Implementing Shared Column Variables

Shan Lee, GlaxoSmithKline, Harlow, UK

ABSTRACT

The purpose of this paper is to illustrate how SAS® PROC REPORT can be used to generate data displays with shared column variables.

I will use an example to explain what I mean by “shared column variables”: let’s suppose we need to create a data display showing the numbers of adverse events for each preferred term, and let’s suppose the preferred terms need to be grouped by system organ class. A classic data display would look like this:

System Organ Class	Preferred Term	Treatment A	Treatment B
Any Event	Any Event	314	326
Ear, Nose and Throat	Any Event	82	90
	Sinusitis	5	4
	…	…	…

If we define system organ class and preferred term to be our “shared column variables”, then we can create a more compact data display like this:

System Organ Class Preferred Term	Treatment A	Treatment B
Any Event	314	326
Ear, Nose and Throat Any Event	82	90
Sinusitis	5	4
…	…	…

This paper will describe the technique for implementing shared column variables which has been used in the HARP Reporting Tools System, developed at GlaxoSmithKline.

INTRODUCTION

I will illustrate the method of implementing shared column variables by explaining the SAS code shown in Appendix 1. The main part of this code consists of the definition of a macro, _display, which generates a data display from an input dataset. The _display macro is a grossly simplified version of a macro that was developed as part of GlaxoSmithKline's HARP Reporting Tools system: it is not intended to be a complete solution to generating data displays, as this would be outside the scope of this paper. Therefore, many features normally associated with good generic code, such as program header, parameter validation, detailed comments etc, have been omitted.

The _display macro calls another macro, _pva, to create a dataset containing meta-data which is used to generate the data display. In Appendix 1, the definition of the _pva macro is blank and the dataset containing meta-data is instead generated by a datastep with hard-coded values: I am assuming you could write your own _pva macro (based on the information presented in this paper) and would be more interested in the code in _display.

At the end of the code in Appendix 1, a test dataset is created and passed to _display, so if you execute all the code in the appendix, then you will see an example of an adverse event data display where system organ class and preferred term have been combined into one column.

PARAMETERS OF THE _DISPLAY MACRO

DSETIN:	name of dataset from which a data display will be generated.
COLUMN:	a PROC REPORT COLUMN statement.
ORDERVARS:	list of variables in the column statement which will be used to sort the rows of the data display; these correspond to variables where the ORDER option has been specified in DEFINE statements of PROC REPORT.
NOPRINT:	list of variables which appear in the COLUMN statement, but should not appear in the data display. Typically, these are variables which have been created solely to enable the data to be sorted correctly.
SHARECOLVARS:	list of variables which should appear in the data display as shared column variables; this may be left blank if you wish to create a normal data display without shared column variables.
OVERALLSUMMARY:	If shared column variables have been specified and if there is an overall summary (eg for adverse event data, total number of subjects with any preferred term for any system organ class), then specifying 'Y' for this parameter will remove the first row of the output and left align the second row, to prevent the overall summary information from being unnecessarily displayed over two rows of the report instead of one.
SKIPVARS:	list of variables – each time the value of one of these variables changes, a blank line will be inserted in the data display.
WIDTH:	widths associated with variables.
SHARECOLVARSINDENT:	indentation factor for shared column variables.
LABEL:	labels associated with variables, specified using SAS notation for a label statement.

CALL TO _PVA AND GENERATION OF META-DATA DATASET

The _display macro calls the _pva macro, passing to it many of its own parameter values, including the name of _display's input dataset. The _pva macro then generates a meta-data dataset, which is similar to the dataset that would be created by calling PROC CONTENTS for the input dataset. However, the meta-data dataset also includes additional variables which provide further information regarding the variables in the input dataset; for example:

ORDER:	set to 1 if the variable appears in the ORDERVARS parameter; set to 0 if the variable doesn't appear in the ORDERVARS parameter.
NOPRINT:	set to 1 if the variable appears in the NOPRINT parameter; set to 0 if the variable doesn't appear in the NOPRINT parameter.
SHARECOLVAR:	set to a positive integer if the variable appears in the SHARECOLVARS parameter (in this case, the positive integer indicates the position of the variable within the SHARECOLVARS parameter); set to missing if the variable doesn't appear in the SHARECOLVARS parameter (the __temp__ variable, described below, is the one exception to this rule).

