Paper PH-03-2014

Case Study: Generating Clinical Trial Summary Plots from an ORACLE database using the SAS® Macro Language

Shannon L. Morrison, M.S., Cleveland Clinic Foundation, Cleveland, OH

ABSTRACT

This paper details creation of summary plots for a clinical trial using data stored in an ORACLE database. The primary recipients for these plots were members of a Data Safety Monitoring Committee (DSMC) - plots needed to be informative and easy to read. After initial data cleaning, the SAS® Macro Language was used to step through the data to create two sets of plots - overall boxplots with jittered data points, and spaghetti panel plots - both showing results over time. Resulting plots were output to PDF files. Both the boxplots and spaghetti plots were grouped by test, and the spaghetti plots were laid out panel-style, with each patient's results contained in an individual panel.

INTRODUCTION

The Department of Quantitative Health Sciences (QHS) at Cleveland Clinic offers research support in the form of statistical analysis, data management services, and statistical programming. For regulatory purposes, projects that are funded by the National Institute of Health (NIH) or the Food & Drug Administration (FDA) typically use data that is stored in an ORACLE database. The layout of this data is often challenging from a programming point of view, as it is not appropriately set up for analysis or reporting straight out of ORACLE.

QHS began work on an FDA regulated trial in August 2010. Since then, the primary objective of the statistical team has been to create 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. These reports were modeled after a patient chart (one patient per table with as much information on that patient as possible per page) and were prepared quarterly to assess safety. The team created macros to generate both reports and analysis data sets for different labs, and this macro was detailed in a previous MWSUG paper (Morrison 2011).

Data collection was completed in early 2014 and the final DSMC meeting was held that spring. For this meeting, in addition to re-running the tables provided at earlier meetings using the complete data set, additional summary plots were requested by the statistician on the project to illustrate efficacy of the treatment.

The first set of plots contains grouped jittered box plots that show test results by time point using code from Harm and Herrera's paper (2012). They illustrate change over time for the entire group as well as individual results using both The Graph Template Language and The SGRender Procedure in SAS 9.3

The second set of plots contains panels of spaghetti plots and use code from Cheng's poster (2008). These plots show results for all individuals on the same axis for each test using The SGPanel Procedure in SAS 9.2.

For both of the plot types, the macro code was used along with the code from Harm and Herrara as well as the Cheng code to generate plots for each patient and test.

Minor data cleaning and manipulation was performed on the ORACLE data sets before using these macros. All plots were generated using PDF Output Delivery System.

The ORACLE Data

ORACLE databases containing the data were transferred into SAS files, which were sent to the data management team during each stage of the project. Our ORACLE team sends databases that contain many automatically generated variables (Clinical Study, Site, Investigator, etc.) as well as multiple versions of numeric variables (the value itself, a "DLV" value, and a "FULL" value), sometimes resulting in data sets containing at least 50 variables. For this analysis, only 5 variables were kept.

A data set containing fake data was created to illustrate the code presented in this paper. A portion of this data (made to represent Blood Chemistry Lab results) in ORACLE table form is shown below:

PT	DATE	CPEVENT	LTST	LVALUEN
P1	20100102	SCREEN	Anion Gap	15.3
P2	20100102	SCREEN	Glucose, random	23.2
P2	20100110	Day 1	Glucose, random	23.7
P2	20100114	Day 4	Total Protein	8.1
P2	20100117	Day 7	Calcium	2.35

In this data set, PT is the unique patient ID, DATE is the visit date, CPEVENT is the visit type (Screen, Month -1, Baseline, Day 1, etc.), LTST is the name of the test that was performed, and LVALUEN is the numeric result of that test. Almost all of the data sets that were needed for the summaries were formatted this way – those that were not needed small changes (for example: adding visit dates to the data set, changing a variable name, etc.). For each visit within each test, days from the Baseline visit were calculated as (Visit Date – Baseline Visit Date), resulting in negative responses when a specific visit took place before Baseline.

Review of the data manipulation macro

The goal for the summaries was, for each test, to generate one jittered boxplot and one set of panel spaghetti plots showing individual patient results. A quick solution was to apply the macro that the team had created in 2011 (Morrison 2011) to quickly step through the data and pick out each individual test so that the individual plots and tables could be created, rather than manually figuring out which tests were in the data set and creating a plot/table for each. The challenge was applying this macro to the new plot code.

