Paper CT04

Let your SDTM or ADaM SAS® code update itself whenever the Specs change

Joseph Hinson, TalentMine, Raleigh NC, USA

ABSTRACT

Statistical programmers typically use SDTM and ADaM specifications laid out in an MS Excel workbook as a guide for developing SDTM or ADaM dataset SAS® programs. But this "Specs" document usually undergoes several changes throughout the study lifecycle. Often the changes are made by the lead programmer without notifying production and validation programmers. The paper uses Proc SQL and the PE domain as examples for updating the variable attributes LABEL, TYPE, and FORMAT whenever the dataset creation program is run. This is made possible by embedding macro calls in the Proc SQL code. The paper describes how the macro calls use DOSUBL and %SYSFUNC to perform virtual "typing" of code keywords in the Proc SQL code. Such a dynamic code update technique can make SDTM and ADaM programs instantly updated whenever run, thereby reducing coding time as well as preventing mismatches between dataset variable attributes and their specifications.

INTRODUCTION

The paper (a) introduces the concept of "digital robots" – small SAS® macros that use the DOSUBL/SYSFUNC utilities to execute code within program statements, and (b) demonstrates how such digital robots embedded in Proc SQL can make SDTM creation programs self-updating in response to changes in the SDTM specifications (usually called "specs").

STRATEGY

The main idea is to get the row position of a variable in the SDTM specs, use that position number to read the values of the attributes, and to use those values to guide "digital robots" which then type the appropriate SQL lines of code. This paper focuses on just the four variable attributes: LENGTH, TYPE, LABEL, and FORMAT. For LENGTH and LABEL, we simply use a small macro, %RV, ("Digital Robot 1") that does ordinary text substitution using a macro variable, a very basic technique in the use of macros. But for the TYPE attribute on which the FORMAT depends, the macro variable approach is not enough. One needs to determine if the variable had been changed from character to numeric type or vice versa, then use the appropriate PUT or INPUT function to update the variable attribute. This calls for extra processing and therefore a more sophisticated macro ("digital robot 2"). Both macros are called "digital robots" because they are present in the Proc SQL code as MACRO CALLS rather than macro variables.

THE PROCESS

A. OBTAINING SPECS INFO

The first step in the process is to read the SDTM specs Excel file and create macro variables: one containing the entire list of variable names as displayed in the Excel document, another containing a corresponding list of variable labels, another containing the corresponding list of lengths, and a fourth list containing the variable types:

:

```
******************READ IN SDTM EXCEL SPECS*******************************;

%let specspath=%str(&path.\StudySpecs);
%let inspecs=Phuse_Paper_SDTM_Specs.xls;
%let domainsheet=PE;
%let specstab="&domainsheet.$A2:E100"n;
%let outspecs=sdtmspecs;

proc import datafile="&specspath.\&inspecs." out=&outspecs. dbms=xls replace;
   getnames=YES;
   mixed=YES;
   range=&specstab;
run;
```

```
**********MACRO VARIABLES FOR ATTRIBUTE LISTS**********;

proc sql noprint;
   select variable_name  into :varnamelist separated by ","
      from sdtmspecs;

   select length  into :len1 - :len&sysmaxlong.
      from sdtmspecs;

   select type  into :typ1 - :typ&sysmaxlong.
      from sdtmspecs;

   select label  into :lab1 - :lab&sysmaxlong.
      from sdtmspecs;
quit;
```

For LENGTH, LABEL, and TYPE, we simply use a small macro, %RV, called “Digital Robot 1”.

B. DIGITAL ROBOT 1

This first macro, %RV, obtains the row position of a variable in a variable list extracted from the SDTM specs Excel worksheet for a particular domain. The row number is then used to read the LENGTH, LABEL, or TYPE into a macro variable &VX the value of which is then “typed” into an SQL line of code:

```
:
%macro %rv (variable, attribute);
%global xattrib separator ampersand rownumber vx;
%let xattrib=%str(&attribute);
%let separator=%str();
%let ampersand=%nrstr(&);
%let rownumber=%sysfunc(whichc(&variable, &varnamelist));
%let vx=%sysfunc(catx(&separator,&ampersand,&xattrib,&rownumber));
&vx.
%mend %rv;
```

IMPLEMENTATION IN THE SQL CODE:

```
proc sql noprint;
create table PE as
      select
/*(1)================================(STUDYID)==========================*/
      strip(protocol) as STUDYID
length=%rv(STUDYID,len)
%sysfunc(ifc((%rv(STUDYID,typ)='Char'),"format=$%rv(STUDYID,len).","format=%rv(STUDYID
,len)."))
label="%rv(STUDYID,lab)",
```

The IFC function is used to pick the correct format based on the variable TYPE.
However this would only be valid if the variable type exactly matches the type already associated with the variable in the program. If the specs had been updated such that the variable type had changed, text substitution by macro variable would not prevent a SAS® error from occurring. One would need to re-establish the variable type in the program using the PUT or INPUT type conversion functions. This is the role played by the Digital Robot 2.

