WUSS 2022 – Session # 100

Finding Your Latest Date

William Coar, Cytel

ABSTRACT

During a clinical trial, participants are often required to visit a clinic multiple times throughout the course of treatment as well as for post-treatment follow-up. Some measurements are taken that day, thus have an association with that visit date. Other data, such as events that occurs between visits, will have different dates, usually the dates at which the events occurred. There is often a need to look for the last known date in which data was collected for a patient. This can involve scanning 20-30 different datasets that contain dates, and then identifying the latest date for which we have data.

This quick tip presentation will utilize macro programming to compartmentalize the identification of the most recent date, including the dataset and variable from which it was obtained. When using standardized data structures, simple imputation can also be performed for partial dates to provide reasonable approximations of a date when working with interim data. The programming concepts presented in this quick tip can be extended to further automate finding a latest date without specifying dataset or variable names.

Programming is done using SAS 9.4M6 in a Windows environment. CDISC standardized data structure will be used to demonstrate the application, though it's use extends to other data structures.

INTRODUCTION

Much data is collected throughout a patient's participation in a clinical trial. Data are captured directly into an EDC by staff where patients receive medical treatment using patient records, or *source documents*. Data can be categorical in nature, continuous, free text fields (comments), or dates.

Reviewing all data captured is key for programmers to know where to start. Reviewing the electronic case report form in addition to variable names and labels in each dataset will help identify potential dates. It may be the case that dates are collected as individual components of month, day, and year. While these will need post-processing, the proposed algorithms are easily updated to accommodate this. Similarly, date information will be captured as a character string in IS08601 format using CDISC standards. The algorithm below makes this assumption thought it is not a limitation to the process.

STEP 1: ALGORITHM GIVEN A SINGLE DATASET

A single dataset does not imply a single date variable. Consider a subject visit dataset (SV). There are two date variables: SVSTDTC and SVENDTC. For simplicity, let's just start with SVSTDTC by creating a dataset with a numeric representation for SVSTDTC. We can then sort by patient and the numeric date, keeping only the last record.

Starting with the end in mind, we create and keep a number of variables in our resulting dataset so that we know what dataset and variable the entry came from. This helps tremendously with debugging. We only have interest in the date component of the IS80601 format, taking only the characters prior to the delimiter T. For simplicity, we begin by assuming complete dates. This can be generalized with imputation, but that is outside the scope of a quick tip.

```
Data ld;
   Set SV;
   Length srcdom srcvar $50 srcval $10;
   srcdom='SV';
   srcvar='SVSTDTC';
   srcval=scan(svstdtc,'T');
   if length(srcval)=10 then ndate=input(srcval,yymmdd10.);
   keep usubjid ndate srcdom srcvar srcval;
run;

proc sort data=ld;
      by usubjid ndate;
      where missing(ndate)=0;
run;

data ld;
      set ld;
      by usubjid ndate;
      if last.usubjid;
run;
```

Understanding that this process needs to iterate over multiple date variables and datasets, macro processing can simplify. In the above example, we can create a straightforward macro to take an input dataset and date variable, and output a new dataset with the latest date for that patient and that single date variable. The above code is updated to use macro processing.

```
%macro lastdate(indata=, dtvar=, outdata=);

Data &outdata;
   Set &indata;
   Length srcdom srcvar $50 srcval $10;
   srcdom="&indata";
   srcvar="&dtvar";
   srcval=scan(&dtvar,'T');
   if length(srcval)=10 then ndate=input(srcval,yymmdd10.);
   keep usubjid ndate srcdom srcvar srcval;
run;

proc sort data= &outdata;
      by usubjid ndate;
      where missing(ndate)=0;
run;

data &outdata;
      set &outdata;
      by usubjid ndate;
      if last.usubjid;
run;
%mend;
```

Once a macro is compiled, then we can easily obtain two datasets; one contains the last date for each patient based on SVSTDTC while the other contains the last date based on SVENDTC. We can then begin processing these two datasets to pick the record with the maximum numeric date.

```
%lastdate(indata=SV, dtvar=SVSTDTC, outdata=ZZ_LDATE1);
%lastdate(indata=SV, dtvar=SVENDTC, outdata=ZZ_LDATE2);

* Using a : is like a wild card. This SET statement will set
together all datasets in the workspace whose name starts with
zz_.;

Data ldate;
   Set zz_:;
run;

proc sort data=ldate;
       by usubjid ndate srcdom srcvar;
run;

data ldate;
       set ldate;
       by usubjid ndate srcdom srcvar;
       if last.usubjid;
run;
```

At this point, we can see how a short macro with a little post processing can obtain a single record for each dataset considering all date variables specified. We also see a programming trick taking advantage of the :, a wild card, discussed in [1] and [2].

