A Tutorial on SAS/IML® Graphics
Linda W. Binkley, SAS Institute Inc., Cary, NC

ABSTRACT
SAS/IML® software, an interactive matrix language, contains both a powerful set of graphics primitives and the ability to create customized graphics routines using these primitives as building blocks. This paper explains how to

- use SAS/IML graphics commands for drawing primitives such as lines, circles, points, and polygons
- use SAS/IML graphics commands for placing text on the graphics output
- assign attributes such as colors, line styles, fonts, and fill patterns to graphics output
- generate advanced graphs interactively and incrementally using the primitives
- arrange one or more graphs on the display device using viewports
- develop a toolkit of customized graphics routines using SAS/IML modules
- save and manage SAS/IML graphics segments in a SAS catalog.

INTRODUCTION
Why would you use SAS/IML graphics? You may be experimenting with different data relationships and find it inconvenient to exit the IML environment to create intermediate and final plots. The graphics commands operate directly on matrix values, making it convenient for you to graphically explore the results of your data calculations immediately without having to reformat the data into an output data set. You may also have a need to develop custom graphics routines not readily available elsewhere. In IML, you can create such routines and then store them as a toolkit in a SAS catalog for use on different data in the current session or a future SAS/IML session. This paper teaches you the fundamentals of the Version 6 SAS/IML graphics subsystem and provides insight into ways that it might be used.

BASIC COMMANDS
SAS/IML graphs are called graphics segments and are automatically stored in a SAS catalog. Each segment consists of one or more primitives. Primitives are simply points, lines, polygons, or character strings geometrically oriented in a two-dimensional world.

Following is a list of some SAS/IML graphics commands and the primitives that they create:

- GDRAW polyline
- GDRAWL individual lines
- GGGRID grid
- GPICE pie slice
- GPOIN point
- GPOLY polygon
- GSCRIPT multiple text strings at any angle
- GVTEXT vertical text
- GXAXIS, GYAXIS axes

A set of commands is available for managing segments:

- GOPEN begins a new segment.
- GCLOSE closes a segment.
- GDELETE deletes a segment from the catalog.
- GINCLUDE includes an existing segment into the current segment.
- GSHOW displays a segment.

INTRODUCTORY EXAMPLE
For example, suppose you want to plot YEAR*FEMALE for population counts in the United States for the years 1950 through 1985. To do this, you must first initialize the IML graphics subsystem:

```sas
call gstart;
```

Set up the input matrices:

```sas
female = [176, 91, 101, 116, 123];
```

Open the graphics segment under the name POP_PLOT:

```sas
call gopen('POP_PLOT');
```

Define the world coordinate system to be used (more about this later):

```sas
call gwiMOw({19110, 118, 1995, 150});
```

And now plot the points using the GPOINT command, using a dot that is 6 units high (with goption gunit=pct) as the plotting symbol:

```sas
call gpoint(year, female)
symbol='dot' height=6 color='pink';
call gshow;
```

The GSHOW command displays the segment. The resulting graph in Figure 1 contains five points: (1950, 76), (1960, 91), (1970, 104), (1980, 116), and (1985, 123). Note that an entire matrix of values is plotted with a single call to GPOINT since SAS/IML graphics commands operate on matrix arguments.

![Figure 1](image-url)
The graphics segment POP_POINT is still open for input, so you can use the GDRAW command to connect the series of points just plotted:

```plaintext
call gdraw (year, female) color='pink';
```
and add axes using calls to GXAXIS and GYAXIS:

```plaintext
call gxaxis origin=[1950,70] length=40 minor=5 format='4.0' height=6;
call gxaxis origin=[1950,70] length=40 minor=4 format='4.0' height=6;
call gshow;
```
The call to GSHOW displays the graph shown in Figure 2. Note that both axes start at the point (1950,70). The x axis is 40 units (years) long, divided with five tickmarks. The y axis is 60 units (K people) long, divided with seven tickmarks. For both axes, tickmark labels are assigned the SAS format 4.0 and a height of 6.

The GSHOW call displays the graph in Figure 3. Note that GVTEXT draws the label for the y axis vertically, starting at (1941, 120). GTEXT draws the label for the x axis horizontally, starting at (1967, 53). GSET assigns each text string a height of 7 and sets the default font to X-swinx.

