Using the FRAME Entry to Develop a Graphical-based Data Review Application.

David A. Bright, Pfizer Central Research, New York, NY

Abstract

The FRAME entry provides a user-friendly environment for creating novel applications that take advantage of object-oriented programming techniques. The FRAME entry provides many pre-defined classes that can be used to create objects. The incorporation of Screen Control Language (SCL) into FRAME entries has enabled users to develop custom applications to take advantage of the analytical tools that the SAS System provides. By using several widget classes and by creating additional customized objects, one can easily build an application.

In this tutorial, a programmer will learn how to create a prototype graphical review tool for the drilldown and analysis of data. Creation of this graphical review prototype will incorporate the exploration of several unique FRAME entry classes. By using the FRAME entry for development of mission-critical applications, a programmer can provide their company with quality analysis in a timely manner.

Introduction

In order to understand the FRAME Entry environment, one should have a grasp of the underlying programming processes that this entry is based upon. The name of this programming technique is called Object-oriented programming (OOP) methodology. Any application created with FRAME entries use many of the concepts and tools of this programming technique. In addition to supporting many of the features of OOP technology including graphical displays and widgets, the FRAME entry supports all the functionality of PROGRAM entries. The objective of this paper is to introduce various ways for using the FRAME entry to accommodate the graphical review of data.

Object-Oriented Programming Methodology

OOP methodology incorporates the ability of an application programmer to design procedural code from individual objects. Each object has a list of prescribed attributes that define the context of the object. Once an object is created, it can be used repeatedly without the need to redefine its attributes for another application. A set of reusable objects that become standardized for any application can be referred to as a toolkit. This toolkit can be ported to any internal operation and can be the source for reusing any tools created from prior applications.

There are several significant benefits for utilizing OOP methodology in your application programming: 1) The creation of object libraries or toolkits can aid in quicker development time by using standardized objects for development 2) Proven validated objects can be reused and assures higher quality 3) The modularity of the objects in the context of the applications require less maintenance 4) Applications can be developed at a faster rate 5) Less manpower needed to support new and existing applications resulting in cost reduction and 6) The ease of adapting to changes in system requirements by the extendability of your objects and toolkits.

OOP Features supported by FRAME entry

The FRAME entry provides several predefined toolkits called classes that you can use to create objects. Once these items are created, you can customize any of the objects and change default attributes to suit the nature of your application.
By using existing classes and by creating additional customized objects, one can easily build an application or a prototype in an expeditious manner. These new objects can be added to your organization's toolkit without having to redefine the attributes that were previously developed. In combination with other tools provided within the SAS System, one can create a refined graphical data review application that can be portable across a variety of platforms.

**Hardware and Software Requirements**

Development of this data review application was performed on an IBM PS/2 model 95 computer and a NEC MultiSync 5FG monitor, with a 80486 microprocessor running at 33 MHz and was configured with 32 MB of RAM. Additional peripheral devices include a Microsoft Mouse and a Hewlett-Packard LaserJet IV printer. The operating system was IBM OS/2 version 2.1 and SAS version 6.08 was used for the development of the data review application. The following SAS products were used in this application: Base SAS, SAS/AF, SAS/GRAPH, SAS/IML, SAS/STAT, and SAS/FSP.

**FRAME Entry Data Review Application**

There are two different graphic types that are supported by the FRAME entry. The first graphic type displays output from a prescribed graphics catalog (GRSEG) entry and is static. The name of this FRAME entry class is called the SAS/GRAPH Output class. The second graphic type enables the user to create and display a variety of plots and charts that are generated dynamically from within the FRAME entry. The FRAME entry class that supports this type of graphic is called the Graphics Class. Unlike the SAS/GRAPH Output Class, the plots and charts that are generated by this class do not already exist in a prescribed graphics catalog.

