PharmaSUG 2018 – Paper DS-13

Global Checklist to QC SDTM Lab Data

Murali Marneni, PPD, LLC, Morrisville, NC
Sekhar Badam, PPD, LLC, Morrisville, NC

ABSTRACT

Laboratory data is one of the primary datasets that we usually see in every submission in clinical trials. It also is the primary dataset that most of the programmers usually encounter with a lot of problems. Due to greater complexity, as it requires more data manipulations to create the SDTM lab data, most programmers completely rely on the Pinnacle report to make sure the SDTM lab data is accurate and can be used for downstream tasks. It is important to realize that passing the Pinnacle issues checks does not mean everything is good. So, it is important to do additional checks to catch the data issues ahead of time since reporting the data issues later costs more money and resources, and sometimes leads to inaccurate results in analysis. This paper talks about effective and useful techniques that help to QC your SDTM Lab data, and will help as a checklist for lab data on your future projects.

INTRODUCTION

In Pharmaceutical/Biotech industry clinical trials, the laboratory dataset is one of the primary datasets to establish drug's safety profile, and the results of laboratory tests play a crucial role in determining the safety and efficacy of the drug being studied. Lab data can be very complex and it is very common to receive data from more than one source and in more than one format, such as in SAS® data format, and non-SAS formats, such as delimited text, Excel® and CSV files. Due to the greater complexity, manipulation of the data is a must in creating SDTM LB data to standardize results, units and ranges to effectively use it in analysis database and in generating reports.

Since it is impossible to receive data without any issues based on nature of the lab data, programmers must put in great efforts to identify these issues, have them resolved and have quality data ready for analysis. Analysis of poor quality data leads to inaccurate conclusions, and it affects safety and efficacy assessments and outcome of the project.

It is very important to develop and run quality checks as a standard practice to improve the data quality. Most of the programmers completely rely on the Pinnacle report to identify and resolve data issues but do not spend enough time initially in pre-processing the data and identifying data issues.

It is very important to realize that resolving issues from Pinnacle report does not mean that data is perfect as Pinnacle checks are driven by CDISC compliance standards and these checks do not check for manual data entry errors, outliers and other data issues. So, along with Pinnacle checks, it is important to implement additional checks initially to isolate, identify data issues ahead of time and to avoid significant delays because of reruns, resource issues and to reduce costs.

This paper provides QC checks that are effective and useful in pre-processing of the data and producing quality SDTM lab data. Some of these checks are present in Pinnacle 21 but the main idea of this paper is to have all the checks placed in one area while programming SDTM LB domain. It will be very beneficial to the study teams and gives adequate time to identify and resolve data issues initially, minimizes time spent further down investigating data issues and ensures high quality data is used in analysis and report generation.

Another important benefit of writing these checks in initial programming phase is it helps to communicate the data issues with Data Management or external vendors in efficient way to avoid any miscommunications. All checks that are specified in this paper have a programming approach and is explained in two approaches. First approach is to write the issues to the log and second approach code is to write the issues to the Excel file that used for communication purpose with Data Management or with

external vendors. There are some additional cautions that are presented in this paper, that helps programmers to decide which checks requires manual review or more attention.

QC CHECKS

Check 1: LBTEST, LBTESTCD Lengths must be 40, 8 characters respectively.

It is very common to receive data with test names longer than 40 characters. TEST, TESTCD must be mapped to Controlled terminology and within specified lengths. Per CDISC guidelines, TEST and TESTCD lengths must be 40 and 8 respectively.

```
/*----Check for lengths of _ _TEST, _ _TESTCD-----*/

proc contents data = Lbinput out=check1 (keep= NAME LENGTH) noprint;
run;

data _null_;
  set check1 (where= (NAME in ('LBTEST','LBTESTCD')));
  if NAME ='LBTEST' and LENGTH >40
   then put "ALERT_R: Length of _ _TEST is more than 40, must be <=40";
  else if NAME ='LBTESTCD' and LENGTH >8
   then put "ALERT_R: Length of _ _TESTCD is more than 8, must be <=8";
run;
```

