PharmaSUG 2015 - Paper BB11

Generic Macros for Data Mapping

Qian Zhao, J&J Consumer Companies, Inc., Morris Plains, NJ

Jun (John) Wang, J&J Consumer China Ltd, Shanghai, China

Ruofei Hao, J&J Consumer Companies, Inc., Morris Plains, NJ

ABSTRACT

Data mapping and its QC can be a daunting task. When data are mapped, e.g., to SDTM, the variable values, variable names, variable types, and data structures, etc., may change. Writing generic macros to perform data mapping and QC of the mapping that work for any data set may seem impossible. This paper explores a technique to use the combination of external control files and generic macros to perform data mapping and QC. This technique can be extended to other uses.

This paper uses two examples to illustrate the concept.

In the first example, we will demonstrate how to QC a data set mapped from a horizontal structure to a vertical structure by isolating a single variable at a time and circulating through all variables.

In the second example, we will demonstrate how to write a macro against data mapping specification to generate mapping program.

INTRODUCTION

Data mapping and its QC can be a daunting task. When data are mapped, e.g., to SDTM, the variable values, variable names, variable types, and data structures, etc., may change. Writing generic macros to perform data mapping and QC of the mapping that work for any data set may seem impossible. This paper explores a technique to use the combination of external control files and generic macros to perform data mapping and QC. This technique can be extended to other uses.

We can illustrate the concept by two examples below.

EXAMPLE 1: QC OF DATA MAPPING

In this first example, we will demonstrate how to QC a data set mapped from a horizontal structure to a vertical structure by isolating a single variable at a time and circulating through all variables.

Assume the vital signs were collected in horizontal structure and then mapped to SDTM:

VS_H (Vital Signs collected in horizontal structure)							
SUBJID	USUBJID	VISITNUM	WTVAL	WTUNIT	HTVAL	HTUNIT	…

→ (mapped to)

→

VS (transposed to conform with SDTM)							
USUBJID	VISITNUM	VSTESTCD	VSORRES	VSORRESU	VSSTRESC	VSSTRESN	VSSTRESU
		WEIGHT					
		HEIGHT					
		…					

After mapping, data structure, and variable value, name, and type changed. So, it is impossible to directly compare the two data sets.

However, if we can carve out a portion, e.g., for WEIGHT:

VS_H (Vital Signs collected in horizontal structure)

SUBJID	USUBJID	VISITNUM	WTVAL	WTUNIT	HTVAL	HTUNIT	…

→ (mapped to)

VS (transposed according to SDTM)

USUBJID	VISITNUM	VSTESTCD	VSORRES	VSORRESU	VSSTRESC	VSSTRESN	VSSTRESU
		WEIGHT					
		HEIGHT					
		…					

The comparison (i.e., QC of the mapping) can be made for the carved-out portion, e.g., WEIGHT in original unit vs WTVAL by the following simple program:

```
Libname data1 "e:\study1\original";
Libname data2 "e:\study1\mapped";

proc sort data=data1.vs_h out=vs_h;
  where wt^=.; * missing weight was not mapped to VS;
  by usubjid visitnum;
run;

proc sort data=data2.vs out=vs;
  where vstestcd='WEIGHT';
  by usubjid visitnum;
run;

data check_mapping_result;
  merge vs_h(in=_a) vs(in=_b);
  * for simplicity in this paper, assuming key variables are the same;
  by usubjid visitnum;
  if round(wt,0.1) ^= round(input(vsorres,best6.),0.1);
run;
```

Following the same steps, we can check HEIGHT in standard unit vs HTVAL:

VS_H (Vital Signs collected in horizontal structure)

SUBJID	USUBJID	VISITNUM	WTVAL	WTUNIT	HTVAL	HTUNIT	…

→ (mapped to)

VS (transposed according to SDTM)

