PharmaSUG 2014 - Paper CC44

A Macro to Create Occurrence Flags for Analysis Datasets

Ed Lombardi, Agility Clinical, Inc., Carlsbad, CA

ABSTRACT

The different paths available for occurrence analysis are fairly straightforward. The path that takes a little longer ends up giving a better end product. This approach is to create occurrence flags in analysis datasets as described in ADaM's ADAE structure. These flags allow for standardized table programs while also providing traceability to allow reviewers to know what records are being used in analysis. The code to create these flags is straightforward and repetitive so it's an excellent situation to use a macro. Otherwise, creating these flags can lead to cluttered sections of analysis dataset programs.

INTRODUCTION

The goal of occurrence analysis is to show when events or interventions occurred for subjects. This type of analysis is often performed on data with a hierarchical structure. It is best practice and industry standard to keep all "occurring" records from the raw data. When performing this incidence analysis, there is a need to reduce each subject to one record within each level of the hierarchy. When using occurrence flags to aid in this type of analysis, each level of the hierarchy requires a separate flag. This means for a typical adverse events analysis, three flag variables are required. One to allow for analysis showing the incidence with any adverse events. A second to show incidence within each body system. And, a third to show incidence with each preferred term within the body system. Examples of the use of these flags can be found in ADaM's ADAE supplement and will be described in a future ADaM occurrence based data model (not released as of the writing of this paper). Occurrence flags can add traceability from your analysis dataset to analysis outputs and to other analysis datasets. Creating these flags using a macro can provide accuracy, consistency and clarity.

MACRO DESCRIPTION

The goal of the macro is to produce one flag variable for every hierarchical level specified by the user. One, two or three levels are most typical in occurrence analysis so the version of the macro presented here can create up to three occurrence flag variables within one macro call. The flags are populated with a 'Y' based on the first occurrence of a user-specified variable within a user-specified sorting algorithm.

Because occurrence analysis can need to follow conditions (e.g., treatment emergent adverse events, serious adverse events), the macro has an option to only populate these flags if records meet a user-specified condition. There is no need to exclude records such as non-treatment emergent adverse events and then add them back on after the flags are created.

The macro can create each hierarchical level within a single macro call. This feature, as opposed to single calls for each hierarchy or single calls for each flag, helps to ensure consistency for a set of flags and simplifies the readability for the overall program. Another feature of the macro is that the flag variables are added onto the original dataset. With large datasets and a large number of flags being created, this approach lessens the number of working datasets which may help with run-time. Additionally, using the same working dataset name as the input and output dataset for the macro allows future calls of the macro to be added without needing to change code later in the program which uses the last data step. The drawback of this approach is the potential for overwritten variables if flag names are not unique throughout the program. However, if the user understands the macro and utilizes it properly, then the macro can be a great tool to aid in occurrence analysis.

MACRO EXECUTION

The full macro code is displayed at the end of this paper. The macro statement is as follows

```
%macro aoccfl(datain,cond=XX, sort1=XX,var1=XX,fl1=XX, sort2=XX,var2=XX,fl2=XX,
sort3=XX,var3=XX,fl3=XX);
```

There is one positional parameter.

datain - this specifies the name of the dataset that macro will use and add flag variables to.

There are 10 keyword parameters. Each keyword parameter defaults to XX and are only used when the user specifies their use. Three of these keyword parameters are used in conjunction and are repeated three times to allow the user to create three sets of flags in one macro call.

cond - this specifies a condition that is required to be met by the row of the first occurrence. An IF statement is used to remove the records not meeting this condition prior to deriving the flags. The records are only removed from intermediate datasets and are added back at the end of the macro to ensure all records are kept in the final working dataset.

sort1 - this specifies the sorting algorithm used in PROC SORT to determine the first occurrence of a specific variable where that row will have a flag record populated with 'Y'.

var1 - this specifies the variable where the macro will flag a first occurrence. This is done in a dataset using "first.&var1" based on the sort from the sort1 macro. The location of the variable in the sorting algorithm is key to ensure only one record is flagged.

fl1 - this specifies the name of the flag variable that will be created

As mentioned above, the sort, var, and fl macro variables are used in conjunction so the user may use one set, two sets or three sets.

EXAMPLES OF MACRO IN USE

If the user only wants to create one flag they can simply call the macro this way

```
%aoccfl(WORK.AE, sort1=USUBJID ASTDT AESEQ,var1=USUBJID,fl1=AOCCFL);
```

Note how the sorting algorithm includes the variables ASTDT and AESEQ. In this example, the addition of these variables ensures that the AOCCFL flag is only populated once for each USUBJID. The use of this sorting algorithm allows a reviewer or QCer to easily replicate the derivation of the flags. Additionally, this sorting algorithm can be documented to provide good traceability.

