Several years ago I developed a low-key statistical report series for annual publication which provides a five year moving window on New Zealand juvenile offending statistics. The series presents each year’s statistics on juvenile offending in an accessible format so that those with an interest in them can readily see whether, relatively speaking, the latest figures from the Department of Social Welfare’s operational statistics represent an increase, a decrease, or some other change in pattern over those for previous years.

Quite different pictures emerge when one looks, say, at court appearances for different ages, at appearances by boys, or by girls, or by Maori youngsters as opposed to non-Maori ones. There is interest in change over time as well as in relativities between these groups, so the task of presenting an overall picture in a compact and accessible way is not as straightforward as it might at first seem.

The approach adopted in the series is to present statistics for these groups in a comparative way using a combination of line charts, pie charts and bar charts, and to banish all statistical tables to the back of the volume where the really keen reader can find them if he or she is so inclined.

Until the beginning of 1987 the graphics were all hand drawn. At the time that the series was developed, my division’s access to machine based graphics software was virtually nil. By the time that the basic graphics had been drawn to publication standard and all the labelling satisfactorily typed onto them, hand drawing generally took three to four weeks of full time effort by one staff member for each volume. The graphs ranged from simple line graphs to some more complex four way bar graphs which combined information for different age, sex and ethnicity groupings (see Figure 1).

The more complex graphs were quite tedious to hand draw as some of them have several hundred data points to plot and the plot had to be begun on A3 sized paper then photo-reduced a number of times as different layers of labelling were typed on.

By the time it had been decided to convert the graphics in the report for machine production, the Department had acquired a number of packages offering some form of graphics software. Symphony* and Open Access* were available on PC along with the dedicated PC graphics packages Harvard* Presentation Graphics and Microsoft* Chart. About to be installed on our graphics PCs was Release 6.03 of SAS/GRAPH®.

As a researcher, all the graphics work I had done had been PC based and of one of two types: quick on-screen graphs as an analysis aid, and some simple histograms and pie charts produced in great haste for illustrating unpublished internal reports.

When I listed the desirable characteristics of any computer based solution to the tedium of

---

*Note: The exact software names and versions mentioned might not be correct as they refer to specific historical software that may not be recognized today.
hand drawing the sort of graphics I have described, it was apparent that any solution would have few features in common with this earlier experience. That had all involved 'following my nose' through menu driven procedures and accepting whatever I could get by way of format without having to spend much time learning the package. The characteristics I listed were as follows:

The solution would:
- need to be 'driveable' by someone with a basic level of the 'core skills' usually acquired by staff in the division;
- preferably not require advanced knowledge of a particular package to make it work;
- need to be cheap to run and maintain;
- need to produce graphs with a uniform page size from year to year with little need for any more input than the necessary raw data for the plot;
- need to allow precise control of text size and placement, given the amount of labelling required on some plots;
- and most importantly, need to accommodate existing graph formats so that the established style of presentation of the series would not be disrupted.

It did not take long to conclude that the other PC packages would not provide an easy solution which met the criteria I have outlined. As they are all menu driven, they require at least a basic knowledge of the package's menu structure to make them go. With the exception of Chart, the formats they offer are relatively limited, they are not terribly flexible in the labelling they allow one to place on the plot, and some knowledge of the package is required in order to import the data for each plot from source files.

After browsing the various manuals I concluded that the only way to achieve all of these features would be to directly program drawing of the more complex graphics myself. This raised the question of whether the effort required for someone with my level of experience to do this would outweigh the cost of continuing to hand draw. Some of the 'other' packages incorporated a form of programming language and/or permitted one to write background macro routines for creating special purpose screen menus which could be customised to a particular task, but I had not previously made use of any of the packages in this way. Those who, like me, are intermittent computer users - and focused on end goals which aren't EDP related - will be familiar with the trade-offs that flash before one's eyes when faced with limited time and the prospect of having to substantially expand one's EDP skills if a computer based solution to a particular problem is to be feasible.

One skill I did already possess was a reasonable facility for using SAS[8] as a programming language. A suite of self-documented batch procedures suggested itself as one way of meeting most of the requirements listed above without getting embroiled in learning another language. The majority of the professional staff in the division have some acquaintance with the SAS[8] System, and the facility the System offers for building documentation into the source code for applications would remove the need for a lot of paper documentation.

There were some potential problems with this route. I had heard impressive claims for Release 6.03 at a New Zealand users' conference towards the end of 1986, but I had not used the SAS[8] System in a PC environment before, nor made much use of SAS/GRAPH[8].

I resolved to try out Release 6.03 on one of the simple graphs as an exploration of whether the concept of a suite of PC based batch graphics procedures was feasible in terms of its production cost and ease of use. If it was, I would then try to program one of the more complex combination histograms like Figure 1. If nothing else this would give me an introduction to the capabilities of the SAS[8] System in a PC environment.

