PhUSE US Connect 2019

Paper DH05

Dynamically harvesting study dates to construct and QC the Subject Visits (SV) SDTM domain

Cleopatra DeLeon, PPD, Austin, TX USA
Laura Bellamy, PPD, Austin, TX USA

ABSTRACT

In clinical trials, dates associated with actual subject encounters are collected and stored across multiple datasets. Over the course of a study, more visits or assessments may be added. Study dates are often plagued with data entry error. Therefore, great pains must be taken when consolidating study dates within the Subject Visits (SV) domain. The author demonstrates a concise methodology leveraging SASHELP views and CALL EXECUTE statements to dynamically harvest dates while building the SV domain. Logical, sequential, and other quality checks (QCs) such as cross-checking key study dates and SDTM Trial Design datasets are proposed. The paper concludes with a discussion to extend this methodology to the Subject Elements (SE) domain.

INTRODUCTION

Throughout the course of a study, visit dates are collected for nearly every subject encounter. On complex studies, combining and summarizing these dates into a single dataset, like the SV domain, is a challenge to describe in a specification, let alone program. Often, analysts use a piecemeal, albeit straightforward method of carefully identifying and programming each study encounter recorded in the clinical database or taken from an external data source for inclusion in the SV domain. This manual approach is tedious, and usually not dynamic enough to keep pace with the fluid nature of many clinical trials.

We believe that a self-documenting, data-driven approach utilizing SASHELP views and CALL EXECUTE statements to programmatically select dates for the SV domain is more robust than manual identification and typical data step programming. Further, the development of the SV domain, combined with a solid understanding of the study design and schedule of events (SOE) provides a convenient opportunity to QC dates and monitor subjects' progression through the study.

CHOOSING DATES FOR SV

When constructing the SV domain, understanding how subjects move through a study is key. Referencing the protocol for study design and SOE provides the backbone for the reason and timing of planned visits and key study dates. Referencing the case report form (CRF) and the database design specifications provides the mechanism and location for collecting dates associated with specific study timepoints.

Both scheduled and unscheduled visits should be included in the SV domain. Typically, dates collected outside of a subject's participation in a study, such as birthdate, and dates for medical history, prior therapies, procedures and medications, should not be considered. In some cases, inclusion of dates for treatment-emergent adverse events (TEAEs) and concomitant medications are necessary. For example, in oncology trials, when deriving the date of last contact, TEAE dates would be included as an unscheduled visit. Further, an adverse event may trigger an unscheduled visit, thus the event date might be considered relevant.

USING SASHELP AND CALL EXECUTE

Utilizing SASHELP views and CALL EXECUTE to extract dates for the SV domain is not as difficult as it may seem. Fundamentally, programming SV requires rerunning similar code across many datasets and variables. SASHELP views and CALL EXECUTE can do this type of programming efficiently while self-documenting in one concise data step.

SASHELP views are essentially queries that store descriptor information about the data in a SAS session. These views are native to the SAS system and need no set up to access them. Information such as SAS libraries, dataset names, and variable attributes are stored automatically at the start of every SAS session. This information is pulled into the session dynamically -- if the data changes, the view will change instantaneously. Another advantage of using a view versus a dataset is that a view requires less working memory space.

CALL EXECUTE mimics the programming in a SAS macro loop. Instead of providing specific parameters, the execution is completely controlled by parameters stored in a dataset. CALL EXECUTE is a natural partner to SASHELP views because of its ability to drive macro type code using dataset descriptor information found in the VCOLUMN view.

METHODOLOGY

First, create a metadata set called ALLDATES. Using the SASHELP.VCOLUMN view, dataset information is filtered using a WHERE clause to point to the location of the data and to identify relevant date variables therein. Once created, a PROC PRINT of the ALLDATES dataset provides a comprehensive list of all datasets and variables considered for the SV domain. This listing may come in handy when specifying or documenting the SV domain in the Define-XML, a required standard for data submissions to the FDA and PMDA.