Check 2: Unique LBSTRESU for LBCAT/LBTEST/LBTESTCD/LBSPEC/LBMETHOD:

Since we get lab units from various sources, it is important to have unique standard units per each lab test. Units must be converted into one standard unit if test is performed by multiple labs for appropriate analysis but there is a lot of confusion around this. For example, same lab test names can differ depending on the lab that performed the test, especially local labs. Units can vary from lab to lab, making it essential for standardization among units and conversions. So, it is important to consider LBSPEC/LBMETHOD to perform this check. If LBMETHOD is not collected, it can be removed in following check. These checks are processed in two ways.

Approach 1: This approach is to write messages to the log, so that it can be easily identified by looking at log and make changes in program if needed.

```
/*Check for unique standard unit per LBCAT/LBTEST/LBTESTCD/LBSPEC/LBMETHOD*/

proc sort data = Lbinput nodupkey out = check2
            (keep = LBCAT LBTEST LBTESTCD LBSPEC LBMETHOD LBSTRESU);
  by LBCAT LBTEST LBTESTCD LBSPEC LBMETHOD LBSTRESU;
run;

data _null_;
  set check2;
  by LBCAT LBTEST LBTESTCD LBSPEC LBMETHOD LBSTRESU;
  if first.LBTEST ne last.LBTEST then do;
   if first.LBTEST then
    put "ALERT_R: Non-Unique Standardized unit present for " LBTEST " test.
    Requires review";
  end;
run;
```

Approach 2: Writing these issues to SAS datasets and then write to Excel file to consolidate issues for communication purpose.

```
proc sort data = Lbinput out = check2
            (keep = LBCAT LBTEST LBTESTCD LBSPEC LBMETHOD LBSTRESU);
  by LBCAT LBTEST LBTESTCD LBSPEC LBMETHOD LBSTRESU;
run;

proc freq data = check2 noprint;
  tables LBCAT*LBTEST* LBTESTCD* LBSPEC* LBMETHOD* LBSTRESU/
   out = check2 list nocum nopercent;
run;

data check2;
  set check2;
  by LBTEST LBTESTCD LBCAT LBSPEC LBMETHOD LBSTRESU;
  if first.LBTEST ne last.LBTEST ;
run;
```

Check 3: LBSTRESU is missing but LBORRESU, LBSTRESC are present.

This check helps to find out if original unit is populated but standard unit is missing and if standard result is populated but standard unit is missing. Standard unit must be populated for all non-missing standard results that are equivalent to original result, or that are converted from original result. If original result is equivalent to standard result, i.e., no conversion required, in those cases we can assign original units to standard units if units are missing. For all other converted results, data should have standard units for all non-missing results for analysis, reports and for valid interpretations.

Approach 1:

```
/*-Check for missing LBSTRESU while LBORRESU, LBSTRESC are not missing-*/
proc sort data = Lbinput
   (where=((LBORRES ne "" and LBSTRESU eq "") or (LBSTRESC ne "" and
           LBSTRESU eq "") ))
    out= check3(Keep= LBCAT LBTEST LBTESTCD LBSPEC LBMETHOD LBSTRESU
                      LBORRESU) nodupkey;
  by LBCAT LBTEST LBTESTCD LBSPEC LBMETHOD;
run;

data _null_;
  set check3;
  by LBCAT LBTEST LBTESTCD LBSPEC LBMETHOD;
  if last.LBTEST then do;
   put "ALERT_R: Standard unit is missing for " LBTEST " test. Requires
   review";
  end;
run;
```

Approach 2: To generate check dataset and use it for producing dataset with consolidated data issues later in the program.

```
data check3;
  set check3;
  by LBCAT LBTEST LBTESTCD LBSPEC LBMETHOD;
  if last.LBTEST;
run;
```

Check 4: Check for duplicate records on key variables

Removing duplicate records is easy but one should make sure they are actual duplicates before deleting them. One subject should have only one record per test per timepoint per visit. If there is more than one record per subject per test per timepoint per visit, it is considered as a duplicate. Investigation of duplicate records is important before deleting them as it may be due to repeated data entry or re-testing of sample due to nature of sample. This is very important check as it may take longer to resolve this type of issues with vendor and data management. Identifying and resolving these issues early will help with accurate analysis and improve data integrity and quality.