In summary, the macro takes the original data set and, using The SQL Procedure, "looks" through the specific variable of interest (LTST in this case, the variable that contains the names of all of the tests from the study) to find each unique entry. The unique entries (test names) are then output to a macro list – if the entry has a space or special character, those are compressed so that good SAS naming conventions are met (for example: "Glucose, random" from the above table would be output as "Glucoserandom"). A second macro list containing the test labels is also output. The total number of unique entries is also counted and stored in a macro variable.

From here, each entry gets its own data set that only contains the patient ID, visit dates, visit name, test label and test result. All of the individual test data sets are then merged on patient ID, visit date and visit name, and the resulting data set now has one record per patient per visit, with each test result being stored in its own variable.

After applying the macro to the above test data set, the final data set for analysis is shown below:

PT	CPEVENT	DATE	AnionGap	Glucoserandom	TotalProtein	Calcium	
P1	20100102	SCREEN	15.3				
P2	20100102	SCREEN		23.2			
P2	20100110	Day 1		23.7			*etc…*
P2	20100114	Day 4			8.1		
P2	20100117	Day 7				2.35	
			etc…				

For these plots detailed in this paper, the final step of merging all of the data sets together was not needed, so the final PROC SQL step in the macro was not used. One new addition was to merge the units for each test into their respective data sets (variable named LABUNI) so that they could be included in the output.

Thus each plot was created using an automatically generated data set for each individual test. Output for the "Glucose, random" test is shown below.

Jittered Boxplots

Original code

The starting point for this set of boxplots is code from Harm and Herrara's paper from 2012. The paper contains code that generates a boxplot along with jittered data points using SAS 9.3. The goal for this project is to generate boxplots like this for each test and output them into one PDF document.

Harm and Herrara's original code uses the Graph Template Language (GTL) to create a template called OVERLAID. In the template, a boxplot is created, along with a scatterplot that is then overlaid on the boxplot. The code is shown below:

```
proc template;
       define statgraph Overlaid;
              begingraph /;
                     EntryTitle "&HGNC. (%trim (&Affy_ID.))";
                     EntryTitle "&Short_Description.";
                     EntryTitle "Group 1";
                     layout overlay/ xaxisopts = (display = none type = linear);
                            BoxPlot x = StudyNo y = Value/
                                   Group = StudyName
                                   Display = (caps median fill)
                                   LegendLabel = "log2 gene expression"
                                   Name = 'BOX';
                            ScatterPlot x = StudyNoJittered y = Value/
                                   markerattrs = (color = gray);
                            DiscreteLegend "BOX"/ Valign = top Border = false;
                     endlayout;
              endgraph;
       end;
run;

proc sgrender data = GraphData (where = (GroupName = 'Group 1'))
       template = Overlaid;
run;
```

This example in the original paper produces side-by-side boxplots with jittered data points for different study numbers. The goal for the study team's project was to produce a side-by-side boxplots with jittered data points for different visits, one per test.

Applying the macro to the original code

Only a small portion of the macro code is needed to complete these plots, specifically the very beginning of the macro that creates the data sets for each test:

```
%macro boxplots (ds = , dsname = , pdfname = );

*Create macro lists containing test names and labels;

proc sql noprint;
       select distinct LTST, compress(LTST, " .%+/(),-"), LABUNI
              into :TestLabel1 - :TestLabel999, :TestName1 - :TestName999, :TestUnit1 -
:TestUnit999
              from &ds.;
       quit;
       %let TotalTests = &sqlobs;
run;

*Check for long test names and truncate them to 28 characters;

%do i = 1 %to &TotalTests;
       %if %LENGTH(&&TestName.&i) > 28 %then %do;
              %let TestName.&i = %SUBSTR(&&TestName.&i, 1, 28);
       %end;
```

```
*Separate each test into its own data set for plotting;

       data &&TestName.&i;
              set &ds.;

              where LTST = "&&TestLabel.&I";
       run;
%end;
```

Each test now has its results contained in an individual data set. Before the plot code can be run, small corrections need to be made to each data set – re-labeling the results variable, coding the visit type (character) to a numeric variable (necessary to work with the plotting axis), and creating a jittered variable using the UNIFORM function in SAS so that the individual points deviate from one another slightly and can be seen more easily.

