Paper 227-26

Hip Hip ARRAY…Expanding and Modifying Records With an ARRAY

Jeanina Worden, PRA International, Lenexa, KS
Lauren Shinaberry, PRA International, Lenexa, KS

ABSTRACT

This paper demonstrates how observations can be added to or modified within a dataset by using a combination of an ARRAY and IF…THEN…DO statements if a single observation meets a specified criterion. The criterion discussed here requires a comparison be made to verify whether a start period and an end period of an event are the same, and the action taken using BASE SAS® v.6.12 if they are not.

INTRODUCTION (OR THE QUICK EXPLANATION)

Before getting started describing the challenge and solution revealed herein, it might help to understand the programming discussed here is taking a database created in a Clinical Data Management system that creates Oracle table then converting a table to a SAS® dataset using BASE SAS v.6.12. The data captured within this dataset represents an event that spans over multiple periods. For analysis purposes these single observations need to be expanded and modified to represent one observation per event for the final output. To do this, new variables are added using INPUT of the original variables, and an INVALUE; a formula can be applied to assign a result that will be used for the comparison. RENAME will maintain the value of the original values for use in the final record, and SELECT can assign values to additional new variables. To avoid errors, and NULL values, PUT is used to note a default assignment within the log. Then the ARRAY allows you to group several possibilities for one variable to account for multiple situations. A DO loop can then be used to determine the number of times the record should be processed. And embedded IF…THEN…ELSE statements can make the variable changes for each new record depending on the ARRAY variable.

THE SPECIFIC CHALLENGE

The original Oracle database contained variables for an event that included a “start period” (PERIOD1), a “stop period” (PERIOD2), and each period was designated by a character value. If these two characters were different, the information within the original observation needed to be expanded with the final dataset showing a record for each period individually. For example, if PERIOD1=P and PERIOD2=3 the record needs expanded to reflect one for the period ‘P’, one for period ‘1’, one for period ‘2’, and one for period ‘3’. In addition to adding the records, each new record must be modified to reflect the correct period it represents, the variable for “ongoing”, AE_STAT, must signify that the event is continuing, the variables that make up the “end date”, AEFINMON, AEFINDAY, AEFINYR, must be blank, and the “visit” should be the one for the given time period. Then the final record should show the original data for all the modified variables and show the accurate visit for the final period. This example would create four records from the original one, as shown in the following diagram:

Original Record

START=P
STOP=2
DATE=MMDDYY
STATUS= ENDED

First Record

START=P
STOP=P
VISIT= ARRAY
VARIABLE
DATE= BLANK
STATUS= CONTINUING

Second Record

START=1
STOP=1
VISIT= ARRAY
VARIABLE
DATE= BLANK
STATUS= CONTINUING

Third Record

START=2
STOP=2
VISIT= ARRAY
VARIABLE
DATE= BLANK
STATUS=
CONTINUING

Final Record

START=3
STOP=3
VISIT= ARRAY
VARIABLE
DATE= MMDDYY
STATUS= ENDED

THE CONQUER

First, PROC FORMAT is used to create an INVALUE format that would be used in the DATA step. The original variable format is a character; the INVALUE assigns a number to the variable that can then be passed through a formula for the comparison:

```
proc format;
  invalue prda 'P'=0
               '1'=1
               '2'=2
               '3'=3
               'F'=4;

  invalue prdb 'P'=1
               '1'=2
               '2'=3
               '3'=4
               'F'=5;
run;
```

Then the good stuff…the original data had to be maintained for the variables that would be changing within the loops, so they were each renamed on the SET line so it would perform the RENAME before execution. Using INPUT, PERIOD1, PERIOD2, and the formats defined in the PROC FORMAT step the variables NUMSTART and NUMSTOP were assigned. By doing this, NUMSTART and NUMSTOP are changed from characters to numbers and could be put into a formula to distinguish their difference. The result from this formula is assigned to NUMPER and is used in the DO statement to determine the number of times the record needs to be processed. To avoid errors, and to account for NULL values, NUMPER was assigned to 1 and a message put into the log so missing values could be identified and investigated.

```
data ad1;
  length aefinmon aefinday $2 aefinyr $4;
  set temp.adverse (rename=(ae_stat=oldstat
aefinmon=oldmon aefinday=oldday
aefinyr=oldyr));
  numstart=input(period1,prda.);
  numstop=input(period2,prdb.);

  numper=numstop-numstart;

  if numper=. then do;
  numper=1;
  put patno=' missing start or stop period
numper set to 1';
  end;
```

