Paper CT07

Accessible Reporting by PROC STREAM

Charumathy Sreeraman, Ephicacy Lifescience Analytics, Bengaluru, India
Samundeeswari Raja, Ephicacy Lifescience Analytics, Bengaluru, India

ABSTRACT

Reporting conformance to ethical and scientific integrity of clinical trials are monitored by various committees. Such monitoring is mainly performed by creating and reviewing the safety reports of a study. Availability of interactive and dynamic representation of the safety data in the form of tables and listings facilitates understanding of the data reported and eases the reviewing/validation process. These reports are commonly shared via email in printable PDF format. The best part of reports in this format is, they are easily printable on many different printers without adjusting margins or worrying about page breaks. PROC STREAM procedure is a powerful tool in a statistical programmer's arsenal because it enables easy generation of reports which clearly stream the, otherwise lumpy safety data. This paper shall discuss about the usage of PROC STREAM procedure, through simple coding for generation of various types of reports in PDF format.

INTRODUCTION

Maintaining high competency of data without compromising quality is an inevitability in the Pharmaceutical industry. We collect massive amounts of data during a clinical study, which are usually summarized as tables, listings and figures for statistical analysis. The safety reports are an imperative part of the study and depict a clear picture of safety of a drug to the patients. These reports are frequently monitored by various committees and regulatory authorities. Creating these reports with drill down hierarchy will make it much more palatable, as huge amount of data are converted as reports. PROC STREAM provides us the easy way achieve these with simple coding technique.

BRIEF INTRODUCTION TO PROC STREAM

PROC STREAM statement specifies an external file with the "OUTFILE=" keyword, as well as options. The OUTFILE=fileref specifies the file where all tokens are written. The arbitrary text is sandwiched inside a "BEGIN" and a four semi-colon ending ";;;;". The text specifies the SAS statements or macros to use with PROC STREAM. There is no "RUN" or "QUIT".

PROCEDURE SYNTAX

```
PROC STREAM OUTFILE= fileref <option(s)>;
BEGIN
<text-1>
<text-n>
;;;;
```

PROC STREAM provides a formal and robust tool to allow a SAS job stream to include any text-based content, including macro variables and macros, by redirecting the results of SAS tokenization of the input file to another location. Any text between a BEGIN keyword (the first token following the PROC STREAM statement) and four adjacent semi-colons (;;;;) is parsed by the SAS word scanner and passed to PROC STREAM for processing.

BUILDING THE REQUIRED MACROS

The macros to be called to make the reports with drill-down hierarchy are

RAW
The macro RAW can be any of the summary tables or listings to which reporting will be done by PROC STREAM.

TREENODES
The macro TREENODES fetches the final dataset to be sent to PROC REPORT, utilizes it as the input dataset to create the nodes for drill-down HTML output by PROC STREAM and also have the JavaScript to convert HTML to PDF output with the drill-down hierarchy.

SAS CODE TO CREATE THE PDF OUTPUT BY PROC STREAM

```
Filename _webout "/path/report/ae.pdf";

proc stream outfile=_webout;
BEGIN
%global pageTitle data hierarchy vars statistic;
<html>
<head>
<title>pageTitle</title>
</head>
<body>
%let rc = %sysfunc(dosubl(%nrbquote(%raw)));
<table><tr><td>
<fieldset>
<legend>&pageTitle</legend>
%treeNodes(data = &data
          ,columns = &vars
          ,columnWidths = 100
          ,textWidth = 125
          ,divPrefix = R
          ,headers = Y
          ,expandedTo = 1
          ,tableAttributes = cellspacing="2" cellpadding="0"
          )
</fieldset>
</td></tr></table>
</body>
</html>

;;;;
```

CONCLUSION

The main intention of the paper is to create the PDF ouput with drill-down hierarchy for wasy review of the lumpy reports. The SAS coding can be series of macros which are pre-prepared and used inside the PROC STREAM. The PROC STREAM can also be used to execute javascript with minimal support coding.

