PharmaSUG 2021 - Paper QT-186

Where's Waldo? An SDTM Crawler

Derek Morgan, CSG, an IQVIA business

ABSTRACT

This is a utility that will quickly find all records for a given USUBJID across the entire SDTM database and can be modified to display all common SDTM data. Originally created to detect false subject visits from a defective clinical data entry system, as written, it lists all --DTC variables from an SDTM data set, as well as VISIT and VISITNUM if present.

INTRODUCTION

It is often useful to have the ability to search a database for specific records based on common criteria across all tables. This method was first developed to allow end users to perform interactive searches in a SAS®-based data entry system deployed across several multi-center observational trials coordinated at Washington University in St. Louis. It has since been modified and placed into a macro for non-interactive use with SDTM, made possible by the common variable naming scheme in SDTM.

This macro was most recently used to detect errors in the SV domain caused by a defective data entry system. This system created a record in the SV domain whenever a subject's record was opened for a given visit, regardless of whether actual data was recorded. Unfortunately, due to the legacy nature of the data, it was not possible to clean these data post-hoc. As a result, many such "phantom" visits were recorded in the SDTM, so SV was unreliable as the documentation of a subject's study participation. We needed a way to quickly determine if a subject truly had a study visit or if the record was generated accidentally. The present version of the macro allows the user to direct the output to one of four ODS destinations: HTML, PDF, LISTING, or as a multiple worksheet Excel workbook. An example of the PDF output (with page breaks removed) can be found in Appendix 1, and a sample of the Excel format is in Appendix 2.

The complete, documented macro code is provided in Appendix 3. The body of this paper will detail code segments that may be of interest to programmers.

WHAT DATA DO I HAVE? DICTIONARY TABLES AS A RESOURCE

This macro uses DICTIONARY.TABLE to count and enumerate all the SDTM datasets in a given SAS library. The dataset count is then used in this step as the upper limit of the indexed macro variables where we store both the fully-qualified SAS data set name and the domain name in those indexed macro variables as shown in Code Fragment 1:

```
1.  PROC SQL NOPRINT;

2.  SELECT STRIP(PUT(COUNT(distinct memname),5.)) INTO :sdtmct
3.  FROM dictionary.tables
4.  WHERE upcase(libname) EQ upcase("&libnam")
5.  ;

6.  SELECT STRIP(catx('.',"&libnam",memname)), memname
           INTO :sds1-:sds&sdtmct, :memnam1-:memnam&sdtmct
7.  FROM dictionary.tables
8.  WHERE upcase(libname) EQ upcase("&libnam")
9.  ;
```

Code Fragment 1: Using DICTIONARY.TABLES

In line 2, the SQL COUNT() function creates a numeric variable containing number of datasets, so the STRIP() and PUT() functions are used to turn that right-justified number into a left-justified character variable. Otherwise, you cannot use it as an index in the INTO clause, because it would see the macro

variable named “&SDS 36”, and this will cause an error because the variable name is invalid. As a matter of course, when using SQL to create macro variables, using the STRIP() function eliminates any impact from leading or trailing blanks. Line 6 uses the count macro variable (&SDTMCT) to set up the range of indexed macro variables for the SAS data set names (&SDS1-&SDSx) and the domain names (&MEMNAM1-&MEMNAMx). Now you have everything you need to iterate through every data set from the library in a macro loop.

