Paper CT11

Automation of quality checks for ongoing CSR data

Savithri Jajam, Covance, Inc, USA
Lavanya Peddibhotla, Covance, Inc, USA

ABSTRACT;

This paper will discuss about utilities to reduce the manual testing for ongoing studies and compare or contrast the changes between two sister studies. First utility will describe how to compare two versions of datasets. They could be source, SDTM, ADaM or TLF output datasets. This procedure is useful to check if previously reported data issues have been resolved or not. Second utility will discuss compatibility of the spec between two sister studies, which will minimize the effort of developing and validating specification for the new study. And finally, third utility will discuss about how to dynamically assign values to ADaM variables when the values are not available in input data, so manual checks are not needed.

INTRODUCTION:

Utilities described below were developed to automate and minimize manual check of source data integrity in an ongoing CSR study and to compare similarities or clearly identify the differences between two sister studies/analysis. These are some of the most common situations in CSR studies so let's dive into the details of the utilities and some scenarios where these utilities were utilized:

Example 1: Resolution of data issues and tracking changes in the data during weekly or monthly data refreshes (i.e. updated or deleted records)

Example 2: Ensure that an update or minor correction in DM before final production run has not impacted any other datasets. In this case we need to perform validation between old and new version of all the SDTM datasets To accomplish this, we can use PROC COMPARE for individual dataset, but it is a laborious and time-consuming process to go over the PROC COMPARE results individually and then summarize the differences between the two libraries.

Example 3: Independent validation errors out or log has warning notes when proc compare is used with incomplete data. Proc compare cannot be excluded from the code to ensure completion of accurate validation of the dataset/output.

In this paper, three different utilities are discussed to minimize manual checks with simple programming so anyone can easily adapt it based on their requirement.

Utility1:

compare_utility.sas: This utility is used to compare the data between two different libraries or two data transfers. This macro works for independent validation of a dataset by substituting the code with Proc compare. Compare goes through only when data is available.

As the macro program is proprietary, full code is not shared but each part of the code from macro is discussed with examples to provide basis of the functionality.

Compare_utility.sas macro compares the dataset(s) in given two libraries and produces report(s) to help summarize the differences. This macro can be used in two different levels.

1) Study level.
2) Metadata Level

Study Level: This macro compares two different libraries and lists out the commonalities or the differences between the two, at a dataset level as well as the observation level.

Metadata level: This macro also provides details about the total number of observations and variables present in both datasets by listing the uncommon variables with attribute differences.

How macro works?

Compare_utility.sas macro will check for

1) Number of datasets uncommon
2) Number of variables differences
3) Number of records different
4) Attributes difference

And creates two different reports for each dataset.

Requirements:

To achieve this, two libnames should be available and type of the compare should be selected. If **type=lib** then it will compare two libraries but if **type=dsn** then it will compare and provide **PROC COMPARE** report for each dataset within the library.

Parameters and sample macro call:

```
% compare_utility (type=lib ,
                    base=C:/Users/Desktop/csr/spec,
                     comp=C:/Users/Desktop/ia/spec);
```

Step by step break down of the code is as below.

Step 1: Call the two libraries and follow below steps to find out list of datasets available in each folder with total number of records in each dataset from **SASHELP.VTABLE.**

```
Proc sql;
create table baseobs as
select distinct memname, nobs as baseobs from sashelp.vtable
where libname eq “BASE” and memtype=”DATA” order by memname;

create table compareobs as
select distinct memname, nobs as compareobs from sashelp.vtable
where libname eq “COMP” and memtype=”DATA” order by memname;
quit;
```

Step 2: Merge base and compare datasets to check for uncommon datasets or unequal observations.

```
data dsn;
 merge baseobs (in=a) compareobs (in=b);
 by memname;
 length dataset $100. obsdiff $200;

if a=b then dataset="Base and Compare are Not Missing";
 else if a and not b then dataset="Base is Not Missing: Compare is Missing";
 else if not a and b then dataset="Base is Missing : Compare is Not Missing";

if dataset^="Base and Compare are Not Missing";

 if baseobs ne . and compareobs ne . and baseobs ne compareobs then  obsdiff = Obervation Difference Between
Base and Compare is ("|| strip(put(abs(baseobs - compareobs),8.))||" )";
else if baseobs = . and compareobs ne . then obsdiff = "Base Dataset is Missing and Compare dataset is Not
Missing ("|| strip(put(compareobs,8.))||" )";
 else IF compareobs = . and baseobs ne . then obsdiff = Base Dataset is Not Missing and Compare dataset is
Missing (")|| strip(put(baseobs,8.))||" )";
 run;
```

Step 3: Create macro variable to select each dataset.

```
Proc sql noprint;
 Select distinct(memname), count(distinct(memname)) into  :ds_list separated BY '~', :ds_cnt
 from dsn  order by memname;
quit;;
%put &ds_cnt &ds_list;
```

