PhUSE EU Connect 2018

Paper CT08

Your Date or Mine?

Melanie O'Neill, PPD, Bellshill, Scotland

ABSTRACT

Creating and validating a dataset can be a difficult and time-consuming task, particularly when information being combined into one variable is pulled from various datasets. During validation being able to easily identify where a mismatching value is sourced can significantly reduce time spent in investigations. This paper presents a macro that can identify the source variable of the value selected and can be used to apply different processing rules and criteria to groups of datasets using last known alive date as an example.

INTRODUCTION

Clinical trials collect data in many different formats which can make evaluating information across multiple datasets challenging. While it is possible to combine various dates from different source datasets using a proc sql statement, when the number of datasets increases, the code can become complex and difficult to manage making debugging while developing the code challenging. The macro detailed in this paper provides a method for not only selecting the maximum date from multiple datasets but also a way to identify the source dataset and variable name of the value selected.

Although this paper focuses on calculating the last known alive date as an example, the macro is equally applicable to different situations. For example, a time to event dataset might need to pull dates from various sources and select either the maximum or minimum date. This macro is also useful when programming with unclean data as it aids investigations into issues.

The 'Last Known Alive' date is a CDISC ADaM variable stored in ADSL which holds the last recorded date for a subject, prior to a (possible) death. It is derived by selecting the maximum date from various source datasets (which should be detailed in the dataset specification documentation) such as start/end dates of adverse events and concomitant medication and disposition events, however it is important to not include dates in the selection related to death information.

Care must always be taken when choosing variables to be considered for last known alive date, to be sure the intended meaning of the data is understood and maintained. Review of the date variables used is usually more time consuming than programming for the selection. For example, Investigator Signature Date on a Subject Summary page can be commonly misunderstood to be the date on which the investigator filled in the data and therefore if death is not listed could be thought that the subject is still alive at that point. However, the Investigator's Signature is usually required to confirm that the Investigator has reviewed all the data reported and deems it to be complete and accurate – a review which may be performed months after a subject has completed the study, and therefore the date should not be considered for the Last Known Alive calculation.

THE %ALIVE MACRO

If double programming is used as the method of validation, then the macro code would either have to be independently validated before being used on both production and validation, or if the macro is not independently validated then the macro should be used either for production or validation and would probably be most useful to the validation programmer. The structure of the datasets created by the macro would allow the validation programmer to see the maximum event date selected from each dataset of interest for every subject and the source of that date. This means when they are validating ADSL, they can provide more information to the production programmer to resolve mismatches. For example, they would be able to see the date they have selected as the maximum was the end date of an adverse event two years in the future, whereas the date the production programmer has selected may match the start date of an adverse event. This narrows the investigation for the validation programmer to the adverse event dataset and shortens the time the production programmer needs to spend reviewing their code to identify any issues.

The macro is designed so only one call per dataset is needed, regardless of the number of event date variables of interest. All names of event date variables are entered as a space delimited list within the call to the macro, which is then parsed and the macro cycles through the code for each event date specified. Due to the number of different source datasets and the potential for the subject identifier to be recorded under different variable names the macro also includes a parameter to specify the variable that should be used as the subject identifier.

The macro variables are set up with no immediate defaults as follows:

```
%macro alive(ds=, date=, sub=);
```

ds = name dataset of containing the date variables
date = name(s) of event date variables of interest. Multiple dates from the same dataset can be listed separated with a space
sub = subject variable used in the dataset e.g. pt or subjid

Within the macro, a reference dataset called death is used to identify the subjects with a death date so that last known alive date is derived for subjects that do not have a death date populated.

Before the macro is called the reference dataset is sorted by subject, and only the subject identifier and reference date are kept. At the end of the process, the reference dataset will be merged with the datasets created by the macro, so the subject identifier is renamed to a standard variable name; in this case, subject.

