Automating the pooling of variables across multiple datasets using Proc SQL and SAS® Macro

Sanjiv Ramalingam, MCS GLOBAL INC.

Abstract

Complex study designs may result in multiple raw datasets being created that have information that is normally present in one dataset present across two or more datasets. This necessitates the creation of a common dataset. To create this common dataset, information first needs to be gathered across studies and a spreadsheet can be created in EXCEL containing the target variables. An algorithm using Proc SQL and SAS Macro is described that reads the spreadsheet and automatically pools the data. The advantage is especially for ISS/ISE studies, code need not be written separately for individual studies.

Building Brick by Brick…

laying down the foundation…

To help understand the technique, a very simple application is considered. Let the objective be to create a mapping of a laboratory dataset. Keeping in mind that when data is "mapped" observations are never lost between the raw and the mapped dataset, the core tasks involved in this mapping spec are to either re-name the source variable or to hard code some of the units for the unit variables for some of the variables. The mapping specification (spec.xls) is then similar in structure to the one below:

protocol	protocol
screen_ing_No	screenno
screen_ing_No	subjid
result_glucose_mg_dL	lbresu1(Hardcode value to mg/dL)
age	age
sex	sex
visit	visit

In the above example the variables in the left are mapped to variable names in the right column. It should also be noted from the above example that there is a one to many mapping, as well as hard coding involved for certain laboratory units. As an example the variable, “result_glucose_mg_dL” will be named in the target dataset as lbresu1 and hardcoded to the unit, “mg/dL”. Alternatively the objective can be stated as re-naming variables and hard coding of certain units.

The above mapping specification in EXCEL is then modified(spec_m.xls) to the one of similar structure as below with three variables named as source, Target and source_o. Basically source_o is the original source variable and source_m is the modified source variable which contains the actual values. In this scenario, variables in the target column will be assigned to the source_m variables and represents the actual name of the variable being mapped. If done manually and for large datasets that involve more than a handful of variables it can be cumbersome trying to type each of the assignments and there also prone to mistakes. The objective then is to automatically make these assignments using the mapping specification document as the source.

Source_m	Target	Source_o
protocol	protocol	protocol
screen_ing_No	screenno	screen_ing_No
screen_ing_No	subjid	screen_ing_No
“mg/dL”	lbresu1	result_glucose_mg_dL
Age	age	age
Sex	sex	sex
Visit	visit	visit

Algorithm and Code:

```
/*
```

a. The attributes (PROC CONTENTS) of the raw dataset are obtained keeping only the name and label. The purpose is explained later in the algorithm.*/

```
proc sort data=raw.glucose out=raw1 ;
by protocol patient_no_ ;
run;

proc contents data=raw1 out=c_spec1(rename
=(name=source_m label=label_m)keep=name label) ;
run;

/*
```

b. The modified specifications are imported as a dataset(varlist) either using PROC IMPORT or the LIBNAME statement. The LIBNAME statement is used as an example here.

```
*/

libname rawd 'xx.xls';
data varlist;
set rawd.'Var_List$'n ;
run;

/*
```

c. Create a variable that basically assigns, Target=Source_m in the varlist dataset. */

```
Var_1=strip(target)||'='||strip(source_m)||';' ;

/*
```

d. Left join the above dataset (varlist) with the c_spec1 dataset on the condition that the source variables from both the datasets are the same to create a new dataset called mod1. This step is implemented so that there is consistency between the variables in the specifications and the actual raw dataset. It also serves as one of the checks for the mapping specification document. */

```
proc sql;
create table mod1 as
select varlist.source_m,target,var_l,source_o,c_spec1.label_m
from varlist
left join
c_spec1
on m1.source_o=c_spec1.source;
quit;
/*
```

e. Using the mod1 dataset as the source dataset macro variables are created using the PROC SQL and into: combination of statements. */

```
/* macro variable for variable list; "target=source;" */
proc sql;
select var_l
into :var_m separated by ' '
from mod1;
quit;

/* macro variable for keep statement */
proc sql;
select target
into :target_v separated by ' '
from mod1;quit;

/*
```

f. Step e can be repeated for creating labels if necessary. Since the macro variables have been the final dataset can now be created. Since a macro variable has been created that has the target=source assignments, the macro variable(var_m) only needs to be invoked. The length statement can be used to make the length assignments if needed. */

```
data final;
set raw1;
&var_m ;
keep &target_v ;
run;
```

Revisiting the Technique

The core concept of the technique involves the following steps:

a. The mapping specification document is modified such that simple assignments can be made.
b. Creating macro variables for making assignment statements, keep statements and label statements.

The bigger picture …

What if the mapping now needs to be implemented not to create just one dataset but to create multiple such datasets and combine all these datasets into one dataset as in the case of Integrated summary of safety and efficacy analysis datasets.

A typical scenario could be in creating an exposure dataset from two raw datasets say OVEX(overall exposure containing a record per subject per treatment with treatment start and end date) and DEEX (detailed exposure containing dose information on a day to day basis). The two datasets are then set together to form a single dataset. As an example only few variables are considered.