Finally to the ARRAY… the five different possibilities for the variable VISNUM are grouped within the ARRAY statement. SELECT (period1) is used to assign 'v' to correspond to the number assigned to the ARRAY variable. The record was to be processed by going through the IF…THEN…DO loops the same number of times as the value of NUMPER, and only if PERIOD1 and PERIOD2 weren't the same. The first DO statement sets 'i' equal to "1 to NUMPER" which is the number of times it should loop through. If this record was not the final, VISIT would be equal to the number that was assigned to VISNUM within the ARRAY, the status would be equal to "ongoing" which is 1, and the "end date" variables would be blank. Looping continues until the final record is reached. The final record still reflects the corresponding VISIT from the ARRAY assignment, but the status of other two variables, "ongoing" and the "end date", are the original data entered into the database. The final ELSE DO is for those records that PERIOD1 and PERIOD2 were originally equal, therefore did not go through the loops. The expanded record is output to the table at the end of each loop by using OUTPUT.

```
if numper^=1 then do;
   do i=1 to numper;

    if i=1 then do;
     visit= visnum{v};
     ae_stat=1;
     aefinmon='';
     aefinday='';
     aefinyr='';
    output;
    end;

    else if i^=numper then do;
     visit= visnum{v+(i-1)};
     ae_stat=1;
     aefinmon='';
     aefinday='';
     aefinyr='';
    output;
    end;

    else do;
     visit= visnum{v+(i-1)};
     ae_stat=oldstat;
     aefinmon=oldmon;
     aefinday=oldday;
     aefinyr=oldyr;
    output;
    end;
    end;
  end;

 else do;
 select(period1);
   when ('P') visit=-1;
   when ('1') visit=1;
   when ('2') visit=2;
   when ('3') visit=3;
   when ('4') visit=120;
   otherwise;
  end;
 ae_stat=oldstat;
 aefinmon=oldmon;
 aefinday=oldday;
 aefinyr=oldyr;
 output;
 end;
run;
```

THE BIG WRAP-UP

To modify records that have multiple possibilities for a variable, the use of an ARRAY is an easy way of grouping those possibilities. This variable can then be used within a series of embedded IF…THEN…DO statements to determine which value to assign, as well as assigning other changes. This paper describes using this idea to expand adverse event records that started in one visit and ended in another, to individual records for each visit.

It also explained how new variables were added using INPUT of the original variables, and an INVALUE was applied, then a formula could be used to assign a result that was used for a comparison. RENAME did maintain the value of the original values for use in the final record, and SELECT could assign values to new variables. As well as, demonstrating how a DO loop can be used to determine the number of times the record should be processed or put a message into the log. And embedded IF…THEN…ELSE statements can make the variable changes for each new record depending on the ARRAY variable.

REFERENCES

SAS Institute Inc. (1990), "SAS Language Reference, Reference Version 6, First Edition," Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1990), "SAS Procedures Guide, Version 6, Third Edition," Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

A big thanks to Darryl Hunter and Sherri Upchurch for putting up with my numerous requests for rereads and input.

CONTACT INFORMATION

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

Jeanina Worden
PRA International
16400 College Boulevard
Lenexa, KS 66111
Work Phone: (913) 577-2883
Fax: (913) 599-2753
Email: wordenjeanina@praintl.com

Lauren Shinaberry
PRA International
16400 College Boulevard
Lenexa, KS 66111
Work Phone: (913) 577-2762
Fax: (913) 599-2753
Email: shinaberrylauren@praintl.com
Web: