Paper PH04-2011

Case Study: Generating a DSMC report from an ORACLE database with SAS® PROC REPORT

Shannon M. Morrison, M.S.
Matthew T. Karafa, PhD.
Quantitative Health Sciences – Cleveland Clinic Foundation, Cleveland, OH

Abstract

The Department of Quantitative Health Sciences at the Cleveland Clinic often works on FDA regulated studies, which store data in ORACLE® databases. ORACLE data is most often not immediately appropriate for reporting or analysis. This paper details methods used to transform data from an ORACLE database (transferred to the data management team as SAS® data sets) into layouts appropriate for both analysis and reporting.

Two macros were created to run data from six different sets of lab results. The first macro transforms data from ORACLE into a layout that can be run in the REPORT procedure and used in future analyses. The macro only requires the user to enter the name of the SAS file containing data from the desired ORACLE table. Problems encountered and discussed include generating valid SAS variable names from potentially long test names or those with special characters.

The second macro uses the resulting data set generated from the data management macro to create the final report for the Data Safety Monitoring Committee (DSMC). The user must enter the name of the SAS file containing the desired ORACLE table, the lab type (to name the PDF output file), and the name of the lab to be printed in the report title. This macro highlights the ability to automatically create PDF files with a customized bookmark pane.

Introduction

The Department of Quantitative Health Sciences (QHS) offers research support in the form of statistical analysis, data management services, and statistical programming to the Cleveland Clinic. Data from projects that are National Institute of Health (NIH) funded, Food & Drug Administration (FDA) regulated, or take place at multiple centers is typically stored in an ORACLE database. The layout of this data can present a unique programming challenge; straight out of ORACLE, data is not appropriately set up for analysis or reporting. QHS began work on an FDA regulated trial in August 2010. During the data gathering phase of the trial, the primary objective for the group was to prepare reports for the 70+ SAS data sets produced by ORACLE to present to the DSMC overseeing the project and prepare the data for analyses to be performed later. Since there were relatively few patients, the DSMC requested tables modeled after a patient chart; one patient per table with as much information about that patient as possible per page. The team was able to create macros to generate both reports and analysis data sets for the different labs.

The first macro deals with data management, transforming what comes from ORACLE into something suitable to be run in PROC REPORT as well as future analyses. The data management macro only requires the user to enter the name of the ORACLE database. Problems encountered and discussed include generating valid SAS variable names from potentially long test names with special characters.

The second macro takes the data set generated from the data management macro and creates the final report. To run the macro, the user must enter the name of the ORACLE database, lab type (used to name the PDF output file), and the name of the lab to be printed in the report title. This macro highlights the ability to automatically create PDF files with a customized bookmark pane and some ways around the common pitfalls that can be encountered when working with clinical data from ORACLE.

In the reports, each test has its own column with visit type, visit date, and test value listed. Based on the specifications the team was given, the output is grouped by patient; each patient has their own page of results, and each page can be accessed using the bookmark pane, which has a bookmark for each individual patient. Generating these listings efficiently was very important, as for this particular study DSMC review was required for each cohort of 4 patients. It is the team's hope that the DSMC will find the reports easy to read and navigate during the committee meetings. Please note that data in this paper is TEST DATA ONLY, and is not real data.

The ORACLE Data

An ORACLE database containing test data was transferred into SAS files, which were sent to the data management team. ORACLE automatically generates many variables (Clinical Study, Site, Investigator, etc.) as well as multiple versions of the numeric values of interest (the value itself, a "DLV" value, and a "FUL" value). The team quickly narrowed down the field of 50 variables and focused only on those variables that were needed for the reports and analysis (5 variables total).

Test Data

A portion of the data straight from the Blood Chemistry Labs ORACLE table is shown below:

PT	DCMDATE	CPEVENT	LTST	LVALUE
T29	20100102	SCREEN	Anion Gap	15.3
T31	20100102	SCREEN	Glucose, random	80.12
D2	20100110	Day 1	Chloride	102.7
X7	20100114	Day 4	Total Protein	8.1
X7	20100117	Day 7	Calcium	2.35

In the above data set, PT is the unique patient ID, DCMDATE is the visit date, CPEVENT is the visit type (Screen, Baseline, Day 1, Day 4, etc.), LTST is the test performed, and LVALUE is the result of that test.