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

data &&TestName.&i;
       set &&TestName.&i;

            if CPEVENT = "SCREEN" then Visit = 1;
       else if CPEVENT = "BASELINE" then Visit = 2;
       …etc…
       else if CPEVENT = "MONTH 6" then Visit = 11;

       VisitJittered = Visit + 0.3*(uniform(2) - 0.05);

       Label
              LVALUEN = "&&TestUnit.&i"
       ;
run;
```

Now that the data sets are ready, the macro variables can be inserted into the original code to output the jittered boxplots. Other options are also inserted into the original code to determine size of the output, how the jittered points on the plots are printed, etc. New options are bolded in the code below.

```
proc template;
       define statgraph lab.Group;
              begingraph / designwidth = 1600px designheight = 1000px;
                     EntryTitle "&&TestLabel.&I (&&TestUnit.&i)";
                     layout overlay/ xaxisopts = (display = none type = linear);
                            BoxPlot x = Visit y = LVALUEN/
                                   Group = CPEVENT
                                   Display = (caps median fill)
                                   LegendLabel = "&&TestLabel.&I (&&TestUnit.&i)"
                                   Name = 'BOX';
                            ScatterPlot x = VisitJittered y = LVALUEN/
                                   markerattrs = (color = black symbol = circlefilled);
                            DiscreteLegend "BOX"/ Valign = top Border = false;
                     endlayout;
              endgraph;
       end;
run;
```

Data is then sorted by Visit so that the boxplots would print in the correct visit order.

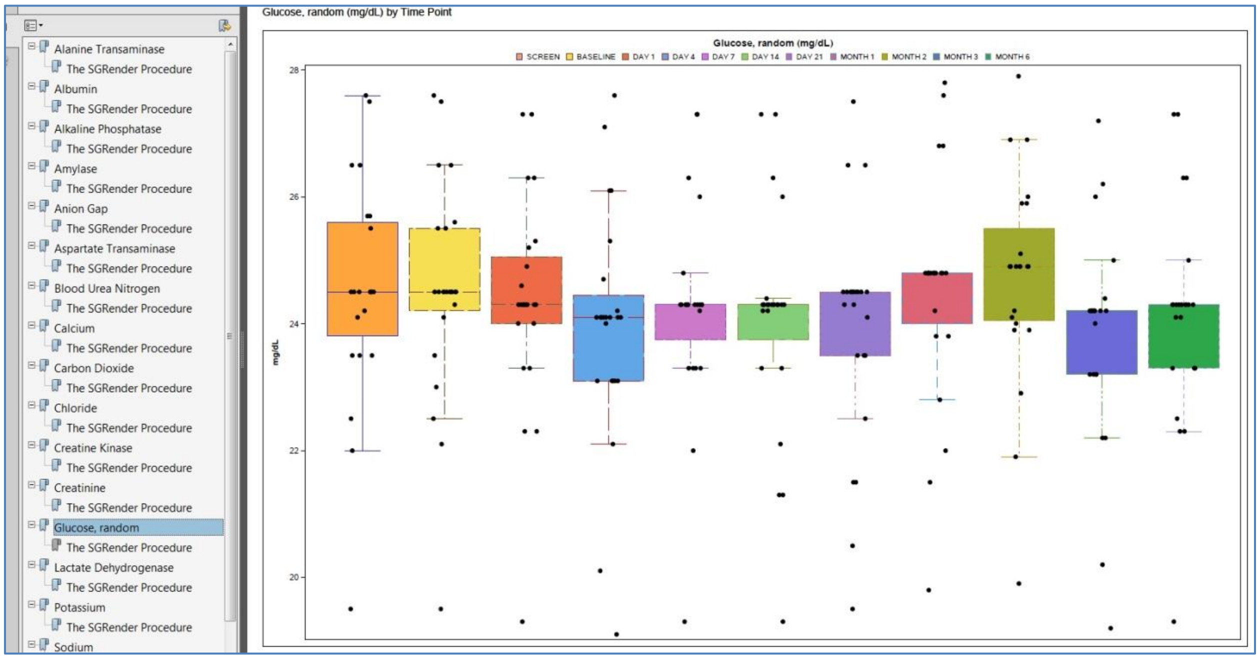
```
proc sort data = &&Testname.&i;
       by Visit;
run;
```

Since these plots were being output to PDF documents using the Output Delivery System. ODS PROC LABEL is used to create a bookmark in the PDF document for each test.

