SAS IN A COURSE FOR BEGINNING COMPUTER USERS
Philip N. Gallagher, Jr., University of North Carolina

Introduction

Several years of SAS experience in our university setting have encouraged me to markedly restructure my philosophy of what can and should be taught in a single one semester terminal course aimed primarily at non-computer science graduate students. For the purposes of our course we consider "student" to be synonymous with "presumptive principal investigator". Among my colleagues SAS (which we frequently describe as "the first of the third generation programming languages") is perceived as having begun a major change in the level at which a non-computer buff can expect to make use of the facilities offered by a large computing center. Before SAS the average graduate student in a discipline not directly related to computer science did not expect to and was not capable of doing much more than performing packaged program analyses such as SPSS or BMD provide upon either small card deck datasets or larger tape/disk files prepared in some mysterious manner by programmers. I note parenthetically that the expectations of the professors who taught that generation of graduate students were somewhat higher (but not much): departmental and computer center programmers can almost universally attest that IBM utilities and similar datasets manipulating routines may have been taught but are still a mystery to most non-professionals.

The point of this paper is that SAS has introduced a revolution into the world of the non-programmer scientist who desires to use the computer as intelligently as possible without being forced to obtain, in addition to expertise in his own field, a secondary specialization as an amateur computer scientist. The old folks among us who learned to fight with computers before SAS had two choices:

1. Learn "FORTRAN", IBM Utilities, SPSS, and as many packages as possible (that is, become a Junior Programmer), or,
2. Submit to the Tyranny of the Programmers, by learning only to manipulate standard packages such as SPSS or BMD on small card decks or programmer prepared tape or disk datasets. (Note that it is still possible to teach or learn SAS in only this restricted way.)

Most experienced data-handling scientists have their own horror stories of the terrible frustrations of the scientist who opted for the second choice, and then had to wait, totally helpless, with his deadline racing towards him like a juggernaut, whilst his overworked programmer(s) kept promising a "cleaned up data file" or the specific customized output demanded by a committee chairman.

The preceding argument has presented only the scientist's view and has ignored the frustrations of the programmer who must attempt to deal with data collected by a scientist who has no sense of the mechanisms necessary to get data from original documents into machine readable form, to provide for editing and updating of the data, and file-linkage, to say nothing of preparing an analysis file with the proper structure for the contemplated analysis.

In the remainder of this paper I will outline the topics which we teach and fondly hope will equip our students:

1. to collect their data in a manner suitable for machine aided analysis, 2. to edit their data in a scientifically respectable manner, and
3. to create a clean(?) analysis file, suitable for the statistical analysis appropriate to the data.

(Although it is very disconcerting for students to hear, we do not "teach SAS" or "teach SPSS" in our course, rather we attempt to teach the philosophy of what must be done more than teaching primarily, SAS as a vehicle for teaching that philosophy.) As much as possible, within the bounds of reason and scientific confidentiality, we present to our students raw data just as they come from our clients. Not only do these data have the glint of genuine research, but they also have the virtue of forestalling complaints that the instructor has diabolically constructed impossible problems in order to be unmercifully cruel to his long-suffering students. Besides that, the real data are almost always much more difficult than anything that anyone but a confirmed sadist would hate the glint of genuine research, but
The Instructor's Expectations

We expect that our students want to have sufficient background to be able to start and complete (with a modicum of advice and guidance) their own computer-aided research problems without having to hire a high-powered programmer. (Should our students have enough money/power to hire programmers, we hope that our students will be able to competently direct the efforts of those programmers.) A high level of programming skill is not necessary to achieve this aim, rather a high level of understanding of what steps in data manipulation are necessary. We assume that any competent doctoral or post-doctoral student who knows what task needs to be done as well as the elementary tools of the trade will be able to keep after any particular task until he can master how to do it. That is how professional programmers succeed, so why not our beginners? For example, how long should it take a statistics student already familiar with SAS to learn how to execute PROC STEPWISE?

