Objective

Develop a graphic procedure that provides data-driven capabilities in producing SAS/IML X*Y plots.

Abstract

Graphical representation of data is by far one of the best methods that we can employ to expedite comprehension of information. Many application systems will often include a graphic subsystem to give its users the ability to plot and/or chart data in a form that not only aids understanding, but adds to presentation value. However, when the situation occurs that involves the plotting of points very dissimilar in magnitude, problems abound. At worst, the graph for some data cannot be produced; at best, the graph is presented in a somewhat distorted display of axes and points.

A graphic subsystem was developed by the CANDA group at Pfizer Inc, CSA Department for such situations. (NOTE: CANDAs are Computer-Assisted New Drug Applications). Using SAS/IML graphics software, a data-driven algorithm was coded which produces a relatively symmetrical X*Y plot of pre- vs. post-treatment data points regardless of their magnitude. Our main application of this graph system is for the FDA review of laboratory data in the CANDA submissions, though it functions as an in-house review tool as well. This subsystem has a wealth of applications when work involves producing aesthetic X*Y plots, even beyond laboratory pre- vs. post-treatment tests. The primary advantage of this facility is the disregard for the varying data measurements involved when generating lab X*Y plots. As a result, development time per CANDA is decreased tremendously since customization of graphing code is not necessary.

Algorithm Specifications

When determining the breadth of what this algorithm should accomplish, a list of requirements were agreed upon by the CANDA group. The goal was to have the graphs maintain the same "appearance" regardless of the data being plotted as well as be an analytical tool for the user. Thus, the program had to display a graph that:

1. is relatively symmetrical
2. has 4-7 major tick marks
3. has 2-4 minor tick marks
4. covers 85-90% of screen display
5. allows for printing of titles and axis labels
6. color codes all aspects of the graph (text, points, axes, lines)
7. draws a reference line (signifying no change between the X and Y values)
8. draws a rectangular grid within the plotting area to help determine values for points (grid lines will be placed halfway between tick marks).

Environment

The initial development of this program was done on an IBM® PS/2 Model 80-20 MHz microcomputer under the OS/2® operating system using Release 6.06 of the SAS® System. The algorithm's average run-time on this machine was approximately 34 seconds. However, the algorithm now resides on an IBM® PS/2 Model 95-33 MHz machine, where the average run-time is 14 seconds. Obviously, we are delighted with the over-50% increase in performance.

Why SAS/IML?

The SAS/IML® Graphics (referred to simply as SAS/IML) package was chosen to develop this program due to a number of factors.

-Call Functions. SAS/IML provides the ability to easily draw lines, generate grids, and scale axes through the use of subroutines.
-Matrices. SAS/IML, with its matrix language, allows for xy coordinates to be stored in matrices and vectors, which is inherent when working in SAS/IML. This gives us the ability to keep "natural" pairs of values together. Since plotting is being done within the X*Y system, points (xy pairs) on the graph, such as line start and end values, can be kept neatly in these structures.
-Control. The control over the entire graph area (including placement of text, grid, and plotting region) provides great flexibility as well as ease of modification.

Also of importance was the fact that much of the previous work on CANDA laboratory graphs were done in SAS/IML. Familiarity with the package as a CANDA development tool figured prominently in the decision to code the algorithm in SAS/IML.
Evaluations of both the DATA Step Graphics Interface (DSGI) and SAS/GRAPH® (GPLOT/ PLOT procedures) were done and the findings were compared with the capabilities of SAS/IML. Brief assessments of each is detailed below.

**DSGI.** In stored, compiled-program form, DSGI graphs can be produced much more quickly than SAS/IML plots. Unfortunately, the compiled program does not allow for the passing of parameters, thereby defeating dynamic capability. Also, the generation of axes, major tick marks with their labels, minor tick marks, and the grid must be done in a more manual way since there are no specific functions that can produce these effects (e.g., lines must be drawn horizontally and vertically to make up the grid). Thus, more program code is needed to facilitate the requirements of the X*Y plot, which increases the run-time in comparison to the SAS/IML algorithm.