Finally, suppose you want to add another set of points for the males. The following code adds such primitives to the POP_POINT segment. It also uses GSCRIPT to add a title and labels for the male and female lines.

```plaintext
/* Male data are in K units: */
male = (75, 88, 99, 110, 116);
call gpoint(year, male) symbol='dot'
    height=6 color='cyan';
call gdraw (year, male) color='cyan';
title = 'Total Population of the U.S. by Sex';
call gscript (1941, 120, title) height=7;
call gscript (1955, 75, 'MALE') height=6;
call gscript (1952, 96, 'FEMALE') height=6;
call gscript (1967, 53, 'MALE') color='pink';
call gclose;
```
The final graph is shown in Figure 4. Note that the call to GCLOSE closes the segment for further input and causes the graph to be saved in a SAS catalog. The default catalog is WORK.GSEG, but you can specify a different catalog in a parameter to GSTART when the graphics subsystem is initialized. Note that SAS/IML graphics catalogs and segments can be shared with other SAS procedures and vice versa.

Total Population of the U.S., by Sex

![Graph of Total Population of the U.S., by Sex](image)

**Figure 4**

**ASSIGNING ATTRIBUTES**

It is simple to assign attributes such as color and line style to graphics output. For individual commands, you can use one or more of the following keyword = arguments:

- **COLOR**: a SAS/GRAPH color name
- **ASPECT**: a real number specifying aspect ratio
- **FONT**: a SAS/GRAPH font name
- **HEIGHT**: a real number specifying character height
- **PATTERN**: a SAS/GRAPH fill pattern
- **STYLE**: an integer specifying a SAS/GRAPH line style
- **SYMBOL**: a SAS/GRAPH plotting symbol.

For example, the command

```sas
call gscript(x, y, title) height=2
font= simplex color='yellow';
```

writes the text string in variable TITLE starting at coordinates (x,y) using a character height of 2. The text is yellow and is written in the Simplex font.

If you want to set attributes throughout the duration of a segment, use the GSET command. For instance,

```sas
call gset('height=2', 2);
```

causes all subsequent output text for the current segment to be displayed using a character height of 2.

**DEFINING WORLD COORDINATES**

SAS/IML software defines a default world coordinate system such that the lower left corner of the display device corresponds to the point (0,0), and the upper right corner corresponds to the point (100, 100). The GWINDOW command is provided to define a new coordinate system corresponding to the data units that are in use in the segment. These units are called the world coordinates, so named because they are the units of measure in the world of the data. GWINDOW defines a rectangular area, in world coordinates, which enables you to conveniently address the graphics device in units of data measurements (years, age, time, miles, and so on) rather than in such device units as cells, pixels, or inches.

For instance, the data in the above example ranged from 1950 to 1985 (years) along the x axis and 75 to 123 (K people) along the y axis. Since GWINDOW arguments take the matrix form

\[
\begin{bmatrix}
MINx, MINy, MAXx, MAXy
\end{bmatrix}
\]

an appropriate GWINDOW call for this data is

```sas
call gwindow([1960, 48, 1995, 150]);
```

Note that the window is enlarged slightly to allow for empty space around the data boundaries. This gives you adequate room for titles, legends, axis labels, and the like.

Alternatively, suppose you want to create a segment that graphs only the data having x values (years) between 1970 and 1985 and y values (K people) between 99 and 123. You could then define the data window to be (1985, 75, 1956, 150). Again, the window is enlarged slightly to allow for space around the data boundaries.

**CREATING A GRAPHICS TOOLKIT**

In SAS/IML software, it is convenient to place repetitive tasks inside subroutine modules. SAS/IML subroutine modules give you the capability to create flexible, highly customized graphics routines, providing you expanded ability to explore data graphically without having to leave the SAS/IML environment. The appendix provides three such user-written modules that create some basic graphs for analyzing population data:

- **BARCHART**: creates a vertical bar chart.
- **PIECHART**: creates a percentage pie chart.
- **GSCENTER**: centers text horizontally about specified coordinates.