REFERENCES (HEADER 1)

Don Henderson, PROC STREAM and SAS® Server Pages: Generating Custom User Interfaces. Paper 1737-2014.
Don Henderson, PROC STREAM and SAS® Server Pages: Generating Custom HTML Reports. Paper 1738-2014.
Joseph Hinson, Proc STREAM: The Perfect Tool For Creating Patient Narratives. PharmaSUG 2015 - Paper AD03
Joseph Hinson, The New STREAM Procedure as a Virtual Medical Writer. PharmaSUG 2015 - Paper AD03
Charumathy Sreeraman and Samundeeswari Raja, Streaming Tables and Listings by PROC STREAM. PhUSE EU Connect 2018 – Paper DV06

ACKNOWLEDGMENTS

We would like to express our sincere gratitude to the management of our organization – “Ephicacy” for the encouragement and support in helping us with all the necessary facilities to write this paper.

We would also like to thank members of the Ephicacy family for their encouragement, insightful comments and hard questions.

Our sincere thanks to our Director Mr. Tyagrajan Swaminathan and our Manager Mr. Srihari Vadlakonda for the unswerving support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Author Name – Charumathy Sreeraman
Company - Ephicacy Lifescience Analytics
Address - : 2nd Main Rd, Sarvobhogam Nagar, Arekere.
City / Postcode - Bengaluru, Karnataka 560076
Work Phone: 080 4146 3195/+91 9620209942
Email: charumathy.sreeraman@ephicacy.com
Web: www.ephicacy.com

Your comments and questions are valued and encouraged. Contact the author at:
Author Name – Samundeeswari Raja
Company - Ephicacy Lifescience Analytics
Address - : 2nd Main Rd, Sarvobhogam Nagar, Arekere.
City / Postcode - Bengaluru, Karnataka 560076
Work Phone: 080 4146 3195/+91 9620209942
Email: Charumathy.sreeraman@ephicacy.com
Web: www.ephicacy.com