The structure of the modified specification document is then similar to the one below:

STUDY	_DRUGD	_DRUG	_FORMD	_FORM	_REGID
A-101	SDDRUGD	SDDRUG1	SDFORMD	SDFORM	SBDREGD
A-102	SDDRUGD		SDFORMD2	SBFORM	SDDREGD
A-103	SBDRUGD	SDDRUG1	SBFORMD	SBFORM	SBDREGD
A-104		SEDRUG2	SDFORMD	SDFORM2	SBDREG
A-105	SDDRUGD	SDDRUG1	SDFORM	SDFORM	SBDREGD
A-106	SBDRUGD	SDDRUG1	SBFORMD	SDFORM2	SDDREGD

The study column contains the different study identifier variables, _drugd contains the drug decode variables, _drug contains the drug code variables, _formd contains the formulation decode variables, _form contains the formulation code variables and _regid contains the regimen decode information.

Algorithm and Code:

a.

/*As explained in the previous example the modified specification document is imported as a dataset either using the LIBNAME statement or by using the PROC IMPORT procedure. */

```
libname rawd 'xx.xls';
data varlist;
set rawd.'Var_List$'n ;
run;
```

b.

/* As many number of datasets are created, a variable(ds1) is created such that each of the values of the variable identifies each of the datasets that are to be created. This variable is later converted into a macro variable so that when each of the datasets is created they can be automatically set together.

A variable var1 is created such that it resembles an invocation of a macro(cds).

*/

```
data varlist;
length ds1 $10 var1 $500;
set varlist;
ds1='ds'||left(trim(i+1));
i+1;
var1=trim('%cds(ds='||strip(ds1)||','||
'study1='||strip(_study)||','||
'_drug1='||strip (_drugd))||','||
'_drug2='||strip(_drug))||','||
'_form1='||strip(_formd))||','||
'_form2='||strip(_form))||','||
'_regi1='||strip(_regid))
||strip(');')) ;
run;
```

/* The values in the column, var1 of the output of this dataset should resemble the one below:

%cds1(ds=ds1,study1=A101,_drug1=SDDRUGD,_drug2=SDDRUG1,_form1=SDFORMD,_form2=

SDFORM,_regi1=SBDREGD);

*/

c.

/* The next step is to create macro variables from the ds1 and the var1 variable. Note that the var1 variable is converted into a macro variable which has the same name as the macro (cds) that creates the individual datasets*/

```
proc sql;
select ds1 into:dsinp1 separated by ' '
from varlist;
quit;
run;


proc sql;
select var1 into:cds separated by ' '
from varlist;
quit;
run;
```

d.

/*The macro that creates each of the individual datasets can now be created. As opposed to the previous example, PROC SQL is used to create the dataset. As illustrated in the mapping specification document certain variables do not have any mapping variable. This is evident in rows two and four where the _drug1 and _drug2 variables are missing respectively. This necessitates the periodic checking for the availability of variables before the SELECT statement can be used. For the above specification document ,three separate if then else conditional statements are used to check for non-missing variables.*/

```
%macro cds(ds=,
study1=,
_drug1=,
_drug2=,
_form1=,
_form2=,
_regi1=,
);

proc sql ;
create table &ds as
/* none of the variables are missing*/
%if (&_drug1^= and &_drug2^= and form1^= and &_form2^= and &_regi1^=) %then %do;
select usubjid,studyno,&_drug1 as drugd,&_drug2 as drugc,&_form1 as formd,&_form2 as
formc,&_regi1 as regimend
%end;

/* when _drug1 is missing */

%else %if (&_drug1= and &_drug2^= and form1^= and &_form2^= and &_regi1^=
) %then %do;
select usubjid,studyno,&_drug2 as drugc,&_form1 as formd,&_form2 as formc,&_regi1 as
regimend
%end;

/* when _drug2 is missing */

%else %if (&_drug1^= and &_drug2= and form1^= and &_form2^= and &_regi1^=
) %then %do;
select usubjid,studyno,&_drug1 as drugd,&_form1 as formd,&_form2 as formc,&_regi1 as
regimend
%end;
```

```
/*final is the dataset that has all the required raw datasets combined as a single raw dataset */

from final
where _study="&study1";
quit;
run;
%mend cds;


e.
/* The macro can be automatically invoked by addressing the macro variable, cds that has already
been created */

&cds;

f.
/* Each of the individual datasets created can be automatically set together using the macro
variable,dsinp1. As needed the lengths for each of the variables can be set and labels assigned. */

data dspool;
set &dsinp1;
run;
```

Improvements/Limitations

One of the more tedious tasks involved in this technique is the permutation and combination of select statements involved if in the mapping specification document there are a number of cells of missing information in many columns. A separate select statement would then have to be written to address each missing cell. Though it is relatively easy as only a lot of cut and paste and slight modifications are required a better approach is sought. There also maybe instances where sub-setting of data is required. This can be achieved by having a condition column in the specification document itself. Also setting all the required raw datasets into a single raw dataset may not always be preferred. Under these circumstances, for each of the columns(variables) in the specification document a column to specify the source dataset can be used.

Conclusion

Refinements can definitely be made to the technique but nevertheless once the technique is implemented it can be modified to suit many other related tasks and make processes less tedious and save a lot of time.

Author contact

Sanjiv Ramalingam

Consultant (MCS GLOBAL Inc.)

Octagon Research Solutions

Wayne, PA-19087

r.sanjiv@yahoo.com