phuse·connect
connect·share·advance
10th–13th November
RAI Amsterdam
The Netherlands
EU 2019
The Clinical Data Science Conference

Paper DH02

SDTM Validation Tool

Anamaria Calai, Cmed, Timisoara, Romania

ABSTRACT

This presentation will describe a validation tool to expand the capabilities of Pinnacle's checks to increase the quality of SDTM deliverables. The tool uses individual programmed checks for each domain and implements cross-domain checks to provide additional value. Limitations of the checks performed by Pinnacle and individual client interpretation of SDTM standards is therefore countered by applying an additional validation to secure the highest quality deliverables.

INTRODUCTION

The overall scope of the Validation Tool is an in-house, SAS macro to detect SDTM inconsistencies versus RAW data and CDISC standards, and to minimize the time and effort required for QC/validation. The macro does not only check the SDTM compliance but also verifies the SDTM datasets versus RAW data.

This paper describes examples of three types of validation of the SDTM datasets:

- **Mapping validation** using the annotated RAW structure and PROC FREQ
- **Timing domains** validation in SE & SV
- **Study specific** conversions in LB validation

These validation checks expand the power of the existing *Community (free) version of the Pinnacle 21 Validator* in respect to sponsor specific requirements, cross-checks between domains and versus RAW data.

SDTM PROCESS FLOW

A high-level flow of a standard SDTM process is illustrated in Figure 1. As shown below, the validation of the SDTM datasets occurs once programming is completed. Any potential issues discovered during the QC process or validation could result in additional time and resource efforts.

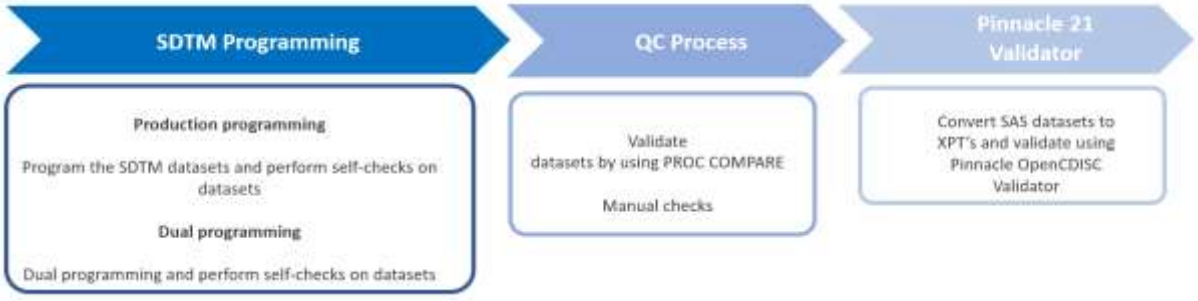

Figure 1 – High-level flow of a standard SDTM process

SDTM VALIDATION TOOL

The necessity for such a tool as this arises from the need for effective processes to produce high quality SDTM datasets. An important aspect of the tool is the comparison with the RAW data; being able to detect any/common programming issues. Moreover, using a Validation Tool to automate the self-checks during programming, as opposed to running the Validation Tool on the final datasets, reduces the time and effort needed to QC the datasets, and the resulting re-work. The output of the macro contains all the issues flagged; gathered in one dataset. This allows a continuation and comparison of the issues from one run of the Validation Tool to the next. The main capabilities of the Validation Tool are illustrated below in Figure 2:

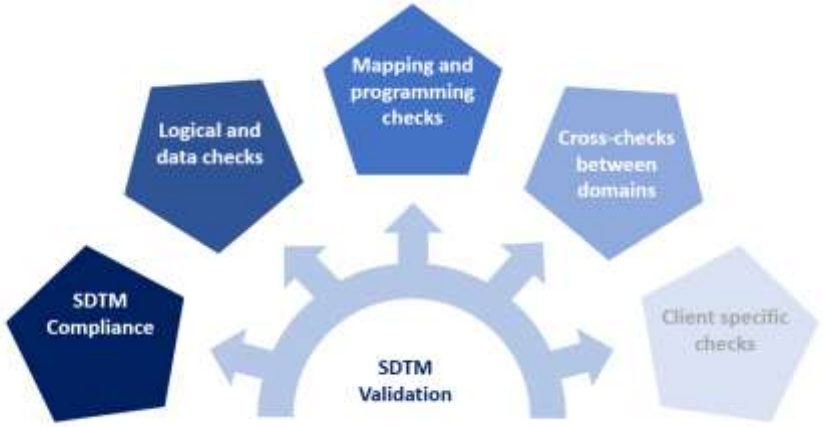

Figure 2 - Validation Tool's capabilities

Applying the Validation Tool results in efficiencies when compared with the standard process. As illustrated below, the QC process takes more time using the standard process, for which the QC includes numerous manual checks in addition to the review of the PROC COMPARE report. Using the Validation Tool, most of the manual checks are replaced and performed automatically. Running the Validation Tool on the SDTM datasets (both production and dual datasets), much earlier in the process, leads to a significant reduction in QC time and also results in a cleaner Pinnacle report, without having to repeat the whole process in order to fix the issues flagged in the report.