C. DIGITAL ROBOT 2

This digital robot simply is a small SAS® macro, %GETTYPE, that is able to execute a PUT or INPUT function in the SQL code based on the variable TYPE attribute, using the DOSUBL function and the %SYSFUNC utility which forces execution by DOSUBL. (This macro %GETTYPE also uses the %RV macro to get the TYPE attribute).

```
%let datasetname=PE;
%macro gettype(variable);
%let rx=%sysfunc(dosubl('
data _null_;
length xtype $5 vartext $50;
dsid=open("&datasetname ");
rows=attrn(dsid,"NOBS");
cols=attrn(dsid,"NVARS");
rc=fetchobs(dsid,1);
xnum=varnum(dsid,"&variable");
xtype=vartype(dsid,xnum);
if (xtype="N") and (%rv(&variable,typ)=%str(Char)) then vartext="put(&variable., 8.)
as &variable. ";
else if (xtype="C") and (%rv(&variable,typ)=%str(Num)) then vartext="input(&variable.,
8.) as &variable. ";
else vartext="&variable.";
rc=close(dsid);
call symputx("vtext", vartext,"g");
run;'));
%str(&vtext.)
%mend gettype;
```

IMPLEMENTATION IN THE SQL CODE:

```
/*(3)==================================(SUBJID)=======================================*/
proc sql noprint;
create table PE as select
       %gettype(SUBJID)
length=%rv(SUBJID,len)
%sysfunc(ifc((%rv(SUBJID,typ)='Char'),"format=$%rv(SUBJID,len).","format=%rv(SUBJID,le
n)."))
label="%rv(SUBJID,lab)",
```

In this case, the %GETTYPE macro is also invoked to execute a PUT or INPUT function for variable TYPE conversion.

All variables can have the %GETTYPE macro call in the SQL code, but in this paper, the use would be limited to just the variable SUBJID, which can have either numeric or character variable TYPE.

One could easily demonstrate that whether the variable type in the SDTM specs is flagged “Num” or “Char”, the PE is created without any SAS® error, with SUBJID appearing as either character or numeric as specified in the specs.

A SAMPLE COPY OF THE SDTM SPECS

Dataset	Variable Name	Label	Type	Length
PE	STUDYID	Study Identifier	Char	15
PE	DOMAIN	Domain Abbreviation	Char	2
PE	**SUBJID**	**Subject Identifier**	**Char**	**10**
PE	USUBJID	Unique Subject Identifier	Char	25
PE	PESEQ	Sequence Number	Num	8
PE	PETESTCD	Body System Examined Short Name	Char	40
PE	PETEST	Body System Examined	Char	40
PE	PEORRES	Verbatim Examination Finding	Char	200
PE	PESTRESC	Character Result/Finding in Standard Format	Char	200
PE	PESTAT	Completion Status	Char	8
PE	VISITNUM	Visit Number	Num	8
PE	VISIT	Visit Name	Char	60
PE	PEDTC	Date/Time of Examination	Char	19
PE	PEDY	Study Day of Examination	Num	8

For clarity, not all the columns of the specs are shown. This paper focuses on just three variable attributes: LABEL, TYPE, and LENGTH. The TYPE attribute also enables the FORMAT to be assigned in the SQL code.

