PharmaSUG 2016 - Paper QT03

Quick and Efficient Way to Check the Transferred Data

Divyaja Padamati, Eliassen Group Inc., North Carolina.

ABSTRACT

Consistency, quality and timelines are the three milestones that every department strives to achieve. As SAS Programmers, we often have to work with data generated by another team or a competitor company. Checking the data for compliance with CDISC or company standards will be a major hurdle in latter cases. In this paper we shall discuss how data compliance can be achieved in a quick and efficient manner. The first step in this process will be checking for redundant data i.e., removing any variables with all missing data or an observation where all the variables are missing. Secondly, we need to make a quick check for the presence of required variables as per the required standards. Finally, we have to create a list of required variables which are absent/not in transferred data. By letting SAS do all the work for us, the result will be faster and efficient. This paper will offer an approach to achieving all these goals by a click of a button. The skill level needed is intermediate to advance.

INTRODUCTION

Improving quality and efficiency in clinical trial data transformation is a never ending process. In this macro, we check for the consistency of the transferred data. We will be checking for variables which are expected to be present in the transferred data but are not present. In order to achieve this list of variables we need to follow below steps,

1. Remove any unwanted/redundant data from the transferred data.
2. Check the variables from transferred data and source data.
3. Create a listing of variables which are not present in the transferred data.

We shall see how each task has been achieved individually first. The entire code with nested macro is present at the end of the paper.

MACRO INPUT PARAMETERS

INDT: Input dataset
INDS: Input dataset used to delete the redundant data.
TYPE: Type of variable to specify in the array definition.
OUTDT: Output dataset with list of variables.
OUTDS: Output dataset after all the modification has been done.
SOURCE: Location of source data.
NUM_CNT: Number of numeric variables.
CHAR_CNT: Number of character variables.
TRANSFER: Location of transferred data.

Step 1: Remove redundant data

To remove all the unnecessary data, we can use the below macro. First we shall create macro variables NUM_CNT and CHAR_CNT which have values of number of numeric and character variables respectively. We define an array for the numeric/character variables in the dataset, and also a flag array for which the dimensions will be same as numeric/character array and initial values will be set as 'Missing' assuming that all numeric/character variables have missing values. This flag array will be updated to "Not Missing" based on the non-missing values present for the numeric/character variables in the dataset. Once we reach the last observation of the input dataset, we check for the values in the flags array. If there are any variables whose corresponding flag value is still "Missing" as we first assigned, then we create a macro variable with the list of these variables. Later we use this macro variable and drop all the character/numeric variables which have missing values for all the observations.

```
data transfer_inds ;
        set &transfer.&inputdt. ; /* Main input dataset */
    run ;

data _NULL_;
        set transfer_inds (obs=1);
        array numarray {*}  _numeric_ ;
        array chararray (*) _char_ ;
/* Creating MV num_cnt for no. of numeric variables */
        call symput ('num_cnt', put(dim(numarray),best.));
/* Creating MV char_cnt for no. of character variables */
        call symput ('char_cnt', put(dim(chararray),best.));
    run;

    %let num_cnt = &num_cnt. ;
    %let char_cnt = &char_cnt. ;

/* Creating macro variables with all character/numeric variable names */

  proc sql ;
    select name into :char separated by ', '
    from dictionary.columns
    where strip(upcase(libname))='WORK'
    and strip(upcase(memname))='TRANSFER_INDS'
    and strip(lowcase(type))='char' ;

    select name into :num separated by ', '
    from dictionary.columns
    where  strip(upcase(libname))='WORK'
    and strip(upcase(memname))='TRANSFER_INDS'
    and strip(lowcase(type))='num' ;
quit ;

/* Deleting the observations where all the variables have missing values */

data transfer_inds ;
  set transfer_inds ;
  array numarray {*} _numeric_ ;
  array chararray (*) _char_ ;
  number_char_missing=cmiss(&char) ;
  number_num_missing=nmiss(&num) ;
  if number_char_missing=dim(chararray) and number_num_missing=dim(numarray) then
delete ;
run ;

    %macro del_vars (type=,inds=,outds=,flgcnt=) ;

    data &outds. ;
        set &inds. end=lastobser;
        length droplist $1000 ;
/* Defining an array for variables to be checked */
        array numarray {*}  &type. ;
/* Defining a flag array. */
        array flgarray {*} $20 flgarray1-flgarray&flgcnt. (&flgcnt. * 'Missing');
/* Updating the value of flag */
        do i=1 to dim(numarray);
            if missing(numarray[i])=0 then flgarray[i]='Not Missing';
        end;
/* Creating the droplist macro variable */
        if lastobser then do;
            do i=1 to dim(flgarray);
                if flgarray[i]='Missing' then droplist=trim(left(droplist)) || ' ' ||
trim(vname(numarray[i]));
            end;
            call symput('droplist', droplist);
        end;
       run;

    data &outds._1 ;
        set &outds. ;
        drop &droplist.  flgarray: droplist i ;
    run ;

    %mend del_vars ;

/* Executing the %del_vars macro to delete numeric variables */
    %del_vars (type=_numeric_,inds=transfer_inds,outds=final1,flgcnt=&num_cnt.) ;

/* As we already removed the numeric variables in the above macro execution, the
output of the above macro will be input to the second execution as we still need to
delete character variables with all missing data */
       %del_vars (type=_char_,inds=final1_1,outds=final2,flgcnt=&char_cnt.) ;
```

Now, in final2_1.sas7bdat we have data with variables which have non-missing data.