**SAS/GRAPH (GPLOT/PLOT Procedures).** The flexibility and control over the graphing region is on par with that of SAS/IML. With no significant difference in run-time execution and possessing very similar capabilities, the main reason for not coding the algorithm in one of these procedures was the generation of the grid. There is no way to produce the grid requirements in the GPLOT or PLOT procedures (positioned inside the axes, lines between each tick mark).

These evaluations are not implying that these other SAS products cannot be used to code a graph algorithm. For our specific criteria, SAS/IML proved to be the best candidate for the job. Further development and enhancement to the algorithm will include the continual assessment of all SAS graphing products. Moreover, it is not unlikely that, in the future, some combination of products is coded to take advantage of certain methods offered by each (e.g., GPLOT’s interpolation, SAS/IML’s grid generation, DSGI’s annotating ability).

**Algorithm Description**

Presented here is a narrative list of what the code within the algorithm should do.

1. Determine the minimum and maximum values for the x and y axes (they will be the same points due to symmetry).
2. Determine the minimum and maximum points for the entire graph area.
3. Determine the appropriate number of major and minor tick marks.
4. Draw and scale axes using range and tick mark data.
5. Plot points and calculate number of points plotted (N).
6. Draw a grid within the axes that is lined up between each major and minor tick mark.
7. Draw a diagonal reference line within the grid, approximately from the origin area to the maximum plotting area (upper right corner).
8. Place titles and axis labels on graph.

Figure 1 below displays a graph produced by the algorithm from lymphocytes data.

![Figure 1: Graph produced from lymphocytes data](image)

**Algorithm Pseudocode**

Pseudocode is presented here to aid in the understanding of the program's construction. Examples of code structures are also given, with SAS/IML functions and pneumonics in bold italics. These steps do not necessarily correspond with the narrative list previously presented.

**Step 1.** With the USE and READ statements, read in the pre- and post-treatment values from the lab dataset.

```plaintext
use dataset-name;
read all var {pretreatment-values-dataset-variable-name}
```
Both pre- and post-treatment matrices will be Nx1 in dimension.

Step 2. Obtain the minimum and maximum for the pre- and post-treatment values together. Remember, the plot is symmetrical. Use the MIN and MAX functions.

\[
\text{lowest} = \min(\text{pretrt-values-matrix-name}, \text{posttrt-values-matrix-name});
\]

Note that the "lowest" variable in the above example is a 1x1 matrix, or scalar. By applying the minimum and maximum values, calculate the range for the axes, the number of major tick marks, and the number of minor tick marks. This can be facilitated easily by using Marcia Murto's programming techniques presented at SUGI 15.

Step 3. Define the world coordinates for the entire graph area. Within SAS/IML, the world coordinate system simply means that the graphing region is defined by the data. A detailed explanation of this concept can be found in the SAS/IML® Software manual, p. 171. It is this system that eases the declaration of the limits to the plotting area in SAS/IML. The minimum and maximum values will be the limits. By horizontally concatenating (symbolized by \( \| \) in the example below) the minimum with itself and the maximum with itself, these values can be used by the GWINDOW function to declare the size of the graph area. Note in the example that the values are vertically concatenated \( (\|) \) also to conform to the GWINDOW function specifics, making the "window" variable a 2x2 matrix.

\[
\text{window} = (\text{min} \| \text{min})/(\text{max} \| \text{max});
\]

Step 4. Draw the x and y axes using the GXAXIS and GYAXIS functions. The four pertinent parameters for these two functions are the origin (minimum horizontally concatenated with itself), the range, and the number of major and minor tick marks. Scaling is done automatically with the use of this information.

\[
\text{call gxaxis(} \text{origin-matrix-name,}
\text{ range-matrix-name,}
\text{ minor-ticks-matrix-name,}
\text{ major-ticks-matrix-name);}
\]

Step 5. Plot the points within the axes using the GPOINT function. Simply use the matrices that hold the pre- and post-treatment values (created in Step 1). The matrix that holds the x-axis values must be the first argument specified in the GPOINT function (in this case, the pre-treatment values); the corresponding matrix for the y-axis values must be the second argument. Note also that a plot symbol and color are specified.

