Paper SI09

# Standardizing SDTM Laboratory (LB) Programming

Murali Marneni, Biogen, RTP, USA

Stephanie Olexovitch, Biogen, Cambridge, USA

## ABSTRACT

In most clinical trials, Laboratory data helps to determine the safety and, in some cases, the effectiveness of drug products. Commonly, laboratory data comes from varied sources such as local labs, central labs, and external vendors. Mapping multiple sources with different file formats and data collection to SDTM.LB dataset requires overcoming varied challenges and complex programming resulting in a time-consuming effort. Lengthy programming steps are required, and multiple programmers may be involved.

To alleviate these concerns, this paper will discuss the process that was created to standardize SDTM LB creation with the help of macros and relatively minimal programming.

## INTRODUCTION

Lab data plays a crucial role in clinical trials. The results are used to make decisions about enrollment, the safety profile and even efficacy. Lab data can be very complex and is commonly received from multiple sources, which adds to the intricacy of mapping the SDTM.LB domain. Standardizing and developing CDISC compliant SDTM.LB to facilitate a Clinical Study Report is the end goal and proves to be time consuming and hard to maintain if it is not done efficiently from the start.

## PROGRAMMING CHALLENGES

Multiple data sources, such as EDC data and external vendors are needed to program Laboratory data making it a constant challenge to programmatically map different styles of data collection into SDTM.LB The volume of data, in addition to, therapeutic area specific tests contribute to the complexity.

To further complicate matters, different interpretation of the specifications and unique styles of programming often result in lengthier programs and can cause excessive run time to process large datasets due to inefficient code.

The following are the areas we focused on when reinventing the current process:

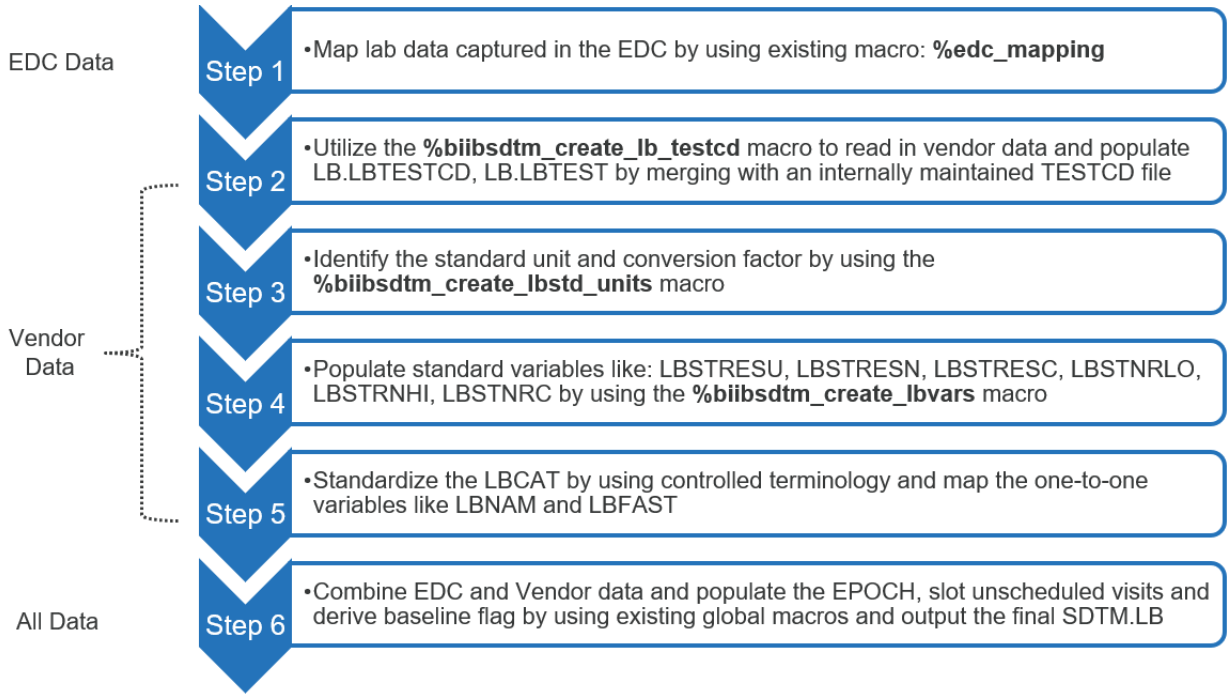
- Lengthy programs increase development time and can create inaccuracies in programming due to different interpretation of the specification.

- Unique data structure for vendors and studies causes low reusability of the programs and costs extra programming hours.
- Varying level of talent within the group and different programming styles often cause issues with the maintenance of complex programs. We have also seen inefficient programs with repetitive code that requires extensive time to research and update throughout the course of the study.