__TEMP__ VARIABLE

There is one further difference between the meta-data dataset generated by _pva and the dataset that would be created by calling PROC CONTENTS for the input dataset: the meta-data dataset generated by _pva always includes an observation corresponding to a variable called __temp__, even though this variable doesn't exist in the input dataset. The __temp__ variable will be created by _display if shared column variables have been specified. In fact, the __temp__ variable will hold the combined values of all the shared column variables (with appropriate indentation etc), and it will be displayed in the final output, instead of the individual share column variables. Also, please note that _pva assigns attributes (i.e. label) to the __temp__ variable based on the last shared column variable.

TEMPORARY TEXT FILES

After the _pva macro has been called, _display executes a %do … %while … loop: within this loop, three temporary text files are created, one after the other; each consists of SAS code which will be used to build a PROC REPORT step at the end of the loop. If shared column variables have not been specified, then there will only be one iteration of the loop: the PROC REPORT step at the end of the loop will generate the final data display. However, if shared column variables have been specified, then there will be two iterations of the loop: at the end of the first iteration, the PROC REPORT step will generate a dataset, rather than a data display; this dataset will be similar to the _display macro's input dataset, but it will also include a new variable, __temp__, containing the combined values of all the shared column variables (with appropriate indentation etc); during the second iteration of the loop, the three temporary text files will be re-created as if shared column variables had never been specified, using the dataset generated by PROC REPORT from the first iteration as the input dataset for the second iteration.

The code that creates each of these three files is described below:

FIRST TEMPORARY TEXT FILE: CREATION OF __TEMP__ VARIABLE

If shared column variables have not been specified, or if this is the second iteration of the %do … %while … loop when shared column variables have been specified, then this file will simply be blank.

If shared column variables have been specified and this is the first iteration of the loop, then the shared column variables are obtained from the meta-data dataset in sequence, and this information is used to create the following statements that will later be included within the PROC REPORT step:

BREAK statements: these will result in additional observations in the dataset generated by PROC REPORT. For example, if system organ class and preferred term are the shared column variables, then for each system organ class, there will be an additional observation where __temp__ stores the name of that system organ class; each of these observations corresponds to a row in the data display which marks the beginning of a group of rows corresponding to the same system organ class.

COMPUTE blocks: within these compute blocks, values will be assigned to __temp__ with appropriate indentation. For example, in the observations created by the break statements described above, __temp__ will store the values of system organ class, left aligned; in all other observations, __temp__ will store preferred term, indented by an appropriate number of spaces.

DEFINE statement: this will be used to specify that __temp__ is a computed variable in the PROC REPORT step.

SECOND TEMPORARY TEXT FILE: CREATION OF DEFINE STATEMENTS

Using the information in the meta-data dataset, DEFINE statements are generated for all variables that are not computed (i.e. all variables except for __temp__). Just before the creation of this file, there is some code which will specifies variables as NOPRINT variables in the meta-data dataset if shared column variables have been specified (and this is the first iteration of the %do … %while … loop). This ensures that the PROC REPORT step will not generate a data display if another iteration of the loop is required.

THIRD TEMPORARY TEXT FILE: CREATION OF BREAK STATEMENTS FOR SKIPVARS

If shared column variables have been specified and this is the first iteration of the loop, then this file will be blank.

If shared column variables have not been specified or if this is the second iteration when shared column variables have been specified, then this will consist of one break statement corresponding to each variable that has been listed in the skipvars parameter.