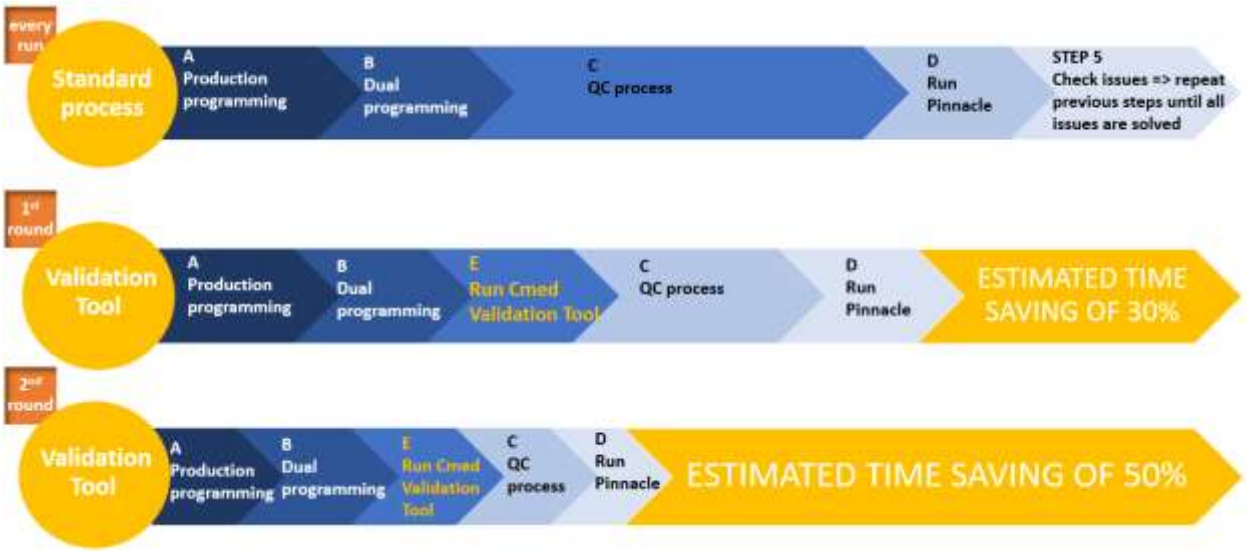

Figure 3 – Comparison between the standard process and the new process (using SDTM Validation Tool)

VALIDATION CHECKS

The creation of the Validation Tool was based on an assessment of manual checks which would remain following a programmatic QC and validation, as well as on possible issues which could be caused by programming and data collection/entry. Three areas were identified (see below Figure 4), where the validation could be easily automated and would have the biggest impact on the quality and efficiency.