Approach 1:

```
/*-Check for identifying duplicate records-*/
proc sort data = Lbinput  tagsort NOUNIQUEKEY
      UNIQUEOUT= Lb (keep = USUBJID LBCAT LBTEST LBTPTNUM VISITNUM LBNAM
                            LBDTC)
             out=check4;
  by USUBJID LBCAT LBTEST LBTPTNUM VISITNUM LBDTC;
run;

data _null_;
  set check4;
  by USUBJID LBCAT LBTEST LBTPTNUM VISITNUM LBDTC;
  if first.USUBJID ne last.USUBJID then
    put "ALERT_R: Duplicate records for subject " USUBJID LBTEST " test.
         Requires review";
run;
```

Approach 2: Check4 dataset generated in above approach can be used for consolidating data issues later in the program.

Check 5: Is Reference range indicator missing while original result/standard result and ranges are present?

Reference range Indicator(LBNRIND) must be populated with respect to reference ranges (LBORNRLO, LBORNRHI, LBSTNRLO, LBSTNRHI, LBSTNRC) when original or standard ranges are not missing. To avoid misunderstanding, it is extremely important to consider factors like whether the reference range indicator values are populated based on standard ranges or original ranges, age, sex of the subject before converting units, ranges to standard format and comparing LBNRIND with upper and lower ranges. Sponsors should specify in the study metadata file (define.xml) under comments column whether LBNRIND refers to standard or original reference ranges and results.

Approach 1:

```
/*-Check for identifying missing LBNRIND records while result and ranges are
present-*/
proc sort data =Lbinput
   (where=(LBNRIND = "" and (LBSTNRLO ne . or LBSTNRHI ne . or LBORNRLO ne ""
           or LBORNRHI ne "" or LBSTNRC ne ""))) tagsort nodupkey
    out= check5 (keep = LBCAT LBTEST LBTESTCD LBNAM LBSTNRHI LBSTNRLO
                        LBSTNRC LBNRIND LBORNRLO LBORNRHI);
  by LBCAT LBTEST LBTESTCD LBNAM;
run;
```

```
data _null_;
  set check5;
  by LBCAT LBTEST LBTESTCD;
  if last.LBTEST  then
  put "ALERT_R: Reference range indicator is missing even though normal
  ranges are present " LBTEST " test. Requires review";
run;
```

Approach 2: Replace _null_ with Check5 in approach 1 and this dataset can be used for consolidating data issues later in the program.

Check 6: Check for derived reference range indicator values.

As discussed in check 5, one should always consider all factors before comparing LBNRIND to normal ranges and make sure LBNRIND is correctly populated. In some cases, LBNRIND is assigned programmatically and it may be populated incorrectly if missing ranges are considered. This check validates LBNRIND value based on normal ranges.

Approach 1:

```
/*-Check for identifying incorrectly derived LBNRIND records -*/
data _null_;
  set Lbinput;
  if (LBSTRESC eq "" or (LBSTNRLO eq  . and LBSTNRHi eq . and LBSTNRC eq ""))
   and LBNRIND ne "" then
   put "ALERT_R: Reference range indicator is populated even though
   result/normal ranges are missing " USUBJID LBSEQ "Requires review";
  if (LBSTRESN ne . and (LBSTNRLO eq  . or LBSTNRHi eq .)) and LBNRIND ne ""
   then put "ALERT_R: Reference range indicator is populated even though
   normal ranges are missing " USUBJID LBSEQ "Requires review";
Run;
```

Approach 2: Replace _null_ with Check6 in approach 1 and this dataset can be used for consolidating data issues later in the program.