\[
\text{call gpoint(} \text{pretrt-values-matrix-name,}
\text{ posttrt-values-matrix-name,}
\text{ "star", } \text{"yellow"});
\]

It is here that the determination of N, the number of points plotted, can be done. However, recognize that it is simply a one-line statement and may be placed elsewhere in the program. Use the NROW function on either one of the plotting matrices. This function will give you the "N" value of the chosen Nx1 plotting matrix. Since each row represents a patient (see Usage Notes), this value is the N of the graph.

\[
\text{N} = \text{nrow(} \text{pretrt-values-matrix-name});
\]

Step 6. Using the value of the space between all tick marks (calculated by the range divided by the major ticks divided by the minor ticks plus 1 for the origin), draw the reference line and the grid. For the reference line, add this "interval" value to the minimum and horizontally concatenate with itself:

\[
\text{min} + \text{interval} \| \text{min} + \text{interval}
\]

The above value would be loaded into a matrix and serve as the start point since the reference line should start one interval length above and to the right of the origin. Do the same with the maximum except subtract the interval value; this is the end-point (one interval length below and to the left of the maximum). These xy pairs can then be used by the GDRAWL function to draw the line.

\[
\text{call gdrawl(} \text{start-point-matrix-name,}
\text{ end-point-matrix-name, } \text{"red"});
\]

For the grid, code a do-group that will load two matrices: a one-column (Nx1) matrix will hold the x values, another one-column matrix will hold the corresponding y values. To position the grid aesthetically and logically, each grid line should be between a tick mark. To do this, the do-group code must iterate between the minimum value (the origin) plus half of the interval to the maximum value minus half of the interval by the interval:
do gpoint = min+(interval*.5) to max-(interval*.5) by interval;

gridpts = gridpts // gpoint;
end;

This sets the entire grid within the axes and its lines between the tick marks. To draw the grid, specify the GGRID function. Again, due to symmetry, the same points (held in matrix "gridpts") can be used for both horizontal and vertical lines.

call ggrid(gridpts,gridpts);

Step 7. Print titles, axes' labels, and the N of the graph. By using the GTTEXT, GVTEXT, and/or GSCRIPT functions along with minimum, maximum, and interval values, text can be positioned anywhere within the graph area. The example below shows one way in which to print the y-axis label, POST TRT:

call gtext(min-(interval*2),
max-(interval*6),
"POST TRT","white");

Summary

The algorithm developed at Pfizer has been tested on lab data measurements from red blood count (x10⁶) to platelets (x10⁶). A graph of RBC data, in which the algorithm generated "nice" endpoints of 3.00 and 7.00 for the axes, is displayed in the next column (Figure 2).

For a graph produced from platelets data (shown in Figure 3), the algorithm gives us endpoints of 100.0 and 400.0; however, the graph remains relatively unchanged in all other aspects. These graphs can be compared to the previously-shown lymphocytes plot to note the consistency in aspects among them.

The X*Y plots are consistently aesthetic as well as meaningful to users, especially FDA reviewers and in-house medical monitors working with lab data. Further, the program code never has to be modified for a different study or project. Given the proper dataset (see Usage Notes), the algorithm will perform seamlessly every time. Thus, the development time per CANDA is expedited. More concentration is spent on the data-processing aspects of these projects rather than the programming. The detailed outline presented in this paper allows any developer to build a SAS/IML graph algorithm customized to their own need and environment.

Usage Notes

1. The input laboratory dataset to the algorithm should be structured in the form of one patient per observation, containing both the pre- and post-treatment values. Thus, each point in the graph represents a patient. The dataset itself must contain all the patients' measurements for one particular lab test (i.e., one dataset for lymphocytes, one dataset for hemoglobin).

2. The graph area can be increased to provide room for more text by simply subtracting a value (e.g., "interval" in the examples) from the minimum data point and/or adding a value to the maximum data point before using the GWINDOW function.
3. The algorithm should be placed in SAS macro form to facilitate parameterization, thereby providing dynamic capability to the program.

References


Acknowledgements

Thanks to David Bright for direction and insight during the development.

Special thanks to Richard Vachal for conceptual input concerning the initial graph display in SAS/IML.