Next, using ALLDATES, the CALL EXECUTE statement dynamically generates a series of PROC SORTs to harvest distinct values of a particular date variable, associated subject-visit information, and, for traceability, the source dataset name. The resulting output datasets are named in a self-documenting manner using the following convention, "SV_[*dataset name*]_[*date variable name*]". Using a KEEP= dataset option, only data source, subject ID, visit-related variables, and date variables are kept. To simplify data handling in the next steps, duplicate records are removed using the NODUPKEY option; and, date variables are renamed as RAWDTC for consistency using RENAME=.

From here, the construction of the SV domain is straightforward. The final steps are to append the output datasets, transpose and store the source dataset names using the SVUPDES variable, and derive the minimum start and maximum end dates (i.e., SVSTDTC and SVENDTC) for each of the subjects' visits. Prior to finalizing the SV domain, we perform quality checks and complete any special handling for unplanned/unscheduled visits as needed per the study requirements.

EXAMPLES

In the following example, we filter by datasets located in the STAGEDR library. The datasets in this location have a consistent structure. Knowing that the variable names ending with "DTC" contain ISO 8601 date formatted values, we harvest date variables using a WHERE clause. Additionally, the WHERE clause is augmented to ignore irrelevant data from MH, SE, PR, and FORMATS datasets, as well as any variables for birthdate. The resulting dataset, ALLDATES contains the location, dataset, and name of all relevant date variables in LIBNAME, MEMNAME and NAME respectively.

```
* output the datasets/variables for QC and documentation *;
data alldates;
  set sashelp.vcolumn(keep=libname memname memtype name type length label

       /* choose a libname from which to gather dates */
      where=(libname='STAGEDR'

             /* remove irrelevant datasets */
             and memtype='DATA' and memname^in('FORMATS' 'MH' 'SE' 'PR')

             /* remove irrelevant date variables */
             and (upcase(label)^?('BIRTH') and name^='BRTHDTC')

             /* subset for ISO date suffix DTC */
             and upcase(reverse(substr(strip(reverse(name)),1,3)))="DTC" ));
```

Next, ALLDATES variables are passed through the CALL EXECUTE statement to harvest the dates for SV. The series of PROC SORTs generated by this CALL EXECUTE statement will remove duplicate date records and create datasets that contain USUBJID, VISIT, VISITNUM, DATAPAGENAME (the source dataset name), and RAWDTC (the source date variable, renamed for consistency).

```
  call execute("proc sort nodupkey data= "
                  || strip(libname) ||"."
                  || strip(memname)
                  || "(keep=usubjid visit visitnum datapagename "
                  || strip(name)
                  || " where=("
                  || strip(name) ||"^=')) out=SV_"
                  || strip(memname)||"_"||strip(name)
                  || "(rename=("
                  || strip(name)
                  ||"=rawdtc)); by usubjid visitnum visit datapagename "
                  || strip(name)
                  || "; run;");
run;

* stack the sv_[dataset name]_[date variable name] datasets *;
data allsv;
  set sv_:;
run;
```

For distinct values of USUBJID, VISITNUM/VISIT, and RAWDTC, the values captured in DATAPAGENAME are concatenated into the SVUPDES variable to preserve all data sources. The clinical database in this example included a variable, DATAPAGENAME which held the name of the source CRF page. Note that the SVUPDES variable should be only used to store the description of what happened during an unplanned visit, in compliance with CDISC Implementation Guidelines. Therefore, upon finalizing the SV domain, SVUPDES should only be left populated for unscheduled visits.

```
* transpose source (datapagename) for traceability – also used for svupdes *;
proc transpose
  out=tsv data=allsv;
  by usubjid visitnum visit rawdtc;
  var datapagename;
run;

data tsv(drop=_: col:);
  set tsv;
  length svupdes $200;
  svupdes=catx('; ',of col:);
run;
```

