EXPERT BASED TRAINING OF SAS USERS: GETTING PRODUCTIVITY FAST

Nelson R. Pardee, Syracuse University

1. Abstract

Many training methods for teaching languages in general (computer languages and SAS in particular) have been tried with varying degrees of success. Because it is desirable that SAS users be productive as soon as possible, choosing the right training method is very important. We accomplish this at the Health Studies Program by giving new SAS users (generally graduate students) a minimal amount of SAS training in classroom, textbook, or CAI fashion, and then giving them a "real" task to do. An expert then works closely with the trainee, providing help which initially may include writing most of the SAS program. Over their tenure, graduate students continue to rely on the expert for help with and approaches to new projects, although in lesser degrees. This method attempts to multiply the expert's skills, and to foster immediate productivity and rapid training for new users.

2. Background

The Health Studies Program does funded research involving analysis of "not small" datasets (up to 1.5 million cases or more). Virtually all of our computer work, including data cleaning, is done in SAS by social science graduate students. On arrival, most of these students have limited data processing expertise, and few have any SAS skills. Tasks routinely given to the students include complex data cleaning, transformation, file matching, merging, and case selection.

Generally, we have about 10 graduate students at any one time, with 3 to 4 new students in any given year. The Academic Computer Center offers excellent short courses, and CAI in SAS, but none are at a sophisticated level. Students often take a short course as an introduction to SAS, or an introduction which is part of a regular course such as statistics, etc.). Otherwise, students begin by reading the SAS Introductory Guide.

3. History

The Health Studies Program did not come into this method of training overnight. In fact, it wasn't planned. Nearly 10 years ago, when the data processing component began, the expert programmer did nearly everything. The reason was simple: the director of the program wanted things done right, done quickly and efficiently. Also, he didn't want tasks left undone because the person doing it didn't know how. It became clear, however, that we had too much work for one, and later two, experts. We needed a way to multiply the knowledge those experts had.

Early, data cleaning and complex file manipulation was done in PL/I, and data analysis was done in SPSS. We began spreading the load by giving the SPSS portions to graduate students. Then SAS became available about three years ago and we found we could do just about anything we needed in SAS. To our surprise, we found that in SAS the graduate students could do most of the programming formerly done in PL/I. So we let them. The only question now was the best way to train them.

4. Characteristics of the Training Method

In retrospect, our training method needed to:

1. Work with users of various backgrounds and skills.
2. Allow learning to proceed at various rates.
3. Foster immediate productivity (within a semester or less).
4. Require minimal class time (difficult to schedule and very time consuming because of teacher/pupil ratio).
5. Make efficient use of expertise available.
6. Provide a basis for continued training.

5. Description of the Training Method

We found that we could achieve all the above objectives by doing the following:

1. Begin with introductory SAS training if possible (short courses, CAI, or an introduction which is part of a regular course such as statistics, etc.). Otherwise, students begin by reading the SAS Introductory Guide.

2. Give the student a "real" task, i.e., a task leading to or generating research results. A faculty researcher wants and will use what the graduate student produces (make work will not do).
3. The student takes the task to a computer expert, either the computer programmer or a skilled graduate student. The expert determines what the student can and cannot do, and does the undoable part in cookbook fashion (this can be as much as 95% early in the student’s tenure with us).

4. Other graduate students, when possible, are formally or informally assigned to the new user as a first line of aid and assistance.

5. The student then attempts to do the task, getting expert help when necessary during the debugging process.

We have been surprised at the complexity of tasks that novices can handle in this fashion. Several reasons make it clear why that is so, however.

1. Generally, the student understands the task and its purpose. They are expert (or are learning to be) in the research task being performed.

2. They are motivated by interest and by supervisor interest to finish the task. Most are developing skills they will apply later when writing their dissertation, so there are perceived future benefits from learning these skills.

3. We limit the amount of learning necessary by teaching nothing superfluous to the task (no one learns JCL except by osmosis; if you don’t need PROC PRINT, you don’t learn it).

4. We foster success when we limit bad advice and dead ends by having expert advice. There is nothing more motivating than success, and nothing more time consuming and frustrating than redoing something or getting wrong advice.

5. We build on past learning. People tend to work in an area of expertise, so after the first task they generally get subsequent tasks based on previous expertise.

If a graduate student stays for two years or more, we find they become very competent SAS programmers. Over time, enough projects are done to round out the persons expertise in and feel for SAS. Typically, the amount of help needed decreases so that the graduate student will need advice less and less as time goes on. We encourage them, however, to “check in” with the expert periodically, just to keep tasks being done as efficiently as possible.