PROC REPORT STEP

The main part of the PROC REPORT step consists of three %include statements, which construct the PROC REPORT step by including the three temporary text files described above. If shared column variables have been specified and this is the first iteration of the %do … %while … loop, then an output dataset name is specified in the PROC REPORT statement and no data display will be generated because all the column variables would have been declared as NOPRINT. In all other situations (i.e. shared column variables not specified, or second iteration when shared column

variables have been specified) no output dataset will be specified and variables that need to be displayed will not be set to NORPINT, so a data display will be generated.

OVERALLSUMMARY

If shared column variables have been specified and if the OVERALLSUMMARY parameter has been set to Y, then the first observation of the dataset created by PROC REPORT will be deleted and the second row left aligned. So, for example, in an adverse event data display, this would avoid having a category called ‘Any System Organ Class’ with just one row, ‘Any Preferred Term’; instead, in the overall summary observation, we would set the preferred term value to ‘Any Event’, so there would be just one row for ‘Any Event’ at the start of the data display.

CONTACT

If you have any comments or questions regarding anything in this paper, then please email:

Shan.2.lee@gsk.com

APPENDIX 1

```
%macro _display(dsetin =
                ,column =
                ,orderVars =
                ,noprint =
                ,shareColVars =
                ,overallSummary =
                ,skipVars =
                ,width =
                ,shareColVarsIndent =
                ,label =
                );

%let overallSummary = %upcase(%substr(&overallSummary, 1, 1));

options missing = ' ';

%local metaData;
%let metaData = metaData;

%_pva(dsetin = &dsetin
     ,column = &column
     ,ordervars = &orderVars
     ,noprint = &noprint
     ,shareColVars = &shareColVars
     ,skipVars = &skipVars
     ,width = &width
     ,shareColVarsIndent = &shareColVarsIndent
     ,label = &label
     ,dsetout = &metaData
     );

%if %length(&shareColVars) eq 0 %then %let shareColVarFlag = 0;
%else %let shareColVarFlag = 1;

filename tmpFile1 'tmpFile1.txt';
filename tmpFile2 'tmpFile2.txt';
filename tmpFile3 'tmpFile3.txt';

%do %while (&shareColVarFlag ge 0);

  /* First temporary text file: creation of __temp__ variable. */

  %if &shareColVarFlag %then %do;

    proc sort data = &metaData
              out = shareColVarDset
             ;
      by shareColVar;
      where .z lt shareColVar le shareColVar0;
    run;

    data _null_;

       file tmpFile1;
       set shareColVarDset end = eof;

       if NOT eof then do;

         put 'Break before ' name '/ summarize;';
         put 'Compute before ' name ';';
             if indent GE 0 then put +2 '    __temp__ = repeat(" ",' indent ')||' name ';';
             else                put +2 '    __temp__ ='                            name ';';
         put 'endcomp;';

       end;
       else do;

         put 'Define __temp__ / id computed width=' width 'noprint;';
```

```
      put 'Compute __temp__ / character length =' width ';';
      put +2 'if _break_ eq "" then ' @;
          if indent GE 0 then put '__temp__ = repeat(" ",' indent ')||' name ';';
          else                put '__temp__ = '                          name ';';
      put 'endcomp;';

      /* Add __TEMP__ to the columns parameter */

      length string result $5000;
      string = symget('COLUMN');
      rx = rxparse('$p <'||trim(name)||'> $s TO =1 " __TEMP__"');
      call rxchange(rx,1,trim(string),result);
      call rxfree(rx);
      call symput('COLUMN',trim(result));

     end;

  run;

