A Quick and User-Friendly Comprehensive Front-End for the SAS/GRAPH® Procedures.

Bob Day, S.A. Medical Research Council.
Retha Keyser, S.A. Medical Research Council.

INTRODUCTION

It is the role of the Computing Centre of the South African Medical Research Council (MRC) to encourage the use of computers (mainframe and PC) wherever they might prove beneficial within the broad spectrum of medical research in southern Africa. One area of scientific computing yet to fulfill its potential appears to be graphics. To quote Donald Greenberg

"If computer graphics is to have a role in improving the future of our civilization, the real value will be in its application to science, engineering, and design."

The influential 1987 report on "Visualization in Scientific Computing" stressed the following points about the potential for computer graphics:

"The gigabit bandwidth of the eye/visual cortex system permits much faster perception of geometric and spatial relationships than any other mode..."

"As a technology, Visualization in Scientific Computing promises radical improvements in the human/computer interface and may make human-in-the-loop problems approachable. As a tool for applying computers to science, it (Visualization in Scientific Computing) offers a way to see the unseen..."

One of our major aims, therefore, is to promote the use of high quality graphics by all our mainframe users, who include academic medical researchers, research technicians, medical bio-statisticians, research students and research administrators. SAS/GRAPH®, which is available in the MRC VM/CMS mainframe environment (IBM 4381), could provide an excellent tool for such graphic output, but it is our experience that it is being sadly underutilised. Less than 50% of our users use SAS®, and particularly PROGRAM screens, since these...}

CONCEPT

When the proposed graphics utility (ESIGSAS) was initially discussed with our users about a year ago, their most prominent specifications were:

1. No end-user SAS® coding
2. Wide range of SAS/GRAPH® output
3. Quality output, quick response
4. Clear, interactive, "PC-like" Menus
5. Flexibility/user-freedom throughout
6. Error checking before SAS submission
7. Retention of structured final SAS code

We, the developers, are academics who have adopted computers because of their proven ability (and further potential) to enhance our research. It was to be expected, therefore, that our design methods would differ from those used by developers with qualifications in Computer Science. Before talking any language considerations into account, we simply applied our general research protocol development methods to this particular problem, i.e. we broke it down into ever smaller components and devised solutions for each of them. We have since been told by suitably qualified people who have seen ESIGSAS that "it has a modular design using top-down functional decomposition". We were relieved to hear that this was a good thing!!! It is certainly reassuring to find that, irrespective of jargon, established and well proven problem solving methods can be applied in new problem domains. The primary functional modules of the proposed ESIGSAS utility are shown in Figure 1.

PROTOTYPING

The utility design emphasized freedom of user selection throughout, with interactive menus being the chosen medium for obtaining such user input. Since SAS Institute promotes SAS/AF® as their speciality tool for interactive screen support, we decided to use it to build a prototype of ESIGSAS. A major hurdle in enabling users with no SAS coding experience to produce SAS/GRAPH® output is the writing and running of the SAS code, in particular the automatic creation of the essential SAS DATASET. We could not easily find a way of doing this using SAS alone, but REXX® (IBM's Restructured EXtended Executor language) proved ideal for this purpose. The automatic creation of a SAS DATASET necessitated the analysis of the parent CMS file, looking in particular for consistent structure and data types. The powerful EXECIO command (almost a language in itself) enables this to be done quickly and transparently. The REXX modules developed work efficiently and should prove equally valuable in other SAS utilities developed for a VM/CMS environment.

The concurrent prototyping in SAS/AF did not progress as smoothly. The problems included the delays involved in editing and debugging MENU and particularly PROGRAM screens, since these...
could not be test run from the PROC BUILD development environment. It became clear that SAS/AF necessitated substantial extra learning (beyond basic SAS), and the lack of an interactive debugging/editing environment made it a difficult and slow process. Added to this were the irritating delays experienced in running the SAS/AF prototype.

The main cause of many of our prototyping problems took some time to be clearly identified. It has been well argued at previous SUGI conferences that the PROC and DATA steps of basic SAS lend themselves well to structured design methods. It was something of a shock, therefore, to discover that our problems were often due to the fact that SAS/AF is not very compatible with structured applications. This was clearly stated by John Rinehart at SUGI'83 who identified the MENU screen (which dominated the ESIGSAS prototype) as the primary culprit. We learned this only after we had found our own solution since the Proceedings did not reach us in Cape Town until the end of 1988.

We decided to investigate how many of these problems could be circumvented by attempting to write an equivalent structured prototype in REXX. We soon discovered that the structure, flexibility, power and interactive debugging capabilities of REXX made it an excellent utility development tool, unquestionably preferable to the current version of SAS/AF. An indicator of the improvement was that the REXX code which produced similar results to the SAS/AF prototype was less than 5% in size! For these reasons the SAS/AF prototype was abandoned, and the full utility was successfully developed in REXX.