After identifying variables of interest, the team discovered another difficulty; “Test data” meant only that the fields met the individual data field criteria. Data was numeric in numeric fields and character in character fields, but was not necessarily similar to what would be presented from a real patient. Intervariable relationships and multiple visits were not provided as part of the test data that the team was given, thus the team had to make educated guesses as to what the “real” data would look like.

Goals

The Primary Investigator and DSMC requested that the data management team attempt to model the lab reports after patient charts; all results would be laid out on one page (with each patient having their own page). The DSMC would need to be able to quickly look through the results for each patient and easily spot any Adverse Events (AEs – abnormal test results) which may be grounds for stopping the trial. Each test name would become a row header, and under each test, the visit, date and test result would be listed. Tests would be performed at multiple visits, so in addition to easily spotting AEs, DSMC members also needed to see how results changed over time.

Data Management Macro

In looking through the lab data sets, it was clear that the layout of the electronic Case Report Forms (eCRFs) and set-up of the data sets (including variable names) for the six lab data sets was the same. Thus it made sense that the resulting lab reports would look the same, and that macros could be created to quickly handle all six data sets. Automating the process was extremely important with 70+ data sets to work with and reports that needed to be generated quarterly. Any part of the data management/report generating process that could be automated would be a huge bonus for the team.

Before reporting began, the data needed to be transformed with the assumption that after this transformation, the data would resemble the final data set for both the DSMC report and final analysis. In this macro, the user only needs to enter in the lab data set to be run:

```
%macro DataManagement (labds = );
```

Handling the test names

Information from the data sets are stored into a series of macro variables:

- Test names from the ORACLE variable LTST are stored in the macro list `LabName`…if test names have spaces or special characters (such as “Glucose, random” in the data snippet above), the names are compressed and special characters are stripped so that they are valid data set/SAS variable names.
- The actual Test Names (including any spaces and special characters) are stored in the macro list `LabLabel` for use later on.
- The number of unique tests in the data is counted and stored in the variable `TotalLabsinData`.

  Since SAS will only use what it needs, and it is known that there are less than 1000 tests, the macro array is created using 1 through 999:

```
proc sql noprint;
   select distinct LTST, compress(LTST, " .%+/(),-")
      into :LabLabel1 - :LabLabel999, :LabName1 - :LabName999
      from &labds.;
   quit;
   %let TotalLabsinData = &sqlobs;
run;
```

We can now refer to `&&LabLabel&i.` or `&&LabName&i.` to get the ith macro variable; the above sql code creates `&LabLabel1.` through `&LabLabel999.` A double & is used to refer to these variables because the macro variable first resolves the &i portion and then leaves us with the &LabName1. portion (for more information on the macro language, refer to "SAS Macro Language Referece"). A loop in the macro then checks for lab test names longer than 30 characters, which are truncated to 30 characters long in accordance with SAS naming conventions:

```
%do j = 1 %to &TotalLabsinData;
   %if %LENGTH(&&LabName&j) > 30 %then %do;
      %let LabName&j = %SUBSTR(&&LabName&j, 1, 30);
   %end;
```

Creating data sets for each test

Each data set is named for the appropriate lab test (for example, the data set WORK.Albumin contains Albumin results only) and contains patient ID, date, visit type and the value of the test. The value variable, originally named LVALUE, is renamed to the test name (matching the data set name) using the macro variable `&&LabName&j`:

```
   Data &&LabName&j;
      set &labds;

      if LTST = "&&LabLabel&j";
      rename LVALUE = &&LabName&j;
   run;

   Data &&LabName&j;
      set &&LabName&j;
      keep PT DCMDATE CPEVENT &&LabName&j;
   run;

%end;
```

Merging the data sets together

The final data set, which can be used for analysis, is created. The merging of the lab data sets is contained within a loop, and a macro variable is created:

```
%do j = 1 %to &TotalLabsinData;

%let new = %eval(&j + 1);
```

Before merging all lab data sets, the first lab data set is renamed WORK.temp1:

```
%if &j = 1 %then %do;
   data temp&j;
      set &&LabName&j;
   run;
%end;
```

Beginning with data set `temp1`, all of the data sets are merged together on patient ID, visit date and visit type, with each data set being named `temp(j+1)` until all of the test data sets have been merged:

```
   proc sql noprint;
      create table temp&new as
         select a.*, b.* from
            temp&j a left outer join &&LabName&j b
            on        a.PT = b.PT and
                 a.DCMDATE = b.DCMDATE and
                 a.CPEVENT = b.CPEVENT;
         quit;
   run;

%end;
```

```
run;
```