These modules are placed in a permanent SAS catalog using the SAS/IML STORE command:

```sas
store module = (barchart piechart gscenter);
```

The following SAS/IML LOAD command brings them back into the current SAS/IML session:

```sas
load module = (barchart piechart gscenter);
```

For example, suppose you want to create two vertical bar charts displaying the age ranges of U.S. adult males and females in the year 1985. Since the modules are already loaded into the workspace, you simply define the input data and invoke the BARCHART module once for each set of data. The following code graphs the female age range:

```sas
col = {R1B_2I11ft, *25-34, *35-54, *55+};
histdata=(111119 20673 21516 21518, 1000);
color = 'pink';
title = 'U.S. Adult Females in 1985';
ylabel = '100K';
call barchart('F_VBAR', histdata, color, title, -S, col, 'AGE RANGE', ylabel, '1.1');
```

Next create a bar chart for the male age range:
hhtdata=(13695 20184 26181 21391) / 10000;
color=cyan;
title = 'U.S. Adult Males in 1985';
call barchart('X.VBAR', hhtdata, color, title, 'X', col, 'AGE RANGE', ylabel, 'X.1');

Two graphics segments are created, F.VBAR and M.VBAR. F.VBAR is shown in Figure 5.

Suppose you now want to create some pie charts that show the percentage of married, widowed, divorced, and single men and women in 1985. Again, define the data, and then invoke the PIECHART module once for each set of data:

patterns = 's';
colors = ['white' 'red' 'cyan' 'yellow'];
title = 'U.S. Adult Females in 1985';
status = 'Single', 'Married', 'Divorced';
female = 18.2 60.4 12.6 8.7;
call piechart(female, 'F.PIE', title, status, patterns, colors);
call piechart(female, 'F.PIE2', title, 'NOLABEL', patterns, colors);

title = 'U.S. Adult Males in 1985';
status = 'Single', 'Married', 'Divorced', 'Widowed';
male = 25.2 65.7 2.6 6.5;
call piechart(male, 'M.PIE', title, status, patterns, colors);
call piechart(male, 'M.PIE2', title, 'NOLABEL', patterns, colors);

Four graphics segments are created: F.PIE and F.PIE2 for the females and M.PIE and M.PIE2 for the males. F.PIE is displayed in Figure 6. F.PIE2 is a shaded version of F.PIE that is unlabeled.

In the next section, you learn how to superimpose the pie chart created in F.PIE2 on the bar chart stored in F.VBAR. This is done using SAS/IML viewports.

Using Viewports

Sometimes you may want to place two or more graphs on the display device. This is easily done through the use of viewports. A viewport is a rectangular area of the device onto which a graphics segment is mapped. Viewport coordinates are defined using the normalized coordinate system. The normalized coordinate system defines points relative to your display device whereby all x and y values vary between 0 and 100. The lower left hand corner of the screen maps to the point (0,0), and the upper right hand corner of the screen maps to the point (100,100).

In SAS/IML software, viewports are defined by the GPORT command using the following argument matrix:

{MINx, MINy, MAXx, MAXy}

For instance, if you want to place the F.PIE segment shown in Figure 6 in the upper right-hand quadrant of the display, use the following GPORT call:

call gport(50 50 100 100);

and then use the GINCLUDE command to place the graph in that viewport:

call ginclude('F.PIE');

Suppose you want to superimpose a pie chart (saved in F.PIE2) on the bar chart (saved in F.VBAR) and shown in Figure 5. The following code creates a new segment, FPBAR, that places the pie chart on the right-hand side of the bar chart:

call gopen('FPBAR');
call ginclude('F.VBAR');
call gport([73 38 100 65]);
call ginclude('F.PIE');
call gclose;
The resulting graph is shown in Figure 7. Of course, the pie chart shown in Figure 7 has no accompanying legend. You can use GPOLY to create a legend:

```plaintext
status = ['Divorced' 'Widowed' 'Married' 'Single'];
patterns = ['a' 'b' 'c' 'd'];
colors = ['yellow' 'cyan' 'red' 'white'];
x1 = -30;
x2 = 0;
call gopen('plegend');
call gwindow((0 0 120 100));
call gscenter("Legend", 60, /l0, 9, 'xswiss', 'white');
do i = 1 to 4;
x1 = x1 + 30;
x2 = x2 + 30;
x = x1 x2 1 x1;
call gpoly(x, [70 70 80 50] i);
   pattern = patterns(i);
   color = colors(i);
call gccenter(status(i), x1 - 55, 7,
   'xswiss', 'white');
end;
call gclose;
```

Suppose you want to create a single graph that contains the following segments:

- **PLEGEND**, the pie chart legend
- **POP_PLOT**, the plot for **YEAR**|**MALE** and **YEAR**|**FEMALE** shown in Figure 4
- **FPIEBAR**, the bar and pie chart for females shown in Figure 7
- **MPIEBAR**, the bar and pie chart for males.

Figure 8 presents such a graph. The segment is created using SAS/IML graphics by stacking and popping viewports, as discussed in the next section.