**Power and its Price**

The speed with which I was able to develop and document a publication standard line graph using the GPLOT procedure proved encouraging. But what about those combination histograms?

Figure 1 looked as good a place to start as any. It has quite a bit of labelling, some of the label positions are data dependent, various text sizes are used, and four histograms are placed on the page at different origins and orientations.

It did appear possible from a browse of the manual that using an ANNOTATE dataset in combination with some file transformation statements and the GLOSE procedure might provide an answer.

ANNOTATE is a very flexible route by which the SAS[8] system offers the user absolute power over what is drawn on a display area, and where. Unlike politics though, with the power goes real responsibility - responsibility for calculating the coordinates which define the "where". ANNOTATE does take responsibility for providing the statements which make a device perform drawing functions like moving a plotter pen about the plot area, drawing lines, bars, circles, pies and points; positioning and writing labels, drawing frames, and so on. But first, how to
calculate the coordinates at which these functions are to be performed?

If one does the arithmetic for a graphic like Figure 1, it soon becomes apparent that there are rather a lot of points to define if absolute coordinates are to be provided for the location of every action. There are the coordinates which locate:

- five groups of seven bars x 4
- five groups of seven age labels x 4
- five x 2 group labels

We're up to nearly 300 coordinates even before all the points necessary to locate the axes and their labels have been defined.

The solution to all this tedium is provided by the SAS Institute in the form of a documented suite of annotate macros which come with the installation diskettes for Release 6.03. These accept variable values for transformation into coordinates, permit relative positioning, and best of all, operations you wish to perform, regardless of the number of data points to be processed. Those called to produce the example figure were:

%system(5, 5, 5); /*set annotate lengths */
length text $ 16; /*for xsys ysys and huys*/
%system(5, 5, 5); /*set plot window top */
retain ymin1 ymin2 ymax1 ymax2 ysys1 ysys2 xsys1 xsys2;

The two MACROS which are referenced define the length and nature of the variables which will be output to the annotate dataset and declare that the coordinate value system for the job is based on percentages of the plot window. This makes it possible to dictate relative proportions for the different components of the graphic without having to calculate hard coordinates for each area. The package then does the calculation for you.

%MACRO bar( xl, yl, x2, y2, color, bartyp, pattern );
%* NOTE: (Xl,YL) and ( X2,Y2 ) are opposing corners.
%*---------------------------------------------;
%*---------------------------------------------;
%*move ( x1, 6y1 );
X = x2; Y = y2;
STYLE = "6pattern";
IF "color" = "" THEN ;
ELSE color = "color" ;
FUNCTION = "BAR" ;
output;
%MEND bar;

They do look a little forbidding if you haven't used MACRO statements before, but don't be put off. I found that it was a simple matter to copy the fully documented set of them straight into the top of an edit window, delete the ones I didn't need to call, and then get on with the business of referencing them to perform the tasks required.

An ANNOTATE dataset just contains a series of observations each defining a graphics function and a set of coordinates at which the function is to be performed. For the example figure these coordinates were generated within a conventional data step, with DROP and RETAIN statements controlling the fate of all the variables processed to generate the coordinates, but not themselves output to these observations:

data f2cmds (drop= yym1 yym2 yym3 yym4 yymax1 yymax2 yysys1 yysys2 xsys1 xsys2 xbar mrate frate eth year age tic txt yval yval2 yestart1 yestart2 xpos ypos ncolors tpos);
	%colanno; /*set annotate lengths */
	length text $ 16; /*for xsys ysys and huys*/
	%system(5, 5, 5); /*set plot window top */
	retain yym1 yym2 yym3 yym4 yymax1 yymax2 yysys1 yysys2 xsys1 xsys2 ysys1 ysys2 xsys1 xsys2;

The values for the minima and maxima of the various axes represent percentages of the plot area. For example, you can use from the values for XAXMIN and XAXMAX that the X axes will have their origin at 30% of the plot area and extend to 95% of the plot area. Being able to use this percentage based system made it quite simple for me to run a ruler over the handdrawn graphs and get the relative proportions of the various components of the graphs right straight off.

if _n_ = 1 then do;
ncolors=1; yminin=0; mafmin=0; nmaxin=0;
mmaxin=0; mmaxin=1500; mmaxin=1500;

xaxmin=30; xaxmax=66; yymin3=35; yymax4=85;
xaxmin=xaxmin - xaxmin; - 1;
xwidth=round(xaxlen/47,.005);
scale=(yymax4 - yymin3)/(mmaxin + mmaxin) * scale);

If you are starting from scratch to develop a complex graph, then I suggest that you sketch out what you want to achieve beforehand on some

631
graph paper. This would only need to be done accurately enough to fix some proportions — so much for the plot, so much for titles, so much for axis labelling, and so on.

