

Conditional LINEs in PROC REPORT

Derek Morgan, Washington University Medical School, St. Louis, MO

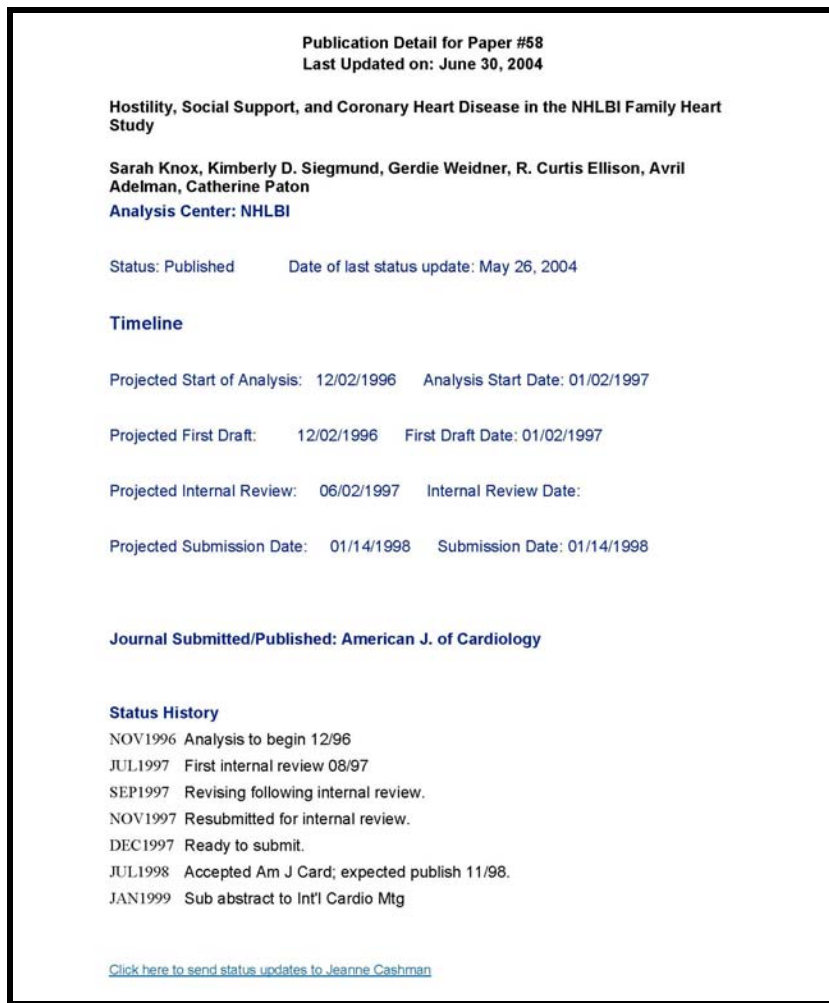
ABSTRACT

Due to the way that PROC REPORT works, you cannot conditionally execute LINE statement within a compute block. If it is important enough in a production environment, you can get around this, as long as the report does not use a stored report definition. While you may not be able to execute LINE statements conditionally, you can conditionally construct the PROC REPORT, and choose which LINE statements are in the PROC REPORT itself. This will show you how in both an application context using SAS component language, and insight on how to do this by using a DATA step.

BACKGROUND

We developed a SAS application that was designed to track the progress of academic publications for our studies. The application runs on our LINUX server and there is a sizable web reporting component. The web-reporting component allows our non-Division of Biostatistics collaborators to access the status of any given publication in a study to which they have access via the World Wide Web. We used PROC REPORT to produce the HTML that is on the WWW. The main dataset is arranged one record per publication. However, there is also a details dataset, arranged one record per event per publication. The question was how best to automate the production of the HTML for this file to produce the following output (HTML reformatted for inclusion in this paper):

FIGURE 1: DETAILS PAGE FOR PUBLISHED PAPERS



The records from the details dataset are at the bottom of the page, while the remainder of the information has been merged from the main dataset. How did we get such a display? Everything above the detail records is a custom header, built with a COMPUTE BEFORE statement in PROC REPORT. That's not a hard thing to do, even from within an AF application. So what's the problem? The above report is fine if a paper has been published and it has detail records. What happens when either of the two doesn't hold?

FIGURE 2: DETAILS PAGE FOR PAPERS IN PROGRESS

Publication Detail for Paper #182
 Last Updated on: June 30, 2004

Conditional LINES in PROC REPORT

Derek Morgan
Analysis Center: Washington University

Status: Draft Date of last status update: June 30, 2004

Timeline

Projected Start of Analysis: 04/30/2004 Analysis Start Date: 04/30/2004

Projected First Draft: 04/30/2004 First Draft Date: 04/30/2004

Projected Internal Review: 07/10/2004 Internal Review Date:

Projected Submission Date: 07/11/2004 Submission Date:

Target Journals

1. Proceedings of the MidWest SAS Users Group
2. Proceedings of the SAS Users Group International Conference

[Click here to send status updates to Jeanne Cashman](#)

A couple of things have happened here: first, the “Journal Published/Submitted” line from figure 1 has been replaced with a “Target Journals” header, and a list of the target journals. Second, there is no “Status History” header for the records from the details dataset (that’s because there aren’t any records in that dataset for this particular publication.) The LINE statements that get executed are different between the two examples.

How can you do that? You can’t execute LINE statements conditionally. It even says so in the documentation:

“Restriction: You cannot use the LINE statement in conditional statements (IF-THEN, IF-THEN/ELSE, and SELECT) because it does not take effect until PROC REPORT has executed all other statements in the compute block.”¹

The documentation is absolutely correct. However, there’s nothing to prevent you from using two different PROC REPORT statements in the two cases, and it’s not that hard, especially from within an application.

HOW DID YOU GET THOSE HEADERS?

Here’s a handy SCL trick: SCL variables can be passed to the PREVIEW buffer as macro variables of the same name. That is, if you have an SCL variable named “ID” in your SCL code, you can use it inside a SUBMIT block as “&ID”. There is no CALL SYMPUT statement necessary. As an example, here is the SCL code snippet for two of the lines for the page header, and the file name for the HTML that is generated.

```
t1 = "Publication Detail for Paper #" || TRIM(LEFT(PUTN(pubin, '4.')));
t2 = "Last Updated on: " || TRIM(LEFT(PUTN(TODAY(), 'WORDDATE18.')));
st = TRIM(SYMPUT('useroot')) || "/secure_html/reports/detail/detail" ||
      TRIM(LEFT(PUTN(pubid, '4.'))) || ".html";
```

This creates three macro variables: &t1, &t2, and &st. These variables are used in TITLE statements, and the ODS HTML statement submitted to the PREVIEW buffer as shown below.

```

SUBMIT CONTINUE;
  TITLE1 BOLD "&t1";
  TITLE2 BOLD "&t2";
  TITLE3 " ";
  ODS LISTING CLOSE;
  ODS HTML FILE="&st" STYLE=stathx;

```

This allows us to assemble the text we want displayed inside of the SCL program, not inside the SAS code submitted for processing. This is important; we have access to the variables we need from the data in the SCL PDV, and an easy way to communicate with the SAS program. Here's the SCL code for the PROC REPORT statement itself.

```

... (The SUBMIT CONTINUE from the block of code above is still in effect) ...
1  PROC REPORT DATA=statushxrpt NOWD NOHEADER;
2  COLUMNS pubid key revdate status_txt;
3  DEFINE pubid / GROUP NOPRINT;
4  DEFINE key / ORDER NOPRINT;
5  DEFINE revdate / DISPLAY F=MONYY7. " " STYLE=[JUST=left CELLWIDTH=10%
   FONT_FACE=bold];
6  DEFINE status_txt / DISPLAY " " FLOW STYLE=[JUST=left CELLWIDTH=90%
   VJUST=top];
7  WHERE pubid EQ &pubin; /* &pubin is a parameter passed to this program module
   to indicate which publication number we are working with */
8  /* Finished the PROC REPORT statement, now let's do the header */

9  COMPUTE BEFORE pubid / STYLE=[FONT_SIZE=3 JUST=left ASIS=yes];
10 /* All macro variables in lines 13-19 have been created inside this program
   module as SCL variables */

11 LINE "<b>&anctr</b>";
12 LINE "&stl";
13 LINE '<font size=+1><b>Timeline</b></font>'; /* Note HTML Code */
14 LINE "&tmln1";
15 LINE "&tmln2";
16 LINE "&tmln3";
17 LINE "&tmln4";
18 LINE ' ';
19 ENDSUBMIT;
20 /* End of standardized part of the header. All HTML detail files are identical
   to this point. After this line, control belongs to the SCL program, NOT the
   PREVIEW buffer */
21 IF journal NE . THEN DO; /* journal is SCL variable */
22   SUBMIT CONTINUE; /* If published in journal, add this to PREVIEW buffer */
23   LINE "<b>Journal Submitted/Published: &jrnl </b>";
24   ENDSUBMIT;
25 END;
26 ELSE DO; /* Not published, add this to PREVIEW buffer instead */
27   IF nt GE 1 THEN DO; /* 1 target; if no targets specified, skip header */
28     SUBMIT CONTINUE;
29     LINE '<b>Target Journals</b>';
30     LINE ' ';
31     LINE "&targ1";
32     ENDSUBMIT;
33   END;

