Paper 349-2013

Using SAS® to dynamically generate SAS® code in order to display both variable label and name as a column header in PROC REPORT or PROC PRINT

Victor A Lopez, Baxter Healthcare Corporation; Heli Ghandehari, Baxter Healthcare Corporation

ABSTRACT

With implementation of data standards such as CDISC SDTM, datasets contain sufficiently meaningful variable names and labels which allow direct reporting from dataset to output (PDF, RTF, and many more). This eliminates the necessity to program lengthy DEFINE statements in PROC REPORT or having to manually assign custom labels in PROC PRINT.

This paper illustrates an innovative approach in using SAS® to dynamically generate SAS® code which will enable us to solve a seemingly easy problem: displaying both the variable label and name as a column header in PROC REPORT or PROC PRINT.

INTRODUCTION

Considerable time and effort is spent on the creation of datasets. With the implementation of data standards, such as CDISC (Clinical Data Interchange Standards Consortium) SDTM (Study Data Tabulation Model), datasets contain sufficiently meaningful variable names and labels which allow direct reporting from dataset to output (PDF, RTF, and many more). This eliminates the necessity to program lengthy DEFINE statements in PROC REPORT or having to manually assign custom labels in PROC PRINT. This paper illustrates an innovative approach in using SAS® to dynamically generate SAS® code which will enable us to solve a seemingly easy problem: displaying both the variable label and name as a column header in PROC REPORT or PROC PRINT.

DATA STANDARDS

Regardless of industry, data standards have become a natural and necessary evolution of how we tabulate collected data into a meaningful structure, format, and definition. Data standards allow programmers to spend significantly less time “learning” the data and more time analyzing the data.

At a minimum, data standards will mandate an adherence to file naming conventions, variable names, and variable labels. The consistent use of these attributes enable programmers to develop standard reusable programs and scripts.

SETTING UP OUR DATA

To illustrate this technique we will be using limited specification for a CDISC SDTM Adverse Event (AE) Case Report Tabulation (CRT) described in Table 1.

Variable Name	STUDYID	DOMAIN	USUBJID	AETERM	…	AESTDTC	AEENDTC
Variable Label	Study Identifier	Domain Abbreviation	Unique Subject Identifier	Reported Term for the Adverse Event	…	Start Date/Time of Adverse Event	End Date/Time of Adverse Event
Row #1	000000	AE	000000-XXXXXX	AE Term 1	…	YYYY-MM-DD	YYYY-MM-DD

Table 1. Simplified specification for an AE SDTM dataset.

We can use the following code to generate our example dataset (2 observations and 6 columns):

```
libname ae xport "<Computer Location>\ae.xpt";

data ae.ae (label="Adverse Events");
  attrib
```

```
    studyid  label = 'Study Identifier'
    domain   label = 'Domain Abbreviation'
    usubjid  label = 'Unique Subject Identifier'
    aeterm   label = 'Reported Term for the Adverse Event'
    aestdtc  label = 'Start Date/Time of Adverse Event'
    aeendtc  label = 'End Date/Time of Adverse Event'      ;

    studyid  = '000000';
    domain   = 'AE';
    usubjid  = '000000-XXXXXX';
    aeterm   = 'AE Term 1';
    aestdtc  = 'YYYY-MM-DD';
    aeendtc  = 'YYYY-MM-DD';
  output;
    studyid  = '000000';
    domain   = 'AE';
    usubjid  = '000000-XXXXXX';
    aeterm   = 'AE Term 2';
    aestdtc  = 'YYYY-MM-DD';
    aeendtc  = 'YYYY-MM-DD';
  output;
run;
```

Now that we have created our dataset we can use the SAS Viewer and select the “Headers and Names” View to browse the dataset. It allows us to simultaneously view the label and variable name (Display 1).

SAS System Viewer - [ae]

File Edit View Window Help

	Study Identifier (STUDYID)	Domain Abbreviation (DOMAIN)	Unique Subject Identifier (USUBJID)	Reported Term for the Adverse Event (AETERM)	Start Date/Time of Adverse Event (AESTDTC)	End Date/Time of Adverse Event (AEENDTC)
1	000000	AE	000000-XXXXXX	AE Term 1	YYYY-MM-DD	YYYY-MM-DD
2	000000	AE	000000-XXXXXX	AE Term 2	YYYY-MM-DD	YYYY-MM-DD

Display 1. SAS System Viewer 9.130 of Adverse Event Dataset

GENERATING A DATA LISTING

Using PROC PRINT or PROC REPORT will enable us to quickly and easily produce a listing using the data we created above. By default PROC PRINT will display variable names as column headings:

```
proc print data = ae.ae noobs;
  var _all_;
run ;
```

In contrast, PROC REPORT requires a NOLABEL option to display the variable names as column headings:

```
options nolabel;
proc report data = ae.ae nowd;
  column _all_;
run;
```

To print all variables (in the order they appear in the dataset, position in observation) we can use the _all_ option. We can easily subset and declare which variables we want to display and control the order by listing the variable names (i.e. STUDYID USUBJID AESTDTC AETERM). PROC PRINT’s NOOBS option is also used to suppress the observation number in the output while in PROC REPORT observation numbers are not defaulted. The NOWD option is used in PROC REPORT to suppress the REPORT window. Both PROCs produce the same **Output 1**.