Generating The Axes

Now to generate the statements which will draw the axes (see below). These statements illustrate how you can combine the functions available through the MACROS with conventional looping structures and transformation statements to feed variable values to the MACROS in an iterative way to perform repetitive draw and label tasks.

(These statements draw one of the two sets of axes in the example figure)

\[
\text{when } = 'A'; \\
\text{tpos} = \text{xaxmin} - 0.75; \\
\text{move}(\text{xaxmin}, \text{yymin3}); \\
\text{draw}(\text{tpos}, \text{yymin3}, \text{black}, 0, 0, 1.75, \text{simplex}, 4); \\
\text{label}(\text{tpos}, 1, \text{ymn1}, \text{mafmax}, \text{black}, 0, 0, 1.75, \text{simplex}, 4); \\
\text{move}(\text{xaxmin}, \text{ymn1}); \\
\text{draw}(\text{xaxmin}, \text{ymax4}, \text{black}, 0, 1); \\
\text{max} = \text{mafmax}; \\
\text{min} = \text{mafmax} + 300; \\
\text{do } I = \text{min} \text{ to } \text{max} \text{ by } 300; \\
\text{ymn1} = (300 * \text{scale}); \\
\text{else if } I = 0 \text{ then } \text{tictxt} = 0; \\
\text{else tictxt} = I; \\
\text{move}(\text{xaxmin}, \text{ymn1}); \\
\text{draw}(\text{tpos}, \text{ymn1}, \text{black}, 0, 1); \\
\text{label}(\text{tpos}, 1, \text{ymn1}, \text{tictxt}, \text{black}, 0, 0, 1.75, \text{simplex}, 4); \\
\text{end;}
\]

Logical expressions which transform an existing coordinate are acceptable coordinate statements in themselves, so this gives you a very flexible way of positioning text in relation to specific data points without getting involved in more calculation. For example, the ticklength for the scale on the Y axis is initialized as a variable value above (tpos), then a relative position to this coordinate is used to locate the labels for the tickmarks.

The components of the \text{DRAW} and \text{LABEL} commands (above) are:

\text{DRAW}: (draw from present pen position, to coordinate X/Y, using a black pen, and line type 1, and line width 1).

\text{LABEL}: (at coordinate X/Y, write the value of ticttxt, using a black pen, without angling the characters, or rotating the characters, using a character height of 1.75, using the simplex font, and beginning the text at orientation '4' to the coordinates)

Defining the Graphic

Now, with all the actions that only have to be performed once out of the way, the dataset containing the values to be plotted can be read in and its observations transformed into draw instructions:

\[
\text{set f2data;} \\
\text{if } _\text{N} = 1 \text{ or } _\text{N} = 36 \text{ then } \text{xstart} = \text{xaxmin} + 0.5; \\
\text{if } _\text{N} = 6 \text{ or } _\text{N} = 15 \text{ or } _\text{N} = 22 \text{ or } _\text{N} = 29 \text{ or } _\text{N} = 43 \text{ or } _\text{N} = 50 \text{ or } _\text{N} = 57 \text{ or } _\text{N} = 64 \text{ then } \text{xstart} = \text{xstart} + (\text{xwidth} * 3); \\
\text{xbar} = \text{xstart} + \text{xwidth};
\]

This block of code (above) just deals with the uneven incrementing which is to go on in relation to the X coordinates for the bars so that they will be drawn in the right groups and on the appropriate X axis.

The next block converts each raw data value into appropriate Y axis coordinates for the bar to represent it. The \text{BAR} MACRO which will actually generate all the draw commands for the bars requires two sets of coordinates for each bar which define opposing corners of the bar. Coordinates for labels for groups of bars are also created.

\[
\text{select (eth);} \\
\text{when } (1) \text{ do;} \\
\text{yval1} = \text{yymin4} + (\text{mrate} * \text{scale}); \\
\text{yval2} = \text{yymax1} - (\text{mrate} * \text{scale}); \\
\text{ystart1} = \text{yymin4}; \\
\text{ystart2} = \text{yymax1}; \\
\text{xpoa} = \text{xstart} + (\text{xwidth} / 2); \\
\text{otherwise do;} \\
\text{yval1} = \text{yymin2} + (\text{mrate} * \text{scale}); \\
\text{yval2} = \text{yymax3} - (\text{mrate} * \text{scale}); \\
\text{ystart1} = \text{yymin2}; \\
\text{ystart2} = \text{yymax3}; \\
\text{xpoa} = \text{xbar}; \\
\text{end;} \\
\text{end;} \\
\text{select (age);} \\
\text{when } (1) \text{ do;} \\
\text{xpoa} = \text{xstart} + (\text{xwidth} / 2); \\
\text{otherwise xpoa} = 0; \\
\text{end;} \\
\text{end;} \\
\text{end;}
\]