The macro begins by reading in the date parameter specified in the macro call and scanning through each variable in the list and executing the code for each variable. The full code is available in Appendix I.

```
%let i=1;
  %let dt=%scan(&date, &i, %str( ));
    %do %while (%superq(dt) ne %str( ));
```

The macro sorts the dataset of interest by subject and descending event date. At this point it is would be possible to adapt the code and apply any specific criterion required, though none is shown in the example. For example, if some datasets of interest require a selection clause, or if the event date of interest is in date/time format and only the date is required. The sorted dataset is then used to create the standard subject variable, as per the reference dataset, and a new variable called ‘origin’ is created by combining the dataset of interest and the event date. The ‘origin’ variable will be the identifier in the final dataset of where the date originated.

```
data &ds._1_&dt.a (keep=subject origin &dt.);
  length subject $100.;
   set &ds._1_&dt.;
     subject=&sub.;
       origin=catx(".", "&ds.", "&dt.");
run;
```

The reference dataset is then merged to the dataset created above and records are kept if they are in the dataset of interest and the event date is less than the reference date or the reference date is missing.

```
data &ds._2_&dt.;
    merge &ds._1_&dt.a(in=a) death(in=b);
      by subject;
        if a and (&dt. < dsdt or dsdt eq .);
run;
```

Once the datasets are merged, the macro checks if the merged dataset has any observations before transposing to display one record per subject and origin. Due to the descending sort used on event date of interest, once the dataset is transposed the first column displays the maximum event date for each subject. The final dataset created in the macro uses the transposed dataset and keeps the subject identifier, the origin and column 1, which is renamed to date.

```
data dataset_&ds._&dt. (keep=subject col1 origin rename=(col1=date));
   set &ds._3_&dt.;
run;
```

The macro will then rescan the date parameter and move onto the next event date of interest in the list.

```
%let i=%eval(&i+1);
 %let dt=%scan(&date, &i, %str( ));
  %end;
```

The macro must be called for each dataset and event date variables as outlined in the dataset specifications. An example of two calls is show below.

```
%alive (ds=addltr, date=lymedt lymsdt, sub=pt);
%alive (ds=assesmnt, date=asesdt , sub=pt);
```

This will produce one dataset per event date variable showing the subject identifier, the origin and the maximum event date variable. The dataset name will be the dataset of interest and event date. The datasets below show the output of running the two macro calls above.

VIEWTABLE: Work.Dataset_addltr_lymedt

	Subject identifier	ORIGIN	DATE
1	0121001	addltr.lymedt	18NOV2016
2	0191002	addltr.lymedt	30MAR2017
3	0211004	addltr.lymedt	30MAY2018
4	0221001	addltr.lymedt	29NOV2017
5	0251002	addltr.lymedt	11JUL2017
6	0251005	addltr.lymedt	04DEC2017
7	0251007	addltr.lymedt	09JUN2017
8	0261001	addltr.lymedt	12OCT2017
9	0281001	addltr.lymedt	03JUL2017
10	0311001	addltr.lymedt	22MAR2017

VIEWTABLE: Work.Dataset_addltr_lymsdt

	Subject identifier	ORIGIN	DATE
1	0121001	addltr.lymsdt	02JUN2016
2	0182001	addltr.lymsdt	16DEC2015
3	0191002	addltr.lymsdt	29MAR2017
4	0211004	addltr.lymsdt	07MAY2018
5	0221001	addltr.lymsdt	27NOV2017
6	0251002	addltr.lymsdt	28DEC2017
7	0251005	addltr.lymsdt	18AUG2017
8	0251007	addltr.lymsdt	16MAY2018
9	0261001	addltr.lymsdt	18NOV2017

VIEWTABLE: Work.Dataset_assesmnt_asesdt

	Subject identifier	ORIGIN	DATE
1	0081001	assesmnt.asesdt	03MAR2017
2	0121001	assesmnt.asesdt	24MAY2016
3	0121002	assesmnt.asesdt	08APR2016
4	0121003	assesmnt.asesdt	02JUN2018
5	0181001	assesmnt.asesdt	15APR2015
6	0182001	assesmnt.asesdt	12JUN2018
7	0191002	assesmnt.asesdt	25JAN2017
8	0191003	assesmnt.asesdt	05DEC2017
9	0211001	assesmnt.asesdt	24APR2018
10	0211002	assesmnt.asesdt	02DEC2016

