Paper PP09

The First Steps in Laboratory Dataset

Karolina Filar, Quanticate, Warsaw, Poland
Paulina Kuczkowska, Quanticate, Warsaw, Poland

ABSTRACT

The laboratory dataset is one of the core safety datasets. At first glance it may appear intimidating, with multiple tests and visits per patient. We will present a few checks that are worth applying at the very beginning of programming work – this may be an input to a standardised process of the domain validation. The first step should be identifying the lab test that will be needed for the reports and finding out whether there have been inconsistencies in coding. It is also a good idea to check specimens of chosen lab tests. Next step is recognising local and central laboratories, since local laboratories often use non-standard units or require different algorithms. It is vital to identify values with non-standard units and develop a process of converting them. Another matter worth considering is what algorithm should be applied for baseline values and how to deal with subjects with no baseline or post-baseline results.

INTRODUCTION

In the accompanying poster we explained the first checks that can be done when working on the laboratory dataset. In this paper we will present some example tools that can be used to make these checks faster and more efficient across studies. Conversion of units may be time-consuming, especially if each non-standard unit is handled separately. This would also significantly increase the size of the SAS program and make the code cumbersome to read. It is therefore worth considering writing a reusable program for unit conversion.

CONVERSION PROCESS

The initial step for the conversion process would be making sure that the laboratory dataset contains the standard unit for each test – the unit that will be used for the reporting. Exceptions are laboratory tests for which units are not required. It is a good idea to keep a list of reporting units in an external file (.txt, .csv, .xls), since in this form it can be easily read in SAS and transformed to a SAS dataset; moreover, all potential updates will require only an update to the external file and rerun of already created code. This is the approach we follow in this article.

CONVERSION FACTORS – DATASET REQUIREMENTS

If the dataset of conversion factors is not provided to the programming team, it can be created by programmers and submitted for clinical review and confirmation. Below are the minimum requirements for this dataset.

For conversions which are not dependent on the test (for example g/dL to g/L), the dataset with conversion factors should contain at a minimum:

- original unit,
- conversion factor,
- reporting unit.

For conversions specific to the lab tests the dataset should additionally contain variable(s) allowing identification of the test, such as the Lab Test Code or the Lab Test Name.

The final dataset should contain unique records only, since the laboratory records may get duplicated otherwise, and if the conversion and standard units were to alter at some point in the future, only one section of the dataset would require updating.

ADDING REPORTING UNITS – EXAMPLE MACRO

The below macro call can be used for merging the laboratory dataset with the dataset containing reporting units:

```
/* Macro for merging laboratory dataset (in_ds) with dataset containing reporting
units (unit_ds) for each test.
Datasets are merged by common variables (byvars) which:
```

```
- are specific to the project,
- identify unique lab test.
Example merge key can be:
- Lab Test Name and Specimen Type, Lab Test Code, LOINC code.
'all' variable specifies the content of the output dataset (out_ds):
- only records with reporting unit found in unit_ds dataset (all=N) or
- all records, irrespective of the corresponding reporting unit found in unit_ds
dataset or not */

%macro std_units(in_ds=lb, byvars=lbtestcd, unit_ds=units, out_ds=lb_unit, all=Y);

       proc sort data=&in_ds. out=&in_ds.s;
              by &byvars.;
       run;

       proc sort data=&unit_ds. out=&unit_ds.s;
              by &byvars.;
       run;

       data &out_ds.;
              merge &in_ds.s(in=a) &unit_ds.s(in=b);
              by &byvars.;

              %if &all.=N %then
                     %do;
                            if a and b;
                     %end;
              %else
                     %do;
                            if a;
                     %end;
       run;

%mend std_units;
```

ADDING CONVERSION FACTORS – EXAMPLE MACRO

Once the reporting units are in the dataset, the next step is to use the conversion dataset to obtain the conversion factor for each pair of original and standard unit, as in the code below:

```
/*
lb - laboratory dataset
conv - conversion dataset
factor - conversion factor variable
org_unit - original unit variable
rep_unit - reporting unit variable
lbtestcd - lab test code variable, in this example it is identifying variable for lab
test. Lbtestcd is used to assign factors for conversion dependent on lab test
(a.lbtestcd=b.lbtestcd); for conversion not dependent on lab test lbtestcd is missing
in conv dataset.
 */

proc sql;
       create table lb_conv as select a.*, b.factor
              from lb as a left join conv as b
                     on upcase(a.org_unit)=upcase(b.org_unit)
                            and upcase(a.rep_unit)=upcase(b.rep_unit)
                            and (a.lbtestcd=b.lbtestcd or missing(b.lbtestcd))
                                   order by a.lbtestcd;
quit;
```

UNIT CONVERSION – EXAMPLE MACRO

A simple macro for unit conversion is included below. This macro should be called within a data step and the conversion is applied only to records where:

- the original unit and reporting unit are not equal,
- the result in original unit and factor variables are not missing,
- the result is numeric (i.e. contains only digits and '.').