APPENDIX

```
%macro treeNodes
      (data = ,                     /* name of input data set that contains the tree
*/
       nestingLevel = nestingLevel, /* variable name that defines the level of the
branch */
       levelText = levelText,       /* variable name that contains the text to label
the node */
       columns =,                  /* optional list of columns to display */
       columnWidths =,              /* width (in pixels) for the columns */
       textWidth = 200,             /* width (in pixels) for the levelText column */
       divPrefix = R,               /* a prefix to define unique ID values */
       headers = Y,                 /* display the column names/labels? blank = No */
       expandedTo = 1,              /* level the tree should be expanded to on initial
display */
       tableAttributes = cellspacing="0" cellpadding="0" border = 0
                                    /* attributes for the HTML table tag */
      );

  <script language="javascript">
function toggleDiv(divid) {
  if(document.getElementById(divid).style.display == 'none'){
  document.getElementById(divid).style.display = 'block';
  }
  else {
  document.getElementById(divid).style.display = 'none';
  }
  if(document.getElementById(divid+'_M').style.display == 'none') {
  document.getElementById(divid+'_M').style.display = 'block';
  document.getElementById(divid+'_P').style.display = 'none';
  }
  else {
  document.getElementById(divid+'_M').style.display = 'none';
  document.getElementById(divid+'_P').style.display = 'block';
  }
}
</script>

%local dsid dsidNext nestingLevelid levelTextid l v nextl var nColumns i rows rc
rowNum lastWidth;

&streamDelim newline;%let plusImage = %str(

MDAwAAAAAAAAAAAP///yH5BAEAAAsALAAAAAALAAsAAAQecMlJJbgY27oXmML0eaC4CShKrkvYja3JUmNmc1w
EADs=
);
&streamDelim newline;%let minusImage = %str(

MDAwAAAAAAAAAAAP///yH5BAEAAAsALAAAAAALAAsAAAQbcMlJJbgY27oX4F5XfaFgmmE6ihRJrl6WgVUEADs
=
);
%let imgWidth = 12;

%let dsid = %sysfunc(open(&data));
%let dsidNext = %sysfunc(open(&data));
%let nestingLevelid = %sysfunc(varnum(&dsid,&nestingLevel));
%let levelTextid = %sysfunc(varnum(&dsid,&levelText));

/* determine maximum depth and number of rows */
%let deep = 0;
%let rows = 0;
%do %while(%sysfunc(fetch(&dsid)) = 0);
    %let l = %sysfunc(getvarn(&dsid,&nestingLevelid));
    %if &deep < &l %then %let deep = &l;
```

```
    %let rows = %eval(&rows+1);
%end;
/* close and re-open to read from beginning */
%let dsid = %sysfunc(close(&dsid));
%let dsid = %sysfunc(open(&data));

/* determine columns */
%let i = 1;
%let var = %scan(&columns,&i,%str( ));
%let lastWidth = &textWidth;
%do %while(%length(&var) gt 0);
     %local vnum&i vtype&i vfmt&i width&i vlabel&i;
     %let vnum&i = %sysfunc(varnum(&dsid,&var));
     %let vtype&i = %sysfunc(vartype(&dsid,&&vnum&i));
     %let vfmt&i = %sysfunc(varformat(&dsid,&&vnum&i));
     %let vlabel&i = %qsysfunc(varlabel(&dsid,&&vnum&i));
     %let vlabel&i = %qsysfunc(coalesceC(&&vlabel&i,&var));
     %let width&i = %scan(&columnWidths,&i,%str( ));
     %let width&i = %sysfunc(coalesceC(&&width&i,&lastWidth));
     %let lastWidth = &&width&i;
    %let i = %eval(&i+1);
     %let var = %scan(&columns,&i,%str( ));
%end;
%let nColumns = %eval(&i-1);

%if %length(&headers) gt 0 and %length(&columns) gt 0 %then
%do;  /* show headers */
    &streamDelim newline;<table &tableAttributes>
    <tr>
    %do i = 0 %to &deep;
       <td width="&imgWidth"> </td>
    %end;
    <td width="&textWidth"> </td>
    %do i = 1 %to &nColumns;
       <th width="&&width&i" align="right">&&vlabel&i</td>
    %end;
    </tr>
    </table>
%end; /* show headers */
%do %while(%sysfunc(fetch(&dsid)) = 0);
     %let rowNum = %eval(&rowNum+1);
     %let l = %sysfunc(getvarn(&dsid,&nestingLevelid));
%if &l lt &expandedTo %then
%do;  /* initially expanded */
    %let display = block;
    %let plus = none;
    %let minus = block;
%end; /* initially expanded */
%else
%do;  /* initially collapsed */
    %let display = none;
    %let plus = block;
    %let minus = none;
%end; /* initially collapsed */
     %let v = %sysfunc(getvarc(&dsid,&levelTextid));
     %let rc = %sysfunc(fetchobs(&dsidNext,&rowNum+1));
     %if &rc = 0 %then %let nextl = %sysfunc(getvarn(&dsidNext,&nestingLevelid));
     %else %let nextl = 0;
&streamDelim newline;<table &tableAttributes>
<tr>
     %do i = 0 %to &l-1;
<td width="&imgWidth"> </td>
     %end;
<td width="&imgWidth">
```

```
     %if &l lt &nextl %then
     %do;
<a href="javascript:;" onmousedown="toggleDiv('&divPrefix&rowNum');">
&streamDelim newline;<img id="&divPrefix&rowNum._M" src="&plusImage" border="0"
style="display:&plus;" alt="Expand">
&streamDelim newline;<img id="&divPrefix&rowNum._P" src="&minusImage" border="0"
style="display:&minus;" alt="Collapse">
&streamDelim newline;
     %end;
     %else %str( );
</td>
<td width="&textWidth">&v</td>
     %do i = &l+1 %to &deep;
<td width="&imgWidth"> </td>
     %end;
     %do i = 1 %to &nColumns;
<td width="&&width&i"
align="right">%sysfunc(getvar&&vtype&i(&dsid,&&vnum&i),&&vfmt&i)</td>
     %end;
</tr>
</table>
%if &l lt &expandedTo %then %let display = block;
%else %let display = none;
<div id="&divPrefix&rowNum" style="display:&display">
     %do i = &nextl %to &l;
</div>
     %end;
%end;
%let dsid = %sysfunc(close(&dsid));
%let dsidNext = %sysfunc(close(&dsidNext));
%mend treeNodes;
```