

AUTOMATION OF QUALITY CHECKS FOR ONGOING CSR DATA

March 11th, 2020

Copyright © 2018 Covance. All Rights Reserved

Public



COVANCE
SOLUTIONS MADE REAL®

Savithri Jajam biography

Savithri Jajam holds a Master's degree from Western Illinois University where she studied chemistry.

She is currently working at Covance as Principal statistical programmer.

She has extensive CDISC experience, working with both SDTM and ADaM standards.

About Covance

Covance Inc., the drug development business of Laboratory Corporation of America Holdings (LabCorp), is the world's most comprehensive drug development company.

We are dedicated to advancing healthcare and delivering Solutions Made Real® by providing high quality nonclinical, clinical, commercialization and informatics services to pharmaceutical and biotechnology companies to help increase the speed, precision and effectiveness of drug development.

Covance is currently working on about half of all active clinical trials globally and has been involved in all top 50 best-selling drugs on the market.

Introduction

Utilities described in this paper were developed for some of the most common situations in CSR studies :

- ▶ To automate and minimize manual check of source data integrity in an ongoing CSR study
- ▶ To compare similarities or clearly identify the differences between two sister studies/analysis.

Introduction Cont.

Some of 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.
- ▶ **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.

List of Utilities:

Below are the 3 utilities we will be talking and macro calls:

1) Name: compare_utility.sas

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

2) spec_utility.sas

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

3) Handling future variables.

Utility 1: compare_utility.sas:

Functionality:

- ▶ For each analysis label on right side.
- ▶ 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.
- ▶ 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.

Utility 1: compare_utility.sas: Cont.

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.

Utility 1: How macro works and Requirements

How macro works?

Compare_utility.sas macro will check for below list of differences and creates two different reports for each dataset.

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

Requirements:

Two libnames should be available and type of the compare should be selected.

Utility 1: 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;
```

Utility 1: 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 ("|| s
trip(put(baseobs,8.))||" );

run;
```

Utility 1: 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;
```

Utility 1: Step 4

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;

Utility 1: Step 5 &6

Derive variable to display in the report with attributes differences and Write two reports, one report with dataset from step 2 and a second report with dataset from step 5.

```
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;
```

Utility 2: spec_utility.sas:

Functionality: 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 below differences and creates three different reports for each dataset

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

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

Utility 2: 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;
```

Utility 2: 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., &l., ~);  
  
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();
```

Utility 2: Step 3 &4

Step 3: Follow same as in step 1 and 2 for compare folder.

Step 4: Finally call **compare_utility.sas macro**.

Reports:

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 Label	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 Input. 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

Reports Cont.

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:

- 1) Even though these macros provide the compatibility between two different specification or two different folders, still they have some limitations.
- 2) 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.
- 3) These macros proved 2 or 3 reports for each dataset which may take some time to investigate each report to check for differences.

Utility 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.

Utility 3: Handling future variables Cont.

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 dml;
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 dml;
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;
```

Reports.

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 dml;
1762   merge dm supp;
1763   by usubjid;
1764   length raceall $50;
1765   if race="MULTIPLE" and race1 ne "" and race2 ne "" then
1766     ! raceall=strip(race1)||"\"||strip(race2);
1767   else if race="MULTIPLE" and race1 ne "" and race2="" then raceall=strip(r
1768   else raceall=strip(race);
1769   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 and save the programmer and company valuable time.

Questions?

Savithri Jajam
Covance, Inc.
4000 Centergreen Way # 300
Cary, NC 27513

Savithri Jajam: savithri.jajam@covance.com

Lavanya Peddibhotla: lavanya.peddibhotla@covance.com



Covance Inc., headquartered in Princeton, NJ, USA, is the drug development business of Laboratory Corporation of America Holdings (LabCorp). COVANCE is a registered trademark and the marketing name for Covance Inc. and its subsidiaries around the world.