After all this, the statements below which actually draw the bars themselves and label them are something of an anticlimax.

\[
\text{when } = 'B'; \\
\text{bar}(\text{xstart}, \text{ystart1}, \text{xbar}, \text{yval1}, \text{black}, 0, 1); \\
\text{bar}(\text{xstart}, \text{ystart2}, \text{xbar}, \text{yval2}, \text{black}, 0, 1); \\
\text{bar}(\text{xstart}, \text{ystart3}, \text{xbar}, \text{yval3}, \text{black}, 0, 1); \\
\text{bar}(\text{xstart} + 1, \text{yval1} + 0.5, \text{age}, \text{black}, 0, 0, 1.5, \text{simplex}, 1); \\
\text{bar}(\text{xstart} + 1, \text{yval2} + 0.5, \text{age}, \text{black}, 0, 0, 1.5, \text{simplex}, 1); \\
\text{when } = 'A'; \text{ xstart} = \text{xbar} ;
\]
Group labels are then generated if the data point for a particular iteration is a 'centre' bar for a group:

```plaintext
if xpos > 0 then do;
  %move(xpos, ypos);
  %label(xpos + 2.5, ypos, "Rate For Each", black, 0, 0, 2.0, simplex, 1);
  %label(xpos, ypos, "Age 10-16", black, 0, 0, 1.8, simplex, 5);
  %label(xpos, ypos, "Rates", black, 0, 0, 1.8, simplex, 6);
end;
run;
```

As you can see from the labelling statements in this last block of code, the label macro is just as happy with literal text enclosed in quotes as it is with variable values.

### Titles

All that remains is to take care of the title for the graph and some of the explanatory text for the Y axes. This was done within the GSLIDE procedure using TITLE and NOTE statements:

```plaintext
title ' ' f=simplex c=black m=(10,-1) pct h=.4cm u=2 'FIGURE 2: ' m=(20,+0) u=2 'Rates' ' For Instances Of Coming To Official Notice ' 'By Age, Sex and Ethnicity 1980-1984 ' ' (Tables 5-9).';
note 1&=30 pet j=1 f=simplex h=.30 em Rates 'for' Instances; note 'of coming' MAORI MALE; note 'to official' ; note 'notices for' ; note 'offending' ; note 'per 10,000' ; note 'population' ; note 'for each' ; note 'age 10-16' ; note 1=8 pct j=1 f=simplex h=.30 cm MAORI FEMALE;
```

The computer drawn version of the graph which results from all this (Figure 2) looks like a neater version of the hand drawn one you have already seen.

The basic techniques which I have illustrated were used to draw a number of other four way histograms for the report volume. The PC SAS Editor makes it easy to copy blocks of code from complete and tested job streams and to rapidly edit these to suit a different display, and this cuts down the development time for subsequent graphs. Another of those from the Patterns of Juvenile Offending volume is Figure 3 (over).

![FIGURE 2: Rates For Instances Of Coming To Official Notice By Age, Sex and Ethnicity, 1981-1985 (Tables 5-9).](image-url)
Batch graphics procedures developed using Release 6.03 of SAS/GRAPH[2] did provide a convenient and manageable solution to the graphics task I was faced with. For reasons of space, the layout of code used to illustrate this paper has been considerably compressed in comparison to the layout which appears on the screen when the application is in normal use. There, one code statement appears per line, with each logical block of statements labelled with /* comments */ to reveal their purpose. This not only provides an explanation of what is going on for a user with no experience of the techniques, but also keeps everything 'out front' so that the code for a particular figure can be subsequently serve other users as a convenient template for developing further graphics.

With most of the documentation embedded in the jobstreams themselves, users who have never used SAS/GRAPH[1], but who know enough base SAS[1] to be able to issue an INCLUDE command from the edit window, make simple editing changes and hit the submit key, are able to complete all the graphics for the report to publication standard in a couple of hours instead of the three to four weeks required previously.

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

(1) Trademark Acknowledgements:

SAS and SAS/GRAPH are registered trademarks of SAS Institute Inc., Cary, NC, USA.

HARVARD is a registered trademark of Software Publishing, Mountains View, LA, USA.

MICROSOFT is a registered trademark of Microsoft Corporation, Redmond, WA, USA.

OPEN ACCESS is a registered trademark of Software Products International Inc., San Diego, LA, USA.

SYMPHONY is a registered trademark of Lotus Development Corporation, Cambridge, MA, USA.

Ron Lovell is a Senior Research Officer with the New Zealand Department of Social Welfare. Enquiries concerning this paper may be addressed to him through PO Box 27-015, Wellington, New Zealand. Telephone (Bus) (04) 846 209 or FAX (04) 852 893.