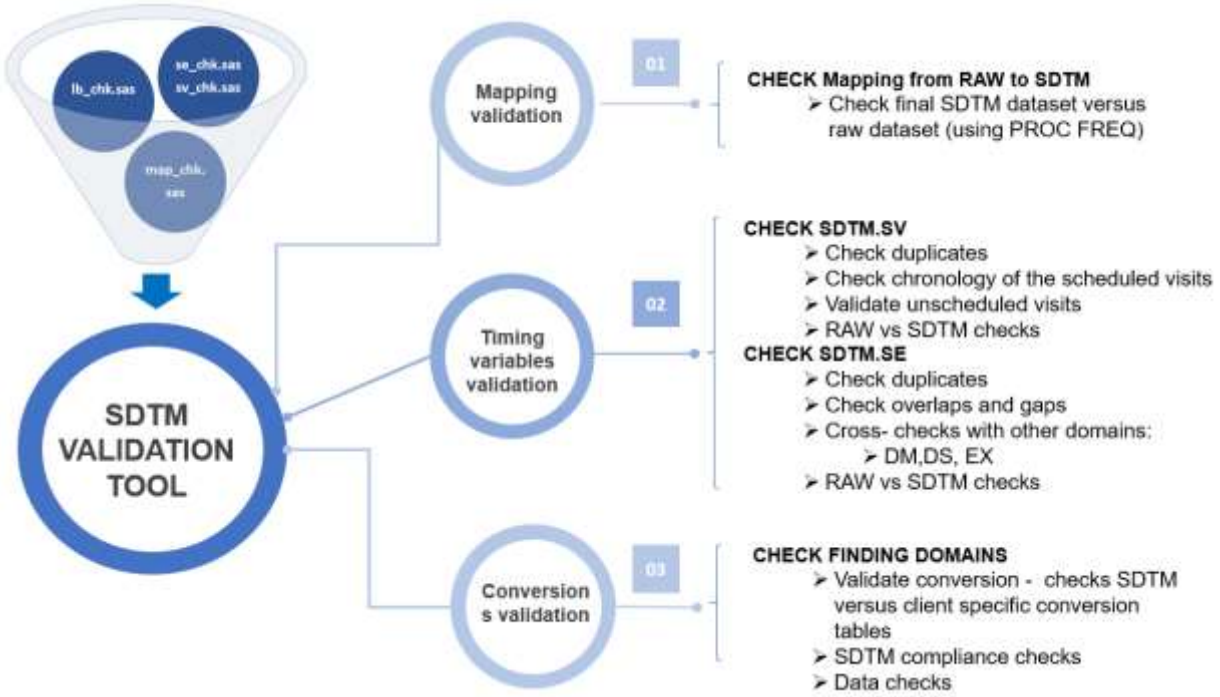

Figure 4 – Main areas addressed by the Validation Tool

METHOD 1: MAPPING VALIDATION

Most commonly used SDTM validation tools address the validity and compliance of final SDTM datasets, however issues can occur due to incorrect mapping or programming which are impossible to identify in the final datasets. Therefore, it was concluded there is a need to double check the values in the SDTMs by comparing against the source data. This feature of the Validation Tool was built in to ensure RAW/SDTM traceability and includes an exhaustive check of all the variables/values. This extra set of checks is able to spot variables and values not being mapped, and vertically validate the SDTM datasets using PROC FREQ (summary per variable, not per record).

Testing started with the AE domain, where almost all mappings are easy derivations based on changing the variable type, renaming or applying a format. An additional way to validate the AE domain was developed using the following methods:

- RAW PROC CONTENTS with two columns added containing the corresponding SDTM variable and dataset:

RAW		SDTM	
RAW dataset	**RAW variable**	**SDTM dataset**	**SDTM variable**
AE	AEACN	AE	AEACN
AE	AE_PTXT	AE	AEDECOD
AE	AEDIS	SUPPAE	AEDIS

➢ PROC FREQ performed on the RAW dataset

```
* RAW FREQUENCIES;
ods output OneWayFreqs=rawfreq;
proc freq data=ae_der;
  tables _all_/ missing;
run;
ods output close;

data rawfreq2(keep= variable value Frequency f_:
              rename=(Frequency=freqraw0));
 length value variable $200;
 format _all_;
 set rawfreq;
 variable=strip(upcase(tranwrd(upcase(table),"TABLE ","")));
 value=strip(coalescec(&rawvars.));
run;
```

VARIABLE	VALUE	RAW FREQUENCY
AEACN	Dose not changed	34
AEACN	Not applicable	12
AEDIS	No	46
AETERM	NEUTROPHIL COUNT DECREASED	5

➢ Mappings required for each unique value (not for all records). For example, for AEDIS the sponsor's metadata indicated the NOYES codelist so the first letter only needed keeping:

```
if cformat eq 'NOYES' then value=upcase(substr(strip(value),1,1));
```

➢ PROC FREQ performed on the SDTM dataset:

```
* SDTM FREQUENCIES;
ods output OneWayFreqs=sdtmfreq;
proc freq data=work.ae_tr;    /* AE_TR is SDTM.AE merged with SUPPAE transposed */
  tables _all_/ missing;
run;
ods output close;

data sdtmfreq2 (keep= variable value Frequency rename=(Frequency=freqsdtm));
 length value variable $200;
 set sdtmfreq;
 variable=strip(tranwrd(upcase(table), "TABLE ",""));
 value=strip(coalescec(&sdtmvar.));
run;
```

VARIABLE	VALUE	SDTM FREQUENCY
AEACN	DOSE NOT CHANGED	34
AEACN	NOT APPLICABLE	12
AEDIS	N	46
AETERM	NEUTROPHIL COUNT DE	5

➢ Merged the datasets from the two PROC FREQs and output all inconsistencies:

```
data mappings (drop= f_: where= (flag ne ""));
 length flag $200;
 merge rawfreq5s(in=raw) sdtmfreq2s(in=sdtm);
 by variable value;
   if raw and not sdtm then flag="Value is in RAW, but not in SDTM";
 else if not raw and sdtm then flag="Value is in SDTM, but not in RAW";
  else if raw and sdtm and freqraw ne freqsdtm then flag="Value present, but different
counts";
run;
```

Upon review of the final dataset, it became very easy to identify the incorrect mappings.

