Applying Metadata Attributes in Analysis Data Sets Using Macro Variables and Proc SQL in SAS®

Rita Tsang, Averion International Corp., Southborough, Massachusetts

ABSTRACT

In clinical trials, we create analysis data sets from the case report form (CRF) data to enhance the efficiency of the statistical analysis. We may derive new variables in these analysis data sets. Metadata attributes such as data set labels, variable labels and formats are important and useful to users. Specific metadata attributes are often required as an industry standard, such as the CDISC Study Data Tabulation Model (SDTM), for the regulatory submission.

The mapping process however can be cumbersome and manual. This paper will show you how this process can be made easy by using macro variables and PROC SQL in SAS. The SAS macro facility is a tool for text substitution. Using SAS MACRO to apply metadata attributes makes good sense. It can accomplish repetitive tasks quickly and efficiently. The automated process can ensure the accuracy of future updates.

INTRODUCTION

This paper will walk you through a macro program called ADS that applies metadata attributes, specifically data set labels, variable labels and formats, to the final analysis data sets in a data library. The variable labels and formats are pre-defined in an analysis data set specification.

The basic premise of the ADS macro is that we can store information of a data library, such as the data set names, the number of variables within each data set, variable names, variable labels, and formats into macro variables. The ADS macro utilizes these stored macro values to do repetitive tasks for each data set within the data library, e.g. assigning a data set label for each data set. It also performs repetitive tasks on a data set level, such as assigning variable labels and formats for each variable.

The ADS macro utilizes extensively the INTO: clause in PROC SQL to create macro variables because the SQL procedure is an excellent tool for creating macro variables on a metadata level.

The Analysis Data Set Specification

We start with the analysis data set specification that contains information such as the variable name (vname), variable label (vlabel), and format (vformat). This document can be created in a format like Microsoft Excel. We can read the analysis data set specification into SAS using Proc Import or DDE.

Obs	domain	Vname	vlabel	vformat
1	DM	STUDYID	Study Identifier	$9.
2	DM	DOMAIN	Domain Abbreviation	$2.
3	DM	USUBJID	Unique Subject Identifier	$20.
4	DM	SUBJID	Subject Identifier for the Study	$3.
5	DM	RFSTDTC	Subject Reference Start Date/Time	$16.

Obs	domain	Vname	vlabel	vformat
6	DM	RFENDTC	Subject Reference End Date/Time	$16.
7	DM	SITEID	Study Site Identifier	$2.
8	DM	BRTHDTC	Date/Time of Birth	$10.
9	DM	AGE	Age in AGEU at RFSTDTC	8.
10	DM	AGEU	Age Units	$6.
11	DM	SEX	Sex	$1.
12	DM	RACE	Race	$20.
13	DM	ARMCD	Planned Arm Code	$5.
14	DM	ARM	Description of Planned Arm	$50.
15	DM	COUNTRY	Country	$3.
16	DM	DMDTC	Date/Time of Collection	$16.

The Macro Program

```
%macro ads;

proc sql noprint;
  select memname into:names separated by '|'
  from dictionary.tables
  where libname eq 'RAWDATA' and memname not in ('DEFINE');
quit;

%put &names;

%let i=1;
%do %while(%scan(&names,&i,|) ne );
  %put ********************;
  %put ********************;
  %put *  ;
  %put *  %scan(&names,&i,|);
  %put *  ;
  %put ********************;

  proc sql noprint;
    select domain into:dslabel
    from rawdata.define
    where domain eq "%scan(&names,&i,|)";
```

```
    select compress(put(count(*),8.)) into:cn
    from rawdata.define
    where domain eq "%scan(&names,&i,|)";

    select vname, vlabel, vformat
            into:column1 - :column&cn,
                :label1 - :label&cn,
                :vform1 - :vform&cn
    from rawdata.define
    where domain eq "%scan(&names,&i,|)";
  quit;


   data derdata.%scan(&names,&i,|) (label="&dslabel" );
    set rawdata.%scan(&names,&i,|);

    %do j=1 %to &cn;
       label &&&column&j ="&&&label&j";
       format &&&column&j &&&vform&j..;
    %end;
   run;

 %let i=%eval(&i+1);

%end;
%mend ads;
```

The Macro Program Deciphered

```
%macro ads;
```

/*
The macro program ADS first reads the name of all data sets in the data library RAWDATA as a macro variable &NAMES with the exception of the DEFINE data set using the "into:" clause in PROC SQL. In our case, we only have one data set DM. Note that a delimiter "|" is used to separate the different data sets.
***/**

```
  proc sql noprint;
   select memname into:names separated by '|'
   from dictionary.tables
   where libname eq 'RAWDATA' and memname not in ('DEFINE');
  quit;
```

/***
The macro then loops through each data set in &NAMES using "%do %while" and "%end" statements. The macro function %SCAN will select the data set stored in &NAMES based on the loop until it reaches a blank. The macro variable &I is first set to 1 and it is incremented by 1 at the end of each loop.

***/

```
%let i=1;
%do %while(%scan(&names,&i,|) ne );
```

/***
A macro variable &DSLABEL is created to store the value of the data set label for the DM data using PROC SQL. In our example, the value of the domain variable "DM" in the define data set will be used as the data set label.
***/

```
proc sql noprint;
 select domain into:dslabel
 from rawdata.define
 where domain eq "%scan(&names,&i,|)";
```

/***
A macro variable &CN is created for the number of variables in DM using PROC SQL. The macro variable &CN will be resolved to 16 because there are 16 variables in DM.
***/

```
 select compress(put(count(*),8.)) into:cn
 from rawdata.define
 where domain eq "%scan(&names,&i,|)";
```

/***
The following PROC SQL code will create 16 macro variables for the variable names (from &COLUMN1 to &COLUMN16), variable labels (from &LABEL1 to &LABEL16), and variable format (from &VFORM1 to &VFORM16) for the DM data.
***/

```
 select vname, vlabel, vformat
        into:column1 - :column&cn,
             :label1 - :label&cn,
             :vform1 - :vform&cn
 from rawdata.define
 where domain eq "%scan(&names,&i,|)";
quit;
```

/**********

Table 1: The following table shows the macro variables, their resolved values, and the source variables in define.sas7bdat.