THE FULL PROC SQL CODE

```
data rawPE;
infile datalines;
input Protocol $ Domain $ Subjid $ seq ExamSite $ Result $ Status $ VisitName $
ExamDate date9. ExamDay;
datalines;
abc pe 101 1 Head Normal Done Baseline 21Mar1884 -3
abc pe 101 2 Skin Normal Done Baseline 21Mar1884 -3
abc pe 101 3 Neck Normal Done Baseline 21Mar1884 -3
abc pe 102 1 Head Normal Done Baseline 21Mar1884 -3
abc pe 102 2 Skin Acne Done Baseline 21Mar1884 -3
abc pe 102 3 Neck Normal Done Baseline 21Mar1884 -3
abc pe 103 1 Head Normal Done Baseline 21Mar1884 -3
abc pe 103 2 Skin Rash Done Baseline 21Mar1884 -3
abc pe 103 3 Neck Normal Done Baseline 21Mar1884 -3
abc pe 104 1 Head Normal Done Baseline 21Mar1884 -3
abc pe 104 2 Skin Normal Done Baseline 21Mar1884 -3
abc pe 104 3 Neck Normal Done Baseline 21Mar1884 -3
;
run;


%macro CreateSDTM();
```

```
******************READ IN SDTM EXCEL SPECS************************************;

%let specspath=%str(C:\Users\admin\Desktop);

%let inspecs=Phuse_Paper_SDTM_Specs.xlsx;
%let xlsheet=PE;
%let specstab="&xlsheet.$A1:E20"n;
%let outspecs=sdtmspecs;

proc import datafile="&specspath.\&inspecs." out=&outspecs. dbms=xlsx replace;
getnames=YES;
mixed=YES;
range=&specstab;
run;

***************MACRO VARIABLES FOR LIST OF ATTRIBUTES************************;

proc sql noprint;
       select variable_name  into :varnamelist separated by ","
              from sdtmspecs;

       select length  into :len1 - :len&sysmaxlong.
              from sdtmspecs;

       select type  into :typ1 - :typ&sysmaxlong.
              from sdtmspecs;

       select label  into :lab1 - :lab&sysmaxlong.
              from sdtmspecs;
quit;

*====================(DIGITAL ROBOT 1)======================;

%macro rv(var, att);
%global xatt sep amp n vx;
%let xatt=%str(&att);
%let sep=%str();
%let amp=%nrstr(&);
%let n=%sysfunc(whichc(&var, &varnamelist));
%let vx=%sysfunc(catx(&sep,&amp,&xatt,&n));
&vx.
%mend rv;

*============================(DIGITAL ROBOT 2)==================================;
%let dsname=rawPE;
%macro gettype(variable);
%let rx=%sysfunc(dosubl('
data _null_;
length xtype $5 vartext $50;
dsid=open("&dsname");
rows=attrn(dsid,"NOBS");
cols=attrn(dsid,"NVARS");
rc=fetchobs(dsid,1);
xnum=varnum(dsid,"&variable");
xtype=vartype(dsid,xnum);
if (xtype="N") and (%rv(&variable,typ)=%str(Char)) then vartext="put(&variable., 8.)
as &variable. ";
else if (xtype="C") and (%rv(&variable,typ)=%str(Num)) then vartext="input(&variable.,
8.) as &variable. ";
```

```
else vartext="&variable.";
rc=close(dsid);
call symputx("vtext", vartext,"g");
run;'));
%str(&vtext.)
%mend gettype;


*-------------------------------------------------------------------------------
                    MAIN SDTM CREATION SECTION
-------------------------------------------------------------------------------;

proc sql noprint;
create table PE as
       select
/*(1)=========================(STUDYID)=============================================*/
              strip(protocol) as STUDYID
length=%rv(STUDYID,len)
%sysfunc(ifc((%rv(STUDYID,typ)='Char'),"format=$%rv(STUDYID,len).","format=%rv(STUDYID
,len)."))
label="%rv(STUDYID,lab)",

/*(2)=========================(DOMAIN)==============================================*/
              DOMAIN
length=%rv(DOMAIN,len)
%sysfunc(ifc((%rv(DOMAIN,typ)='Char'),"format=$%rv(DOMAIN,len).","format=%rv(DOMAIN,le
n)."))
label="%rv(DOMAIN,lab)",

/*(3)=========================(SUBJID)==============================================*/
              %gettype(SUBJID)
length=%rv(SUBJID,len)
%sysfunc(ifc((%rv(SUBJID,typ)='Char'),"format=$%rv(SUBJID,len).","format=%rv(SUBJID,le
n)."))
label="%rv(SUBJID,lab)",

/*(4)=========================(USUBJID)=============================================*/
              compress(strip(protocol))||'-'||strip(SUBJID) as USUBJID
length=%rv(USUBJID,len)
%sysfunc(ifc((%rv(USUBJID,typ)='Char'),"format=$%rv(USUBJID,len).","format=%rv(USUBJID
,len)."))
label="%rv(USUBJID,lab)",