The final temp data set is then renamed to be used in the reporting macro:

```
%let TotalNum = %eval(&TotalLabsinData + 1);

Data &labds._report;
   set temp&TotalNum;
run;
```

The DATASETS procedure is used to delete unneeded temporary data sets:

```
%do j = 1 %to &TotalLabsinData;
   proc datasets;
      delete &&LabName&j;
   run;
%end;

proc datasets;
   delete temp;
run;

%do j = 1 %to &TotalNum;
   proc datasets;
      delete temp&j;
   run;
%end;

%mend DataManagement;
```

Result

Using the Blood Chemistry data set (LB1_CHEM), the data set WORK.LB1_CHEM_REPORT was created. The data set contains unique combinations of patient ID, visit type and visit date, as well as variables for each test, which contain the test results. This data set can be used for analysis of the Blood Chemistry data.

A portion of the original dataset from ORACLE is shown below:

PT	**DCMDATE**	**CPEVENT**	**LTST**	**LVALUE**
T29	20100102	SCREEN	Anion Gap	15.3
T29	20100102	SCREEN	Albumin	12
T31	20100102	SCREEN	Alanine Transaminase	1234
T31	20100102	SCREEN	Alkaline Phosphatase	123
T31	20100102	SCREEN	Glucose, random	80.12
D2	20090206	Day 1	Chloride	102.7
D2	20090206	Day 1	Albumin	56.25
D2	20090206	Day 1	Alanine Transaminase	99.60
D2	20090206	Day 1	Alkaline Phosphatase	79.91
X7	20100114	Day 4	Total Protein	8.1
X7	20100117	Day 7	Calcium	2.35
etc…				

A portion of WORK.LB1_CHEM_REPORT is shown below:

PT	**CPEVENT**	**DCMDATE**	**AlanineTransaminase**	**Albumin**	**AlkalinePhosphatase**	*etc…*
D2	SCREEN	20090115		8.8		
D2	Baseline	20090203	88.12	31.10	34.25	
D2	Day 1	20090206	99.60	56.25	79.91	
D2	Day 4	20090210	43.48	71.25	19.31	
D2	Day 7	20090213	65.27	18.59	73.14	
E1	SCREEN	20100101				
T29	SCREEN	20100102		12		
T30	SCREEN	20100116	1234			
T31	SCREEN	20100102	12345		123	
etc…						

Reporting Macro

PROC REPORT is used to further transform the resulting data from the data management macro into a patient chart-style layout.

Handling the test names

As with the data management macro above, data set information was stored in a series of macro variables:

- Test names from the variable LTST are stored in the macro list `LabName`…if test names have spaces or special characters (such as "Glucose, random" in the data snippet above), the names are compressed and special characters are stripped so that they are valid SAS variable names. Long lab names are truncated to the first 30 characters.
- The actual Lab Names (including any spaces and special characters) are stored in the macro list `LabLabel` for use later on.
- The number of unique labs in the data is counted and stored in the variable `TotalLabsinData`.
- In the macro definition, `testname` names the output file, `labds` is the same data set used in the data management macro, and `testlabel` is used in the report title:

```
%macro RunLabReport (testname = , labds = , testlabel = );

proc sql noprint;
   select distinct trim(left(LTST)), compress(trim(left(LTST)), " .+%/(),-")
      into :LabLabel1 - :LabLabel999, :LabName1 - :LabName999
      from &labds.;
   quit;
   %let TotalLabsinData = &sqlobs;
run;

%do j = 1 %to &TotalLabsinData;

   %if %LENGTH(&&LabName&j) > 30 %then %do;
      %let LabName&j = %SUBSTR(&&LabName&j, 1, 30);
   %end;
%end;
```

Creating Custom Bookmarks

Reports are outputted to PDF files with one page per patient. After consulting with a well-known expert in the field (see Acknowledgements below), a macro was found on *sasCommunity.org* to assign custom bookmarks. The PDF destination is assigned and unique patients are stored in a macro list similar to the lab macro lists above:

```
%macro custombookmarks;

   ods listing close;
   ods pdf file = "&SUGDIR./others/&testname..pdf";

   proc sql noprint;
      select distinct PT into :p1-:p999
      from &labds.;
      quit;
   run;
   %let numberOfPTS = &sqlobs;
```