DICTIONARY.COLUMNS gives you access to all the information SAS maintains about table columns. The execution speed depends on how many SAS data libraries are defined within the session, how many tables are in those libraries, and how many variables are in each of those tables. Anything you can do in the WHERE clause to narrow the SQL search will rapidly improve execution time. Code Fragment 2 demonstrates how this is used to get a list of all the --DTC variables from a data set:

```
1. PROC SQL NOPRINT;
2. CREATE TABLE dtvars AS
3. SELECT STRIP(catx('.',"&libnam ",memname)) AS dsname, name AS varname
   FROM dictionary.columns
4. WHERE UPCASE(libname) EQ UPCASE("&libnam") AND memname EQ "&&memnam&i"
         AND name LIKE '%DTC'
5. ;
6. QUIT;
```

Code Fragment 2: Using DICTIONARY.COLUMNS

We will be using the list of --DTC variables to construct our date filter. We can also use the &SQLOBS automatic macro variable to check if any --DTC variables were found. If it’s zero, then this data set (represented as “MEMNAM&I”) is excluded from the output. This prevents errors from occurring when data sets without subject information are tested, such as the trial design data sets. The DTVARS data set is used to build the date filter and the list of date variables, then store them into macro variables in a succeeding DATA _NULL_ step. It is possible to build the filter, list of date variables and their corresponding macro variables within this PROC SQL step using the SEPARATED BY clause, but separating it made for easier debugging when this code was originally developed.

TAKING ADVANTAGE OF VARIABLE INFORMATION FUNCTIONS

Code Fragment 3 was used to find if the SDTM data set contained the VISIT variable (and VISITNUM, assuming they are always paired.) Although I haven’t benchmarked it, this is probably the fastest way to determine if a variable exists in a SAS data set, because it doesn’t require processing any records in the data set:

```
1. %LET dsid=%SYSFUNC(OPEN(&&sds&i));
2. %LET vn=%SYSFUNC(VARNUM(&dsid ,VISIT)); /* Test upper and lower */
3. %LET vn2=%SYSFUNC(VARNUM(&dsid ,visit));/* case variable names */
4. %IF &vn GT 0 OR &vn2 GT 0 %THEN
5.     %LET visvar=VISIT VISITNUM ;
6. %ELSE
7.     %LET visvar= ;
8. %LET rc=%SYSFUNC(CLOSE(&dsid));
```

Code Fragment 3: Scanning for Specific Variables

We start and end using the OPEN() and CLOSE() functions. We do not need any record-level information; therefore, no DATA step or SAS procedure is necessary. Everything we want is in the header section of the SAS data set. We use the VARNUM() function in lines 2 and 3 (in case the variable name of “visit” is all lowercase) to see if it exists because the VARNUM function returns 0 if it doesn’t find a variable by that name. If it’s found, then put VISIT and VISITNUM in a macro variable for later use, otherwise, leave that macro variable blank. It is important to make sure you CLOSE() any data set you open in this way. If you do not, you could create a memory leak, and risk getting/modifying the wrong information.

You can find the variable length, format, informat, type, label, and whether it exists without processing any records in a data set by using the variable information functions in SAS. Table 1 lists these functions. Note that except for the VARNUM() function, all these require the variable number, not the variable name. You may also substitute "VARNUM(*data-set-id,variable-name*)" for the variable number.

VARNUM(*data-set-id,variable-name*)	Get the variable number (order within data set). Returns 0 if not found.
VARLEN(*data-set-id,variable-number*)	Get the length of a variable.
VARFMT(*data-set-id,variable-number*)	Get the format assigned to a variable.
VARINFMT(*data-set-id,variable-number*)	Get the informat assigned to a variable.
VARTYPE(*data-set-id,variable-number*)	Get the type of a variable.
VARLBL(*data-set-id,variable-number*)	Get the label of a variable.
VARNAME(*data-set-id,variable-number*)	Get the name of a variable.

Table 1: Variable Information Functions

CONCLUSION

As stated in the introduction, the complete macro is in the appendix, below. This macro can be used by data management, or as I have used it, to find data for a specific subject during ADaM programming and data quality checking. If you need a tool that will allow you to quickly look up all the records for a subject across all SDTM domains, and the data are stored in SAS data sets, this is relatively quick and reliable.

ACKNOWLEDGMENTS

Although I didn't need to contact them for this paper, I'd like to thank the Technical Support Division at SAS Institute for all the help they've given me over the course of my career. As far as I'm concerned, you're still the best in the business.