Check 7: Check for Origin of SDTM variable.

If a variable origin is from multiple sources, it must be documented under Origin column. It is very common to receive lab data from multiple sources and origin should be updated to ‘CRF’,’eDT’, as applicable, per data collected source and documented in Define.xml. Some of instances where origin is from more than one source are LBDTC, LBSPEC, LBNAM, LBREFID, LBTEST, LBCAT, LBORRES, LBORRESU.

Check 8: LBSTRESC is expected to be populated for Baseline records (LBBLFL='Y').

Standard result value must be populated for all records that are flagged as baseline. It is very common to incorrectly assign baseline flag by considering records with missing result, as sample may be collected but not analyzed due to several reasons, and considering missing collection dates. This check helps in identifying incorrectly derived baseline flags in the program.

Approach 1:

```
/*-Check for identifying incorrectly derived LBBLFL records -*/
data _null_;
  set Lbinput  (where=(LBBLFL ='Y' and LBSTRESC =="")) ;
  if _n_ > 1 then put "ALERT_R: Record is flagged as baseline even though
   test is not done/result is missing " USUBJID LBSEQ "Requires review";
run;
```

Approach 2: Replace _null_ with Check8 in approach 1 and this dataset can be used for consolidating data issues later in the program.

Check 9: Check for records with missing LBSTRESC while LBORRES is provided.

Standard character result (LBSTRESC) must be populated when original result is not missing. This check is specifically helpful when standard result is populated by converting original result using conversion factors. If conversion factors are different for each patient (age/sex) and each test then there is possibility of overlooking tests for conversion and programmatical errors when conversions are not dynamic.

Approach 1:

```
/*-Check for identifying missing LBSTRESC records while LBORRES not missing-
*/
proc sort data = Lbinput (where=(LBORRES ne "" and  LBSTRESC eq "")) tagsort
          nodupkey out = check9 (keep = LBTEST LBTESTCD LBORRES LBSTRESC);
  by LBTEST LBTESTCD;
run;

data _null_;
  set check9;
  if _n_ > 1 then put "ALERT_R: Standard result is missing while original
   result is not missing" LBTEST  "Requires review";
run;
```

Approach 2: Replace _null_ with Check9 in approach 1 and this dataset can be used for consolidating data issues later in the program.

Check 10: Check for missing LBSTRESN while LBSTRESC is provided and represents a numeric value.

For all standard results (LBSTRESC) that represents a numeric value, LBSTRESN must be populated. As discussed in Check 9, this is specifically helpful when standard result is populated by converting original result using conversion factors and LBSTRESN is derived from LBSTRESC.

Approach 1:

```
/*-Check for identifying missing LBSTRESN records while LBSTRESC is not
missing-*/
data check10;
  set Lbinput  (where=(LBSTRESC ne ""));
  NRES = input(lborres,??best.);
run;
```

```
proc sort data = check10 (where=(NRES ne "" and LBSTRESN= .)) tagsort
      nodupkey out = check10 (keep = LBTEST LBTESTCD LBSTRESN NRES LBSTRESC);
  by LBTEST LBTESTCD;
run;

data _null_;
  set check10;
  if _n_ > 1 then put "ALERT_R: Standard numeric result is missing while
   character result is not missing" LBTEST "Requires review";
run;
```

Approach 2: Replace _null_ with Check10 in approach 1 and this dataset can be used for consolidating data issues later in the program.

Check 11: Missing value for LBSTRESC, while LBSTRESU is provided

Units should be populated only when result is populated. This can be programmatically fixed by assigning units to missing if result is missing. However, this issue can be resolved in early stages of the project by reviewing study design specifications, CRF, and making sure units are populated only when result is populated while study is in user acceptance testing (UAT). The following QC check helps in identifying the issue.