Laying out the report

After indicating the title of the report and that page numbers should be printed, the report itself is laid out using the data set `&labds._report`, the data set that was created in the Data Management macro earlier. A loop within the report code and use of the WHERE statement in the PROC REPORT call indicate that separate reports will be run for each patient, all of which are contained in the same PDF document. ODS PROCLABEL creates the bookmark names using the patient macro list created above:

```
%do i= 1 %to &numberOfPTS;

   ods proclabel="Patient &&p&i";
   ods escapechar = '~';
   option nobyline;

   title1 "Test Data - &testlabel. Report";
   title2 "for patient &&p&i";
   footnote 'Page ~{thispage} of ~{lastpage}';

   proc report data = &labds._report
      nowindows colwidth = 4 spacing = 3 headline headskip split = "^" contents = '';
      where PT = "&&p&i";
```

Variables are then selected for the report. In addition to patient ID, unique visit date and type combinations are printed for each lab test, along with the result. These variables are printed in such a way that the visit type, date and result line up under the lab name, which spans the three columns. This is accomplished using a loop within the COLUMN statement in PROC REPORT and the lab name/label macro variables previously created:

```
      column PT
        %do j = 1 %to &TotalLabsinData;
            ("&&LabLabel&j" CPEVENT DCMDATE &&LabName&j)
         %end;;
```

When defining the variables in the report, a loop is used to print the appropriate lab results:

```
      define PT            / order group "Patient" noprint;
      define CPEVENT       / "Visit";
      define DCMDATE       / "Date";

      %do j = 1 %to &TotalLabsinData;
        define &&LabName&j   / "Value";
      %end;
   run;

%end;
```

Finishing the macro

All that is left is to close ODS PDF and clear out the titles and footnotes:

```
   ODS PDF CLOSE;

   footnote;
   title1;
   title2;

%mend custombookmarks;
```

The custombookmarks macro must then be run within the RunLabReport macro:

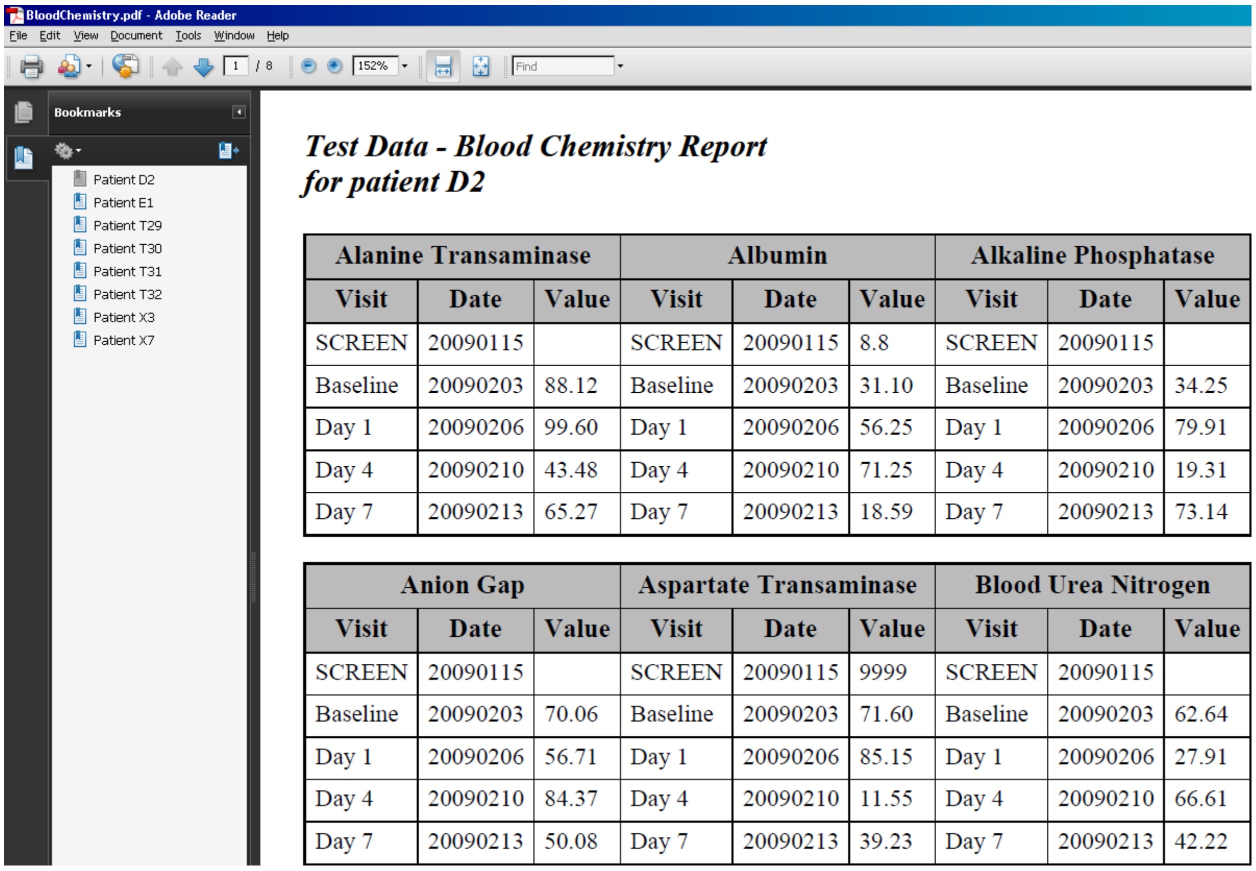
```
   %custombookmarks;
%mend RunLabReport;
```