The student will need to be able to face doing or directing the research data management on a moderately complex project (i.e., no more than roughly 50 form types, a million records, and 2000 variables). This may seem like an enormous undertaking, and it would have been so before SAS76, but now it is at least feasible (if not usually desirable) as doctoral research. In order to even begin such tasks, the user must have at least some hands-on experience with:

1. The design of forms for computer analysis.
2. The design of questions for computer analysis.
3. "Punching" of data - to cards, disk, or tape, including, if possible, OCR.
4. Range and consistency checks.
5. Record linkage.
6. Updates.
8. File documentation.
10. Custom output.

If the user can master the elements of the above tasks, then the programmer may be used for tasks more suitable to his higher-level skills. Clearly the beginning user will need access to a source of good advice for especially complicated problems, especially in matters of research data management, but we intend our course to provide a useful base upon which to implement that advice. I note in passing an observation of long standing: the programmer is only rarely sophisticated in the scientific content of the scientist's discipline and expects to be evaluated on the elegance of his programs, whereas the scientist/user is primarily interested in elegant data, correctly obtained and analyzed. In fortunate situations both elegant programs and elegant data are realized, but most scientists would prefer to sacrifice elegant programs if the price were known beforehand to be long delays while beautiful code is written, producing programs incomprehensible to any but the programmer/author. We hope that the scientist/user who writes (or directs the writing of) his own programs will be able to include the subtleties of his discipline while managing to prevent unjustified expenditures of time on slightly prettier code.

The Fundamental Goals of the Course

Our basic premise from which the course stems is that we want our students to at least be able to:

- organize their research to produce records suitable for computer-aided analysis
- convert their raw data reliably to machine readable form
- edit, backup, document, and display their data
- analyze their data
- prepare an acceptable report.

Because we have so few lecture hours in our course (28), we cannot give separate lectures to each of these topics and must attempt to build them in as illustrations in discussions of more specific subjects. Therefore we regularly seize upon every opportunity to give hints and advice on how and why to do certain tasks:

- How to design forms for keying by keypunch operators rather than a PhD
- How to design records for uniqueness and ease of access
- How to face the interrelated problems of edit checking, record linkage, and updating

Documentation

Backup of raw data, raw records, update files, and final files

Supervision of programmers:

- Insistence on good data now rather than elegant programs much later
- Insistence on good programming style in editing, documentation,
comments, and backup, referring to Brooks (1975) and Kernighan & Plauger (1974) as necessary.

Insisting on realistic time estimates for all tasks contemplated.

Among their assignments, the students are given at least six unlaundered datasets, just as they came from the field; each assignment requires (at increasing levels of difficulty) that the data be read, edited, documented, analyzed, and reported (a standard scientific report, not just raw computer printout and F-statistics). In order for them to be able to do this we must teach:

The mechanics of keypunch and terminal usage (both CALL-OS and TSO)

Job Control Language, including JOB, EXEC, DD, and comment statements, as well as elements of catalogued and instream procedures

File Manipulation/Dataset Management -- that is, a great deal of tape and disk dataset management, which most beginners learn so late in the game.

SAS, including regular usage of good program structure

Computer center facility usage (library & user service)

Other packages (SPSS, BMD, FORMAT) with minimal instruction in order to instill confidence in the students that they have the ability to undertake tasks in programs which they have not studied in a formal course.

Because our course is short we have not been able to devote as much time to the report writing/output dataset generation capabilities in SAS as we would have liked to have done. I have been very pleased to find that many former students have been able to learn this material on their own when the need arose.

Summary

Because SAS is readily understood and mastered by beginning users, a course that is constructed around the data management and report writing capabilities of SAS as well as the statistical analysis procedures can easily be integrated with instruction on tapes, disks, JCL, and keypunch/terminal usage. The non-programmer scientist can now plan to conduct/supervise his entire research without the necessity of spending years becoming an amateur computer scientist, and he can expect that his research design will be improved because he understands the paths that his data must follow in going from the original forms to the final analysis file. It is now possible to teach a satisfactory course for the non-programmer scientist as a user.

References