/*(5)=========================(PESEQ)===============================================*/
              seq as PESEQ
length=%rv(PESEQ,len)
%sysfunc(ifc((%rv(PESEQ,typ)='Char'),"format=$%rv(PESEQ,len).","format=%rv(PESEQ,len).
"))
label="%rv(PESEQ,lab)",

/*(6)=========================(PETESTCD)============================================*/
              substrn(strip(upcase(ExamSite)),1,8) as PETESTCD
length=%rv(PETESTCD,len)
%sysfunc(ifc((%rv(PETESTCD,typ)='Char'),"format=$%rv(PETESTCD,len).","format=%rv(PETES
TCD,len)."))
label="%rv(PETESTCD,lab)",

/*(7)=========================(PETEST)==============================================*/
              strip(ExamSite) as PETEST
```

```
length=%rv(PETEST,len)
%sysfunc(ifc((%rv(PETEST,typ)='Char'),"format=$%rv(PETEST,len).","format=%rv(PETEST,le
n)."))
label="%rv(PETEST,lab)",

/*(8)=========================(PEORRES)=========================================*/
             strip(result) as PEORRES
length=%rv(PEORRES,len)
%sysfunc(ifc((%rv(PEORRES,typ)='Char'),"format=$%rv(PEORRES,len).","format=%rv(PEORRES
,len)."))
label="%rv(PEORRES,lab)",

/*(9)=========================(PESTRESC)========================================*/
             strip(result) as PESTRESC
length=%rv(PESTRESC,len)
%sysfunc(ifc((%rv(PESTRESC,typ)='Char'),"format=$%rv(PESTRESC,len).","format=%rv(PESTR
ESC,len)."))
label="%rv(PESTRESC,lab)",

/*(10)=========================(PESTAT)==========================================*/
             status as PESTAT
length=%rv(PESTAT,len)
%sysfunc(ifc((%rv(PESTAT,typ)='Char'),"format=$%rv(PESTAT,len).","format=%rv(PESTAT,le
n)."))
label="%rv(PESTAT,lab)",

/*(11)=========================(VISITNUM)========================================*/
             whichc(visitname,"Baseline", "Week1","Week2") as VISITNUM
length=%rv(VISITNUM,len)
%sysfunc(ifc(%rv(VISITNUM,typ)=%str(Char),"format=$%rv(VISITNUM,len).","format=%rv(VIS
ITNUM,len)."))
label="%rv(VISITNUM,lab)",

/*(11)=========================(VISIT)===========================================*/
             VisitName as VISIT
length=%rv(VISIT,len)
%sysfunc(ifc((%rv(VISIT,typ)='Char'),"format=$%rv(VISIT,len).","format=%rv(VISIT,len).
"))
label="%rv(VISIT,lab)",

/*(12)=========================(PEDTC)===========================================*/
             case
                    when not missing(ExamDate)
                           then put(ExamDate,IS8601DA.)
                           else ''
             end as        PEDTC
length=%rv(PEDTC,len)
%sysfunc(ifc((%rv(PEDTC,typ)='Char'),"format=$%rv(PEDTC,len).","format=%rv(PEDTC,len).
"))
label="%rv(PEDTC,lab)",

/*(13)=========================(PEDY)============================================*/
             ExamDay as PEDY
length=%rv(PEDY,len)
%sysfunc(ifc((%rv(PEDY,typ)='Char'),"format=$%rv(PEDY,len).","format=%rv(PEDY,len)."))
label="%rv(PEDY,lab)"


from rawPE
```

```
      order by STUDYID, USUBJID, PETESTCD,  VISITNUM, PEDTC;
quit;

%mend CreateSDTM;
%CreateSDTM();.
```

CONCLUSION

The paper demonstrates the possibility of making a SAS® program update itself based on changes from an external source. This is because macro calls can be embedded in programming statements. Complex algorithms can be executed as macro calls using the %SYSFUNC and DOSUBL utilities of SAS®, making such macro calls "digital robots". The technique was applied to the creation of the SDTM Physical Examination dataset, PE, based on specifications laid out on an Excel worksheet. Because these digital robots always read the external specs, any updates made in the specs are automatically reflected in the virtual typing made by the digital robots.
In future, it is quite conceivable that the technique can be extended to the derivations column of the SDTM specs worksheet.

RECOMMENDED READING

Rick Langston, "Submitting SAS® Code On The Side",
https://support.sas.com/resources/papers/proceedings13/032-2013.pdf

CONTACT INFORMATION

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

Joseph Hinson, PhD.
joehinson @ outlook . com

.