If the user is creating three flags macro call looks like this.

```
%aoccfl(WORK.AE, sort1=USUBJID ASTDT AESEQ,var1=USUBJID,fl1=AOCCFL,
sort2=USUBJID AEBODSYS ASTDT AESEQ,var2=AEBODSYS,fl2=AOCCSFL,
sort3=USUBJID AEBODSYS AEDECOD ASTDT AESEQ,var3=AEDECOD,fl3=AOCCPFL);
```

If the user is using a condition, the macro call looks like this.

```
%aoccfl(WORK.AE, cond=AESER = 'Y',
sort1=USUBJID ASTDT AESEQ,var1=USUBJID,fl1=AOCC01FL,
sort2=USUBJID AEBODSYS ASTDT AESEQ,var2=AEBODSYS,fl2=AOCC02FL,
sort3=USUBJID AEBODSYS AEDECOD ASTDT AESEQ,var3=AEDECOD,fl3=AOCC03FL);
```

For this condition, only records where AESER = 'Y' can be flagged.

If the user is calling the macro multiple times, it might look like this.

```
%aoccfl(WORK.AE, sort1=USUBJID ASTDT AESEQ,var1=USUBJID,fl1=AOCCFL,
sort2=USUBJID AEBODSYS ASTDT AESEQ,var2=AEBODSYS,fl2=AOCCSFL,
sort3=USUBJID AEBODSYS AEDECOD ASTDT AESEQ,var3=AEDECOD,fl3=AOCCPFL);

%aoccfl(WORK.AE, cond=AESER = 'Y',
sort1=USUBJID ASTDT AESEQ,var1=USUBJID,fl1=AOCC01FL,
sort2=USUBJID AEBODSYS ASTDT AESEQ,var2=AEBODSYS,fl2=AOCC02FL,
sort3=USUBJID AEBODSYS AEDECOD ASTDT AESEQ,var3=AEDECOD,fl3=AOCC03FL);
```

Note how in both macro calls, the same dataset, WORK.AE, is used. If the user wanted to do more than two sets of flags, they could keeping using the dataset WORK.AE.

FULL MACRO CODE

```
%macro aoccfl(datain,cond=XX, sort1=XX,var1=XX,fl1=XX, sort2=XX,var2=XX,fl2=XX,
sort3=XX,var3=XX,fl3=XX);

*create working dataset for sorting and remove records that do not meet the
condition*;
%if &cond ne XX %then %do;
  data &datain &datain.sort;
    set &datain;
```

```
    if &cond then output &datain.sort;
       else output &datain;
  run;
%end;
%else %do;
  data &datain.sort;
    set &datain;
  run;
%end;

*create the first flag - sort and set*;
proc sort data=&datain.sort;
  by &sort1;
run;

data &datain.sort;
  set &datain.sort;
  by &sort1;
  if first.&var1 then &fl1 = 'Y';
run;

*create the second flag - sort and set*;
%if &sort2 ne XX %then %do;
  proc sort data=&datain.sort;
    by &sort2;
  run;

  data &datain.sort;
    set &datain.sort;
    by &sort2;
    if first.&var2 then &fl2 = 'Y';
  run;
%end;

*create the third flag - sort and set*;
%if &sort3 ne XX %then %do;
  proc sort data=&datain.sort;
    by &sort3;
  run;

  data &datain.sort;
    set &datain.sort;
    by &sort3;
    if first.&var3 then &fl3 = 'Y';
  run;
%end;

*set back records onto the original working dataset name*;
%if &cond ne XX %then %do;
  data &datain;
    set &datain &datain.sort;
  run;
%end;
%else %do;
  data &datain;
    set &datain.sort;
  run;
%end;

%mend aoccfl;
```

CONCLUSION

The macro presented in this paper is very simple. It uses basic code and logic. The task it aids in performing is actually easily done without the macro. However, the use of a macro in this situation helps the user avoid long repetitive code. It can make the readability of the program more easily followed. The flags which can be produced by using the macro are an important part of good traceability for occurrence analysis. This macro was written to help streamline the process for this important part of analysis dataset creation.

REFERENCES

Analysis Data Model (ADaM). http://www.cdisc.org/adam

ACKNOWLEDGMENTS

The author would like to thank his friend and colleague James Hasson who is a good source of philosophical discussions on programming and local San Diego cuisine.

CONTACT INFORMATION

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

Ed Lombardi
elombardi@agility-clinical.com