Step 2: Check the variables from transferred data and source data.

Next step, is to check for variables which are not present in the transferred data. In order to do this we shall make use of CONTENTS procedure and create data which contains the names of the available variables.

```
/*Getting the list of variables from source data and transferred data */

proc contents data = final2_1 out=transfer(keep=name type);
proc contents data = &SOURCE..&INDT. out=source(keep=name type);
```

Once we get the meta-data of the required datasets, we MERGE the transferred and source data to identify the variables which are not present or extra in transferred data.

Step 3: Create a listing of variables which are not present in the transferred data.

```
/*identifying the Variables which are present in source data but not in transferred
data and vice versa. */

    data &outdt. ;
          merge source(in=a) transfer(in=b) ;
          by name type ;
          if (a and not b) or (b and not a) ;
    run;
```

CONCLUSION

Quality check is a major task in any industry and working with transferred data is a very common scenario. This macro provides an extra layer of quality assurance in such cases. This macro can be used to lay ground work and give preliminary review of the data.

REFERENCES

SAS® 9.2 Macro Language: Reference. Cary, NC: SAS Institute Inc.
SAS® 9.2 Language Reference: Dictionary. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

Eliassen Group Biometrics and Data Solutions
1005 Slater Rd #100,
Durham, NC 27703

CONTACT INFORMATION

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

Name: Divyaja R Padamati
Enterprise: Eliassen Group Biometrics and Data Solutions
Address: 1005 Slater Rd #100
City, State ZIP: Durham, NC 27703
E-mail: dpadamati@eliassen.com
Web: egbiometrics.com

```
/* Purpose of the macro: To identify the non-missing variables which are present in
source data but not in transferred data.
Step 1. Delete the variables from transferred data where all the values are missing.
ie., a variable for which  all observations are missing.
Step 2. Create 2 new datasets with only the variable names and their types using
procedure contents.
Step 3. Merge the source data and transferred data by name and type to find any
unnecessary or extra data. */

%macro varcheck (inputdt=,outdt=,source=,transfer=);

    libname transfer "&transfer." access=readonly ;
    libname source "&source." access=readonly ;

/* Clearing the work library */

    proc datasets lib=work kill ;
    quit ;

/*Step 1. Deleting redundant data i.e., variables/observations with all missing data.
*/

    data transfer_inds ;
        set &source..&inputdt. ;
    run ;

    data _NULL_;
        set transfer_inds (obs=1);
        array numarray {*}  _numeric_ ;
        array chararray (*) _char_ ;
        call symput ('num_cnt', put(dim(numarray),best.));
        call symput ('char_cnt', put(dim(chararray),best.));
    run;
    %let num_cnt = &num_cnt. ;
    %let char_cnt = &char_cnt. ;

/* Creating macro variables with all character/numeric variable names */

  proc sql ;
    select name into :char separated by ', '
    from dictionary.columns
    where strip(upcase(libname))='WORK'
    and strip(upcase(memname))='TRANSFER_INDS'
    and strip(lowcase(type))=’char’ ;

    select name into :num separated by ', '
    from dictionary.columns
    where  strip(upcase(libname))='WORK'
    and strip(upcase(memname))='TRANSFER_INDS'
    and strip(lowcase(type))='num' ;
quit ;

/* Deleting the observations where all the variables have missing values */

data transfer_inds ;
  set transfer_inds ;
  array numarray {*} _numeric_ ;
  array chararray (*) _char_ ;
  number_char_missing=cmiss(&char) ;
  number_num_missing=nmiss(&num) ;
  if number_char_missing=dim(chararray) and number_num_missing=dim(numarray) then
delete ;
run ;
```

```
    %macro del_vars (type=,inds=,outds=,flgcnt=) ;

    data &outds. ;
        set &inds. end=lastobser;
        length droplist $1000 ;

        array numarray {*}  &type. ;
        array flgarray {*} $20 flgarray1-flgarray&flgcnt. (&flgcnt. * 'Missing');

        do i=1 to dim(numarray);
            if missing(numarray[i])=0 then flgarray[i]='Not Missing';
        end;

        if lastobser then do;
            do i=1 to dim(flgarray);
                if flgarray[i]='Missing' then droplist=trim(left(droplist)) || ' ' ||
trim(vname(numarray[i]));
            end;
            call symput('droplist', droplist);
        end;
    run;

    data &outds._1 ;
        set &outds. ;
        drop &droplist.  flgarray: droplist i ;
    run ;

    %mend del_vars ;

    %del_vars (type=_numeric_,inds=transfer_inds,outds=final1,flgcnt=&num_cnt.) ;
    %del_vars (type=_char_,inds=final1_1,outds=final2,flgcnt=&char_cnt.) ;

/*Step 2. Create 2 new datasets with only the variable names and their types using
procedure contents.*/

    proc contents data=final2_1 out=transfer(keep=name type);
    proc contents data=&source..&indt. out=source(keep=name type) ;

/*Step 3. Identifying the variables which are present in source data but not in
transferred data and vice versa.

    data &outdt. ;
          merge source(in=a) transfer(in=b) ;
          by name type ;
          if (a and not b) or (b and not a) ;
    run;

%mend varcheck ;
```

SAMPLE CALL:

```
%VARCHECK (indt=ae, outdt=ae_varchk, source=%str(/stats/study/data),
transfer=%str(/stats/study/transfer/data) ) ;
```