PharmaSUG 2014 - Paper BB03

That SAS®sy Lab Data

Amie Bissonett, inVentiv Health Clinical, Minneapolis, MN

ABSTRACT

Working with laboratory data can be a daunting task. Data can come from central labs, local labs, or both on the same study. Receiving lab data from different labs can result in values within a lab test to have differing units and normal ranges, which need to be reconciled prior to analyzing and displaying results so that values are accurate and consistent. CDISC standards bring additional issues to the table that need to be considered while programming the SDTM and ADaM data sets. This paper displays some data issues to look for and how to handle them, as well as how to handle programming issues for SDTM, ADaM, and CTCAE grading.

INTRODUCTION

Laboratory data is generally the largest set of data in a pharmaceutical trial. Patient safety is of utmost importance, and lab information is an objective way to determine any adverse reactions a patient has to an investigational product. Analyzing lab data is pretty straight forward. The difficulty is getting the data in analysis ready format.

LOCAL AND CENTRAL LABORATORIES

The ideal situation in any trial is to get all of the labs for all subjects done at laboratories utilizing the same set of units and normal ranges. Unfortunately, this is often not the case. If a trial utilizes laboratories local to the study sites, many different standards may be used and data is received in differing units, which cannot be analyzed together.

There are two main standard conventions for lab values, Conventional Units (U.S. standard) and the International Standard of Units (SI). Laboratories in the United States generally collect their labs in conventional units; however, many trials choose to analyze the data in SI units. Therefore, the lab values must all be converted from conventional units to SI units. Conversion factors exist to transform the data, and many data management departments have systems in place to do this conversion.

LB SDTM DATA SET

Generating the LB SDTM data set is very similar to creating a lab analysis data set without CDISC standards. This paper is written based on use of CDISC standards; however, all information can be transferred to programming for non-CDISC data sets.

The raw lab data set received from local and/or central laboratories should contain certain variables, such as the full analyte name, ANALYTENAME, a shorter analyte name, usually called SASNAME, the analyte values, units, and normal ranges. This paper is assuming that conventional units are collected and that conversions are applied and SI unit variables are added and in the raw data received by programming, so that two sets of values, units, and ranges are included.

The LB data set is then derived based on the Case Report Form (CRF) annotations and a data definition table. For CDISC, a standard template exists in the SDTM Implementation Guideline document and can and should be used as a starting point. There are specific formats and default mappings to derive the variables. Most of the variables can simply be assigned to the corresponding variable in the raw data set.

Some companies have applied their own standard for lab test name and the lab test code values; however, CDISC does have its own list of standard values that should be used if possible. There are a few descriptive variables that need to be created for each lab test, including category and sub-category. A simple way to derive these variables is to create formats for all analyte names for all of these variables, such as LBTESTCD, LBTEST, LBCAT, and

LBSCAT. The ANALYTENAME variable values are standard, allowing these formats to be used across multiple projects.

```
lbtestcd = put(analytename, $lbtestcd.) ;
lbtest = put(analytename, $lbtest.) ;
lbcat = put(analytename, $lbcat.) ;
lbscat = put(analytename, $lbscat.) ;
```

ADLB ADAM DATA SET

The ADLB ADaM data set is a Basic Data Structure (BDS) data set, which contain one or more records per subject, per analysis parameter, per analysis time point. ADaM data sets are designed to be analysis ready so that little to no additional programming is required in the table, figure or listing program. There are variables that support the analysis, such as subject identifier, treatment group, and covariates.

The analysis parameter variables, PARAMCD and PARAM, are the basis from which the analysis is grouped on. The specifications for these variables require that there is a one-to-one relationship between PARAMCD and PARAM. PARAMCD is the short name of the analysis parameter and set to SDTM LBTESTCD. PARAM must include the long parameter value, units (if applicable), specimen type, location, position and any other applicable qualifying information. Each clinical trial will need to determine the information required. This is a difference from SDTM LB data, which has separate variables for each of these values.

Only information required for analysis should be kept in the ADLB data set. If lab tests are to be analyzed in both conventional and SI units, one record for each unit must be present in ADLB. Due to the one-to-one relationship for PARAMCD and PARAM, these values must have some value to differentiate them between the two records, such as adding a single character to the end of the PARAMCD and PARAM values, i.e. PARAMCD = ‘ALB’, PARAM = ‘Albumin (g/L)’ for SI unit record, and PARAMCD = ‘ALBC’, PARAM = ‘Albumin (g/dL)’ for conventional unit record.