In addition to these two graphic classes, the data review application uses two other FRAME entry classes: the Hotspot Class and the general utility class called Widget Class. The SAS/AF FRAME entry provides over 20 other classes that one can use to assign attributes to objects. These can easily be combined within the same FRAME entry to form a dynamic and versatile application.

There are two procedures that one can use to create a FRAME entry for the display of graphical data. The first procedure is to create the entire object using the class dialog window which provides fill-in boxes to assign attributes. This may initially be the easiest procedure to learn about the object's attributes. The second procedure is to create all the object's attributes from within the SCL methods. Each class provides the programmer the ability to use customized methods for displaying graphics. The advantage to using class methods is the ability to pass SCL global macro variables through the code. This makes each FRAME entry much more flexible for coding structure reusability.

**The SAS/GRAPH OUTPUT Class**

The SAS/GRAPH Output Class allows one to display a cataloged graphic segment within a FRAME in order to have the ability to assign intelligence behind this graph. The graphic segment must be cataloged and is therefore a static element. This particular class would be most useful for an EIS (Executive Information System) where there are a finite number of graphs that you would want to display. Another technique that could be utilized is to create a temporary graphic segment from SAS/GRAPH, SAS/IML or DSGI (Data Step Graphic Interface) and load this temporary graphic segment into your FRAME application. The graphic objects within this class allow one to designate special areas within the graph called hotspots. These hotspots make areas around the graphic segment sensitive to input from the keyboard or the mouse.
The design of an application to take advantage of this class requires the selection of a cataloged plot. To illustrate this class, a cataloged graphic segment generated from SAS/IML will be used for the application. After invoking AF, edit the FRAME entry called PLOTOUT.FRAME and using the second mouse key (or PMENU selection) choose MAKE, and then SAS/GRAPH Output. Once the size of the region for this object has been drawn, put the cursor in the middle of the object and use the second mouse button to display a new menu. Selection of the OBJECT ATTRIBUTES will start the dialog boxes for the SAS/Graph Output Class (Figure 1). The name of the graphics object will be PLOT2 and the cataloged graphic name will be SUGILABS.LABGRAPH.GRSEG.

Once a SAS/Graph Output object is selected, hotspots for the object can be created using the HOTSPOT Class. Positioning the cursor anywhere on the graph and using the second mouse key select TURN HOTSPOT MODE ON and then position the cursor on any part of the graphic region and press the second mouse key to get a pop-up menu. Select MAKE and HOTSPOT from this menu and size the hotspot box area to the region of interest. Once the hotspot area has been completed, point inside the hotspot area and use the second mouse button to assign the HOTSPOT attributes from the pop-up menu. Name the hotspot SPOT1 (Figure 2) and select the CYAN color for the hotspot. Under the hotspot's additional attributes window, one can assign several command processing strings that would be useful in an application (Figure 3).

Another way of presenting the same processing without going through the dialog window procedure is documented and shown in Appendix 1. The SCL method approach to programming facilitates modular and reproducible coding structures. Figure 4 shows the run-time application of this FRAME entry using the SAS/Graph Output class to produce a plot with hotspot selections that drill down to a screen (Figure 5) of data that can be used as an aid for elaborating on significant changes in pre-vs post-data values.

The GRAPHIC Class

The Graphic Class allows the programmer the ability to create plots and charts dynamically. Once a dataset is chosen and axis variables selected, fifteen plot types can be chosen to display graphical data. After a plot is displayed, users can retrieve data for any selected point by moving the cursor to the point of interest and clicking. This will open the Data Information window that will permit users to observe values associated with a particular data point on the graph.