SELECTING THE LAST KNOWN ALIVE DATE

Once all macro calls have completed, the resulting datasets (dataset_&ds._&dt.) are all set together and sorted by subject number and descending date.

```
data all;
 set dataset_:;
run;

proc sort data=all out=all_s;
 by subject descending date;
run;
```

VIEWTABLE: Work.All_s

	Subject identifier	ORIGIN	DATE
1	0081001	assesmnt.asesdt	03MAR2017
2	0121001	addltr.lymedt	18NOV2016
3	0121001	addltr.lymsdt	02JUN2016
4	0121001	assesmnt.asesdt	24MAY2016
5	0121002	assesmnt.asesdt	08APR2016
6	0121003	assesmnt.asesdt	02JUN2018
7	0181001	assesmnt.asesdt	15APR2015
8	0182001	assesmnt.asesdt	12JUN2018
9	0182001	addltr.lymsdt	16DEC2015
10	0191002	addltr.lymedt	30MAR2017
11	0191002	addltr.lymsdt	29MAR2017
12	0191002	assesmnt.asesdt	25JAN2017
13	0191003	assesmnt.asesdt	05DEC2017
14	0211001	assesmnt.asesdt	24APR2018
15	0211002	assesmnt.asesdt	02DEC2016
16	0211003	assesmnt.asesdt	01JUN2018
17	0211004	addltr.lymedt	30MAY2018
18	0211004	addltr.lymsdt	07MAY2018
19	0211004	assesmnt.asesdt	25SEP2017
20	0211005	assesmnt.asesdt	16APR2018

The dataset above now has the maximum event date from all dataset/variable combinations for each subject sorted by descending date. To find the maximum date and therefore the last known alive date the first record for each subject is selected.

```
data max_date;
 set all_s;
  by subject descending date;
   if first.subject;
run;
```

VIEWTABLE: Work.Max_date

	Subject identifier	ORIGIN	DATE
1	0081001	assesmnt.asesdt	03MAR2017
2	0121001	addltr.lymedt	18NOV2016
3	0121002	assesmnt.asesdt	08APR2016
4	0121003	assesmnt.asesdt	02JUN2018
5	0181001	assesmnt.asesdt	15APR2015
6	0182001	assesmnt.asesdt	12JUN2018
7	0191002	addltr.lymedt	30MAR2017
8	0191003	assesmnt.asesdt	05DEC2017
9	0211001	assesmnt.asesdt	24APR2018
10	0211002	assesmnt.asesdt	02DEC2016
11	0211003	assesmnt.asesdt	01JUN2018
12	0211004	addltr.lymedt	30MAY2018
13	0211005	assesmnt.asesdt	16APR2018
14	0211006	assesmnt.asesdt	14MAY2018
15	0221001	addltr.lymedt	29NOV2017
16	0251001	assesmnt.asesdt	19APR2018

CONCLUSION

Macros can provide a quick and effective solution to repetitive code and have the functionality to incorporate different rules and criteria for different datasets or variables. As discussed above, the macro can be applied to any analysis that requires selecting from multiple dates in multiple sources. This example was from a study that was not fully CDISC compliant, however depending on the situation the code could be altered to take advantage of the SDTM and ADaM naming convention and sashelp.vcolumn to automatically identify all date variables in the database for processing.

The %alive macro discussed here helps to identify which dataset and date variable combination has been selected from the large number specified in the derivation of last known alive date in ADSL. When discussing any discrepancies in the values used for last known alive date, the most useful dataset to refer to is the all_s dataset shown above, as this shows every date variable for every subject in descending order and has the unique identifier which displays the source dataset and date variable of each value.

One recommendation would be to add in code to delete the individual datasets coming from the %alive macro once these have all been set together to create the dataset all_s as shown above. This will free up memory within SAS and will help making navigating to datasets easier. This has been commented out in the code provided in the appendix for ease of traceability for the unfamiliar user.

