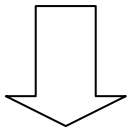

TABLE 3: Result of Merged Data (using SQL)

ID	EpisodeDate	Startspan_dt	Endspan_dt
001	5/13/02	5/1/01	7/15/03
001	7/29/03	5/1/01	7/15/03
002	1/3/00	11/1/99	2/28/00
003	11/9/01	1/15/02	3/15/02
003	11/9/01	7/1/03	8/1/03
003	11/9/01	11/1/03	12/10/03
003	8/15/03	1/15/02	3/15/02
003	8/15/03	7/1/03	8/1/03
003	8/15/03	11/1/03	12/10/03

TABLE 4: Result of *Retain* and *Initialize*

ID	EpisodeDate	Startspan_d	Endspan_dt	FirstStart	FirstEnd	BetBeg	BetEnd	When
001	5/13/02	5/1/01	7/15/03	.	.	5/1/01	7/15/03	When Elig
001	7/29/03	5/1/01	7/15/03	.	7/15/03	.	.	After Elig
002	1/3/00	11/1/99	2/28/00	.	.	11/1/99	2/28/00	When Elig
003	11/9/01	1/15/02	3/15/02	1/15/02	.	.	.	Before Elig
003	11/9/01	7/1/03	8/1/03	1/15/02	.	.	.	Before Elig
003	11/9/01	11/1/03	12/10/03	1/15/02	.	.	.	Before Elig
003	8/15/03	1/15/02	3/15/02	.	3/15/02	.	.	After Elig
003	8/15/03	7/1/03	8/1/03	.	8/1/03	.	.	After Elig
003	8/15/03	11/1/03	12/10/03	11/1/03	8/1/03	.	.	Between Elig

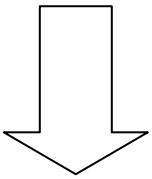

TABLE 5: Output of *LAST*

ID	Episode Date	Startspan_dt	Endspan_dt	FirstStart	FirstEnd	BetBeg	BetEnd	When
001	5/13/02	5/1/01	7/15/03	.	.	5/1/01	7/15/03	When Elig
001	7/29/03	5/1/01	7/15/03	.	7/15/03	.	.	After Elig
002	1/3/00	11/1/99	2/28/00	.	.	11/1/99	2/28/00	When Elig
003	11/9/01	11/1/03	12/10/03	1/15/02	.	.	.	Before Elig
003	8/15/03	11/1/03	12/10/03	11/1/03	8/1/03	.	.	Between Elig

The final record per person contains the last Startspan and Endspan date associated with that ID, not necessarily the Startspan and Endspan date associated with that Episode Date. This is because *RETAIN* allowed us to keep the needed information in the new, *retained* variables, without changing the Startspan and Endspan dates each time.

CODE

```
proc sql;
create table one as
select *
from EligFile, EpisodeFile
where EligFile ID = EpisodeFile.ID ;
;
QUIT;


data two;
        retain firststart firstend betbeg betend When;
set one ;

by ID EpisodeDate;

format firststart mmddyy10. firstend mmddyy10. betbeg
mmddyy10. betend mmddyy10.;

if first.EpisodeDate then
 do;
        firststart=.;
        firstend=.;
        betbeg=.;
        betend=.;
        When= '                    ';
 end;

 * TAG DURING SPAN HIT *;
if EpisodeDate ge startspan_dt and EpisodeDate le
endspan_dt then
 do;
        betbeg=startspan_dt;
        betend=endspan_dt;
        When= 'When Elig    ';

 end;


 * TAG POST ELIGSPAN ENDDATE *;

if EpisodeDate gt endspan_dt then do;

        if firstend eq . then do;
                firstend = endspan_dt;
                When= 'After Elig  ';  end;

        else if endspan_dt gt firstend then do;
        firstend = endspan_dt;
                When= 'After Elig  ';  end;

                end;

* TAG PRIOR ELIGSPAN BEGIN DATE *;
if betbeg eq . then do;

        if EpisodeDate lt startspan_dt then do;

        if firststart eq . then do;
                firststart=startspan_dt;
                When= 'Before Elig  ';  end;

        else if startspan_dt lt firststart then do;
        firststart=startspan_dt;
           When= 'Before Elig  ';  end;

                end;

* TAG BETWEEN SPAN HIT *;

if betbeg eq . and firstend ne . then do;

        if EpisodeDate gt firstend and EpisodeDate lt
firststart then do;
                When= 'Between Elig  ';
                end;

end;
end;


if last.EpisodeDate then
 do;
        output;
 end;

RUN;
```

CONCLUSION

Using variables and options readily available in SAS, creating a unique record per event per person from a many-to-many merge can be accomplished.

TRADEMARKS

References

Delwiche L.D. and Slaughter S.J. (2000) *"The Little SAS Book," Second Edition*, Cary, NC: SAS Institute Inc.

Mehra, S, et al. (2002) "*Span Span Away! Creating One Unique Record for Overlapping Admissions and Discharges from Multiple Inpatient Hospital Stays.*" Proceedings of Twenty Seventh Annual SAS Users Group International Conference, 214-27.

Acknowledgments

A special thanks to the PSRDC team, in particular, Keith Vossberg and Rebecca Larsen.

Contact Information:

Policy and Services Research Data Center (PSRDC)
Mental Health Law and Policy
Louis de la Parte Florida Mental Health Institute
University of South Florida
13301 Bruce B Downs Blvd
Tampa FL 33612

Sonal Pathak
Phone: 813-974-0449
e-mail: spathak@fmhi.usf.edu

Shabnam Mehra
Phone: 813-974-9315
e-mail: smehra@fmhi.usf.edu