STEP 2: IDENTIFY DATA WITH DATES

Knowing the dataset and variable name, a space delimited list can be obtained where each element in the list contains the dataset name and variable name. With the implementation of data standards, it becomes easier to define a list of potential dates across datasets. Besides, as good programmers, we should know this because we know our data.

STEP 3: ONE STEP FURTHER WITH MACROS

Once a list is defined, macro processing will allow us to dynamically determine the number of date variables to be used for looped. With each iteration of the loop, we select a dataset.variable name from the list, and call our ***%lastdate*** macro. A set of output datasets can be created, set together, and processed using the same technique (with a wildcard :) described earlier.

```
%macro datelist(dlist=, dout=);

%let nlist=%sysfunc(countw(&tst,' '));
%do j=1 %to &NLIST;
       %let thisj=%scan(&DLIST,&J,' ');
       %let thisjd=%scan(&THISJ,1,'.');
       %let thisjv=%scan(&THISJ,2,'.');
       %lastdate(indata=&THISJD, dtvar=&THISJV, outdata=zz_ldate_&J);
%end;

data &dout;
       set zz_ldate_:;
run;

proc sort data=&dout;
       by usubjid ndate srcdom srcvar;
run;

data &dout;
       set &dout;
       by usubjid ndate srcdom srcvar;
       if last.usubjid;
run;

proc datasets library=work;
       delete zz_ldate_: / memtype=data;
run;
quit;
%mend;
```

Using the above macro we can simplify programming to obtain an value of a latest date in the data for each patient.

```
* Find the last known date.;

%datelist(dlist= dm.rfstdtc dm.rfendtc ds.dsstdtc lb.lbdtc sv.svstdtc sv.svendtc
ec.ecstdtc ec.ecendtc ex.exstdtc ex.exendtc vs.vsdtc, dout=lptvis);
```

CONCLUSION

Certainly there are many programming techniques and programming styles that can be utilized to determine a last date. The steps presented start with a one-at-a-time approach, then proceed with steps to automate. The approaches can easily extend to cases that include date imputation, or even automatically scanning datasets to dynamically create a list of dataset.variable names that exist in an entire library. They can easily be modified to identify an earliest date for a subset of records for each patient, such as a earliest dosing date, or an earliest start of subsequent therapy.

REFERNCES

[1] Coar, W. The Lost Sibling of the Semi-colon, WUSS 2012

[2] Tinazzi, A. Mind the Gap: Make Sure you are Upskilled with SAS 9.x, PHUSE EU Connect 2020

CONTACT INFORMATION

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

William Coar
Axio Research, A Cytel Company
William.coar@cytel.com

APPENDIX:

LASTDATE

```
%macro lastdate(indata=, dtvar=,outdata=);

data &outdata;
       set &indata;
       length srcdom srcvar $50 srcval $10;
       srcdom=upcase("&INDATA");
       srcvar=upcase("&DTVAR");
       srcval=scan(&dtvar,1,"T");
       if length(srcval)=10 then ndate=input(srcval,yymmdd10.);
       keep usubjid ndate srcdom srcvar srcval;
run;

proc sort data=&outdata;
       by usubjid ndate;
       where missing(ndate)=0;
run;

data &outdata;
       set &outdata;
       by usubjid ndate;
       if last.usubjid;
run;
%mend;
```

DATELIST

```
%macro datelist(dlist=, dout=);

%let nlist=%sysfunc(countw(&dlist,' ‘));
%put Number in list: &nlist;

%do j=1 %to &NLIST;
       %let thisj=%scan(&DLIST,&J,' ');
       %let thisjd=%scan(&THISJ,1,'.');
       %let thisjv=%scan(&THISJ,2,'.');
       %put This data = &THISJD;
       %put This variable = &THISJV;
       %lastdate(indata=&THISJD, dtvar=&THISJV, outdata=zz_ldate_&J);
%end;

data &dout;
       set zz_ldate_:;
run;

proc sort data=&dout;
       by usubjid ndate srcdom srcvar;
run;

data &dout;
       set &dout;
       by usubjid ndate srcdom srcvar;
       if last.usubjid;
run;

proc datasets library=work nodetails noprint;
       delete zz_ldate_: / memtype=data;
run;
quit;
%mend;
```