```
options orientation = landscape papersize = a3;
ODS GRAPHICS ON;
ODS PDF FILE = "&BASEDIR./others/Boxplots&pdfname..pdf";

ods proclabel "&ds.";

proc sgrender data = &&TestName.&i template = lab.Group;
       title "&&TestLabel.&i (&&TestUnit.&i) by Time Point";
run;
%end;

title;

ODS PDF CLOSE;
ODS GRAPHICS OFF;

%mend;
```

Results

The resulting sex of side-by-side boxplots for the "Glucose, random" test is shown below. Again, using the macro code, each test will have its own set of boxplots as part of the overall PDF document. Individual data points are jittered and overlaid, allowing outliers to be easily spotted while, at the same time, giving an overall summary of the data.



Panel Spaghetti Plots

Original code

For this set of plots, code from Cheng's paper is used. The original code is shown below:

```
proc sgpanel data = labdata (where = (labtest = "Creatinine"));
       panelby subjid / rows = 4 columns = 4;
       series x = visit y = result;
       colaxis values = (-6 to 105 by 7);
```

```
run;
```

This example code produced a panel of individual spaghetti plots, one for each subject, showing their Creatinine results. Each of the plots share a common axis scale for easy comparison.

Applying the macro to the original code

As with the above boxplot, macro code is applied to create individual data sets for each test. The Output Delivery System is also used for these plots.

```
%macro lineplots (ds = , dsname = , pdfname = );

*Create macro lists containing test names and labels;

proc sql noprint;
       select distinct LTST, compress(LTST, " .%+/(),-"), LABUNI
              into :TestLabel1 - :TestLabel999, :TestName1 - :TestName999, :TestUnit1 -
:TestUnit999
              from &ds.;
       quit;
       %let TotalTests = &sqlobs;
run;

*Check for long test names and truncate them to 28 characters;

%do i = 1 %to &TotalTests;
       %if %LENGTH(&&TestName.&i) > 28 %then %do;
              %let TestName.&i = %SUBSTR(&&TestName.&i, 1, 28);
       %end;

*Separate each test into its own data set for plotting;

       data &&TestName.&i;
              set &ds.;

              where LTST = "&&TestLabel.&i";
       run;
%end;

Options orientation = landscape papersize = a3;
ODS GRAPHICS ON;
ODS PDF FILE = "&SUGDIR./LinePlots&pdfname..pdf";
ods proclabel "&ds.";

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

For each test, the results variable is re-labeled so that the test unit prints out on the y-axis. The data is then sorted first by patient ID, then by days from the Baseline visit so that panels are printed and displayed in the proper order.

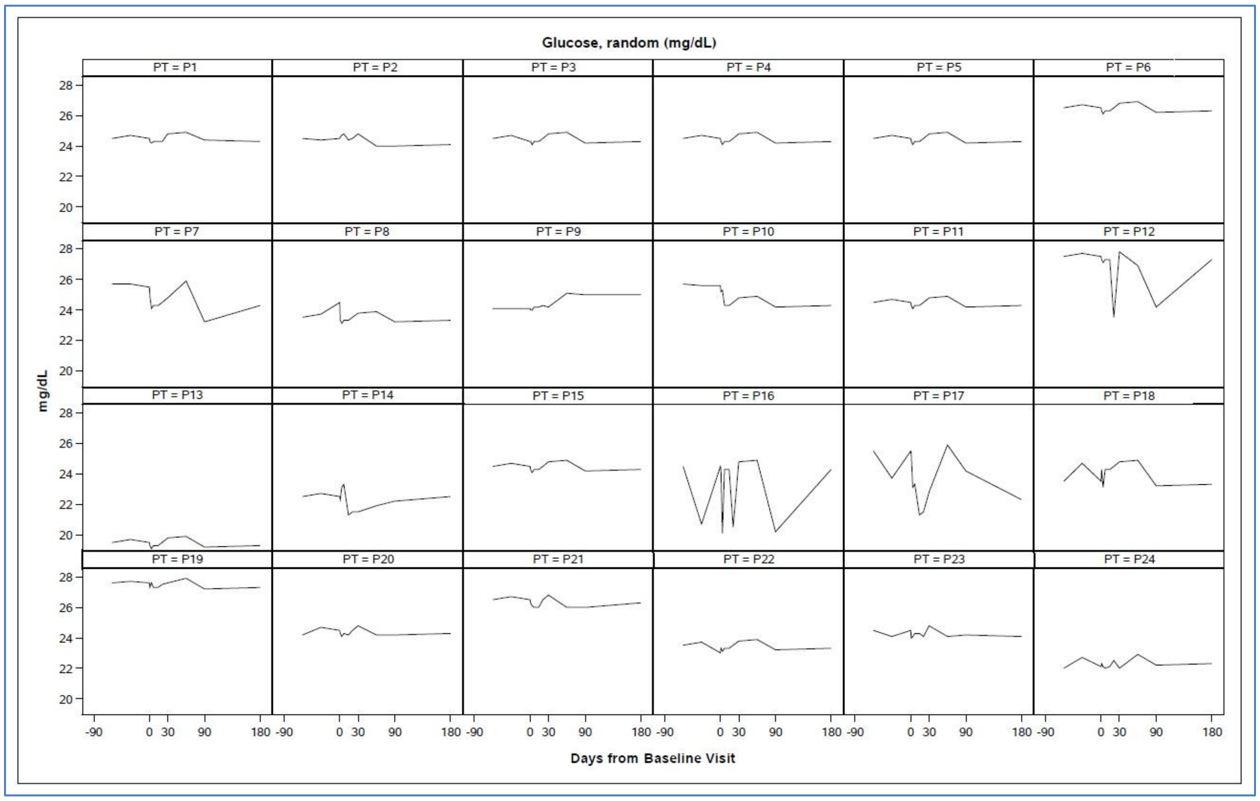
```
data &&TestName.&i;
       set &&TestName.&i;

       label
              LVALUEN = "&&TestUnit.&i"
       ;
run;

proc sort data = &&TestName.&i;
       by PT days_baseline;
run;
```

