PharmaSUG 2022 - Paper QT-199

Monitoring SDTM Compliance in Data Transfers from CROs

Sunil Gupta, Principal SAS Programmer, Experis

ABSTRACT

Typically, CROs transfer SDTMs to sponsors on a regular basis and sponsors accept the new SDTMs. Instead of just blindly accepting SDTMs, it is better to monitor SDTM compliance after each SDTM transfer. The benefits include higher SDTM compliance, higher data quality and error-free programs downstream. With each data transfer, smarter sponsors are proactive to build and compare metadata between the specifications, original and new SDTMs. This method assures more consistency in SDTM data structure as well as study level demographic and safety statistics. Sponsors can get early alert of any major changes in study baselines. This presentation shows how to create standard and custom SDTM metadata as well as methods to identify and track SDTM metadata differences.

OUTLINE

The outline reviews potential data issues, compliance gate testing, standard and custom metadata and SAS examples.

- Potential Issues with New Data
- Data Transfer Compliance Gate
- Standard Metadata
- Custom Metadata
- Snipit SAS Examples

POTENTIAL ISSUES WITH NEW DATA

At least once, sponsors can expect issues with new data transfers. The types of issues include new baseline, structure, duplicates, holes, missing and invalid values. In general, during the collection of clinical data and internal dataset mergers, data issues may inadvertently occur.

DATA TRANSFER COMPLIANCE GATE

Compliance acceptance testing is similar to database lock acceptance testing. Sponsors should decide to accept or reject new data transfers based on defined domain cross checks. If sponsors reject new data transfers then they can give advance notice to CROs to correct the data. Unexpected variable structure changes include new, drop, rename, length, label and variable type. Unexpected value changes may also occur while site query updates are expected. New data is also expected.

STANDARD METADATA

Patient data metadata and statistics should also be compared before accepting data refresh. Standard SDTM metadata includes: Count, Names, Size, # Variables, # of OBS, # unique USUBJID, Date Stamps, Required, Duplicates, Variable (Type, Label, Order, New, Missing) and Codelist (New, Missing, Invalid, Case-sensitive, Blanks).

SAS offers many tools to create standard metadata for all SDTMs and variables.

- SASHELP.VTABLE
  - Confirm # of datasets, observations and datetime stamps
  - Confirm required datasets and variables
- SASHELP.VCOLUMN
  - Confirm # and type of variables and attributes
  - Confirm order of variables
- Macro variables
  - Create to store sort and store information across records
  - Compare macro variable values

CUSTOM METADATA

Study Customization metadata includes: DM (Min/Max Age, # Males/# Females, #/Min/Max Random Dates, # Completed Study) and AE (# SAFFL, # Deaths). This custom metadata represents important study milestones, primary and secondary endpoints.

Comparing the base and updated metadata empowers sponsors to identify in advance any changes to key study milestones before refreshing ADaMs, tables, lists and graphs. This type of check fits well within SAS Life Science Analytical Framework (LSAF). The study team can be communicated of issues to be better informed.

Proc SQL is ideal to help build custom metadata since there are many options for subsetting, grouping and variable creation. Each intermediate dataset can be appended or merged. Sponsors can create and build data transfer metadata.

Data Transfer Metadata

- Confirm required files, datasets and variables
- Confirm total number of datasets and datetime stamps
- Confirm total number of observations and patients
- Confirm total number and type of variables and attributes
- Confirm required non-missing variables
- Confirm no duplicate records
- Confirm order of variables
- Confirm new variable lengths are not less than maximum variable lengths
- Confirm complete codelist dictionary, ex. lab units
- Monitor descriptive statistics on key categorical and continuous variables
- Compare and contrast previous datasets and attributes to monitor increase or decrease in datasets, records or variables

Examples of metadata application SAS examples include comparing two SDTM metadata vitals, attributes and values, confirm data quality (non-missing, zero or duplicate records and complete codelists) as well as cross-referencing raw data, SDTMs, ADaMs with data transfer specifications.

SNIPIT SAS EXAMPLES

Samples illustrate the variety of applications for monitoring data transfers from CROs.