VARIABLE	VALUE	FLAG	RAW FREQUENCY	SDTM FREQUENCY
AETERM	NEUTROPHIL COUNT DECREASED	Value is in RAW, but not in SDTM	5	
AETERM	NEUTROPHIL COUNT DE	Value is in SDTM, but not in RAW		**5**

The method described above results in an improved way to validate the dataset, countering the potential of new values or variables being missed if the mapping specifications are not updated correctly. For example, if both production and dual programmer use the derivations from the mappings specifications, both would obtain the same result, but this would not mean the final dataset was correct. Therefore, this method is also a validation of the derivations, not only the output dataset.

The advantage of this method is that there is not the need to manually program all the assignment statements, because for each RAW variable there is the corresponding SDTM variable, and this is sufficient for the program to compare the frequencies for all the values per variable. Another important aspect is that numeric-character conversions are not needed for performing this QC check, as the PROC FREQ converts everything to character. This enables identification of incorrect conversions and saves the time required to convert anything, in order to test the mappings.

TABLE	AEHLGTCD (numeric)	F_AEHLGTCD (character)	Frequency
Table AEHLGTCD	10003018	10003018	1
Table AEHLGTCD	10014938	10014938	2

METHOD 2: TIMING DOMAINS VALIDATION

The programming of SE & SV can often be challenging, thus the need for an efficient validation is crucial. If the standard approach is to create the SV and SE first and use these for other domains to reference, then the programming of these two domains is very important for the success of the SDTM programming.

SUBJECT ELEMENTS (SDTM.SE) CHECKS

Checking the SE domain can be challenging when looking only at the final SDTM SE dataset. Instead, looking at the RAW data and cross-checking against the other SDTM datasets is arguably the most effective way to validate this domain. The definition for the start and end dates per element, programmed as defined in the mapping specifications cannot guarantee the accuracy of this domain. Based on previous experience in challenges encountered, a list of checks was added to the SE validation macro (included in Figure 5).

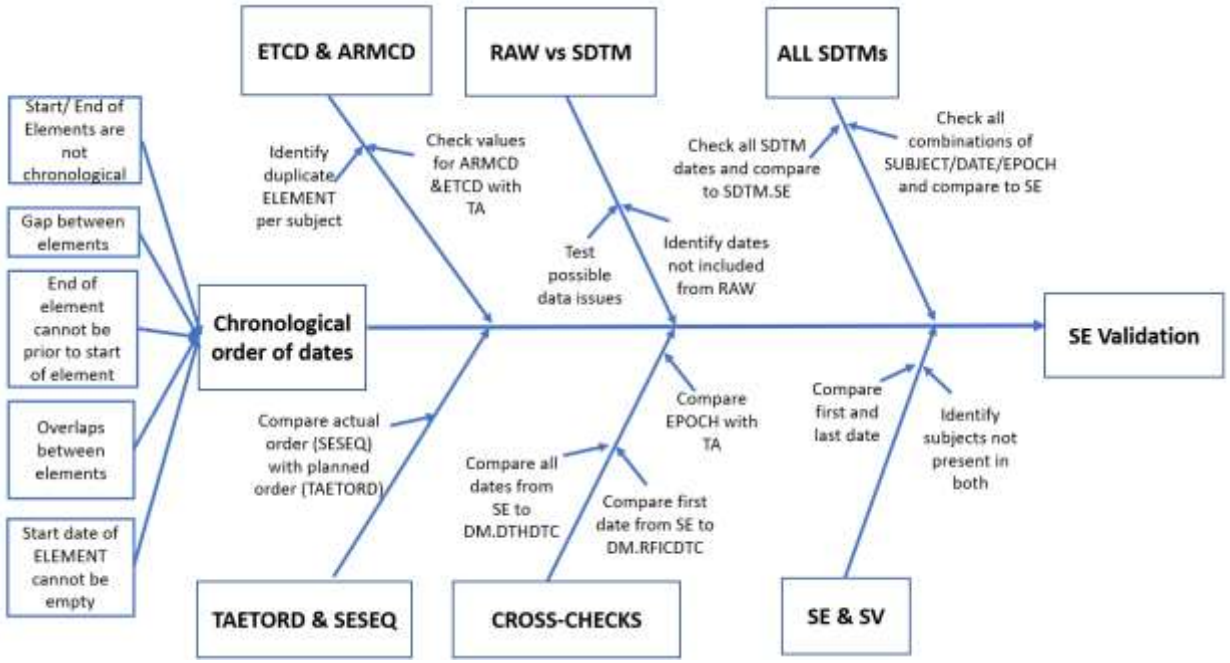

Figure 5 – Diagram with the SE checks addressed by the Validation Tool

These checks can flag potential data or programming issues with ease, allowing a problem to be identified and corrected before it creates issues in other domains (e.g. overlapping elements can create duplicate records in other domains).