  %let originalMetaData = &metaData;

  data tempMetaData;
    retain noprint 1;
    set &metaData (drop = noprint);
  run;

  %let metaData = tempMetaData;

%end;
%else %do;

  data _null_;
    file tmpFile1;
    put ' ';
    stop;
  run;

%end;

/* Second temporary text file: creation of define statements. */

data _null_;

  file tmpFile2;
  set &metaData;
  if NOT computed;

  put 'define ' name '/ ' @;
  if order then put 'order ' @;
  else put 'display ' @;
  if noprint then put 'noprint ' @;
  if width ne . then put 'width = ' width @;
  if label ne '' then put '"' label +(-1) '" ' @;
  put ';';

run;

/* Third temporary text file: creation of break statements for skipvars. */

%if not &shareColVarFlag %then %do;

  data _null_;
    file tmpFile3;
    set &metaData;
    where skipvar;
    put 'break after ' name ' / skip;';
  run;

%end;
%else %do;
```

```
    data _null_;
      file tmpFile3;
      put ' ';
      stop;
    run;

  %end;

  proc report data = &dsetin
              %if &shareColVarFlag %then out = tempDsetin;
             nowindows
             list
             missing
             headline
             headskip
             ;
    column &column;
    %include tmpFile1;
    %include tmpFile2;
    %include tmpFile3;
  run;

  %if &shareColVarFlag eq 0 %then %do;

    %let shareColVarFlag = -1;

  %end;
  %else %if &shareColVarFlag eq 1 %then %do;

    data &metaData;;
      set &originalMetaData;
      if .z lt shareColVar le shareColVar0 then noprint = 1;
      if upcase(name) eq '__TEMP__' then do;
       computed = 0;
        display = 1;
      end;
    run;

    %if &overallSummary eq Y %then %do;
      data tempDsetin;
        set tempDsetin;
        if _n_ eq 1 then delete;
        else if _n_ eq 2 then __temp__ = left(__temp__);
      run;
    %end;

    %let dsetin = tempDsetin;

    %let shareColVarFlag = 0;

  %end;

%end; /* %do %while (&shareColVarFlag ge 0) */

%mend _display;

%macro _pva(dsetin =
           ,column =
           ,orderVars =
           ,noprint =
           ,shareColVars =
           ,skipVars =
           ,width =
           ,shareColVarsIndent =
           ,label =
           ,dsetout =
           );
%mend _pva;

data metaData;
  length name label $40;
```

```
  infile cards delimiter = ',';
  input name $ order computed noprint shareColVar shareColVar0 skipvar width indent label
$;
  cards;
aesocSort, 1, 0, 1,  , 2, 0,   ,   ,
aeptSort,  1, 0, 1,  , 2, 0,   ,   ,
aesoc,     1, 0, 0, 1, 2, 1,   , -1, System Organ Class
aept,      1, 0, 0, 2, 2, 0, 50,  1, System Organ Class/   Preferred Term
treatA,    0, 0, 0,  , 2, 0, 11,   , Treatment A
treatB,    0, 0, 0,  , 2, 0, 11,   , Treatment B
__temp__,  0, 1, 0, 3, 2, 0,   , -1, System Organ Class/   Preferred Term
;

data ae;
  length aesoc aept $50;
  infile cards delimiter = ',';
  input aesocSort aeptSort aesoc $ aept $ treatA treatB;
  cards;
1, 1, Dummy              , Any Event,        314, 326
2, 1, Ear nose and throat, Any Event,         82,  90
2, 2, Ear nose and throat, Sinusitis,          5,   4
2, 2, Ear nose and throat, Throat irritation, 11,  11
2, 2, Ear nose and throat, Rhinitis,           4,   5
3, 1, Cardiovascular     , Any Event,         70,  65
3, 2, Cardiovascular     , Hypertension,      35,  29
3, 2, Cardiovascular     , Aneurysms,         27,  26
;

%_display(dsetin = ae
         ,column = aesocSort aesoc aeptSort aept treatA treatB
         ,ordervars = aesocSort aesoc aeptSort aept
         ,noprint = aesocSort aeptSort
         ,shareColVars = aesoc aept
         ,skipVars = aesoc
         ,width = aesoc 50 treatA treatB 11
         ,shareColVarsIndent = 2
         ,label = aesoc = "System Organ Class" aept = "System Organ Class/   Preferred
Term" treatA = "Treatment A" treatB = "Treatment B"
         ,overallSummary = Y
         )
```