- Example 1. Proc SQL Outer Union Corr appends 13 metadata variables
- Example 2. Compare Two SDTM Metadata Attributes
- Example 3. SDTM Data Discrepancy Report

- Example 4. Confirm Non-Missing Values in Required Variables
- Example 5. Confirm SDTM is not Zero Records
- Example 6. Confirm Non-Duplicate SDTM Records
- Example 7. Cross Reference SDTM Data Transfer Specs

EXAMPLE 1. PROC SQL OUTER UNION CORR APPENDS 13 METADATA VARIABLES

1. USUBJID patient count
2. SAFFL patient count
3. USUBJID duplicate records check
4. ENRLDT minimum and maximum dates

5 – 7. ARM Codelist

8 – 12. APHASE Codelist

13. Cohort ENRLDT minimum date

Obs	adsl_vrs	obs	num1	char1	date1	date2
1	USUBJID	187	187		Counts	
2	SAFFL	.	146			
3	USUBJID Duplicate Records	.	.	No Duplicate Records	No Duplicates	
4	ENRLDT (Min, Max)	.	.		17APR2015	25SEP2017
5	ARM	.	146	KTE-C19		
6	ARM	.	15	Not Assigned	Unique values	
7	ARM	.	26	Screen Failure		
8	APHASE	.	11	Phase1		
9	APHASE	.	21	Phase2		
10	APHASE	.	81	Phase2 Cohort1	Unique values	
11	APHASE	.	32	Phase2 Cohort2		
12	APHASE	.	42	Phase2 Cohort3		
13	101-006-003 Cohort/Enrollment Date	.	1	Phase2 Cohort3	31OCT2016	

EXAMPLE 2. COMPARE TWO SDTM METADATA ATTRIBUTES

```
proc sql;
  create table dc as
  select name, memname, type, length, label, format, count(name) as dupcnt
  from sashelp.vcolumn where upcase(libname) in ('WORK') group by name
having calculated dupcnt > 1; quit;
proc sort datadata=dc;  by name memname; run;
data dc1;
length lag_type $8. lag_length 8. lag_label $8. lag_format $8. flag $1.;
set dc;
by name memname;
lag_type=lag(type); lag_length=lag(length); lag_label=lag(label);
lag_format=lag(format);
if first.name then do;
  lag_type=type; lag_length=length; lag_label=label; lag_format=format;
end;
  if last.name and lag_type^=: type or lag_length^=: length or lag_label^=:
label or lag_format^=: format then flag='Y';
  drop lag_type lag_length lag_label lag_format; run;
```

Metadata Attributes

File Attributes
- NOBS, NVARS, Size and Date

Variable Attributes
- Type, Length, Label, Format

Discrepancy Report

Proc SQL Full Outer Join
- All records from both datasets
- Variable name match
- Display all values

Differneces in common vars

Obs	flag	name	memname	type	length	label	format	dupcnt
1		Age	CLASSA	num	8			2
2		Age	CLASSB	num	8			2
3		Height	CLASSA	num	8			2
4		Height	CLASSB	num	8			2
5		Name	CLASSA	char	8			2
6		Name	CLASSB	char	8			2
7		Sex	CLASSA	char	1			2
8	Y	Sex	CLASSB	num	8			2
9		Weight	CLASSA	num	8			2
10		Weight	CLASSB	num	8			2
11		format	DC	char	49	Column Format		2
12		format	DC1	char	49	Column Format		2
13		label	DC	char	256	Column Label		2
14		label	DC1	char	256	Column Label		2
15		length	DC	num	8	Column Length		2
16		length	DC1	num	8	Column Length		2
17		memname	DC	char	32	Member Name		2
18		memname	DC1	char	32	Member Name		2
19		name	DC	char	32	Column Name		3
20		name	DC1	char	32	Column Name		3
21	Y	name	_PRODSAVAIL	char	8			3
22		type	DC	char	4	Column Type		2
23		type	DC1	char	4	Column Type		2

Two records from Proc SQL – CLASSA, CLASSB

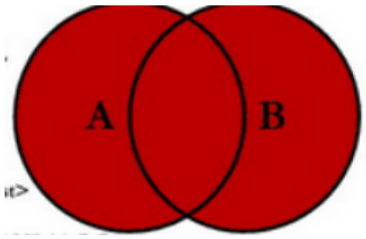
LAG() compares two records to flag differences