The Report

The macro was run using the Blood Chemistry data set, called LB1_CHEM. The resulting report was output to BloodChemistry.pdf:

```
%RunLabReport (labds = lb1_chem,
               testname = BloodChemistry,
               testlabel = Blood Chemistry);
```

A screenshot of the PDF document is shown below:



BloodChemistry.pdf - Adobe Reader
File Edit View Document Tools Window Help
1 / 8
152%
Find
Bookmarks

Patient D2
Patient E1
Patient T29
Patient T30
Patient T31
Patient T32
Patient X3
Patient X7

Test Data - Blood Chemistry Report
for patient D2

Alanine Transaminase			Albumin			Alkaline Phosphatase		
Visit	**Date**	**Value**	**Visit**	**Date**	**Value**	**Visit**	**Date**	**Value**
SCREEN	20090115		SCREEN	20090115	8.8	SCREEN	20090115	
Baseline	20090203	88.12	Baseline	20090203	31.10	Baseline	20090203	34.25
Day 1	20090206	99.60	Day 1	20090206	56.25	Day 1	20090206	79.91
Day 4	20090210	43.48	Day 4	20090210	71.25	Day 4	20090210	19.31
Day 7	20090213	65.27	Day 7	20090213	18.59	Day 7	20090213	73.14

Anion Gap			Aspartate Transaminase			Blood Urea Nitrogen		
Visit	**Date**	**Value**	**Visit**	**Date**	**Value**	**Visit**	**Date**	**Value**
SCREEN	20090115		SCREEN	20090115	9999	SCREEN	20090115	
Baseline	20090203	70.06	Baseline	20090203	71.60	Baseline	20090203	62.64
Day 1	20090206	56.71	Day 1	20090206	85.15	Day 1	20090206	27.91
Day 4	20090210	84.37	Day 4	20090210	11.55	Day 4	20090210	66.61
Day 7	20090213	50.08	Day 7	20090213	39.23	Day 7	20090213	42.22

As desired, bookmarks for each patient are automatically created. The test names (with appropriate spacing and special characters, where appropriate) span the visit type, date and test result, and each patient has their own page of results. Page numbers are printed along the bottom of the report (not shown here).

Conclusion

This paper describes methods used to transform data from an ORACLE table into a SAS data set that can be used for analysis and reporting. The layout of an ORACLE database may seem confusing, but with a little patience and some macro knowledge, programmers can shape the data to make it more SAS-friendly.

References

(1) Carpenter A. Carpenter's Complete Guide to the SAS® REPORT Procedure. Cary, NC: SAS Institute Inc., 2007.
(2) Anonymous. "Customizing PDF By Variable Bookmarks." sasCommunity.org. <http://www.sascommunity.org/wiki/Customizing_PDF_By_Variable_Bookmarks> (April 19, 2010).

Acknowledgements

The authors wish to thank Jeffery Cohen, MD and Peter Imrey, PhD for their input on this paper. The authors also wish to thank Art Carpenter for his quick response to our cry for help and for his recommendation of the coding used above (see Report Macro Custom Bookmark section).

Contact Information

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

Name:	Shannon Morrison, M.S.
Enterprise:	Cleveland Clinic Foundation
Address:	9500 Euclid Avenue / JJN3-01
City, State ZIP:	Cleveland, OH 44195
Phone:	(216) 636-5413
Fax:	(216) 444-8021
E-mail:	morriss2@ccf.org
Web:	http://lerner.ccf.org/qhs