Because the Graphic class graphic segment is created within the FRAME entry, there is no need to reference any cataloged graphic segment. The initial sequence for creating this FRAME entry is similar to that of creating a FRAME entry using the SAS/Graph Output Class. After invoking AF, edit the FRAME entry called LABPLOT.FRAME and using the second mouse key (or PMENU selection) choose MAKE, and then GRAPHICS. Once the size of the region for this object has been drawn, put the cursor in the middle of the object and use the second mouse button to display a new menu. Selection of the OBJECT ATTRIBUTES will start the dialog boxes for the Graphics Class (Figure 6). The name of the object will be PLOT1 and the name of the underlying dataset will be SUGILABS and the plot type will be Scatter. Select PREHCT for the X-variable and PSTHCT for the Y-variable and suppress the legend. Additional graphic attributes can be assign from this window (i.e colors and titles). To enable the Data Information window, put the cursor anywhere within the graphic region and using the second mouse key select View Data On from the pull-down menu.

To attain a similar level of functionality without going through the Graphic class dialog boxes is shown in Appendix 2. This program documents the raw SCL methods that were used to create
the data application. Figure 7 illustrates the runtime application of this FRAME entry using the Graphics class to produce "on the fly" scatter plots of data with the ability to identify each individual's plotted values within the plot and to drilldown on other information pertinent to the point selected (Figure 5).

**Summary**

The FRAME entry provides a user-friendly environment for creating novel applications that take advantage of object-oriented programming techniques. All the applications developed are platform independent and utilize the latest hardware and software technologies.

The FRAME entry enables a programmer to develop reusable, modular coding structures by taking advantage of object-oriented programming methodology. Within the FRAME entry, there are two classes that enable users to develop applications that utilize graphic output: the SAS/Graph Output and the Graphics classes. By defining objects within the FRAME entry using SCL methods or dialog windows, several applications with the ability to drilldown on key segments of your data have been presented. Development of rapid prototypes, similar to the data review applications presented, can provide a company with quality analysis in a timely manner.

The author may be contacted at:

Pfizer Central Research
235 E. 42nd Street
219-4-20
New York, NY 10017
(212) 573-4553

**References**


**Acknowledgements**

The author would like to thank Daniel Gronell and Yvonne Selby for their assistance in the development of this FRAME entry tutorial.

Base SAS, SAS/STAT, SAS/AF, SAS/GRAPH, SAS/IML and SAS/FSP are registered trademarks of SAS Institute Inc., Cary, NC, USA.

IBM, PS/2 and OS/2 are registered trademarks of International Business Machines Corporation, Armonk, NY, USA.

Microsoft and Microsoft Mouse are registered trademarks of Microsoft Corporation, Redmond, WA, USA.

Hewlett-Packard Laserjet IV is a product of Hewlett-Packard Company, Mountain View, CA, USA.

NEC is a registered trademark of NEC Technologies, Inc., Boxborough, MA, USA.
Figure 1: SAS/Graph Output Class Dialog Box that displays object attributes.

Figure 2: Hotspot Class Dialog Box that displays object attributes.

Figure 3: Command String assignment window

Figure 4: Run-time application of SAS/Graph Output class.
Figure 5: Patient Safety Profile Screen results from drilldown.

Figure 6: Graphic Class Dialog Box that displays object attributes.

Figure 7: Run-time application with Data Information Window for Graphic class
APPENDIX I : SCL METHODS FOR SAS/GRAPH OUTPUT AND HOTSPOT CLASS

******************************************************************************
• Program name: PLOTOUT.FRAME
• Developer: DAVID A. BRIGHT
• Date: 12/8/92
• Purpose: To display cataloged laboratory scatterplots with the ability to use hotspots to drilldown on subsetted patients to address safety issues.

******************************************************************************

This program illustrates how to design and build a patient lab safety drilldown system using the FRAME entry and the SAS/graphics Output and Hotspot class widgets. Although one could use the FRAME entry dialog windows to code many aspects of this system, this gives an alternative way to do modularized coding.