Dupcnt > 1 indicates common variables between SDTMs

Flag = 'Y' indicates attribute differences in common variables

EXAMPLE 3. SDTM DATA DISCREPANCY REPORT

```
proc sql;
create table dc as
select coalesce(a.name, b.name) as name
, ifc(a.name > '', 'Y', 'N') as a_usubjid label="a_name"
, ifc(b.name > '', 'Y', 'N') as b_usubjid label "b_name"
, a.age as a_age, b.age as b_age
, a.sex as a_sex, b.sex as b_sex
from sashelp.class as a
full outer join class as b on strip(a.name)=strip(b.name);
quit;
```



Source A and B may have different records which need to be reviewed to determine which records are correct.

Full Outer Join joins records from both SDTMs

EXAMPLE 4. CONFIRM NON-MISSING VALUES IN REQUIRED VARIABLES

```
proc sql;
create table dc as
select unique 'ADAE' as domain,
  compbl("SUBJECTID / ASTDT") as key_grp_vr
  length=50 , case when nmiss(ASTDT) >= 1 then
 'FAIL: Required Variable has at least one Missing Value'
  else 'PASS: Required Variable Does Not have any
 Missing Values' end as dc_rslt label="Data Check
 Results For:" length=200 from adae;
quit;
```

Page Break

Obs	domain	key_grp_vr	dc_rslt
1	ADSL	SUBJECTID / AGE	PASS: Required Variable Does Not have any Missing Values

Page Break

Obs	domain	key_grp_vr	dc_rslt
1	ADAE	SUBJECTID / AEDECOD	PASS: Required Variable Does Not have any Missing Values

Page Break

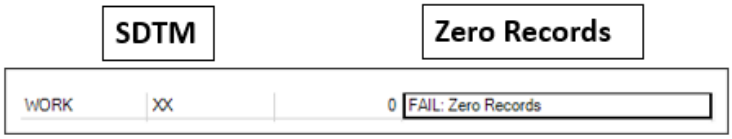
Obs	domain	key_grp_vr	dc_rslt
1	ADAE	SUBJECTID / AEDECOD	PASS: Required Variable Does Not have Missing Values

EXAMPLE 5. CONFIRM SDTM IS NOT ZERO RECORDS

```
* Condition test to show zero records check works for WORK directory;
proc sql;
create table xx like sashelp.class;
```

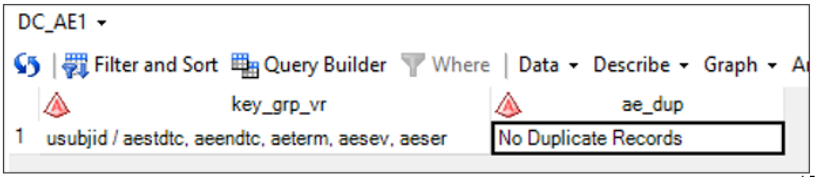
```
run;

proc sql;
create table dc as
select unique libname, memname, nobs
  , case when nobs = 0 then 'FAIL: Zero Records'
  else 'PASS: Records Exist' end as dc_rslt label="Data Check Results For:"
length=75
  from dictionary.tables where libname in ('SDTM' 'ADAM' 'WORK');
quit;
```



EXAMPLE 6. CONFIRM NON-DUPLICATE SDTM RECORDS

```
proc sql;
create table dc_ae1 as
select unique
  compbl("usubjid / aestdtc, aeendtc, aeterm, aesev, aeser")
  as key_grp_vr length=50
  , case when count(usubjid) = 1 then 'PASS; Duplicate Records
  do Not Exist'
  else 'FAIL; Duplicate Records Exist' end as ae_dup label="Work:
 AE duplicate records" length=75
 from work.ae group by usubjid, aestdtc, aeendtc, aeterm, aesev,
 aeser;
quit;
```