As an example, an issue which can be easily missed is when one of the RAW datasets is not included while creating the SDTM.SE. Usually this would be manually updated in the program. To avoid study specific updates and to ensure inclusion of all the RAW datasets in the SE programming, a "DO loop" is used to search through all the RAW datasets and select all the dates, using the naming conventions for date variables.

```
data vcolumns;
  set sashelp.vcolumn(where=(LIBNAME = "RAW" ));
run;

data all_dates;
set vcolumns;
 name_=compress(name, '1234567890');
 len=length(name);
 if memname not in ("AE","DD","CM","MH") and index(name,'ENDT') eq 0 and /*check DAT*/
 ((upcase(substr(name_,length(name_)-2)) in 'DAT'));
run;

*get the list of all datasets where we want to pick the dates from;
proc sql noprint;
```

```
  select memname into: names separated by ' '  from all_dates;
quit;
```

The output dataset contains the dataset name (MEMNAME) and the date variable name (NAME). Please see the example below in Figure 6:

MEMNAME	NAME
DS_EOS	DSSTDT
DS_SCR	DSSTDT
EG	EGDT
EX	EXSTDT
LB	LBDT
SV	SVSTDT
VS	VSDT

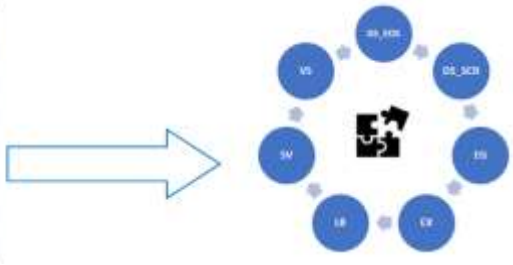

Figure 6 - Create a loop for all the domains which we need dates from

All the dates can be gathered for each subject and compared with the SDTM.SE dataset, to flag all RAW dates which are not included in the SE programming.

The example below shows an example of where the LB date should have been included in the SE dataset.

RAW.LB	
SUBJECT	**LBDT**
1001008	30OCT2017

SDTM.SE			
USUBJID	**ETCD**	**SESTDTC**	**SEENDTC**
STUDY_1001008	SCR		2017-11-13
STUDY_1001008	BAS	2017-11-13	
STUDY_1001008	ABCIVD1	2017-11-17 T08:00	2018-03-09

With manual checks, it is time-consuming to identify the missing "piece of the puzzle" (the LB date missing from SDTM.SE dataset). The Validation Tool flags it (CHECK07 below). The same approach is repeated in CHECK08 (see below), for all SDTM datasets. This check will loop through all SDTM datasets and flag when it finds a date which is not yet included in SDTM.SE dataset (or within the range of dates per element):

USUBJID	CODE	FLAG
STUDY_1001008	CHECK07	CHECK07: Date cannot be found in the SDTM.SE. Please check dataset **RAW.LB** and date 2017-10-30 (RFICDTC=2017-10-30)
STUDY_1001008	CHECK08	CHECK08: Date is not in both SDTMs and SDTM.SE. Dataset=**SDTM.LB** and Date=2017-10-30

A further useful check developed involves the testing of the chronology of the SE dates per subject. The code below checks the chronology of the SE dates and allows for easier identification of any inconsistencies, by transposing all start and end dates and then merging them back by subject ID.

```
data allse check05 (keep= studyid usubjid code flag);
   length code flag $200;
   merge arm_etcd (in=arm) &dset._st (in=st) &dset._end (in=end);
   by usubjid;
   if arm;

   array elem_st (*) %do i=1 %to %eval(%sysfunc(strip(&max.)));  sestdtc&i. %end; ;
   array elem_end (*)  %do i=1 %to %eval(%sysfunc(strip(&max.))); seendtc&i. %end; ;
```

```
 do i=2 to dim (elem_st) by 1;
 * output if there is any element with a date prior to the previous element;
 if elem_st[i] lt elem_st[i-1] and cmiss(elem_st[i],elem_st[i-1] ) eq 0 then do;
       code="CHECK05";
       flag=cat("CHECK05: Start Elements are not chronological. Check ",
strip(elements[i]), " and the previous one");
       if flag ne '' then output check05;
end;

if elem_end[i] lt elem_end[i-1] and cmiss (elem_end[i],elem_end[i-1]) eq 0 then do;
       code="CHECK05";
flag=cat("CHECK05: End of Elements are not chronological. Check", strip(elements[i]),
"and the previous one");
      if flag ne '' then output check05;
end;

*check if the end of one element is the start of the next;
if elem_st[i] ne elem_end[i-1] and cmiss(elem_st[i] , elem_end[i-1]) eq 0 then do;
       code="CHECK05";
       flag=cat("CHECK05: Gap found between element ",strip(elements[i]),
       " and the previous one. SESTDTC=",strip(elem_st[i]),
       " while end of the previous is SEENDTC=", strip(elem_end[i-1]));
       if flag ne '' then output check05;
end;

*check if the end of one element is missing, but we have a start for the next one;
if elem_end[i-1] eq "" and elem_st[i] ne "" then do;
      code="CHECK05";
      flag="CHECK05: the end of element: "!!strip(elements[i-1])!!
" cannot be missing if start of next element: "!!strip(elements[i])!!" is populated";
      if flag ne '' then output check05;
    end;
   end;
  output allse;
run;
```