### STACKING AND POPPING VIEWPORTS

The **GPORTSTK** command stacks viewports; in other words, it defines a new viewport relative to the current viewport so that you have a viewport within a viewport. The stacked viewport is popped (deleted) by a call to **GPORTPOP**.

For example, in Figure 8, note that our **POP_PLOT** segment (previously shown in Figure 4) is located in a viewport in the lower left quadrant of the display. The new viewport, (0 0 100 100), is defined relative to the current (default) viewport, (0 0 100 100), using **GPORTSTK**. **GINCLUDE** places the **POP_PLOT** segment in the new viewport:

```plaintext
call gopen('fig8');
call gportstk((0 0 100 100));
call gwindow((0 0 100 100));
call ginclude('pop_plot');
gPOLY draws a box around the viewport defined by **GPORTSTK**:
   x = [0 100 0 100];
y = [0 0 100 100];
call gpoly(x, y);
color = 'white';
Next, **GPORTPOP** deletes the viewport so that **GDRAWL** can be used to draw projection lines across the current viewport of (0 0 100 100) which spans the entire device:
   call gportpop;
call gdrawl((10 10, 10 40, 10 70),
   (60 10, 60 40, 60 70))
   style = '12 color = 'cyan';
```

Now define a new viewport so that the **MPIEBAR** segment can be placed in the lower right quadrant of the display. Use **GPOLY** again to draw a box around the viewport:

```plaintext
call gportstk((60 0 100 90)));
call ginclude('mpiebar');
call gpoly(x, y);
color = 'cyan';
```

**GPORTPOP** pops the viewport so that a second set of projection lines can be drawn across the entire display area (viewport (0 0 100 100)):

```plaintext
call gportpop;
call gdrawl((30 10, 30 40, 30 70),
   (60 10, 60 40, 60 70))
   style = '33 color = 'pink';
```

Similarly, the **FPIEBAR** segment is placed in the upper right quadrant, and the **PLEGEND** segment is placed in the upper left quadrant of the display:

```plaintext
call gportstk((60 0 100 90));
call ginclude('fpiebar');
call gpoly(x, y);
color = 'pink';
call gportpop;
call gportstk((60 0 100 90));
call ginclude('plegend');
call gportpop;
```

The **GPORTPOP** command above takes you back to the default viewport (0 0 100 100). You now finish the segment by adding some text:

```plaintext
title = 'U.S. POPULATION STUDY';
call gccenter(title, 58, 95, 6,
   'xswiss', 'white');
source = '
   SOURCE: THE WORLD ALMANAC AND BOOK OF FACTS, 1988';
call gscript(1, source) begin=2
   font = 'xswiss' color = 'white';
call gclose;
```

### CONCLUSION

The SAS/IML graphics subsystem enables you to create high-quality, customized presentation graphics. This paper has covered the basics of Version 6 SAS/IML graphics. By using the commands and principles discussed, you can create graphics displays and routines tailored to current and future applications, while remaining in the interactive and powerful SAS/IML programming environment.
U.S. POPULATION STUDY

Legend

Total Population of the U.S., by Sex

U.S. Adult Females in 1985

U.S. Adult Males in 1985

SOURCE: THE WORLD ALMANAC AND BOOK OF FACTS, 1988

Figure 8
APPENDIX

/* BARCHART: Create a vertical bar chart */
call gscript(x-lenIO.S, It str)
nbar, ncol(histdata);
call gl/hrt;
goptions gunit .. pd; " character height

/* Compute xh,yh points for bars. */
call gscenter (xtlabel ,mid, origin( 121 ) _ (wspace 
end;
ypoint = origin( 121) - wspace1t.4;
k = nbar-2;

/* Label axes. */
call gscenter (Xlabel, xlength/2, ypoint, 7 , 'xswiss' , 'wbi te' ) ;
call gyaxis (or igin, ymax-or i'1in[2), 8)
call gxaJ1s{origin,xlength,nbar" 1);
origin., 0
call gset ('color','white');
call gscenter (ti tIe, 50,92,8, 'xswiss' • 'wbite' ) ;
incr = 1;

/* Each bar is 0.6 units wide. */
Create x and y nes.
Compute next midpoint for bar */
currmid = currmidtlncr;

/* Draw the bars. */
SAS, SAS/GRAPH, and SAS/IML are registered trademarks of SAS Institute Inc., Cary, NC, USA.