USUBJID	VISITNUM	VSTESTCD	VSORRES	VSORRESU	VSSTRESC	VSSTRESN	VSSTRESU
		WEIGHT					
		HEIGHT					
		…					

```
Libname data1 "e:\study1\original";
Libname data2 "e:\study1\mapped";

proc sort data=data1.vs_h out=vs_h;
  where ht^=.; * missing height was not mapped to VS;
  by usubjid visitnum;
run;

proc sort data=data2.vs out=vs;
```

```
    where vstestcd='HEIGHT';
    by usubjid visitnum;
  run;

  data check_mapping_result;
    merge vs_h(in=_a) vs(in=_b);
    * for simplicity in this paper, assuming key variables are the same;
    by usubjid visitnum;
    * original Height in inches, standard Height in centimeters;
    if round(htval,0.1)^=round(vsstresn/2.54,0.1);
  run;
```

Looping through all pairs (with the proper conversion as needed, e.g., when checking height in standard unit):

- WTVAL vs VSORRES (where VSTESTCD=”WEIGHT”)
- WTVAL vs VSSTRESC (where VSTESTCD=”WEIGHT”)
- WTVAL vs VSSTRESN (where VSTESTCD=”WEIGHT”)
- WTUNIT vs VSORRESU (where VSTESTCD=”WEIGHT”)
- WTUNIT vs VSSTRESU (where VSTESTCD=”WEIGHT”)
- HTVAL vs VSORRES (where VSTESTCD=”HEIGHT”)
- HTVAL vs VSSTRESC (where VSTESTCD=”HEIGHT”)
- HTVAL vs VSSTRESN (where VSTESTCD=”HEIGHT”)
- HTUNIT vs VSORRESU (where VSTESTCD=”HEIGHT”)
- HTUNIT vs VSSTRESU (where VSTESTCD=”HEIGHT”)
- …

the QC of mapped vital signs data can be completed.

We can specify the above pairs in an external control file (data mapping QC specification), e.g., a spreadsheet:

lib1	lib2	dt1	dt2	by1	by2
e:\study1\original	e:\study1\mapped	vs_h	vs	usubjid visitnum	usubjid visitnum
e:\study1\original	e:\study1\mapped	vs_h	vs	usubjid visitnum	usubjid visitnum
e:\study1\original	e:\study1\mapped	vs_h	vs	usubjid visitnum	usubjid visitnum
e:\study1\original	e:\study1\mapped	vs_h	vs	usubjid visitnum	usubjid visitnum
e:\study1\original	e:\study1\mapped	vs_h	vs	usubjid visitnum	usubjid visitnum
e:\study1\original	e:\study1\mapped	vs_h	vs	usubjid visitnum	usubjid visitnum
e:\study1\original	e:\study1\mapped	vs_h	vs	usubjid visitnum	usubjid visitnum
e:\study1\original	e:\study1\mapped	vs_h	vs	usubjid visitnum	usubjid visitnum
e:\study1\original	e:\study1\mapped	vs_h	vs	usubjid visitnum	usubjid visitnum
e:\study1\original	e:\study1\mapped	vs_h	vs	usubjid visitnum	usubjid visitnum

cont’d

whr1	whr2	var1	var2
wt^=.	vstestcd='WEIGHT'	wt	vsorres
wt^=.	vstestcd='WEIGHT'	wtunit	vsorresu
wt^=.	vstestcd='WEIGHT'	wt	vsstresc
wt^=.	vstestcd='WEIGHT'	wt	vsstresn
wt^=.	vstestcd='WEIGHT'	wtunit	vsstresu
ht^=.	vstestcd='HEIGHT'	ht	vsorres
ht^=.	vstestcd='HEIGHT'	htunit	vsorresu
ht^=.	vstestcd='HEIGHT'	ht	vsstresc
ht^=.	vstestcd='HEIGHT'	ht	vsstresn
ht^=.	vstestcd='HEIGHT'	htunit	vsstresu

cont’d

ChkExpression	ChkID	Message
if round(wtval,0.1) ^= round(input(vsorres,best6.),0.1)	WT01	Weight in Original Unit mapping discrepancy
if upcase(wtunit) ^= vsorresu	WT02	Weight Original Unit mapping discrepancy