CONTACT INFORMATION

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

Derek Morgan
CSG, an IQVIA business
mrdatesandtimes@gmail.com

APPENDIX 1: SAMPLE RTF/PDF OUTPUT

Note: Each data set is printed on its own physical page(s). The page breaks have been removed to display this example.

Subject Search SDTM Results
Dataset: sdtm.DM

Unique Subject Identifier	Subject Reference Start Date Time	Subject Reference End Date Time	Date Time of First Study Treatment	Date Time of Last Study Treatment	Date Time of Informed Consent	Date Time of End of Participation	Date Time of Death	Date Time of Birth	Date Time of Collection
STUDY01-0001					2017-09-14	2017-10-11T05:24		1967-08-06	2017-09-14

Subject Search SDTM Results
Dataset: sdtm.DS

Unique Subject Identifier	Date Time of Collection	Start Date Time of Disposition Event
STUDY01-0001	2017-09-14	2017-09-14
STUDY01-0001	2017-10-10	2017-10-10

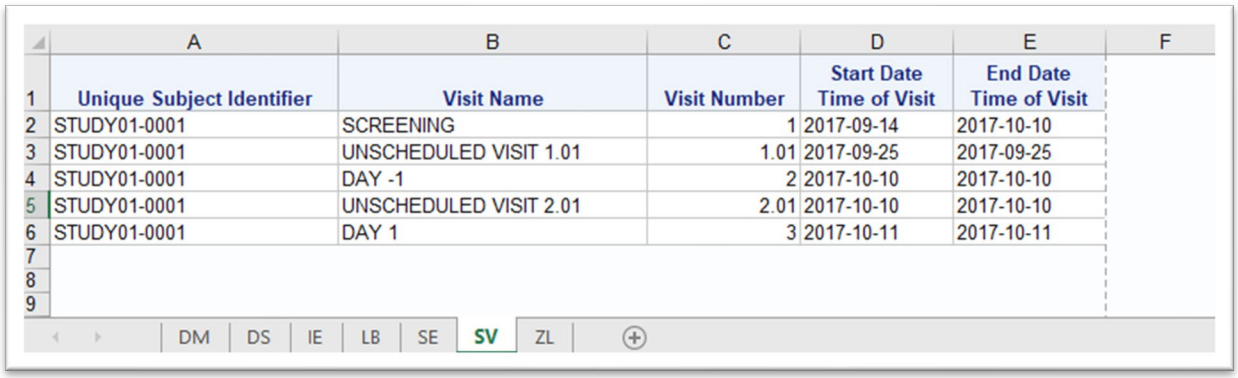
Subject Search SDTM Results
Dataset: sdtm.IE

Unique Subject Identifier	Visit Name	Visit Number	Date Time of Collection
STUDY01-0001	SCREENING	1	2017-10-10

Subject Search SDTM Results
Dataset: sdtm.LB

Unique Subject Identifier	Visit Name	Visit Number	Date Time of Specimen Collection
STUDY01-0001	SCREENING	1	2017-09-14T10:04
STUDY01-0001	SCREENING	1	2017-09-14T10:04
STUDY01-0001	SCREENING	1	2017-09-14T10:04
STUDY01-0001	SCREENING	1	2017-09-14T10:04
STUDY01-0001	SCREENING	1	2017-09-14T10:04
STUDY01-0001	SCREENING	1	2017-09-14T10:04
STUDY01-0001	SCREENING	1	2017-09-14T10:04

APPENDIX 2: SAMPLE EXCEL OUTPUT



	A	B	C	D	E	F
1	Unique Subject Identifier	Visit Name	Visit Number	Start Date Time of Visit	End Date Time of Visit	
2	STUDY01-0001	SCREENING	1	2017-09-14	2017-10-10	
3	STUDY01-0001	UNSCHEDULED VISIT 1.01	1.01	2017-09-25	2017-09-25	
4	STUDY01-0001	DAY -1	2	2017-10-10	2017-10-10	
5	STUDY01-0001	UNSCHEDULED VISIT 2.01	2.01	2017-10-10	2017-10-10	
6	STUDY01-0001	DAY 1	3	2017-10-11	2017-10-11	
7						
8						
9						