CONTACT INFORMATION

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

Melanie O'Neill
PPD
Fleming House
Strathclyde Business Park
Bellshill, ML4 3NJ
Work Phone: +44 (0) 1698 575097
Email: Melanie.o'neill@ppdi.com
Web: www.ppdi.com

APPENDIX I

```
*******************************************************************************
* FILENAME:    ALIVE.SAS
* DEVELOPER:   Melanie ONeill
* PLATFORM:    SAS 9.3
*
* PARAMETERS:
* ds    = is the dataset of containing the date variables
* date = event date(s) of interest.
* sub   = subject variable used in the dataset e.g. pt or subjid
*
*******************************************************************************
*** ALIVE macro to identify last known alive date ***;
*** create standard subject variable within reference dataset (death dataset) and sort
by subject. This dataset can then be merged to datasets being called within the alive
macro ***;
   proc sort data=trans.dth out=death (keep=pt dsdt rename=(pt=subject));
     by pt;
   run;

%macro alive (ds=, date=, sub=);
 %let i=1;

*** scan through each value entered in the date string within the macro call***;
 %let dt=%scan(&date, &i, %str( ));

*** check this has not returned an empty string, if not then continue with process***;
 %do %while (%superq(dt) ne %str( ));

*** sort dataset of interest by subject and descending event date, the first record
will hold the maximum available event date ***;
       proc sort data=trans.&ds. out=&ds._1_&dt.;
         by &sub. descending &dt.;
        where not missing(&dt.);
       run;

*** create standard subject variable and assign a standard length, as multiple
datasets are being used there could be various lengths that will cause a log message
when all datasets are combined
create origin variable that combines dataset of interest and event date. this will be
used as an identifier within the final overall dataset ***;
       data &ds._1_&dt.a (keep=subject origin &dt.);
         length subject $100. origin $50.;
           set &ds._1_&dt.;
          subject=pt;
         origin=catx(".", "&ds.", "&dt.");
       run;

*** merge reference dataset and keep records that are in dataset of interest where
reference date is missing. ***;
       data &ds._2_&dt.;
         merge &ds._1_&dt.a(in=a) death(in=b);
           by subject;
                if a and (dsdt eq .);
       run;

*** check that merged dataset has records before attempting to transpose ***;
       proc contents data=&ds._2_&dt.
        out=contents(where=(upcase(name)="SUBJECT")) noprint;
       run;

       proc sql noprint;
         select NOBS into:_obs from contents;
       quit;

       %if &_obs. > 0 %then %do;
```

```
*** transpose dataset to get one record per subject ***;
             proc transpose data=&ds._2_&dt. out=&ds._3_&dt.;
               by subject origin;
                 var &dt.;
             run;

*** keep first column from transposed dataset as this will have the latest available
event date ***;
             data dataset_&ds._&dt. (keep=subject col1 origin
                                     rename=(col1=date));
               set &ds._3_&dt.;
             run;

      %end;

*** cycle to next variable in date string ***;
      %let i=%eval(&i+1);
      %let dt=%scan(&date, &i, %str( ));
 %end;

*** remove intermediate datasets (optional) can be used once program has been
developed to help clear up library space ***;

/*proc datasets lib=work;*/
/*delete &ds._:;*/
/*quit;*/
/*run;*/

%mend alive;

%alive (ds=addltr, date=lymedt lymsdt, sub=pt);
%alive (ds=assesmnt, date=asesdt , sub=pt);

*** set together all datasets created by the alive macro, these datasets will have the
maximum event date from each dataset of interest ***;
   data all;
     set dataset_:;
   run;

*** sort by descending event date across all datasets of interest ***;
   proc sort data=all out=all_s;
     by subject descending date;
   run;

*** select the first record per subject, this will be the maximum even date available
across all datasets of interest ***;
   data max_date;
     set all_s;
      by subject descending date;
        if first.subject;
   run;
```