EXAMPLE 7. CROSS REFERENCE SDTM DATA TRANSFER SPECS

```
proc sql;
create table dc as
select unique
count(unique USUBJID) as tu_ptc label="USUBJID count"
, b.*, c.*, d.*, e.*, f.*, g.*
from sdtm.tu as a,
(select count(unique USUBJID) as tr_ptc from sdtm.tr) as b,
(select count(unique USUBJID) as rs_pct from sdtm.rs)  as c,
(select count(unique USUBJID) as tu_tr_rs_pct from tu_tr_rs)  as d,
(select unique input(scan(put(crdate, DATETIME16.), 1, ':'), date7.)
format=date9. as tu_stamp from sashelp.vtable
  where upcase(libname)="SDTM" and upcase(memname)=upcase("TU")) as e,
```

```
(select unique input(scan(put(crdate, DATETIME16.), 1, ':'), date7.)
format=date9. as tr_stamp from sashelp.vtable
  where upcase(libname)="SDTM" and upcase(memname)=upcase("TR")) as f,
(select unique input(scan(put(crdate, DATETIME16.), 1, ':'), date7.)
format=date9. as rs_stamp from sashelp.vtable
  where upcase(libname)="SDTM" and upcase(memname)=upcase("RS")) as g;
quit;
```

	1	2	3	4	5	6	7
	tu_ptc	tr_ptc	rs_pct	tu_tr_rs_pct	tu_stamp	tr_stamp	rs_stamp
1	164	164	147	164	09JUL2018	09JUL2018	09JUL2018

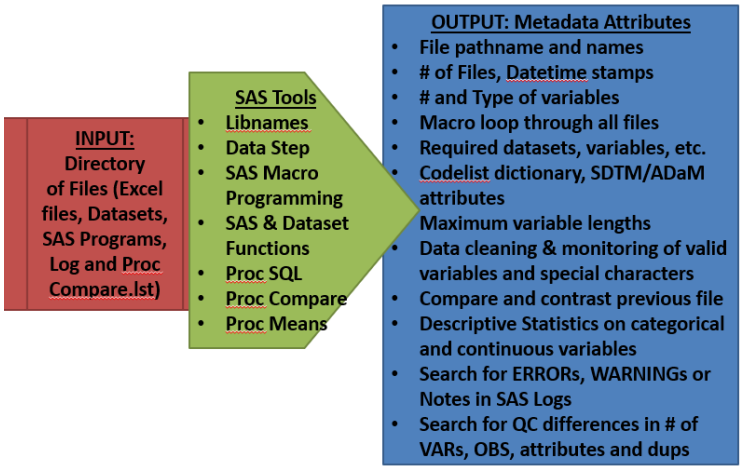
- Use Proc SQL to create and merge all 7 intermediate datasets
- Each dataset is an independent SDTM and metadata

1. TU Patient Count
2. TR Patient Count – same as TR but larger than RS
3. RS Patient Count – lower than TU and TR, could be due to lag time
4. TU_TR_RS Patient Count – unique patient count across TU, TR and RS
5. TU Date Stamp
6. TR Date Stamp
7. RS Date Stamp – same date as TU and TR

SUMMARY

In summary, proactive sponsors build systems to continuously check new data transfers for all of their studies. These sponsors are better informed of unexpected changes in data structure and key metadata and so can better manage updates needed to meet study milestones. Proc SQL plays a key role to help build standard and custom metadata with minimum maintenance. Over time, SAS can extract intelligence information from metadata and macro processing.

- Identify Controls on New Data
- Create Data Transfer Compliance Gate
- Leverage Proc SQL to apply Standard Metadata
- Leverage Proc SQL to build Custom Metadata



CONTACT INFORMATION

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

Sunil Gupta
Experis
Sunil.Gupta@Experis.com

Sunil Gupta is a Principal SAS Programmer at Experis. Sunil is also an international speaker, best-selling author of five SAS books, and a global SAS and CDISC trainer. Sunil has over twenty-five years of experience in the pharmaceutical industry. Most recently, Sunil is teaching a CDISC online class at the University of California at San Diego and classes on Data Science using SAS at UCLA and UCSD Extensions. In 2019, Sunil published his fifth book, Clinical Data Quality Checks for CDISC Compliance Using SAS and in 2011, Sunil launched his unique SAS mentoring blog, SASSavvy.com, for smarter SAS searches. Sunil has an MS in Bioengineering from Clemson University and a BS in Applied Mathematics from the College of Charleston.