ChkExpression	ChkID	Message
if round(wtval,0.1) ^= round(input(vsstresc,best6.)*2.2046,0.1)	WT03	Weight (C) in Standard Unit mapping discrepancy
if round(wtval,0.1) ^= round(vsstresn*2.2046,0.1)	WT04	Weight (N) in Standard Unit mapping discrepancy
if wtunit ^= put(vsstresu,$kgtolb.)	WT05	Weight Standard Unit mapping discrepancy
if round(htval,0.1) ^= round(input(vsorres,best6.),0.1)	HT01	Height in Original Unit mapping discrepancy
if upcase(htunit) ^= vsorresu	HT02	Height Original Unit mapping discrepancy
if round(htval,0.1) ^= round(input(vsstresc,best6.)/2.54,0.1)	HT03	Height (C) in Standard Unit mapping discrepancy
if round(htval,0.1) ^= round(vsstresn/2.54,0.1)	HT04	Height (N) in Standard Unit mapping discrepancy
if trim(htunit)||'cm' ^= 'in'||vsstresu	HT05	Height Standard Unit mapping discrepancy

Table 1. Data Mapping QC Specification

and wrap the above simple program into a macro:

```
%macro Data_Mapping_QC;

* tot_var is the total number of rows in the above QC specification;
%do i=1 %to &tot_var;

libname data1 "&&lib1&i";
libname data2 "&&lib2&i";

proc sort data=data1.&&dt1&i out=&&dt1&i;
 %if %length(&&whr1&i)>0 %then %do;
  where &&whr1&i; * to identify the matching observations as in dt2;
 %end;
  by &&by1&i;
run;

proc sort data=data2.&&dt2&i out=&&dt2&i;
 %if %length(&&whr2&i)>0 %then %do;
  where &&whr2&i; * to identify the matching observations as in dt1;
 %end;
  by &&by2&i;
run;

data check_mapping_result&i;
  merge &&dt1&i(in=_a) &&dt2&i(in=_b);
  * for simplicity in this paper, assuming by1 and by2 are the same;
  * if not, extra steps are needed;
  by &&by1&i;
  &&ChkExpression&i
  then do;
    …
    statements (e.g., to make the check output more user friendly)
    …
    ChkExpression = "&&ChkExpression&i";
    ChkID = "&&ChkID&i";
    Message = "&&Message&i";
    output;
  end;
run;

* release libname association;
libname data1;
libname data2;
```

```
%end; * end of loop for tot_var;

%mend Data_Mapping_QC;
```

The macro parameters can be passed in by importing the QC specification (the external control file). The number of loops is controlled by the number of rows in the specification. Another way to view the external control file is that the columns define macro parameters, and the rows pass parameter values to each macro call.

Notice the four sets of columns in the QC specification: lib, dt, by + whr, and var. These are the four-dimensions to identify a data field, i.e., location (drive:\path, lib), data set (dt), row (observation, by+whr) and column (variable name, var). With these four dimensions identified, the comparison between the original and the corresponding mapped data point can be performed.

Also notice that the macro handles the part that is common, or unchanging, from data set to data set, e.g., the sorting, sub-setting, merging; while the control file handles the part that is changing or unknown to the macro. In other words, the macro handles the function of each column of the control file; while the control file handles the contents of each column.

Samples of the data, QC specification, macro and QC summary are attached to this paper. (to view attachment: View -> Show/Hid -> Navigation Panes -> Attachments)

EXAMPLE 2: DATA MAPPING

In this second example, we will demonstrate how to write a macro against data mapping specification to generate mapping program.

Assume we have the following data mapping specification to map VS data to SDTM:

MAPPING_ID	MAPVAR	VARORD	SRCVAR	TGTVAR	VARMAP
0		1	STUDYID	STUDYID	studyid
0		2	assigned	DOMAIN	"VS"
0		3	STUDYID, SUBJID	USUBJID	trim(studyid)||subjid
0		12	VISITNUM	VISITNUM	visitnum
0		13	VISITDAT	VSDTC	put(visitdat,e8601da.)
0		14		VSTPT	
0		15		VSTPTNUM	
1	WTVAL	4	assigned	VSSPID	"01"
1	WTVAL	5	assigned	VSTESTCD	"WEIGHT"
1	WTVAL	6	assigned	VSTEST	"Weight"
1	WTVAL	7	WTVAL	VSORRES	compress(put(wtval,6.1))
1	WTVAL	8	WTUNIT	VSORRESU	upcase(wtunit)
1	WTVAL	9	WTVAL	VSSTRESC	compress(put(wtval*0.45359237,6.1))
1	WTVAL	10	WTVAL	VSSTRESN	wtval*0.45359237
1	WTVAL	11	WTUNIT	VSSTRESU	put(upcase(wtunit),$LbToKg.)
2	HTVAL	4	assigned	VSSPID	"02"
2	HTVAL	5	assigned	VSTESTCD	"HEIGHT"
2	HTVAL	6	assigned	VSTEST	"Height"
2	HTVAL	7	HTVAL	VSORRES	compress(put(htval,6.1))
2	HTVAL	8	HTUNIT	VSORRESU	upcase(htunit)
2	HTVAL	9	HTVAL	VSSTRESC	compress(put(htval*2.54,6.1))
2	HTVAL	10	HTVAL	VSSTRESN	htval*2.54
2	HTVAL	11	HTUNIT	VSSTRESU	put(upcase(htunit),$IncToCm.)

cont'd

VARLABEL	VARTY	VARLEN	ENABLED
Study Identifier	Char	20	Y
Domain Abbreviation	Char	2	Y
Unique Subject Identifier	Char	30	Y
Visit Number	Num		Y
Date/Time of Measurements	Char	20	Y
Planned Time Point Name	Char	20	N

VARLABEL	VARTY	VARLEN	ENABLED
Planned Time Point Number	Num		N
Sponsor-Defined Identifier	Char	10	Y
Vital Signs Test Short Name	Char	8	Y
Vital Signs Test Name	Char	40	Y
Result or Finding in Original Units	Char	40	Y
Original Units	Char	20	Y
Character Result/Finding in Std Format	Char	40	Y
Numeric Result/Finding in Standard Units	Num		Y
Standard Units	Char	20	Y
Sponsor-Defined Identifier	Char	10	Y
Vital Signs Test Short Name	Char	8	Y
Vital Signs Test Name	Char	40	Y
Result or Finding in Original Units	Char	40	Y
Original Units	Char	20	Y
Character Result/Finding in Std Format	Char	40	Y
Numeric Result/Finding in Standard Units	Num		Y
Standard Units	Char	20	Y

Table 2. Data Mapping Specification

We can write a macro against this specification to generate a mapping program for VS:

```
proc format;
  value $LbToKg  'LB' ='kg'; * weight unit conversion;
  value $IncToCm 'IN' ='cm'; * height unit conversion;
run;

%macro mapping(
  mapping_spec=,
  domain=,
  libin=,
  dsin=,
  libout=,
  dsout=,
  pgm_out=);

proc import datafile="&mapping_spec"
            out=mapping
            dbms=xls
            replace;
            sheet="&domain";
            textsize=2048;
run;

proc sql noprint;
  * produce variable list and label;
  select distinct
         strip(upcase(tgtvar)),
         strip(upcase(tgtvar))||"='"||strip(varlabel)||"'", varord
         into :varlist separated by ' ',
              :varlab separated by ' ', :varord1 separated by ' '
    from mapping
    where enabled="Y" and varlabel^=' '
    order by varord;

  * produce length for character variables;
  select distinct
         strip(upcase(tgtvar))||" $"||strip(put(varlen,best3.)), varord
         into :varlen separated by ' ', :varord2 separated by ' '
    from mapping
    where enabled="Y" and varty="Char" and varlen^=.
    order by varord;
```

```
quit;

proc sort data=mapping out=mapping1;
  by mapping_id varord;
  where enabled="Y";
run;

filename pgm "&pgm_out"; * program to be generated below;

* The step below is to generate a mapping program according to the mapping
specification passed in;
data _null_;
  set mapping1 end=eof;
  by mapping_id;

  file pgm;

  length sas_code test_condition $1000;

  if _n_=1 then do;
     put @1 "libname in ""&libin"";" /
         @1 "libname out ""&libout"";" //
         @1 "data out.&dsout;" /
         @3 "format &varlist;" /
         @3 "length &varlen;" /
         @3 "set in.&dsin;" /
         @3 "keep &varlist;" /
         @3 "label &varlab;" /;
  end;

  sas_code = catx("=", tgtvar, varmap);

  if mapping_id=0 then do; * mapping_id=0: variables not to be transposed;
     put @3 sas_code +(-1) ";";
     * write a blank line to separate coding blocks;
     if last.mapping_id then put;
  end;

  else do;
     test_condition = "if not missing (" || strip(mapvar) || ") then do;";
     if first.mapping_id then
        put @3 test_condition;
     put @6 sas_code +(-1) ";";
     if last.mapping_id then
        put @6 "output;" /
            @3 "end;" /;
  end;

  if eof then
     put @1 "run;" //
         @1 "libname in clear;" /
         @1 "libname out clear;";
run;

%include "&pgm_out";
filename pgm clear;

%mend mapping;

%mapping(
  mapping_spec=e:\Data Mapping Spec.xls,
  domain=VS,
  libin=e:\study1\original,
```

```
dsin=vs_h,
libout=e:\study1\mapped,
dsout=vs,
pgm_out=e:\VS Mapping.sas)
```