Approach 1:

```
/*-Check for identifying missing LBSTRESC records while LBSTRESU is not
missing-*/
proc sort data =Lbinput
    (where=((LBSTRESC eq "" and LBSTRESU ne "") or (LBORRES eq "" and
             LBORRESU ne "" )))   tagsort nodupkey
    out = check11(keep = LBTEST LBTESTCD LBSTRESU LBORRESU LBORRES LBSTRESC);
  by LBTEST LBTESTCD LBSTRESU LBORRESU;
run;

data _null_;
  set check11;
  by LBTEST LBTESTCD;
  if first.LBTEST ne last.LBTEST then
   put "ALERT_R: Units are populated while results are missing" LBTEST
   "Requires review";
run;
```

Approach 2: Replace _null_ with Check11 in approach 1 and this dataset can be used for consolidating data issues later in the program.

Check 12: Check to see if lab result is present and it is numeric but with missing corresponding original unit.

This check helps to find out the missing original unit records while reported result value is a numeric. It is necessary to have checks to find the missing unit observations as unit is very important for analysis purpose, and to determine any toxicity flags that are related to lab result values. Sometimes, this check draws confusion, for example: LBTEST = ‘pH’, this test result is mostly reported in numeric but they do not have any units reported. So, what is importance of this check? Even though this check does not apply to every test result that is reported in numeric, average studies indicate that mostly 95% of lab tests

that have results in numeric requires units. So, this check has more advantages compared to drawbacks and of course we can write a subset condition in the check to avoid Lab tests like 'pH' from the check.

Approach 1:

```
/*-Check for identifying missing LBORRESU records while LBORRES is provided*/
data check12;
  set Lbinput ;
  lborresn = input(lborres,??best.);
  if lborresn ne . and lborresu = ' ';
  put "Alert_R:" USUBJID " have records with reported result that has numeric
   result values but do not have corresponding unit";
run;
```

Approach 2: Replace _null_ with Check12 in approach 1 and this dataset can be used for consolidating data issues later in the program.

TRACKING THE ISSUES

It is important to track data issues after finding them. The above checklist can be macrotized to output the issues to Excel file with a tab for each check to track these issues. It is also useful for efficient communication of data issues with data management and external vendors. Since we are dealing with a large quantity of data, this helps in automation of the communication process of data issue reporting and gives metrics on issues which helps to know the status and health of the data. See figure 1 as an example of a report.