The conversion of units is done by multiplying the result in the original unit by the specified conversion factor. As a result of calling below macro values are assigned to reporting unit, numeric and character result in the reporting unit, reference ranges for lower and upper limits. The macro parameter &conv_ln can be used to switch off the conversion of lower and upper limits – the default option below is that the conversion is done. Additionally the variable convfl is created to flag records that have been converted and to aid quality checks of the conversion.

In the example below we do not require any specific format for the results, however, there may be standard-specific or study-specific numeric result precision expected.

```
/*
conv_ln – macro parameter, specify if lower and upper limits of range should be
converted (conv_ln=Y)
rep_unit – reporting unit
factor – conversion factor
convfl – flag for converted observations
standard CDISC variables:
lborres – result or finding in original units
lborresu – original units
lbsrtesu – standard units
lbstresn – numeric result/finding in standard units
lbstresc – character result/finding in standard format
lbornrlo – reference range lower limit in original unit
lbornrhi – reference range upper limit in original unit
lbstnrlo – reference range lower limit – standard units
lbstnrhi – reference range upper limit – standard units
*/

%macro conversion(conv_ln=Y);
       %let conv_ln=%upcase(&conv_ln);

       if upcase(lborresu) ne upcase(rep_unit) and cmiss(factor, lborres)=0 and
       findc(lborres, '.', 'dkt')=0 then
              do;
                     lbstresu=rep_unit;
                     lbstresn=input(lborres, best.)*factor;
                     lbstresc=strip(put(lbstresn, best.));

                     %if &conv_ln. = Y %then
                            %do;
                                   lbstnrlo=input(lbornrlo, best.)*factor;
                                   lbstnrhi=input(lbornrhi, best.)*factor;
                            %end;

                     convfl='Y';
              end;
%mend conversion;
```

After conversion of units it is a good practice to check the dataset for outliers as they may indicate data issues in the laboratory dataset. An example of data issue is wrong recording of prefix 'micro' in the unit and using incorrect symbol 'm' instead of 'u'. Unit micromole per litre should be written as 'umol/L' however it may be wrongly assigned as 'mmol/L', resulting in 1000 times higher result than the actual value. Such cases should be reported to Data Management.

CENTRAL AND LOCAL LABORATORIES - UNITS CONCATENATED WITH THE RESULT

While central laboratories ensure a standard approach and provide values in standardised units, it is not always the case with local laboratories. The macro below may be used to separate units concatenated to the value in the result variable (i.e. unit is included in the result variable and the unit variable is missing). Macro should be called within the data step for tests where numeric result is expected. It is assumed that:

- result value begins with number or '.' in the case of missing result,
- unit value starts with a letter – macro will not work properly for units like '10^9/L' or '10^6/UL'.

```
/*
condition - macro parameter, used to specify subset of tests for macro
sepfl – flag for separated observations
old_lborres - holds value of lborres before separation
lborres, lborresu – standard CDISC variables, see code above
*/

%macro separate(condition);
       if &condition. then
              do;
                     lborres_=strip(lborres);

                     if anyalpha(lborres)>0
                     and (anydigit(substr(lborres_,1,1))=1 or substr(lborres_,1,1)='.')
                     and missing(lborresu) then
                            do;
                                   old_lborres=lborres;
                                   lborresu = substr(lborres_, anyalpha(lborres_));
                                   lborres = substr(lborres_, 1, anyalpha(lborres_)-1);
                                   sepfl='Y';
                            end;
              end;

       drop lborres_;
%mend separate;
```

Example calls may be:

```
%separate (lbtestcd in ('ALT' 'CA' 'BILI' 'K'))
%separate (lbcat ne 'URINALYSIS')
```

ACKNOWLEDGMENTS

Many thanks to our internal reviewers for their valuable comments: Fiona Brock, Fabien Du Fraisse, Pritesh Acharya, Bibas Timilsina, Tejendra Parikh, Jedrzej Kuskowski.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. The authors can be contacted as follows:

Karolina Filar, Paulina Kuczkowska
Quanticate LTD
Hankiewicza 2
Warsaw, 02-103
Poland
karolina.filar@quanticate.com
paulina.kuczkowska@quanticate.com
http://www.quanticate.com