An issue to be aware of when deriving the PARAM value in ADLB is that in the SDTM LB data set, it is common practice to set the unit variable to missing when the lab value is missing. The expected unit should be present in PARAM, which requires some additional programming to generate the non-missing unit values and assign to the PARAM values for all records within a lab test. This code shows using PROC SQL to derive the non-missing conventional units (in light blue) and the SI units (in red) and merging them on to the lab data by LBTESTCD. This join will populate the values for all records of LBTESTCD. Note that LBORRESU and LBSTRESU are dropped from the first LB data set so that PROC SQL does not give a warning for the variables already existing when they are joined on from the sub-queries.

```
PROC SQL ;
  create table lb1 as
    select a.*, lborresu, lbstresu
      from lb (drop=lborresu lbstresu) a
      LEFT JOIN
      (select distinct lbtestcd, lborresu
        from lb
        where lborresu ne ''
      ) b
      on a.lbtestcd=b.lbtestcd
      LEFT JOIN
      (select distinct lbtestcd, lbstresu
        from lb
        where lbstresu ne ''
      ) c
      on a.lbtestcd=c.lbtestcd
    order by usubjid ;
quit ;
```

DATA CHECKS

While there are edit checks for lab data, there are some simple checks that can be done in programming in addition to the edit checks to catch data errors, multiple units within a lab test, and conversion errors. It is also a good idea to check the ANALYTENAME and SASNAME values as incorrect values for these occasionally arise.

CHECK FOR DIFFERING UNITS, NORMAL RANGES AND OUTLIER VALUES

PROC SQL is a great tool to quickly and easily generate checks on units, normal ranges, and outlier values. Similar checks can be performed using other procedures, such as PROC FREQ or PROC MEANS.

This first query gives a frequency listing of the conventional and standard units for each lab test. The outer query (in grey) can be added for a quick look to see if a lab test has more than one original or standard unit.

```
PROC SQL ;
  select *, count(distinct lborresu) as nlborresu,
            count(distinct lbstresu) as nlbstresu
   from (
    select distinct lbtestcd, lbtest, lborresu, lbstresu, count(*) as n
      from lb
      group by lbtestcd, lbtest
   ) ;
quit ;
```

The first query gives a frequency listing of the original and standard units for each lab test. The outer query (in grey) can be added to count the number of different units for a lab test for a quick look for possible data errors.

LBTESTCD	LBTEST	LBORRESU	LBSTRESU	n	nlborresu	nlbstresu
ALB	Albumin			1162	1	1
ALB	Albumin	g/dL	g/L	1162	1	1
ALP	Alkaline Phosphatase			1162	1	1
ALP	Alkaline Phosphatase	U/L	U/L	1162	2	1
ALP	Alkaline Phosphatase	U/dL	U/L	1162	2	1
BASO	Basophils			1166	1	1
BASO	Basophils	%	10^9/L	1166	1	1

Table 1. Output results from PROC SQL query checking lab units

The next two queries check the normal ranges and the minimum and maximum values, looking for possible outliers, for each lab test. One query displays the values for conventional units and the other displays SI units. An outer query can be added, similar to above, to add a count of differing low and high values within a lab test to quickly determine if there are multiple ranges present. Lab tests with multiple normal ranges are highlighted in the output results.

```
PROC SQL ;
  select distinct lbtestcd, lbornrlo, lbornrhi, min(lborres) as minval,
      max(lborres) as maxval
    from lb
    group by lbtestcd
  ;
```

```
  select distinct lbtestcd, lbstnrlo, lbstnrhi, min(lbstresn) as
minval, max(lbstresn) as maxval
    from lb
    group by lbtestcd
  ;
quit ;
```

LBTESTCD	LBORNRLO	LBORNRHI	minval	maxval
ALB	3.5	5.2	3.3	49
ALP	35	104	16	144
ALP	40	129	16	144
BANDS	0	5	1	1
BASO	0	2.4	0	2

Table 2. Output results from PROC SQL query checking range and outliers in conventional units

LBTESTCD	LBSTNRLO	LBSTNRHI	minval	maxval
ALB	35	52	33	490
ALP	35	104	16	144
ALP	40	129	16	144
ANC	1.1	7.6	0.999	13.5575
ANC	1.5	8.1	0.999	13.5575

Table 3. Output results from PROC SQL query checking range and outliers in SI units

LAB RANGES AND COMMON TERMINOLOGY CRITERIA FOR ADVERSE EVENTS (CTCAE) GRADING

There are two important variables used to quickly determine if an investigational product is causing adverse reactions in a subject. There is a lab value indicator that simply flags whether the lab value is within, below or above the normal range for that lab test, and a CTCAE grade variable which uses published ranges from the U.S. Department of Health and Human Services.

