Paper CC16

Dynamic reporting – a data driven approach

Shan Bai, Eli Lilly and Company, Indianapolis, IN

ABSTRACT

One of the essential processes in clinical trial is to generate Tables, Figures & Listings (TFLs) by using SAS® for presenting and communicating the study results. How to make the program very flexible to meet different reporting requirements from various studies is a challenge. This paper will present a data driven approach to the issue of dynamic reporting by taking advantage of one of SAS features: SASHELP dictionary tables.

INTRODUCTION

In today's world, every pharmaceutical company faces the same pressure to increase productivity and reduce cost in every step of their R&D activities. Creating Tables, Figures and Listings (TFLs) for clinical trial analyses and reporting is an important part of drug development. By generating different TFLs with the same program, dynamic reporting could increase productivity in TFLs creation, and at the same time consistently preserve the quality of TFLs across many studies. This paper will discuss one of the approaches to achieve dynamic reporting - a data driven approach. Examples will be presented to illustrate this approach.

DATA DRIVEN DYNAMIC REPORTING

In general, you need 1) a reporting dataset, and 2) the SAS code with either PROC REPORT or DATA _NULL_ to create a TFL. The TFL then is generated by executing the SAS code against the reporting dataset. The key here is: all the contents of the TFL are contained in the reporting dataset, SAS code merely is a method to present the data in a desired form. The idea of data driven reporting is to use the information in the reporting dataset to dynamically change the SAS code, therefore produce different TFLs. To implement this idea we need to have a grasp on the information about the reporting dataset. But we do not get a chance to look at the dataset, since in almost all the cases it is an interim dataset during the execution of the whole SAS program for analyses and reporting. Fortunately SAS provides us a way to look up the dataset information during SAS sessions. The information we need could be found in dictionary tables. These tables are automatically created by the SAS System and can be found in the SASHELP library. After a SAS session started, the dictionary tables tell us the state of the session and track items and their attributes, such as dataset members, dataset variables. You can access the contents of these tables through PROC SQL. For example, the following program can be used to get all the variable names in dataset REPORT.

```
PROC SQL;
   SELECT NAME
   SEPARATED BY ' '
   FROM DICTIONARY.COLUMNS
   WHERE LIBNAME='WORK' AND
         MEMNAME='REPORT';
QUIT;
```

In the following sections, we will present several examples to show how the SASHELP dictionary tables can be used to communicate the dataset information to PROC REPORT or DATA _NULL_ to dynamically generate reporting tables.

DYNAMIC P-VALUE REPORTING

In clinical trial, requirements on reporting p-values could be different across different studies even they may have similar study design. One may require only reporting overall treatment comparison p-value; others may require additional pair wise comparison p-value or p-value from different test. Even for the pair wise comparison, there could be different pair wise comparisons being specified. The desire is to have the same SAS program able to pick up different p-value requirement automatically, and then create different report accordingly. Since there is no way for us to know in advance what the reporting requirement will be when we are coding the program, we have to find a way to get the information during SAS execution time. Using SASHELP dictionary tables to access the dataset information becomes handy here. The following program could be used to get the list and the count of p-value variables for reporting from dataset REPORT.

```
PROC SQL NOPRINT;
   SELECT NAME INTO :PVLIST
   SEPARATED BY ' '
   FROM DICTIONARY.COLUMNS
   WHERE LIBNAME='WORK' AND
         MEMNAME='REPORT' AND (UPCASE(NAME) CONTAINS 'PVAL');

   SELECT COUNT(NAME) INTO :PVCOUNT
   FROM DICTIONARY.COLUMNS
   WHERE LIBNAME='WORK' AND
         MEMNAME='REPORT' AND (UPCASE(NAME) CONTAINS 'PVAL');
QUIT;
```