**********/

Macro Variable	Purpose	Resolved value	Source Variable in DEFINE.SAS7BDAT
&NAMES	Data sets in the RAWDATA library	DM	NA
&DSLABEL	Data set label	DM	DOMAIN
&CN	Number of variables in DM	16	NA
&COLUMN1 - &COLUMN16	Variables in DM	Values of VNAME from DM, e.g. &COLUMN1 is resolved to **STUDYID**	VNAME
&LABEL1 - &LABEL16	Variable labels in DM	Values of VLABEL from DM, e.g. &LABEL1 is resolved to **Study Identifier**	VLABEL
&VFORM1 - &VFORM16	Formats in DM	Values of VFORMAT from DM, e.g. &VFORM1 is resolved to **$9**	VFORMAT

/**********

The ADS macro assigns the data set label in the data setp.

**********/

```
data derdata.%scan(&names,&i,|) (label="&dslabel" );
 set rawdata.%scan(&names,&i,|);
```

/**********

A nested do-loop is used to assign the variable labels and formats to the variable names stored in &COLUMN1 - &COLUMN16.

Note that SAS uses multiple &'s to resolve nested macro variables. These multiple &'s are resolved in pairs from left to right. In the example of the macro variable &&&column&j, first && is resolved to &, then && is resolved to &, and finally &J is resolved to 1, therefore we have &COLUMN1 in the first loop.

Same logic applies to macro variables &&&label&j, and &&&vform&j.

```
label &column1 ="&label1";
format &column1 &vform1.;
```

Referencing back to Table 1, the macro variable &column1 is resolved to STUDYID, and &label1 is resolved to Study Identifier, and &vform1 is resolved to $9. As a result, we have the final label and format statements in the first loop as:

```
label STUDYID ="Study Identifier";
format STUDYID $9.;
```

**********/

```
    %do j=1 %to &cn;
       label &&&column&j ="&&&label&j";
       format &&&column&j &&&vform&j..;
    %end;
   run;



   %let i=%eval(&i+1);
  %end;

%mend ads;
```

The do-loop will repeat 16 times within the data set DM as illustrated in the following MPRINT in the SAS log.

```
MPRINT(ADS):   data derdata.DM (label="DM    " );
MPRINT(ADS):   set rawdata.DM;
MPRINT(ADS):   label STUDYID = "Study Identifier";
MPRINT(ADS):   format STUDYID $9.;
MPRINT(ADS):   label DOMAIN = "Domain Abbreviation";
MPRINT(ADS):   format DOMAIN $2.;
MPRINT(ADS):   label USUBJID = "Unique Subject Identifier";
MPRINT(ADS):   format USUBJID $20.;
MPRINT(ADS):   label SUBJID = "Subject Identifier for the Study";
MPRINT(ADS):   format SUBJID $3.;
MPRINT(ADS):   label RFSTDTC = "Subject Reference Start Date/Time";
MPRINT(ADS):   format RFSTDTC $16.;
MPRINT(ADS):   label RFENDTC = "Subject Reference End Date/Time";
MPRINT(ADS):   format RFENDTC $16.;
MPRINT(ADS):   label SITEID = "Study Site Identifier";
MPRINT(ADS):   format SITEID $2.;
MPRINT(ADS):   label BRTHDTC = "Date/Time of Birth";
MPRINT(ADS):   format BRTHDTC $10.;
MPRINT(ADS):   label AGE = "Age in AGEU at RFSTDTC";
MPRINT(ADS):   format AGE 8.;
MPRINT(ADS):   label AGEU = "Age Units";
MPRINT(ADS):   format AGEU $6.;
MPRINT(ADS):   label SEX = "Sex";
MPRINT(ADS):   format SEX $1.;
MPRINT(ADS):   label RACE = "Race";
MPRINT(ADS):   format RACE $20.;
MPRINT(ADS):   label ARMCD = "Planned Arm Code";
MPRINT(ADS):   format ARMCD $5.;
MPRINT(ADS):   label ARM = "Description of Planned Arm";
MPRINT(ADS):   format ARM $50.;
MPRINT(ADS):   label COUNTRY = "Country";
MPRINT(ADS):   format COUNTRY $3.;
MPRINT(ADS):   label DMDTC = "Date/Time of Collection";
MPRINT(ADS):   format DMDTC $16.;
MPRINT(ADS):   run;
```

CONCLUSION

This paper has introduced a macro called ADS that applies data set labels, variable labels and formats within a data library. We use a data set DM which has 16 variables as an illustration. It is not hard to imagine how the mapping process can be greatly enhanced if we are working on a large database that has a lot more data sets and variables.

By default, the SAS macro facility performs text substitution in a repetitive manner. Therefore it makes perfect sense to use macro variables in conjunction with PROC SQL to apply metadata attributes in analysis data sets.

REFERENCES

Clinical Data Interchange Standards Consortium (CDISC) (2005), Study Data Tabulation Model Implementation Guide: Human Clinical Trials, Austin, TX: CDISC Inc.

SAS Institute (2008), SAS Online Documentation for SAS 9.1.3 release, Cary, NC: SAS Institute Inc.