## LB PROGRAM STANDARDIZATION

In order to resolve the challenges and issues, we came up with a solution to standardize the programs, making the programs more user-friendly and significantly reduced programming hours. We identified that developing new global macros and embedding them in a standard template program allows the team to create the study level programs with greater efficiency, maintainability, reusability, and understandability irrespective of vendor file structure.

Standard Program template was created in six steps as below.



## STANDARD PROGRAM TEMPLATE

```
/* **Setup macro call** */

%let chrProgram=biibxyz;
%let chrProtocol=study123;
%let chrAnaltype=sdtm;

%include "/setup.sas";


*************************************************************
Step1: EDC data mapping
*************************************************************;

/***mapping lab data that captured in EDC ***/

%edc_mapping(sdtmdomain = lb);

*************************************************************
VENDOR data mapping
*************************************************************;


/* **Step2: Call Lab testcd macro to pull the SDTM_TESTCD, SDTM_TEST** */


%biibsdtm_create_lb_testcd(dsin= lb_vendor
                          ,dsout= lb_vendor_testcd
                          ,lbtestcd=lbtestcd
                          ,lbtest=lbtest
                          ,lbcat=lbcat
                          ,stdlib=stdlib
                          ,stddsin=biib_master_testcd_test_map_guid
                          ,provider_name="LAB VENDOR"

            );


/* **Step3: Unit Conversions ** */

data lb_vendor_testcd;
set lb_vendor_testcd;

** Standardize LBSPEC into ct:specimen submission values ***;
             if index(lbspec,'Whole Blood')>0 then lbspec = 'BLOOD';
             else lbspec = strip(upcase(scan(lbspec,1,'')));
run;

%biibsdtm_create_lbstd_units(dsin=lb_vendor_testcd
                            ,dsout=lb_vendor_all
                            ,lborresu=lborresu
                            ,lbspec=lbspec
                            ,stdlib=stdlib
                            ,stddsin=BIIB_LAB_CONVERSION_FACTORS
                                        );
```

```
/* ***Step4: Creating SDTM.LB.LBORRESU, LBORNRLO, LBORNRHI, LBSTRESN,
LBSTRESC, LBSTRESU, LBSTNRLO, LBSTNRHI, LBSTNRC**; */

%biibsdtm_create_lb_vars(dsin= lb_vendor_all
                            ,dsout=vendordata
                            ,lbtestcd=lbtestcd
                            ,lbtest=LBTEST
                            ,lborres=LBORRES
                            ,LBORRESU=LBORRESU
                            ,LBORNRLO=_lbornrlo
                            ,LBORNRHI=_LBORNRHI
                            ,LBSTNRC=
                                  );
```

4

```
/* ***Step5: mapping one on one mapping variables based on specification****/
data all_vendor;
       %attrib(vars =  DOMAIN SUBJID LBCAT LBORRES LBREFID LBTSTDTL LBNRIND
LBNAM LBSPEC LBDTC LBLOINC LBDTC LBSPCCND LBSTAT LBREASND);
      set vendordata;

      /*    LBNRIND*/
             if alrtfl ne '' then lbnrind=strip(put(alrtfl,$alrtfl.));
             if alrtfl ne '' and lbnrind = '' then put "warning: Please check
      the format of alrtfl. where more values need to be added";

      /*one to one mapping variables*/
                  LBNAM = strip(lbnam);
                  lbfast = strip(lbfast);
                  if _lbrefid ne . then lbrefid = strip(put(_lbrefid,best.));

      **  LBLOINC **;
                  LBLOINC = strip(LBLOINC);

      ** LBDTC            **;
                  LBDTC = STRIP(LBDTC);

      ** LBSPCCND **;
      LBSPCCND = STRIP(SPECCND);

run;
```

5

```
/* ***Step6: Creating Baseline flag, slotting Unscheduled visits, epoch
variables***/


***Combine EDC and Vendor datasets ***;

data edc_and_vendor;
Set edclb
    all_vendor;
run;
```

6

```
***Merging with dm dataset to get reference start date for deriving variables
***;

%biibsdtm_merge_dm_v2(dsin= edc_and_vendor, dsout=lb_dm, dmloc=crtdir.dc,
dmvars=usubjid rficdtc rfstdtc rfxstdtc rfxendtc rfendtc, dmsubj=Y, debug=N);

***Calling EPOCH MACRO** *;
%epoch(date = lbdtc);

** **UNSCEDULED slotting based on scheduled visit macro
%biibsdtm_unscheduled_visit_v2(DSIN=epoch
      ,DSOUT=lb_uns
      ,UNSCTEXT=%STR(UNS)
      ,VISITVAR=_visit_
      ,BASENUM=100000
      ,DATE= lbdtc
      ,STDY=lbdy
      ,SORTVAR=
      );


*****Assign baseline flag using standard macro. update timevars or groupby
variables as needed****/

%biibsdtm_create_blfl(dsin= lb_uns , dsout=lb_base, xx=lb, timevars= lbdtc ,
groupby=usubjid lbtestcd lbcat lbspec lbmethod lbtstdtl,
      date=lbdtc, result= lborres, rule=, refdsn=crtdir.dm, refdt=rfxstdtc);


******Derive sequence variable ***;

%seq(dsin=lb_base,dsout=final_lb, byvars=&attrib_dssort., seqby=usubjid,
seqvar=lbseq, dropvars=);

***Output SDTM.LB*****;

data output.lb;
      set final_lb;
run;

************************************************************
End of the program
************************************************************;
```

## STANDARD PROGRAM TEMPLATE DETAILS

### STEP 1

Use an existing macro to map the EDC data to and interim LB dataset.