DM | DS | IE | LB | SE | SV | ZL

APPENDIX 3: THE COMPLETE MACRO

```
/* Include any required library/environment setup code */

%INCLUDE "mysetupfile.sas";

%MACRO waldo(
   usubjid_prefix=,  /* for data where usubjid is composed of a common
                        prefix concatenated with the subject ID */
   zero_just=,       /* width of subject ID field for leading zeroes
                        (SUBJID, not USUBJID). Leave blank if leading
                        zeroes aren't used */
   subjid=,          /* subject identifier */
   libnam=SDTM,      /* SAS libname for SDTM data location */
   outdir=,          /* output directory name */
   outtyp=,          /* output file type: PDF, HTML, EXCEL, leave blank for
                        LISTING destination */
   outfiltr=,        /* date filter - search for a specific date across
                        domains and all --DTC variables */
   debugyn=N         /* turns on MPRINT, MLOGIC and SYMBOLGEN */
) ;

%LOCAL _subjid _outtyp;

%IF &debugyn EQ Y or &debugyn EQ y %THEN %DO;
   OPTIONS MPRINT MLOGIC SYMBOLGEN;
%END;
%ELSE %DO;
   OPTIONS NOMPRINT NOMLOGIC NOSYMBOLGEN;
%END;

DATA _NULL__;
%IF &zero_just ne %THEN %DO;  /* Justify subject IDs with leading zeroes */
   LENGTH subjid $ &zero_just;
   zj = CATS('z',"&zero_just",'.');
   subjid = PUTN(SYMGET('subjid'),zj);
%END;

%ELSE %DO;  /* If not zero-justified, create subjid as character to avoid
               leading spaces */
   subjid = SYMGETC('subjid');
%END;
outtyp = LOWCASE(STRIP(SYMGET('outtyp'))); /* standardize output type */

/* Use CALL SYMPUTX instead of CALL SYMPUT to remove trailing blanks from
   macro variable */

CALL SYMPUTX(('_subjid',subjid);
CALL SYMPUTX(('_outtyp',outtyp);
RUN;
```

```
/* use dictionary.table to find out how many SDTM data sets are in the
   library, and what the names are */

PROC SQL NOPRINT;
/* STRIP() removes leading spaces from numeric so it can be used as
   index suffix in following SQL step
*/

SELECT STRIP(PUT(COUNT(distinct memname),5.)) INTO :sdtmct
FROM dictionary.tables
WHERE upcase(libname) EQ upcase("&libnam")
;

/* store SDTM data set and domain names in indexed macro variables */

SELECT STRIP(catx('.',"&libnam",memname)), memname
       INTO :sds1-:sds&sdtmct, :memnam1-:memnam&sdtmct
FROM dictionary.tables
WHERE upcase(libname) EQ upcase("&libnam")
;
QUIT;


/* Assign output filename - default is "sdtm_waldo" concatenated with
   subject ID */

%LET outfn = sdtm_waldo&_subjid;

/* Output destination selection */

ODS _ALL_ CLOSE;
%IF &_outtyp EQ pdf %THEN
    ODS PDF FILE="&outdir/&outfn..pdf";
%ELSE %IF &_outtyp EQ html %THEN
    ODS HTML FILE="&outdir/&outfn..html";
%ELSE %IF &_outtyp EQ excel %THEN
    ODS EXCEL FILE="&outdir/&outfn..xlsx";
%ELSE
    ODS LISTING FILE="&outdir/&outfn..lst";

; /* terminate ODS output statement */
```

```
/* loop through data sets */

%DO i=1 %TO &sdtmct;
    %IF &debugyn EQ Y or &debugyn EQ y %THEN %DO;
        %PUT dataset=&&sds&i; /* debug - what dataset am I checking? */
    %END;