For distinct values of SUBJECT and VISIT, the SVSTDTC and SVENDTC date ranges are calculated as the minimum and maximum RAWDTC for each grouping. The resulting dataset provides the basic structure for SV.

```
* get min/max start/end dates *;
proc sql noprint;
create table sv as select distinct
  usubjid, visitnum, visit, svupdes,
  min(scan(rawdtc,1,'T')) as svstdtc length=20,
  max(scan(rawdtc,1,'T')) as svendtc length=20
  from tsv
  group by usubjid, visitnum, visit, svupdes;
quit;
```

Before finalizing SV depending, slotting (or renumbering) unscheduled visits may be considered depending on specific sponsor requirements and study designs.

Visit or Date Anomalies

At this point, one might consider the major development of SV complete. However, having the entirety of a study's subjects, visits and associated dates at one's fingertips offers a convenient opportunity for checks, not just on data cleanliness, but also on subjects' study participation. Indeed, as per intended, the SV domain allows reviewers to get a good feel for the breadth of the study data.

Checking for illogical or outlying dates is uncomplicated and efficient in SV. The following data step checks for dates occurring (1) after date of death, (2) after data extraction, or (3) prior to a subject's informed consent or participation

start date. There may be some circumstances were visits occur after a death, such as a follow-up or an autopsy. Such exceptions should be judged individually per study design. Generally, these dates are unexpected, and as such, should be scrutinized as a data anomaly.

```
data ck1;
  * merge in key study dates – may be used later for SE development (?!) *;
  merge tsv dm(keep=usubjid rficdtc rfstdtc rfendtc dthdtc);
  by usubjid;

  * (1) QC check: dates past Death date *;
  if ^missing(dthdtc) and dthdtc < svstdtc then
       put 'NOTE: DM QUERY - Dates past death date '
             USUBJID= 'SOURCE=' SVUPDES VISIT= SVSTDTC= 'DEATH DATE=' DTHDTC= ;

  * (2) QC check: dates past extract date *;
  if ^missing(svstdtc) and put("&g_extract."d,yymmdd10.-l) < svstdtc then
       put 'NOTE: DM QUERY - Dates past extract date '
             USUBJID= 'SOURCE=' SVUPDES VISIT= SVSTDTC= 'EXTRACT DATE=' "&g_extract.";

  * (3) QC check: dates prior to informed consent date *;
  if ^missing(rficdtc) and svstdtc < rficdtc then
       put 'NOTE: DM QUERY - Dates prior to informed consent '
             USUBJID= 'SOURCE=' SVUPDES VISIT= SVSTDTC= RFICDTC= ;
run;
```

Having all visit dates in one place allows the comparison of visits across and within different CRF forms or pages. Typically, visits are expected to occur sequentially. In SV, the visit dates should appear in a linear pattern, and may be checked to ensure all dates from one visit occurred prior to the next, as shown in the data step below. First, sort by SUBJECT, visit number (VISITNUM), and then, by start date (SVSTDTC). Using the LAG function, flag records where the previous date is greater than the next. To report the subset of dates that appear out of order, flagged dates are merged back to the original dataset and printed for ease of review. While comparing out of order dates, another check on the number of days between two visits may be calculated and examined as well.

```
data ck;
  set sv;
  by usubjid visitnum visit svstdtc;
  lagdtc=lag(svstdtc); * create variable with lagged svstdtc values *;
  if first.usubjid then lagdtc=''; * null lagdtc from different subject *;
  if ^missing(svstdtc) and lagdtc>svstdtc then OutofOrder='Y'; * flag dates *;

  * calculate the number of days diff between two visits *;
  diffsv = input(svstdtc,yymmdd10.-l) - input(lagdtc,yymmdd10.-l);
run;

proc sql noprint; * merge flagged dates, keeping both overlapping records *;
create table dmquery as select distinct
  ck.*
  from ck, ck(keep=usubjid lagdtc outoforder where=(outoforder='Y')) as x
    where ck.usubjid=x.usubjid and (ck.outoforder='Y' or ck.svstdtc=x.lagdtc);
quit;

proc print width=min
  data=dmquery(drop=lag:);
  title 'Data check - Subjects with dates out of visit order';
run;

proc freq
  data=ck;
  title 'Data check – Distribution of days between two visits';
  tables visitnum*visit*diffsv/ missing;
run;
```