VI. Suitability of This Training Method for SAS

This training method does not work with most traditional programming languages, primarily due to the large amount of up-front learning required (e.g., PL/I, FORTRAN, COBOL). It does work for those languages designed to be simple for beginners, such as BASIC, but capability is not usually available to do complex tasks with powerful built-in features such as SAS.

Following is a short list of characteristics which contribute to this suitability for SAS:

1. Initial simplicity. The new user can do real things with a limited knowledge.

2. A powerful language. This minimizes program size and complexity. A major component of this is the non-procedural character of PROCs.

3. The language fits the task. The tasks we do are generally straightforwardly expressible (at best) or generally understandable (at least) in SAS. The code may be long, but not torturous or opaque.

4. Easy debugging. A combination of factors, including natural modularity of the language. Easy use of PUT’s (for checking data values), and built-in housekeeping contribute to this. This minimizes initial programming errors and makes it easier to remove them. In a traditional language, a simple logic error in a moderately complex program may prove far too difficult for a novice to isolate. With SAS, we have found the novice can isolate, if not always correct, many problems.

VII. Objections to this Training Method

The major objection I can see would state that this method puts a great deal of pressure on the novice, and gets him/her in over his/her head. Another is that it is difficult to know if the results are correct; the novice may not know. The keys here are adequate supervision by the expert and correct research methodology. Adequate supervision will make sure the novice is not in too deep. For some tasks, the novice or recent beginner may serve as little more than a gopher, doing little on their own volition. Nevertheless, they will still be learning. Correct research methodology will have and check hypotheses, know (or be carefully scrutinizing in order to know) the data, and incorporate checks on the work all along the way, regardless of who does the programming.

One shortcoming I often feel in our training is a limited perspective on what SAS really can do. This approach is, particularly for us, very problem-solving oriented. There are large areas of SAS’s capabilities that people never learn because they don’t need it to solve our problems; or they can solve the problem (less efficiently) through known SAS features (it can be argued that the expert should keep close enough track to stop this from happening). From a pragmatic point of view, this probably doesn’t matter; we get the job done. However, as one who believes in the intrinsic value of learning and the value of knowing a little more than one absolutely has to know, I feel some desire to teach a little more broadly; perhaps to equip for that next task that will require greater knowledge than before.
One amusing problem, if you will, is that near the end of the school year, the graduate students get so good that the experts may need to do little programming, per se, at all. They only give advice. We need to remember that the expert needs to occasionally "get his hands dirty," or he will "lose the touch." The expert needs to keep learning.

VIII. The Role of the Expert

For this training method to work, the expert(s) must have several characteristics:

1. Know how to do the kinds of tasks users will request.
2. Understand user requests (not the same as above).
3. Know SAS (and some things about computing in general) in order to select the optimum way(s) of performing tasks.
4. Communicate with users and programmers (here, graduate students) at their individual level.
5. Maintain communication with the expertise at the computer center (or the next level of computer expertise available).
6. Continue to grow in computing skills.

Knowledge is not enough. Communicating knowledge and tasks with a broad spectrum of people is also necessary. An atmosphere conducive to this communication needs to be nurtured (one graduate student commented on her appreciation of an atmosphere in which it was "OK to ask questions").

Some knowledge transfer theory would argue that training should be done by peers, e.g., graduate students train graduate students. Reasons for this include the need to speak the same language; and the need to feel comfortable and not threatened by training from someone outside the peer group. We do this to some extent by having them work together and assigning new students to experienced students. We also do not have formal organizational structures which limit communication between people who are not peers (the "experts" and the "graduate students"). The key is that useful, timely, and expert information to do the task is being disseminated and used.

IX. Benefits and Conclusion

We have found not only a training method for teaching people SAS, but a way of multiplying and disseminating expertise. For us, it didn't come out of educational theory, but out of necessity. Facets of it are used everywhere, everyday; it isn't new with us, but I don't recall seeing it often thought of as a training program. And, it does work for us. Much research and many trained graduate students attest to its success. The key is having an expert — a real expert — who can communicate both with those who need programming done and with the person, often a novice, to do it. The result: a lot of work done, a lot of people trained.

Informally, this kind of training continually occurs in in organizations where people communicate effectively with each other. The challenge is that, once we know how it can work, we plan to provide and use expertise to the greatest advantage, decreasing training time and increasing productivity.

For further information:

Nelson R. Pardee
Health Studies Program
The Maxwell School
Syracuse University
712 Ostrom Avenue
Syracuse, New York 13210
(315) 423-4700