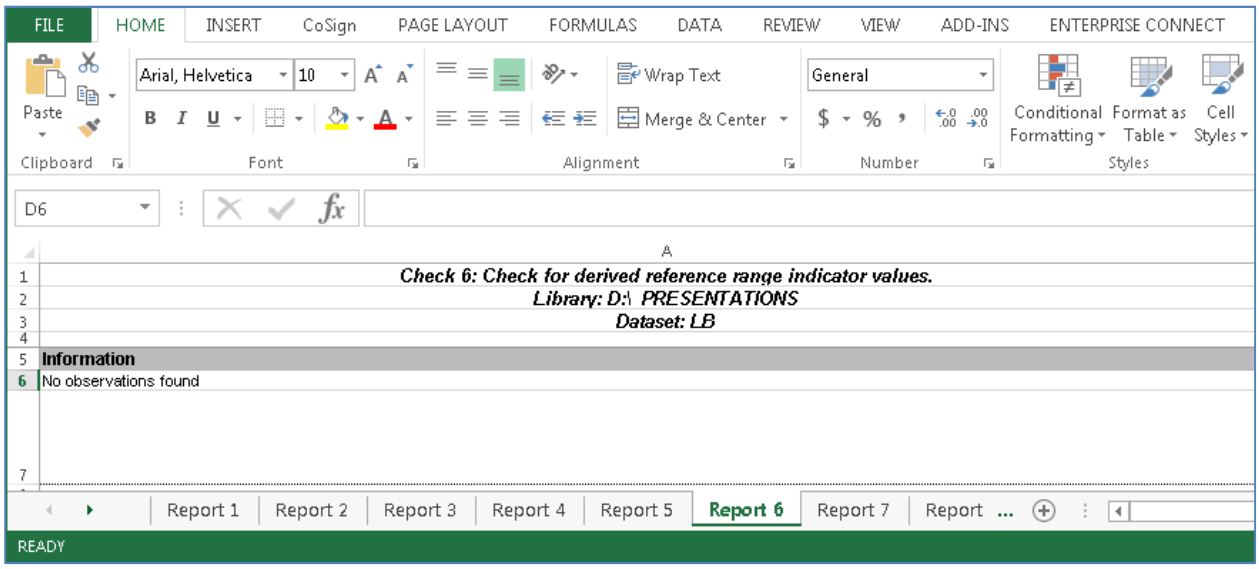

Figure 1: Example of consolidated report.

CAUTIONS

Caution 1: Check to see if derived standard result values are based on conversion factor that have rounding on decimals:

After converting to standard units with respect to original units, standard result shows more decimals than the original values, we often see that most of the programmers just round the values to 2 or 3 decimals so to speed up the validation process. This type of rounding should be avoided in SDTM level as this causes

more disadvantages to the study, especially when creating grouping, flagging of normal/abnormal or CTCAE grading. Sometimes, this type of rounding in initial stages may also lead to incorrect p-value and may show negative impression towards the study. So, it is recommended to do the rounding of lab results based on statistician input and after discussion with broader study teams to avoid any misinterpretations.

Caution 2: Do not assume conversion factors if it was not provided:

Conversion factors can differ from one analyte to another analyte, and it is totally depending on molecular weight and density of an analyte. If it is not provided then do not consider or assume other analyte conversion factor even though the units appear to be the same. Here are some examples: Uric Acid conversion factor for mg/dL to umol/L is 59.48, Bilirubin conversion factor for mg/dL to umol/L is 17.1, Creatinine conversion factor for mg/dL to umol/L is 88.4. It is recommended to utilize standard lab conversion metadata sheet and review it with lab expert, statistician and client. Once a standard spreadsheet is in place, it can be macrotized to pull this information to convert the values as well as to handle the number of significant digits preferred for reporting purposes.

CONCLUSION

There may be other checks which could be applied along with above discussed checks for more efficient programming. The checklist presented in this paper helps to build and standardize checks, that leads to successful and efficient programming of huge datasets. This also brings consistency throughout the submission of projects. These methods will allow statistical programmers to detect the raw data issues early, which will lead to high quality data at database lock. The study team can draw reliable conclusions on both the safety and efficacy data, which will benefit the patients and the client.

REFERENCES

- http://www.pinnacle21.net/blog/fda-final-guidance-webinar-qa
- STUDY DATA TECHNICAL CONFORMANCE GUIDE. Guidance for Industry Providing Regulatory Submissions in Electronic Format – Standardized Study Data v3.3. FDA CDER, CBER https://www.fda.gov/ForIndustry/DataStandards/StudyDataStandards/default.htm

ACKNOWLEDGMENTS

We would like to thank each one from PPD Phase 2-4 department, RTP for continuous support and especially thanks to Catherine Edwards, Ken Borowiak, Hunter Everton, Ryan Wilkins, Ajay Gupta, Evangela Hager for their inputs, comments. I would like to thank to my family for support.

CONTACT INFORMATION

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

Murali Marneni, M.S
PPD, LLC
3900 Paramount Parkway
Morrisville, NC 27560
E-mail: Murali.Marneni@ppdi.com, marneni02@gmail.com

Sekhar Badam, M.S
PPD, LLC
3900 Paramount Parkway
Morrisville, NC 27560
E-mail: Sekhar.Badam @ppdi.com, sekhar0445@gmail.com

DISCLAIMER

The content of this paper are the works of the authors and do not necessarily represent the opinions, recommendations, or practices of PPD, LLC.

The interpretations of CDISC standards presented are not necessarily correct and do not represent the CDISC consortium.