The example below shows how the Validation Tool flags missing/incorrect dates relating to ELEMENTS.

SUBJID	SESTDTC1	SEENDTC1	SESTDTC2	SEENDTC2	SESTDTC3	SEENDTC3
008	⚠ **-start of ELEMENT cannot be missing**	2017-11-13	2017-11-13	⚠ **-missing end of ELEMENT when start of next ELEMENT is populated**	2017-11-17T08:00	2018-03-09
013	2017-11-13	2017-11-14 T08:10	2017-11-14 T08:10	2018-03-06	2017-10-30 ⚠ **- gap between ELEMENT 2&3 -start of ELEMENT 3 is before the start of ELEMENT 2**	2017-11-13

The Validation Tool picks up all the issues highlighted above and displays them in the output dataset presented below. The Tool helps to collect and present all possible data/programming issues in one place.

USUBJID	CODE	FLAG
STUDY_1001008	CHECK01	CHECK01: Start date of an element cannot be empty. ETCD=SCR
STUDY_1001008	CHECK05	CHECK05: the end of element: BAS cannot be missing if start of next element: ABCIVD1 is populated
STUDY_1001013	CHECK05	CHECK05: Start Elements are not chronological. Check ABCIVD1 and the previous one
STUDY_1001013	CHECK05	CHECK05: End of Elements are not chronological. Check ABCIVD1 and the previous one
STUDY_1001013	CHECK05	CHECK05: Gap found between element ABCIVD1 and the previous one. SESTDTC=2017-10-30 while end of the previous is SEENDTC=2018-03-06

SUBJECT VISITS (SDTM.SV) CHECKS

Initial creation of an accurate SV SDTM dataset supports successful programming of all the SDTM domains, is a key reference for the overall visit timing in the study, and for identification of assessments outside the scheduled visits. A validation of SV therefore ideally identifies data or programming issues at a very early stage of the process, to avoid extra effort at a later stage. Several possible issues were identified and included in the validation checks, as seen below in Figure 7:

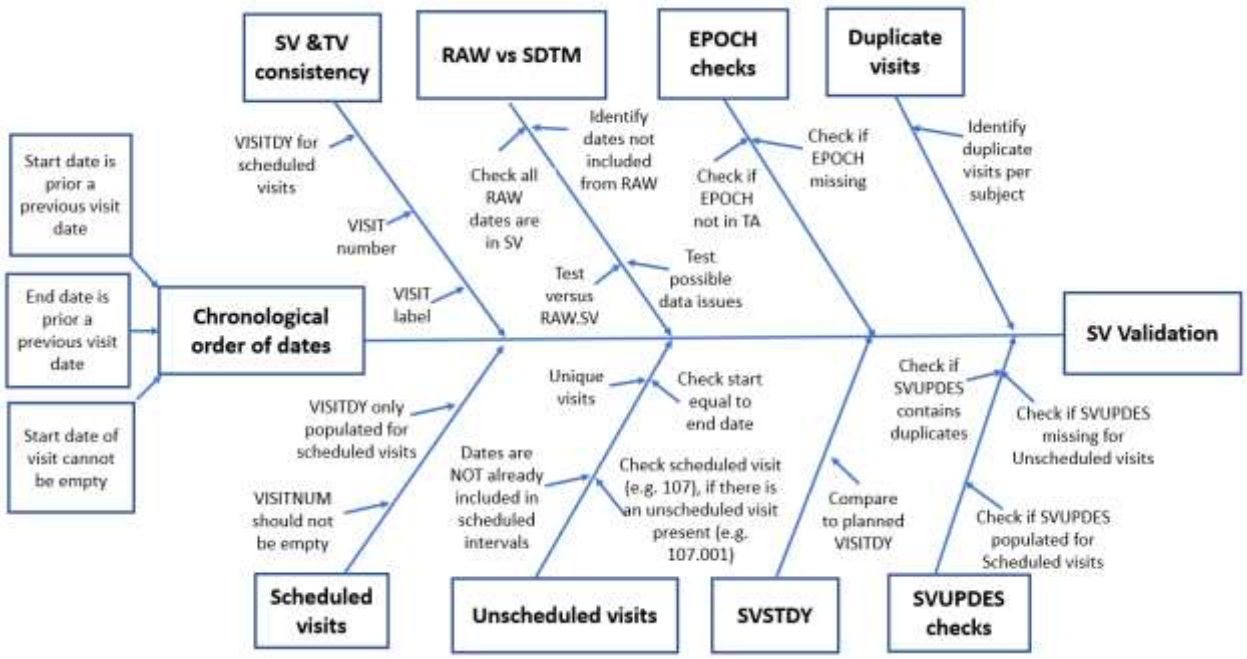

Figure 7-Diagram with the SV checks addressed by the Validation Tool

Special attention is paid to unscheduled visits, and the checking of whether the unscheduled records are truly dates outside of the scheduled window (and have the SVUPDES populated correctly). Two examples are shown below.

USUBJID	SVUPDES	VISITNUM	VISIT	VISITDY	SVSTDTC	SVENDTC
STUDY_1001051 ⚠ **date already included in scheduled** ⚠ **unscheduled visit should not be an interval** ⚠ **missing SVUPDES**		107.010		57	2018-04-03	2018-05-07
STUDY_1001095 ⚠ **duplicates in SVUPDES** ⚠ **SVUPDES should be missing for scheduled visits**	LB, LB, LB	1.000	Screening	-28	2018-03-23	2018-04-09

The issues are correctly flagged by the Validation Tool, as seen in the below table:

USUBJID	CODE	FLAG
STUDY_1001051	CHECK04	CHECK04: Unscheduled date for visit 107.01 is included already in the scheduled visit 109
STUDY_1001051	CHECK06	CHECK06: Check unscheduled visit as start and end are different 107.01 and date SVSTDTC= 2018-04-03 and SVENDTC= 2018-05-07
STUDY_1001051	CHECK07	CHECK07: Unscheduled visit does not have SVUPDES. VISITNUM= 107.01 and date SVSTDTC= 2018-04-03 and SVENDTC= 2018-05-07
STUDY_1001051	CHECK13	CHECK13: Planned study day VISITDY should not be populated for unscheduled visits. USUBJID=STUDY_1001051 and VISITNUM=107.01
STUDY_1001095	CHECK14	CHECK14: SVUPDES should be missing for scheduled visits. USUBJID=STUDY_1001095 and VISITNUM=1
STUDY_1001095	CHECK11	CHECK11: SVUPDES contains duplicates. SVSTDTC= 2018-03-23 and source repeated is: LB and VISITNUM= 1

METHOD 3: CONVERSIONS VALIDATION

The validation of the Findings SDTM datasets is completed in Pinnacle 21 Validator based on the CDISC compliance for the domain, but this cannot account for sponsor specific rules. Sponsor specific checks have been included in the Validation Tool, using the conversion tables provided by a sponsor. The focus is to check all conversions and precisions, based on the reference tables provided by the sponsor. Validation checks were developed on the test code and name, original and standardized unit, original and standardized result (conversion) and precision.

Checking the conversions versus the standards provided by the sponsor allows extension of the available CDISC compliance checks, and an additional check of the conversion to supplement the dual programmed validation.

An example of sponsor specific Reference Tables is presented below, containing all the information needed for the conversions required in the Findings domains.

TESTCD	TEST	CAT	SCAT	MSRUNIT Measured unit	PREUNIT Sponsor preferred unit	CNVFCR Converted factor (SI)	CNVUNIT Converted unit (SI)	PCSNUM Precision
HOMA2IR	HOMA2-Insulin Resistance	CHEMISTRY			NO UNIT			
APTT	Activated Partial Thromboplastin Time	HEMATOLOGY	COAGULATION	sec	s	1	s	1
LYM	Lymphocytes	HEMATOLOGY	CEREBROSPINAL FLUID	Leuko/ul	cells/uL	0.001	10E9/L	0
FIBRINO	Fibrinogen	HEMATOLOGY	COAGULATION	mg/dL	mg/dL	0.01	g/L	2
SODIUM	Sodium	CHEMISTRY	ELECTROLYTES	mmol/L	mmol/L	1	mmol/L	0
UWBCQ	Urine white blood cells(value)	URINALYSIS			**HPF**			

Below is an example of 3 errors identified by the Validation Tool.