STUDYID	DOMAIN	USUBJID	AETERM	AESTDTC	AEENDTC
000000	AE	000000-XXXXXX	AE Term 1	YYYY-MM-DD	YYYY-MM-DD
000000	AE	000000-XXXXXX	AE Term 2	YYYY-MM-DD	YYYY-MM-DD

Output 1. PROC PRINT/REPORT output – variable names

In PROC PRINT we have the LABEL option to enable variable labels as column headings:

```
proc print data = ae.ae noobs label;
  var _all_;
run ;
```

In PROC REPORT the default is to display the labels as column headings since OPTIONS LABEL is default. However, we can control the behavior by manually setting OPTIONS LABEL.

```
options label;
proc report data = ae.ae nowd;
  column _all_;
run;
```

Both PROCs produce the same **Output 2**.

Study Identifier	Domain Abbreviation	Unique Subject Identifier	Reported Term for the Adverse Event	Start Date/Time of Adverse Event	End Date/Time of Adverse Event
000000	AE	000000-XXXXXX	AE Term 1	YYYY-MM-DD	YYYY-MM-DD
000000	AE	000000-XXXXXX	AE Term 2	YYYY-MM-DD	YYYY-MM-DD

Output 2. PROC PRINT/REPORT output – variable labels

There is no SAS® OPTION or PROC REPORT/PRINT option to enable the display of both variable name and label.

GENERATING DYNAMIC CODE

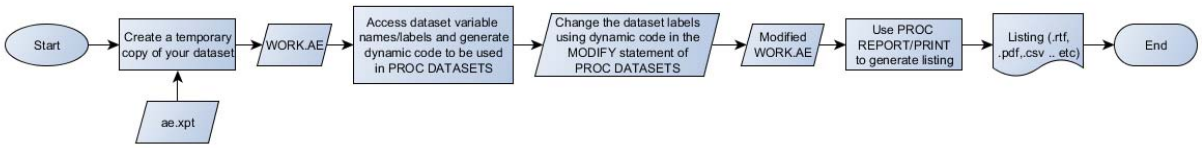
The following code will allow us to display both the variable label and name using PROC PRINT/REPORT. We will use what is "free" in PROC PRINT/REPORT (either variable name or label) rather than defining lengthy programming statements which will be specific to the input dataset (e.g. DEFINE/LABEL statements for *each* variable). For example, while the following would be necessary under the conventional approach…

```
label='Reported Term for the Adverse Event (AETERM)' …
```

```
define aeterm / 'Reported Term for the Adverse Event (AETERM)' …
```

…we can program general code to work with *any* input dataset.

PROGRAM FLOW CHART



We begin by creating a temporary copy of your dataset

```
proc copy in=ae out=work;
run;
```

In order to modify the dataset labels we will use PROC DATASETS which requires the following syntax:

```
proc datasets lib=<library name> ;
  modify <member name>;
  label variable_name_1 = 'variable 1 label'
        variable_name_2 = 'variable 2 label'
                          .
                          .
                          .
        variable_name_n = 'variable n label';
quit;
```

We generate the code above dynamically by directly accessing variable names and labels using PROC SQL and accessing the DICTIONARY.COLUMNS table. We create macro variable &varlist which will contain a space separated list of all variables from the dataset WORK.AE. Meanwhile, macro variable &varlabelcode will generate the SAS® code necessary to modify a dataset using a LABEL statement in PROC DATASETS (variable=<'label'>):

```
proc sql noprint ;
  select name
        ,strip(name)||"='"||strip(label)||"("||strip(upcase(name))||")'"
        into :varlist      separated by ' ',
             :varlabelcode separated by ' '
   from dictionary.columns
   where libname='WORK' and memname = "AE";
quit ;
```

Macro variable	Macro variable resolves to:
&varlist	STUDYID DOMAIN USUBJID AETERM AESTDTC AEENDTC
&varlabelcode	STUDYID='Study Identifier(STUDYID)' DOMAIN='Domain Abbreviation(DOMAIN)' USUBJID='Unique Subject Identifier(USUBJID)' AETERM='Reported Term for the Adverse Event(AETERM)' AESTDTC='Start Date/Time of Adverse Event(AESTDTC)' AEENDTC='End Date/Time of Adverse Event(AEENDTC)'

Table 2. Macro variables used

We now update dataset WORK.AE using PROC DATASETS

```
proc datasets lib=work;
  modify ae;
  label &varlabelcode ;
quit;
```

The dataset modification allows us to pass the “new” dataset through PROC REPORT or PROC PRINT and display our custom column header.

```
proc report data = work.ae nowd;
column &varlist;
run ;

proc print data = work.ae label noobs;
  var &varlist;
run ;
```

Both PROCs generate the same report displayed in Output 3.

Study Identifier (STUDYID)	Domain Abbreviation (DOMAIN)	Unique Subject Identifier (USUBJID)	Reported Term for the Adverse Event(AETERM)	Start Date/Time of Adverse Event(AESTDTC)	End Date/Time of Adverse Event(AEENDTC)
000000	AE	000000-XXXXXX	AE Term 1	YYYY-MM-DD	YYYY-MM-DD
000000	AE	000000-XXXXXX	AE Term 2	YYYY-MM-DD	YYYY-MM-DD

Output 3. PROC PRINT/REPORT output – variable name and label

CONCLUSION

Although the examples presented were prepared using PROC PRINT/REPORT we could apply this methodology and pass in modified datasets to other SAS Procedures which rely on SAS Label for display/reporting. More importantly using SAS® to dynamically generate SAS® code is an efficient programming technique which when combined with data standards allow for the development of standard and reusable SAS® programs.

CONTACT INFORMATION

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

Victor A Lopez
Baxter Healthcare Corporation
1 Baxter Way
Westlake Village, CA 91362
(805) 372-4048
E-mail: victor_lopez1@baxter.com