```

```

34 IF nt GE 2 THEN DO; /* 2 targets; bypasses blank line if only one target */
35     SUBMIT CONTINUE;
36     LINE "&targ2";
37     ENDSUBMIT;
38 END;
39 IF nt EQ 3 THEN DO; /* 3 targets; bypasses blank line if two or fewer targets
*/
40     SUBMIT CONTINUE;
41     LINE "&targ3";
42     ENDSUBMIT;
43 END;
44 END;
45 /* End of publication/no publication switch; only appropriate LINE statements
based on SCL variable values have been sent to the PREVIEW buffer for
processing. */
46 IF key NE 0 THEN DO; /* Any detail records for this publication? If yes, print
"Status History" header */
47     SUBMIT CONTINUE;
48     LINE ' ';
49     LINE '<font size+1><b>Status History</b></font>';
50     ENDSUBMIT;
51 END;
52 /* Conditional processing over, do the following for every record */
53 SUBMIT CONTINUE;
54 ENDCOMP; /* Close COMPUTE block in PROC REPORT */
55 RUN;
56 ODS HTML CLOSE;
57 ODS LISTING;
58 ENDSUBMIT;
59 RETURN; /* Control sent to SAS System - executes SAS code in the PREVIEW buffer
*/

```

All we've done here is to pick and choose which code gets sent to the PREVIEW buffer using conditionals inside the SCL program. These statements were pulled verbatim from the application SCL code, with only the comments in blue added. For shorter headers, you could pick from appropriately formatted TITLE statements instead of LINE statements in COMPUTE blocks. If the conditional piece is in a COMPUTE AFTER block, you could use FOOTNOTE statements. However, remember that you can have a maximum of ten FOOTNOTE or TITLE statements. This would only work if you have fewer than ten lines of either custom header or footer information, and any titles and footnotes that aren't customized must be counted towards those ten lines.

That's all very nice if you're working with an application. Besides, we cheated by using SCL and its automatic-macro variables. What happens if you have a SAS/IntrNet application, and you'd like to customize headers based on user selections from an HTML page? Do you need to use SAS/AF somehow? Can you even do this?

Of course you can. I wouldn't recommend it for a one-time program, but this is a production application we are talking about. You can use SAS to write its own code using DATA _NULL_, and PUT statements. The trickiest part is going to be recognizing where to put the report code so that the CGI broker can find it once it's been written. You can also write it into a series of macro variables, and execute that from the program that wrote the code. If you write it to an external file, use a %INCLUDE statement in the program to execute the code that it wrote. This way, you can use the macro variables returned from your HTML page to pick and choose which LINE (or TITLE or FOOTNOTE) statements get used. If nothing else, you can execute an SCL entry from SAS/IntrNet, and use the SCL method described above.

SUMMARY

You can create custom headers in PROC REPORT inside of COMPUTE blocks. Unfortunately, the LINE statement cannot be executed conditionally (with an IF-THEN construction), so your header will not be changed based upon the data used. However, you can get around this by constructing your COMPUTE block conditionally. If you are using SCL inside of an application, you can put SUBMIT blocks containing your LINE statements inside of IF-THEN clauses, causing only the LINE statements you want to be submitted for processing from the PREVIEW buffer.

If you are trying to do this in another environment, you can use DATA _NULL_ and PUT statements to write the SAS code to a file, and conditionally build your COMPUTE block from there. You also have the option of executing an SCL entry from SAS/IntrNet. Overall, it requires a small amount of extra work, and it is a question of whether the functionality is worth the effort. In a production environment, this may help you build prettier reports, but unless it's necessary, I would not recommend doing this.

REFERENCES:

Reprinted with permission of SAS Institute Inc. from *SAS Procedures Guide, Version 8, Volumes 1 and 2*. Copyright 1999 by SAS institute, Inc.

ACKNOWLEDGEMENTS

This work has been partially funded by: NHLBI grants HL56567, HL54473, HL67899, HL72524, and HL72507.

CONTACT INFORMATION:

Further inquiries are welcome to:

Derek Morgan
Division of Biostatistics
Washington University Medical School
Box 8067, 660 S. Euclid Ave.
St. Louis, MO 63110
Phone: (314) 362-3685 FAX: (314) 362-2693
E-mail: derek@wustl.edu

This and other SAS System examples and papers can be found on the World Wide Web at:

<http://www.biostat.wustl.edu/~derek/sasindex.html>

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.