CODE	FLAG
CHECK03	SI unit is different from Reference Tables. PARM=UWBCQ TESTCD=UWBCQ LBCAT=URINALYSIS LBMETHOD= LBSPEC=URINE **LBSTRESU=/ul,** PREUNIT=HPF
CHECK04	Conversion is not correct (compare re-derived RESC with SDTM.LBSTRESC), please check PARM=LYM, LBTESTCD=LYM, PREF_UNIT=Leuko/ul, LBSTRESU=10E9/L, CNVFCR=0.001, PCSNUM=2, RESC=2.72, **LBSTRESC=2717.00**, LBORRES=2717
CHECK05	Precision is different from Reference Tables, please check PARM=APTT LBTESTCD=APTT PCSNUM=1 LBSTRESU=s **SDTM_PREC=0**

The Validation Tool also has a consistency flag between result and unit, considering that when the LBSTRESU unit is "CODED", the expectation would be that the result is also coded. Similarly, if the result is "POSITIVE" or "NEGATIVE", the expectation would be for the unit to be "CODED". Programming these flags helps to double check the appropriate parameter codes from the CRF.

CODE	FLAG
CHECK07	Check if result is measurable, as the unit suggests we should not have numeric results. TESTCD=HOMA2IR LBCAT=CHEMISTRY LBMETHOD= LBSPEC=PLASMA/SERUM **LBORRES=6.37** LBSTRESU=NO UNIT
CHECK07	Check if result is measurable, as the unit suggests we should not have numeric results. TESTCD=UBILST LBCAT=URINALYSIS LBMETHOD=DIPSTICK LBSPEC=URINE **LBORRES=0.2** LBSTRESU=CODED
CHECK08	Check if result should be coded, as the result is not numeric. TESTCD=UCASTQ LBCAT=URINALYSIS LBMETHOD= LBSPEC=URINE **LBORRES=NEGATIV**E LBSTRESU=mg/dL

OUTPUT FROM THE VALIDATION TOOL

The Validation Tool creates output as a dataset containing all the identified issues consolidated. The validation checks are further output to a Microsoft Excel spreadsheet in which the user can add comments alongside the issues detected (the last two columns from the example below: STATUS and COMMENT). This enables comparison of the issues from one run to the next and allows the highlighting of previous issues which were resolved or under investigation, but also inclusion of any new issues.

An example of the Validation Tool output showing a comparison between the SDTM datasets created from two different extractions is presented below. This allows the focus to be on new issues and avoids wasting time investigating older/known issues. This is also a good option to keep track of old data issues/anomalies and their resolution from one extract to another.

COMP FLAG	STUDYID	OLD_ EXTRACT	NEW_ EXTRACT	USUBJ ID	TYPE	FLAG	STATUS	COMMENT
OLD AND NEW	ABC123	30AUG19 :07:41:36	03SEP19: 15:58:06	001	SE	Date is not in both SDTMs and SDTM.SE. Dataset= SDTM.LB Date=2019-02-18	ONGOING	Date queried with DM
OLD ISSUE, BUT FIXED	ABC123	30AUG19 :07:41:36		002	DV	Comparison RAW/SDTM PROC FREQ not done, as SDTM.DV is empty.	TO BE UPDATED	
NEW ISSUE	ABC123		03SEP19: 15:58:06	003	SV	Unscheduled date for visit 2102.01 is included already in the scheduled visit 2103		

VALUE ADDED

The advantages of using the SDTM Validation Tool compared to validating the SDTM datasets only using Pinnacle 21 Validator are:

- Able to implement sponsor-specific checks
- Able to perform cross-checks against the RAW data
- Fixing issues earlier in the process of SDTM development
- Having a full report of the issues from all datasets gathered in one place
- Being able to compare several reports and spot new issues easier, but also track old issues resolution

CONCLUSION

In summary, whilst there are many ways of validating SDTM datasets, the Validation Tool is based on analysis of the most common, previously encountered issues in programming and validating SDTMs. Without the tool, data anomalies can be easily missed. The Validation Tool helps by eliminating the need for time consuming thorough manual QC checks. This paper recommends running the Validation Tool to supplement the QC process. The Validation Tool is not intended to replace Pinnacle 21 Validator or other similar tools, but rather to enhance their power.

The Validation Tool is designed to be considered as a living organism with the flexibility to be updated to satisfy ever evolving needs. Thus, ensuring the highest quality SDTM deliveries and customer satisfaction.

REFERENCES

- https://www.lexjansen.com/pharmasug/2018/DS/PharmaSUG-2018-DS13.pdf

ACKNOWLEDGMENTS

I would like to thank all the reviewers of this paper for their valuable comments and suggestions.

CONTACT INFORMATION

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

Author Name: Anamaria Calai
Company: Cmed SRL
Address: Str. Coriolan Brediceanu, Nr. 10 City Business Centre Building A, Second Floor
City / Postcode: Timisoara, Timis, 300011, Romania
Work Phone: +40 (0)356 433 929
Email: acalai@cmedresearch.com
Web: https://www.cmedresearch.com/