Then the macro variables PVLIST & PVCOUNT could be used to pass the information to PROC REPORT or DATA _NULL_ for generating different reports. Following are two sample reports with different p-value reporting requirement.

```
Sample Report 1
 ------------------------------------------------------------------------------------
                       TRT1        TRT2       TRT3      p-value
                      (N=XX)      (N=XX)     (N=XX)     --------
 Event                n   (%)     n   (%)    n   (%)    Overall
 ------------------------------------------------------------------------------------

 Event1               XX (XX.X)  XX (XX.X)  XX (XX.X)    .XXX
 Event2               XX (XX.X)  XX (XX.X)  XX (XX.X)    .XXX
 Event3               XX (XX.X)  XX (XX.X)  XX (XX.X)    .XXX


Sample report 2
 ------------------------------------------------------------------------------------
                       TRT1        TRT2       TRT3              p-value
                      (N=XX)      (N=XX)     (N=XX)     --------------------------
 Event                n   (%)     n   (%)    n   (%)    Overall  1 VS. 2  1 VS. 3
 ------------------------------------------------------------------------------------

 Event1               XX (XX.X)  XX (XX.X)  XX (XX.X)    .XXX     .XXX     .XXX
 Event2               XX (XX.X)  XX (XX.X)  XX (XX.X)    .XXX     .XXX     .XXX
 Event3               XX (XX.X)  XX (XX.X)  XX (XX.X)    .XXX     .XXX     .XXX
```

CONTROL TREATMENT NAME IN THE REPORT DYNAMICALLY

How user friendly is one of the most important attributes of a good SAS program. Giving user the control of some aspect of the report makes the SAS program more flexible, and easy to use. Often in clinical study, not the full medical term of a treatment is used in the report but an abbreviated one instead. Especially when grouping several treatments into one treatment in the report, the new treatment name will not be the combination of all the treatment names, but a short meaningful name to represent the combination in order to make it fit in the report. All the information about the treatment names is already contained in the report dataset. SASHELP dictionary tables could be used to access that information. The following program will get the format information about the treatment, and put it in a macro variable TRTSORTFMT.

```
PROC SQL NOPRINT;
   SELECT FORMAT INTO :TRTSORTFMT
   FROM DICTIONARY.COLUMNS
   WHERE LIBNAME='WORK' AND
         MEMNAME='REPORT' AND (UPCASE(NAME) = "TRTSORT");
QUIT;
```

TRTSORT is the corresponding numerical code of each treatment. If there is a format associated with TRTSORT, then the formatted value of TRTSORT will be applied to TRT as shown in the following.

```
%IF &TRTSORTFMT NE %THEN %DO;
  DATA REPORT;
     SET REPORT;
     TRT=PUT(TRTSORT, &TRTSORTFMT);
  RUN;
%END;
```

In this way you will have control of what treatment name to be shown in the result table. Following are two sample tables with identical layout but different treatment names.

```
Report Example 1
 ------------------------------------------------------------------------------------------
                       TRT1         TRT2        TRT3       p-value
                      (N=XX)       (N=XX)      (N=XX)      --------
 Event                n    (%)     n    (%)    n    (%)    Overall
 ------------------------------------------------------------------------------------------

 Event1               XX (XX.X)  XX (XX.X)  XX (XX.X)     .XXX
 Event2               XX (XX.X)  XX (XX.X)  XX (XX.X)     .XXX
 Event3               XX (XX.X)  XX (XX.X)  XX (XX.X)     .XXX


Report Example 2
 ------------------------------------------------------------------------------------------
                       AAA          BBB         CCC        p-value
                      (N=XX)       (N=XX)      (N=XX)      --------
 Event                n    (%)     n    (%)    n    (%)    Overall
 ------------------------------------------------------------------------------------------

 Event1               XX (XX.X)  XX (XX.X)  XX (XX.X)     .XXX
 Event2               XX (XX.X)  XX (XX.X)  XX (XX.X)     .XXX
 Event3               XX (XX.X)  XX (XX.X)  XX (XX.X)     .XXX
```

CONCLUSION

TFLs are an integral part of clinical study report (CSR) for presenting clinical trial results in an easy to read and understand way to people who are interested in the study. The reported results data is the center piece of the TFLs, and could determine the layout of the TFLs. Using SASHELP dictionary tables to access the reporting dataset information allows the program to automatically decide what kind of report to create during the SAS execution time. This approach gives us the ability to generate different TFLs dynamically by using the same SAS program. Beside the simple examples presented in this paper, it could further serve as a building block for more advanced TFLs automation in clinical trial reporting.

REFERENCES

Michael Davis, “You Could Look It Up: An Introduction to SASHELP Dictionary Views,” Proceedings of SUGI 26, 17-26

ACKNOWLEDGMENTS

The author would like to thank Connie Bryant and other members of CSA team for their comments and support

CONTACT INFORMATION

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

Shan Bai
Eli Lilly and Company
Lilly Corporate Center
Indianapolis, IN 46285
Work Phone: (317) 277-6082
E-mail : shan_bai@lilly.com