A major facet of the REXX utility is its ability to create and alter files of SAS code (using EXECIO) and then submit them whenever the user requested it. However, the initial design dictated that the SAS files were run in batch mode (Figure 2) which was criticised because of the wasted user and cpu time caused by repeatedly entering and exiting the SAS environment.

Figure 2: Batch Mode of REXX/SAS Utility
Ideally, the utility should run within the SAS environment to eliminate these delays, but implementing this proved problematic because SAS does not return control to the REXX utility following the interactive running of SAS code. The solution was to recall the REXX utility at the end of each SAS job with additional pointers so that the user was returned to the point in ESIGSAS from which the SAS code was originally submitted (Figure 3).

This produced a faster and more efficient utility, which otherwise looked identical to the user. But the utility was no longer truly modular, and the ease of maintenance and further development characteristic of the batch mode design was partially lost.

The ESIGSAS UTILITY

Initially, ESIGSAS establishes whether the user is just browsing or has data to plot. If the latter, the data file is analysed and, if necessary (and if the parent data file is sufficiently structured) a SAS DATASET is automatically created.

The first level menu allows the user to select the most appropriate main graphic category for his data (i.e. line, bar, pie, or three dimensional). Examples of these may be viewed if required. The second level menus guide the user through selecting the most appropriate output of the chosen main graphic category, allowing examples of the alternatives available in SAS to be viewed at each stage if required.

Once the desired output has been decided upon, the user is shown the top few lines of his SAS DATASET above a menu from which he selects the variable(s) to be plotted. This interactive menu (Figure 4) may be changed as often as the user wishes, and incorrect variable names are not accepted.

![Figure 4:](image)

A further set of interactive menus follow for the user to choose the necessary parameters for Titles, Legends, Axes and Symbols (Figure 5). Suitable default values are displayed, so the user need not make entries unless he wishes.

![Figure 5:](image)

When the number of the parameter to be changed is entered, the acceptable values available are displayed from which the user should select and enter his choice. A warning results from an incorrect selection, and the user is prompted to try again (Figure 6).

![Figure 6:](image)

The resultant SAS/GRAPH output is then displayed, unless errors occurred. If the errors are of a simple nature (e.g. VPOS or HPOS needing to be enlarged), the utility can correct and resubmit the SAS code without user intervention. If the errors are more complex, however, the relevant SASLOG is displayed in browse mode. The next menu (Figure...
7) gives the user the choice of leaving the utility (and retaining the final SAS code), printing the graphic, working from another data file, or improving the current graphic.

Figure 7:

If further improvements are required, however, (or even a different graph type) then a menu (Figure 8) enables the user to choose to interactively alter all and any aspects of the graphic until he/she is satisfied.

Figure 8:

Once a user has acclimatised to the format of ESIGSAS, the graphic response time can be impressive. If a user selects the default values throughout, ESIGSAS displays his data in the graphic format of his choice in less than a minute!

CONCLUSIONS

ESIGSAS is proving popular with new and experienced SAS users alike. It is introducing academic researchers to SAS, and making them aware that SAS is much more than "just" a powerful statistical package. It saves user and CPU time, and improves the quality and range of the users' graphic output. Its success has (inevitably) resulted in suggested extensions, including:

1. A further level of ESIGSAS (available only after the basic graphic has been successfully displayed) enabling the user, via similar interactive menus, to improve the output with, for example, GOPTIONS, ORDER, and ANNOTATE statements.

2. Similar access to specialised graphic procedures, e.g. GPEARSON and GBLAND.

Although ESIGSAS was developed as a front-end for SAS/GRAPH, many of the REXX modules work independently of the type of SAS PROC steps used. This indicates that we have (unwittingly) developed a REXX 'shell' for further menu driven SAS utilities of any description, or size!

We are not aware of comparable languages in other Operating Systems, but we recommend the use of REXX in the VM/CMS environment and think that SAS/AF must be improved substantially to become a viable competitor for the development of large utilities. Our experiences urge us to recommend strongly that prototypes of any REXX:SAS utilities should be modular, and therefore developed with SAS code submission in BATCH mode. The utility may then be converted to the interactive SAS mode, but only after the batch mode version has been fully debugged.

Questions and comments regarding this article would be welcomed by:

Dr Bob Day
Computing Centre
S.A. Medical Research Council
P.O. Box 70
CAPE 7505
R.S.A.

Phone: 021 932 0311

ACKNOWLEDGEMENTS

This research and development was supported by the South African Medical Research Council, who also generously funded the author's participation in SUGI 14. Thanks are also due to our colleagues in the Computing Centre, without whose help and encouragement the utility would not have been developed.
REFERENCES