   /* Are there --DTC variables in this SDTM dataset? */
     PROC SQL NOPRINT;
     CREATE TABLE dtvars AS
     SELECT STRIP(catx('.',"&libnam ",memname)) AS dsname, name AS varname
     FROM dictionary.columns
     WHERE UPCASE(libname) EQ UPCASE("&libnam") AND memname EQ "&&memnam&i"
           AND name LIKE '%DTC'
     ;
     QUIT;

  /* Only proceed if there is a --DTC variable. Without this, you'll get
     an error in datasets that don't have –DTC variables */

     %IF &sqlobs gt 0 %THEN %DO;
       /* Check all --DTC variables in a given dataset */
         DATA _NULL_;
         SET dtvars END=last;
         LENGTH dtfilter dtlst $ 2048;
         RETAIN dtfilter dtlst;
         IF NOT CMISS(SYMGETC('outfiltr')) THEN  /* leave date filter blank
                                                    if not requested */
            dtfilter = CATX(' or ',dtfilter,CATX(' ',varname, 'EQ',
                       QUOTE("&outfiltr")));
         dtlst = CATX(' ',dtlst,varname);
         IF last THEN DO;
            CALL SYMPUTX(('dtfilter',STRIP(dtfilter));
            CALL SYMPUTX(('dtlst',STRIP(dtlst));
         END;
         RUN;

    /* get VISIT and VISITNUM if they exist. Assumption is if VISIT exists,
       then VISITNUM does as well. */

         %LET dsid=%SYSFUNC(OPEN(&&sds&i));
         %LET vn=%SYSFUNC(VARNUM(&dsid ,VISIT)); /* Test upper and lower */
         %LET vn2=%SYSFUNC(VARNUM(&dsid ,visit));/* case variable names */
         %IF &vn GT 0 OR &vn2 GT 0 %THEN
             %LET visvar=VISIT VISITNUM ;
         %ELSE
             %LET visvar= ;
         %LET rc=%SYSFUNC(CLOSE(&dsid));
```

```
    /* Find USUBJID records filtered by date (if provided) */

        DATA found;
        SET &&sds&i (KEEP=usubjid &visvar &dtlst);
        WHERE usubjid EQ "usubjid_prefix.&_subjid"
        %IF &dtfilter ne %THEN %DO;
           AND (&dtfilter)
        %END;
        ;   /* close WHERE statement */
        RUN;

       /* For Excel output, one dataset per worksheet, domain names as
          worksheet names */

        %IF &_outtyp EQ excel %THEN %DO;
            ODS EXCEL OPTIONS(SHEET_NAME="&&memnam&i");
        %END;
        %ELSE %DO;
            TITLE "Subject SDTM Search Results";
            TITLE2 "Domain: &&memnam&i";
        %END;
        PROC REPORT DATA=found;
        COLUMNS usubjid &visvar &dtlst;
        RUN;
   %END;
%END;

ODS &_outtyp CLOSE;

OPTIONS NOMPRINT NOMLOGIC NOSYMBOLGEN;
%MEND waldo;

/* ---------------------Sample Call---------------------
   USUBJID are represented as "STUDY05-141-xxx"
   Subject identifiers are zero-justified to 3 places
   Looking for USUBJID STUDY05-141-004
   Send output to /studydir/qc/checks/sdtm_waldo4.pdf (as a PDF file)
   Check this subject for any data on May 18, 2021
   Turn on internal debugging
-----------------------------------------------------------*/

%waldo(usubjid_prefix=STUDY05-141-,
       zero_just=3,
       subjid=4,
       libnam=sdtm,
       outdir=/studydir/qc/checks/,
       outtyp=pdf,
       outfiltr=2021-05-18,
       debugyn=Y
       );
```