

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

Joseph Hinson

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

HOW?

1. Let SAS read the specs and determine the row position of a variable.
2. Use row position to obtain values for LENGTH, LABEL, TYPE.
3. Use digital robot 1 to insert length and label in Proc SQL
4. Use digital robot 2 to determine if there is a type change.
5. If change is from character to numeric, an INPUT function is inserted in Proc SQL. Change from numeric to character triggers a PUT function insertion.
6. Digital robot inserts the correct format statement in Proc SQL.

Typical SDTM Specs (many columns omitted)

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

Create Macro Variable Lists of Variable 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 (%rv) obtains LENGTH and LABEL

```
%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;
```

Digital Robot 1 (%rv) updates LENGTH and LABEL

```
proc sql noprint;  
create table PE as  
    select  
        strip(protocol) as STUDYID  
        length=%rv(STUDYID,len) ,  
        label="%rv(STUDYID,lab) " ,  
  
Etc.,  
Etc.,
```

Digital Robot 2 (%gettype) determines if variable needs type conversion and generates code

```
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.";
```

Proc SQL updated by Digital Robot 2

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

The Use of DOSUBL and %SYSFUNC in %GETTYPE

```
%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;
```

SUMMARY

- A SAS program can update itself if embedded with macro calls (“digital robots”) that can read external dataset specifications and digitally type modifications to the program accordingly.
- To prevent the macro call from instead displaying the macro definition, the macro is wrapped inside %SYSFUNC and DOSUBL function to force immediate macro execution.