The macro variables can now be inserted into the original code to output the jittered boxplots. Other options are also inserted into the original code to determine size of the output, the number of columns per page, and to offset the x-axis points from the sides of the plots. Results are printed at 3 months from Baseline (-90 days), Baseline (0 days), Month 1 (30 days), Month 3 (90 days) and Month 6 (180 days). Changes to the original code are indicated in bold type.

```
title "&&TestLabel.&I (&&TestUnit.&i")
ods graphics / width = 16in height = 10in;
proc sgpanel data = &&TestName.&i;
       panelby PT / rows = 4 columns = 6;
       series x = days_baseline y = LVALUEN;
       colaxis values = (-90 0 30 90 180) offsetmin = 0.06 offsetmax = 0.06;
run;

title;

%end;

ODS PDF CLOSE;
ODS GRAPHICS OFF;
%mend;
```

Results

The resulting panel plot for the "Glucose, random" test is shown below. Again, using the macro code, each test has its own panel as part of the overall PDF document. There is one panel per patient, and all results are on the same axis for easy comparison. This allows investigators to identify potential problems or to see steady progress (for example – most patients are steady except for #12, #16, #17 and #18 – why did they have big changes? Also, why are patient #13's values so low compared to everyone else?).



CONCLUSION

This paper describes a method used to apply the SAS Macro Language to existing code. The layout of an ORACLE database is not appropriate for analysis – the macro code transforms the data, and by using macro variables in the existing SAS code, plots can quickly be generated and used to show clinical data summaries both individually and overall for the cohort in a clinical trial.

REFERENCES

(1) Anonymous. “Customizing PDF By Variable Bookmarks.” sasCommunity.org. <http://www.sascommunity.org/wiki/Customizing_PDF_By_Variable_Bookmarks> (April 19, 2010).
(2) Cheng, Wei. “When Simpler is Better – Visualizing Laboratory Data Using ‘SG Procedures.’” Western Users of SAS Software (WUSS). <http://www.wuss.org/proceedings08/08WUSS%20Proceedings/papers/pos/pos01.pdf> (November 2008).
(3) Harm, Volker and Herrera, Catalina Mejia. “Grouped Jittered Box Plots in SAS 9.2 and SAS 9.3.” PhUSE. < http://www.phusewiki.org/docs/2012/PAPERS/CS/CS03.pdf> (October 2012).
(4) Morrison, Shannon. “Case Study: Generating a DSMC report from an ORACLE database with SAS® PROC REPORT.” MidWest SAS Users Group. < http://www.mwsug.org/proceedings/2011/pharma/MWSUG-2011-PH04.pdf> (September 2011).

ACKNOWLEDGMENTS

The author wishes to acknowledge Matt Karafa for his help with the original data transformation macro.

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
Work Phone:	(216) 636-5413
Fax:	(216) 444-8021
E-mail:	morriss2@ccf.org
Web:	http://lerner.ccf.org/qhs