In this example, the macro serves as a compiler of the mapping specification. For each column in the mapping specification, the macro knows its function, though not its contents, and therefore knows how to translate the specification into the programming language.

The benefits of this approach are:

- Fully utilize the specification, e.g., no need to re-type the mapping expressions, variable labels, lengths, etc., in the program.
- Maintain consistency between the specification and the program; therefore changes need to be made only in one place.
- Make the mapping task easier and streamlined by filling a few blanks, e.g., variables to be mapped (MAPVAR), source variable (SRCVAR), mapping expression (VAR_MAP), and variable length (VARLEN). (VARORD, TGTVAR, VARLABEL and VARTY are SDTM standard and need not to be changed.)
- Allow making changes in a central location (i.e., not scattered in different SAS statements and/or steps).
- Obtain better traceability from the source variable to the mapped variable.
- Enable a higher level standardization due to reusability of the mapping specification template and the macro. (note: SDTMIG in the spreadsheet format can be downloaded and used as the base for the mapping specification template.)
- Potentially require lower level of SAS skills.

Samples of the mapping specification, macro and program generated by this macro are attached to this paper. (to view attachment: View -> Show/Hid -> Navigation Panes -> Attachments)

SUMMARY:

Using external control files, standard programming is possible for what seems otherwise impossible.

The macro can handle the parts that do not change, that are machine interpretable; the external control file handles the parts that are changing, or un-expressible to the macro.

For simplicity of this paper, the two examples are simplified cases. The concept demonstrated can be extended to more complicated cases. One can also enhance the two macros to make them more sophisticated.

Other uses of the concept can include, e.g., standardized data processing, using SAS to do SAE reconciliation (match verbatim between clinical database and safety database), passing a large number of different sub-setting conditions to the same data step or procedure, and passing information from analysis result metadata to table generation program (e.g. table number, title, footnote, analysis variable and model statement).

ACKNOWLEDGMENTS

We would like to thank Brenda Zimmerman and J. Tony McGuire for reviewing this paper and providing valuable comments.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. For any questions or sample codes described in this paper, please contact the authors at:

Name: Qian Zhao
Enterprise: J&J Consumer Companies, Inc.
Address: 185 Tabor Road, Morris Plains, NJ 07950
Work Phone: 973-385-2363
E-mail: qzhao1@its.jnj.com

Name: John Wang
Enterprise: J&J Consumer China Ltd
Address: Worldwide Emerging Markets Innovation Center, 3285 Dongchuan Road, Minhang District, Shanghai 200245, China
Work Phone: 86-21-2416-8380
E-mail: jwang19@its.jnj.com

Name: Ruofei Hao
Enterprise: J&J Consumer Companies, Inc.
Address: 185 Tabor Road, Morris Plains, NJ 07950
Work Phone: 973-385-0561
E-mail: rhao@its.jnj.com