Many protocols designate a planned study day for each scheduled visit in its SOE. When some flexibility is expected, the SOE may also provide a window of *x* number of days. The planned study day for scheduled visits may be included within the Trial Visit (TV) domain's VISITDY variable. The difference between SV.SVSTDY and TV.VISITDY would illustrate the actual study day variance that may be crosschecked against any specified window. While checking the study day window, SV.VISIT may also be crosschecked against TV.VISIT to ensure SDTM compliance.

```
data ck3 tvxck;
  merge sv(in=sv where=(upcase(visit)^? 'UNSCH'));
        tv(in=tv keep=visitnum visitdy);
  by visitnum; * merge SDTM TV record by VISITNUM *;

  * check the expected window, within +/- 7 days of planned study day (tv.visitdy) *;
  tvdiff = abs(svstdy - visitdy) > 7 ;
  output ck3;

  if (in sv and not tv) then output tvxck;
run;

* records outside of the target window *;
proc freq
  data=ck3;
  title 'Data check – Distribution of difference between SV and TV visit days';
  tables visit*tvdiff/list missing;
run;

* Visits in SV but not in TV *;
proc freq
  data=tvxck;
  title 'SDTM compliance check – Visits in SV not in TV';
  tables visit/ missing;
run;
```

APPLICATION TO SE DOMAIN AND CONCLUSION

The SV and SE domains have similarities. While the SV domain presents a comprehensive view of subjects' progress through the study, the SE domain describes a summarized view of subjects' progress through just the major study elements. The EPOCH variable in SE characterizes a category of related VISITs associated with the major study elements.

By incorporating programmatic quality checks in SV that require key dates – key dates which potentially demarcate EPOCHs – we propose that SV and SE are best developed in parallel. Indeed, per SDTM compliance guidelines, the EPOCH variable is expected in the SV domain, necessitating careful planning to avoid any programming circularity. Parallel development using SASHELP.VCOLUMN and CALL EXECUTE to dynamically harvest study dates, ensures an elegant, robust, flexible, and self-documenting method to produce the SV and SE domain.

REFERENCES

- CDISC Study Data Tabulation Model and SDTM IG V3.2 at http://www.cdisc.org/sdtm
- SAS® 9.4 Language Reference: Concepts, Sixth Edition at http://documentation.sas.com/?docsetId=lrcon&docsetTarget=n07vq6hmss6f67n1vhnnn5cw0chh.htm&docsetVersion=9.4&locale=en
- CALL EXECUTE made easy for SAS data-driven programming at https://blogs.sas.com/content/sgf/2017/08/02/call-execute-for-sas-data-driven-programming/

ACKNOWLEDGMENTS

Special thanks to Jesse Pratt, Lynn Clipstone and Ajay Gupta for the gift of their time and experience.

RECOMMENDED READING

- MAPPING OF SV BY Nicola Tambascia
- A Better Perspective of SASHELP Views by John R. Gerlach
- Creating SV and SE First by Henry B. Winsor and Mario Widel

CONTACT INFORMATION

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

Cleopatra DeLeon
PPD – Austin, Texas
+1 910 558-4444
cleopatra.deleon@ppdi.com

Laura Bellamy
PPD – Austin, Texas
+1 910 558-2557
laura.bellamy@ppdi.com

Brand and product names are trademarks of their respective companies.

The code for this paper was generated using SAS software. Copyright © 2019 SAS Institute Inc. SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc., Cary, NC, USA.