INIT:

PLOT2:
call notify('plot2','_SET_GRAPH_','SUGI.LABS.LABGRAPH.GRSEG'); /* Designates graphic segment */
call notify('H0T1',"_SHOW_HOTSPOTS_", "CYAN", "REVERSE"); /* Shows hotspot location */
return;

MAIN:

HOT1:
* For display only to demonstrate METHOD features *;
call notify("HOT1","_GET_VALUE_", ilist); /* Uses SCL list for Hotspot class */
* Diagnostic *
ulist = putlist(ilist);
hots = getnitemc(ilist, "OBJNAME"); /* Returns SCL list value for OBJNAME */
* Diagnostic *
put 'The value of hotspot is' hots;
if hots = 0 then return;
submit continue;
proc fsbrowse data:sugi.labs
  screen = sugi.labs.patprof.screen;
  where (study = 'ST1');
run;
endsubmit;
return;

HOT2:
submit continue;
proc fsbrowse data=sugi.labs
  screen = sugi.labs.patprof.screen;
  where (study = 'ST2');
run;
endsubmit;
return;

HOT3:
submit continue;
proc fsbrowse data=sugi.labs
  screen = sugi.labs.patprof.screen;
  where (study = 'ST3');
run;
endsubmit;
return;

TERM:
call notify('plot2','_CLEAR_'); /* Clears all graphic segments */
return;

1536
APPENDIX 2: SCL METHODS FOR GRAPHIC AND GRAPHIC TEXT CLASS

* Program name: LABPLOT.FRAME
* Developer: DAVID A. BRIGHT
* Date: 12/8/92
* Updates: 2/20/94 Adapted for SUGI 19 Presentation
* Purpose: To display laboratory scatterplots with the ability to drill down on individual patients to address safety issues.

This program illustrates how to design and build a patient lab safety drill down system using the FRAME entry and the Graphics class widgets. Although one could use the FRAME entry dialog windows to code many aspects of this system, this gives an alternative way to do modularized coding.

INIT:

PLOT1:
call notify('plot1', 'SET TITLE', 1, 'Laboratory Graph - Hematocrit'); /* Set title */
call notify('plot1', 'SET FONT', 'ZAPF'); /* Set title font */
call notify('plot1', 'SET JUSTIFY', 'CENTER'); /* Set title justification */
call notify('plot1', 'SET COLOR', 'TEXT', 'WHITE'); /* Set text color */
call notify('plot1', 'SET CMD', 'SUGI.LABS'); /* Define dataset */
call notify('plot1', 'SET_TYPE', 2); /* Scatterplot graph designation */
call notify('plot1', 'SET_INDEP_VAR', 'PREHCT'); /* Selection of X-variable */
call notify('plot1', 'SET_DEP_VAR', 'POSTHCT'); /* Selection of Y-variable */
call notify('plot1', 'SHOW LEGEND', 0); /* Suppress display of Legend */
call notify('plot1', 'SET COLOR', 'AXIS', 'CYAN'); /* Color axis cyan */
call notify('plot1', 'HIGHLIGHT', 1); /* Highlight selected data points */
call notify('plot1', 'SET_COLOR', 'HIGHLIGHT', 'YELLOW'); /* Color highlight yellow */
call notify('plot1', 'SHOW CLICK_INFO', 1); /* Display data values for selected point */
call notify('plot1', '_SET_VALUE', VALUE); /* Returns value of selected point into */
return;

MAIN:

LABLIST = getniteml(VALUE, 'DEPVALUE'); /* Get the X-value from SCL sublist */
PSTVAL = getnitem(LABEL, 'VALUE'); /* Get the X-value from SCL list */
** Diagnostic **;
put 'The value of the pre lab value is' PSTVAL;

PREVAL = getnitem(VALUE, 'INDVALUE'); /* Get the Y-value from SCL list */
** Diagnostic **;
put 'The value of the pre lab value is' PREVAL;

submit continue;
proc fsbrowse data=sugi.labs /* Drilldown into patient safety data */
screen = sugi.labs.patprof.screen; /* by using the pre- and post-treatment */
where (prehct = preval) and (posthct = postval); /* lab values. */
run;
endsubmit;
return;

TERM:

return;