**STEP 2**

**%biibsdtm_create_lb_testcd** is used to merge external vendor Lab data with internally maintained TESTCD mapping guidance file to pull the respective SDTM_TEST, SDTM_TESTCD information which is later mapped to SDTM.LB.LBTEST, LB.LBTESTCD. The macro can be used on any vendor file that has variable names in any naming convention. This macro also outputs an excel file with tests that do not have standard SDTM.LB.LBTEST, LB.LBTESTCD values with respected to the collected TEST/TESTCD value. The excel file is then reported to the governance team, who will provide guidance on the appropriate SDTM.LB.LBTEST, LB.LBTESTCD values.

Description of macro variables:

| | |
|---|---|
| DSIN | Name of input dataset from work library |
| DSOUT | Name of output dataset |
| LBTESTCD | Name of LBTESTCD variable in input dataset |
| LBTEST | Name of LBTEST variable in input dataset |
| LBCAT | Name of LBCAT variable in input dataset |
| STDLIB | Library name of internally maintained TESTCD mapping guidance file |
| STDDSIN | Dataset name in STDLIB Library |
| PROVIDER_NAME | Provide **UPCASE** value of provider name (vendor name) present in STDDSIN dataset based on study external vendor. |

**STEP 3**

**%biibsdtm_create_lbstd_units** was developed to get the conversion factor and standard unit with respect to the original collected unit by merging with an internal Biogen resource file (unit conversion mapping file). This macro takes the output dataset generated from Step 2 as an input dataset. This macro output the excel file with units that do not have standard unit and conversion factor with respect to the collected unit for specific test. This helps to send that report to governance team to get standard unit and conversion factor values with resect to the collected unit for specific test.

Description of macro variables:

| | |
|---|---|
| DSIN | Name of input dataset from work library |
| DSOUT | Name of output dataset |

| | |
|---|---|
| LBORRESU | Name of LBORRESU variable in input dataset |
| LBSPEC | Name of LBSPEC variable in input dataset |
| STDLIB | Library name of unit conversion mapping file |
| STDDSIN | Dataset name in STDLIB Library |

**STEP 4**

**%biibsdtm_create_lb_vars** uses step 3 output dataset as an input dataset, this macro was developed for the purpose of populating the standard variables by using the conversion factor and standard unit.

Description of macro variables:

| | |
|---|---|
| DSIN | Name of input dataset from work library |
| DSOUT | Name of output dataset |
| LBTESTCD | Name of LBTESTCD variable in input dataset |
| LBTEST | Name of LBTEST variable in input dataset |
| LBORRES | Name of LBORRES variable in input dataset |
| LBORRESU | Name of LBORRESU variable in input dataset |
| LBORNRLO | Name of LBORNRLO related variable in input dataset |
| LBORNRHI | Name of LBORNRHI related variable in input dataset |
| LBSTNRC | Name of LBSTNRC related variable in input dataset. If input dataset does not have LBSTNRC related variable then leave it as blank, then macro will create based on character values in original reference low range and high range. |

**STEP 5**

Create the one-to-one mapping variables as per source data collection and mapping specification.

**STEP 6**

Set the intermediate EDC and Vendor datasets together. Populate the EPOCH variable, slot unscheduled visits, create the baseline flag, populate SEQ variable and output the LB dataset to output library.

## CONCLUSION

As Lab dataset programming is often more complex, this programming technique along with a standard programming template helps to resolve some of the programming challenges and utilizes a highly efficient program that is easy to maintain. This technique also helps to achieve our goals and shorten the programming timelines.

## REFERENCES


1. CDISC, SDTM controlled terminology.
2. CDISC, SDTM IG 3.3 package.
3. FDA. Study Data Technical Conformance Guide - Technical Specification Document. March 2019
4. FDA CDER/CBER. Position on Use of SI Units for Lab Tests. October 25, 2013
5. FDA. Recommendations for the Submission of LOINC® Codes in Regulatory Applications to the U.S. Food and Drug Administration. November 2017


## ACKNOWLEDGEMENTS


We would like to acknowledge all team members and leadership team for the support and endorsement.


## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please contact the author at


Name: Murali Marneni
Email: murali.marneni@biogen.com

Name: Stephanie Olexovitch
Email: stephanie.olexovitch@biogen.com



SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.