The raw lab data usually includes a variable with the reference range flag, LABFLAG. This is an important variable to check as errors can occur. To check the raw LABFLAG values, simply create another flag variable by comparing the lab value to the normal range low and high value in the raw data set and compare the values to the raw LABFLAG values. When creating the check variable, be sure to determine whether the conventional or SI unit variables were used in deriving the LABFLAG variable in the raw data as sometimes when the conversion occurs, a value that is near the low or high cutoff may be out of range in one unit and within range in the other. The following code creates the indicator variable and the check variable and a PROC FREQ can be run to check the values for any discrepancies. In this example, the LABFLAG was derived using conventional units, so the check variable is derived using the conventional unit value and range variables.

```
/* set lab value indicator from raw labflag variable */
  if labflag eq '+' then lbnrind = 'HIGH' ;
  else if labflag eq '-' then lbnrind = 'LOW' ;
  else if labflag eq '0' then lbnrind = 'NORMAL' ;
/* create flag from lab range values to check labflag */
  length indck $10 ;
  if n(numericvalue, lablow, labhigh) = 3 then do ;
    if numericvalue < lablow then indck = 'LOW' ;
    else if lablow <= numericvalue <= labhigh then indck='NORMAL' ;
    else if numericvalue > labhigh then indck = 'HIGH' ;
  end ;
```

The CTCAE grade is an important value to analyze as the grades have been determined based on the severity of the clinical symptoms associated with the lab value, which is, in effect, an adverse event. Some lab tests incur adverse reactions when their levels rise, and other when they drop, and some for both directions. Grades range from a value of one, for mild symptoms, to five, which is death related to the adverse event.

CTCAE grades require a comparison of the lab value in a specific unit, many of which allow for comparison in conventional or SI units, however, it is common practice to use the SI unit values. Some of the comparisons are based on specific ranges, and many are based on multiples below/above the lower/upper limit of the normal range for the lab test, i.e. Grade 1 range = >ULN-1.5xULN, Grade 2 range = >1.5xULN-2.5xULN, where ULN is the upper limit of the normal range. While the actual code for this is outside the scope of this paper, a macro to derive the CTCAE grades will be a standard that can be used across projects and automate this task. The lab test and its corresponding ranges for each grade can be entered in to an Excel spreadsheet and converted in to a SAS® data set and easily merged with the lab data to derive the CTCAE grade values.

TABLES, FIGURES, AND LISTINGS (TFLS)

Whether lab data is in a legacy style analysis data set, SDTM or ADaM format, the data should analysis ready when it is used in a TFL program. ADLB will have your analysis variables derived for much simpler programming in TFL programs, whereas, legacy and SDTM data sets will require derivations within the TFL program. Regardless of derivations in the TFL program, one important detail needed in a TFL program for lab data is the precision to use to display each lab parameter. Standard practice for statistical analysis determines decimal places displayed based on the number of decimals in the raw data. There are standard numbers of decimals for each lab parameter; however, values often come in the raw data with differing number of decimals based on the lab parameter units. Formatting the display values can be easily accomplished by creating an Excel spreadsheet with each lab parameter, units and the number of decimal places. This can be read in to a SAS® data set to generate macro variables to use as the formats for display.

This example uses PROC SQL to create the macro variables. It generates a list of macro variables containing all of the lab test values and the corresponding formats. The macro variables for lab test are &lbtest1 through &lbtest*n*, where n is the number of lab tests and derived in to the macro variable &nlb. The corresponding macro variables containing the formats are &lbfmt1 through &lbfmt*n.*

```
PROC SQL noprint ;
  select strip(put(count(distinct lbtestcd), best.)) into :nlb
    from formats
  ;
  select distinct lbtestcd, lbformat into :lbtest1 - :lbtest&nlb.,
:lbfmt1 - :lbfmt&nlb.
    from formats
  ;
quit ;
```

When generating the tables for each lab parameter, the macro variables can be used to subset the data by lab test and apply the corresponding format to the lab values. Notice that two periods are required after the lab format macro variable, one to mark the end of the macro variable, and the second to complete the format name.

```
DATA lab1 ;
  set alllabs (where=(lbtestcd eq "&&lbtest&i")) ;

  format lbstresn &&lbfmt&i.. ;
run ;
```

CONCLUSION

Working with lab data can be a daunting task even for an experienced programmer. There is much more to working with lab data than the programming. Lab data sets are large and encompass many aspects that take time to learn, such as having consistent values for the lab test name and code value, ensuring the category is correct, i.e. Hematology or Chemistry, having the same units and normal range values and possible outliers which could be data errors. This paper gives some tips on several items to check in the lab data to ensure the data being analyzed is accurate and meaningful.

RECOMMENDED READING

Base SAS 9.2 Procedures Guide, PROC SQL

REFERENCES

U.S. Department of Health and Human Services, Common Terminology Criteria for Adverse Events (CTCAE) Version 4.0, 2010

CONTACT INFORMATION

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

Name: Amie Bissonett
Enterprise: inVentiv Health Clinical
E-mail: amie.bissonett@inventivhealth.com