Step 4: Create two different datasets with attributes to compare.

```
Proc sql;
Create table basevar as
Select memname, name, length as blength, label as blabel, format as bformat, informat as binformat, xtype as
btype
from sashelp.vcolumn Where libname=”BASE” and memtype=”DATA” order by memname;
```

```
Create table comparevar as
Select memname, name, length as clength, label as clabel, format as cformat, informat as cinformat, xtype as
ctype
from sashelp.vcolumn Where libname=”COMP” and memtype=”DATA” order by memname;
quit;
```

Step 5: Derive variable to display in the report with attributes differences.

```
data variable;
merge basevar(in=a)  comparevar(in=b );
 length variable type len lable format informat $100;
by memname name;
if a and b then variable="Variable is present in Base and Compare";
else if a and not b then variable="Variable is Present in Base But not Present in Compare";
else if not a and b then variable="Variable is not Present in Base and but Present in Compare";
if blable="" then blable="Missing";
if clable="" then clable="Missing";
if blable=clable then lable="Variable Label Match";
else lable="label Do Not Match : Base Lable="||strip(blable)||", Compare Lable="||strip(clable);
run;
```

Step 6: Write two reports, one report with dataset from step 2 and a second report with dataset from step 5.

Procedure 2: Spec compatibility with reference study.

Writing programs for SDTM or ADaM takes time and attention. However, if we can check the compatibility and similarity with reference study then we can save time as well improve quality. To achieve this **spec_utility.sas** macro will provide a detailed report by checking list of uncommon datasets or list of uncommon variables or list of variable derivation if it is differing with the sister study.

How macro works?

Macro will look for list of datasets/domains and will import each sheet into SAS dataset. After imported **spec_utility.sas** macro will check for

1) Number of datasets uncommon
2) Number of variable uncommon
3) Attributes difference
4) Derivation differences

And creates three different reports for each dataset.

Requirements: To achieve this, specification should have a sheet with list of all datasets/domains.

Parameters and sample macro call:

```
%spec_utility(base=C:/Users/Desktop/csr/spec,
                    comp=C:/Users/Desktop/ia/spec);
```

Step by step break down of the code is as below.

Step 1: Import list of domains sheet from specification and create macro variable to use for each sheet.

```
proc import datafile= "&base./csr-adam-specs.xlsx"  out=base.domain dbms=xlsx replace; sheet='Domains';
run;

proc sql noprint;
select distinct(domain), count(distinct(domain)) into :dm_list separated by “~:, :dm_cnt
from base.domain order by domain:
quit;
%put &dm_cnt &dm_list;
```

Step 2: Use a do loop to import all sheets listed in domains sheet and adjust variable names as needed.

```
%macro doloop();
%do i = 1 %to &dm_cnt;
%let dm = %scan (&dm_list., &I., ~);
proc import datafile= "&base./csr-adam-specs.xlsx"   out=base.&dm(where=(domain ne "")keep=domain name
label type length format  definition
                                        rename=(variable_name=name
                                        variable_label=label
                                        variable_type=type
                                        variable_length=length
                                        sas_format=fomat)) dbms=xlsx replace;
sheet="&dm.";
run;
%end;
%mend doloop;
%doloop();
```

Step 3: Follow same as in step 1 and 2 for compare folder.
Step 4: Finally call **compare_utility.sas** macro.

Reports:

As show in Picture 1 and Picture 2, spec**_utility** and **compare_utility** macros will be creating two reports. But **spec_utility** macro creates one extra report with derivation difference for each variable in the dataset as in Picture 3.

Picture 1: Number of datasets uncommon

Dataset Name	Dataset Difference
ADEX	Base is Not Missing : Compare is Missing
ADTTE	Base is Missing : Compare is Not Missing

Picture 2: Variables with different attributes and list of variable uncommon.

Variable Name	Variable	Variable Lable	Variable Length	Variable Type	Variable Format
AP01EDT	Variable is present in Base and Compare	Variable Label Match	Variable Length is Same	Variable Type is Same	Variable Format is Same
AP01SDTM	Variable is present in Base and Compare	Variable Label Match	Variable Length is Same	Variable Type is Same	Variable Format is Same
AP01STM	Variable is present in Base and Compare	Variable Label Match	Variable Length is Same	Variable Type is Same	Variable Format is Same
AP01STMF	Variable is not Present in Base and but Present in Compare	label Do Not Match : Base Lable=Missing, Compare Lable=Period 01 Start Time Imput. Flag	Variable Length is Not Same: Base Length=., Compare Length=1	Variable Type is Not Same : Base Type=Miss, Compare Type=Char	Variable Format is Same

Picture 3: Each variable Derivation Differences

Study :: Test

Data set	Variable Name	Base Derivation	Comp Derivation
ADSL	AP01EDT	Convert date part of max(SDTM.SV.SVENDTC) to numeric version where SDTM.SV.EPOCH='Randomized Period-Induction Dose' or 'Randomized Period-Maintenance Dose'.If DISCRFL = 'Y', set to be DISCRDT.	Convert date part of max(SDTM.SV.SVENDTC) to numeric version where SDTM.SV.EPOCH='Randomized Period-Induction Dose' or 'Randomized Period-Maintenance Dose'.If DISCRFL = 'Y', set to be DISC01DT.AP01EDT=min(AP01EDT, TRTEDT+56).
ADSL	AP01SDTM	AP01SDT+AP01STM	AP01SDT+AP01STMImpute the time part to be 00:00 if it is missing.
ADSL	AP01STM	Convert time part of min(SDTM.EX.EXSTDTC) to numeric version where SDTM.EX.EPOCH ='Randomized Period-Induction Dose' or 'Randomized Period-Maintenance Dose'	Convert time part of min(SDTM.EX.EXSTDTC) to numeric version where SDTM.EX.EPOCH ='Randomized Period-Induction Dose' or 'Randomized Period-Maintenance Dose' Impute the time part to be 00:00 if it is missing.
ADSL	AP01STMF		Set to be 'H' if time part of AP01STM is imputed.
ADSL	AP02EDT	Convert date part of max(SDTM.SV.SVENDTC) to numeric version where SDTM.SV.EPOCH='Extension Period-Loading Dose' or 'Extension Period-Maintenance Dose'.If DISCFL = 'Y', set to be DISCDT.	Convert date part of max(SDTM.SV.SVENDTC) where SV.SVENDTC<FCDAT. If FCDAT is missing, convert date part of max(SDTM.SV.SVENDTC) where SDTM.SV.EPOCH='Extension Period-Loading Dose' or 'Extension Period-Maintenance Dose'.

Limitations:

Even though these macros provide the compatibility between two different specification or two different folders, still they have some limitations.

1) In **spec_utiliyu.sas** macro, if there is any type or an extra space difference in the derivation still, it will be shown in the report as difference.

2) These macros proved 2 or 3 reports for each dataset which may take some time to investigate each report to check for differences.

Procedure 3: Handling future variables.

While working on ongoing studies, programmers often work with incomplete data. In that case, supplemental dataset might not have any observations but any use of those variable in the ADaM or outputs programming should be dynamic to ensure that these programs don't end with errors.

Example:

As shown in Picture 4, when Race2 is missing and it is been used in RACEALL derivation. Log will have uninitialized note.

To achieve this there are many different methods, but two easy methods are shown below.

Method 1:

```
data dm;
length usubjid race $50;
usubjid="001"; race="WHITE"; output;
usubjid="002"; race="MULTIPLE"; output;
run;

data supp;
length usubjid $50 race1  $200;
usubjid="002";RACE1="ASIAN";output;
run;

data dm1;
merge dm supp;
by usubjid;
length raceall $50 race1 race2 $200;
race1=race1;
race2=race2;
if race="MULTIPLE" and race1 ne "" and race2 ne "" then raceall=strip(race1)||"\"||strip(race2);
else if race="MULTIPLE" and race1 ne "" and race2="" then raceall=strip(race1);
else raceall=strip(race);
run;
```

Method 2:

```
data dm;
set dm;
length race1 race2 $200;
race1="";
race2="";
run;

data dm1;
merge dm supp;
by usubjid;
length raceall $50 ;
if race="MULTIPLE" and race1 ne "" and race2 ne "" then raceall=strip(race1)||"\"||strip(race2);
else if race="MULTIPLE" and race1 ne "" and race2="" then raceall=strip(race1);
else raceall=strip(race);
run;
```

Picture 4:

usubjid	race
001	WHITE
002	MULTIPLE

usubjid	race1
002	ASIAN

usubjid	race	race1	race2	raceall
001	WHITE			WHITE
002	MULTIPLE	ASIAN		ASIAN

```
1760
1761        data dm1;
1762        merge dm supp;
1763        by usubjid;
1764        length raceall $50;
1765        if race="MULTIPLE" and race1 ne "" and race2 ne "" then
1765      ! raceall=strip(race1)||"\"||strip(race2);
1766        else if race="MULTIPLE" and race1 ne "" and race2="" then raceall=strip(r
1767        else raceall=strip(race);
1768        run;

NOTE: Variable race2 is uninitialized.
NOTE: There were 2 observations read from the data set WORK.DM.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

CONCLUSION:

New data extractions and referring sister studies are common during a clinical trial, and it is important to discover changes to the existing, structure, or attributes of the data prior to processing programs. The above macro can assist the programmer in making these comparisons. As the procedures are simple and portable, small changes can be made to address specific issues relevant to the programmer or the study. This program has the potential to detect major issues early on and save the programmer and company valuable time.

References:

1 Kavitha Madduri (2012). PharmaSUG 2012 - Paper AD24, Let's compare two SAS® libraries! http://www.pharmasug.org/proceedings/2012/AD/PharmaSUG-2012-AD24.pdf

2 Michael Stackhouse, Franklin E. Menius Jr (2018). PharmaSUG 2018 - Paper AD-13, An Automated Macro to Compare Data. https://www.lexjansen.com/pharmasug/2018/AD/PharmaSUG-2018-AD13.pdf

Acknowledgments:

Author would like to thank the employer Covance,Inc. for the approval to present this paper at this PHUSE. Special thanks to Vamshi Matta for her valuable feedback on this paper.

CONTACT INFORMATION:

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

Savithri Jajam Covance, Inc.
savithri.jajam@covance.com

Lavanya Peddibhotla Covance, Inc.
lavanya.peddibhotla@covance.com

-0-