Paper 141-29

Successfully Supporting SAS® at the University: Lessons Learned

David C. Steven, The University of Washington, Seattle, WA

ABSTRACT

Support of research computing at large universities in the United States and around the world presents some uniquely difficult challenges. Such universities normally boast a large and highly skilled central computing support division, tasked with providing in-house solutions for the university community. More and more frequently, the paradigm for user support includes open source solutions that are "free." Supporting SAS software requires political savvy, teaching abilities, communication skills, and a dedication to flexibility and excellence.

PREAMBLE: THE LICENSING DECISION

Support of research computing at large Universities in the United States and around the world presents some uniquely difficult challenges. Such universities normally boast a large and highly skilled central computing support division, tasked with providing in-house solutions for the university community. More and more frequently, the paradigm for user support includes open source solutions that are "free". The development of the Pine email system at the University of Washington, the OpenAFS file system, originating at Carnegie Mellon University, and more recently, the development of the R language for statistical analysis by a web of individuals, mostly comprised of university professors and staff supported by universities and colleges around the world is testament to the power and success of this paradigm.

The challenge presents itself in those cases where a proprietary software system, such as SAS, is the appropriate solution, or the solution specified by certain faculty. At this point, support shifts from an activity that is purely academic and technical to one that involves a budget and interaction with a software vendor. In contrast to supporting the R statistical language, which can be initiated easily and without seeking budget approval by downloading and installing the open source software, support for vendor-supplied software, such as SAS, carries with it budget considerations that require review and approval: Do we purchase the software? How much can we afford? What machines can we afford to install it on?

The first challenge, thus, is to determine the demand for the software and whether a funding source can be identified—or gently persuaded! Nearly all universities in the United States already license SAS software, at least to some degree. Some universities, such as Penn State, manage a university-wide institutional licensing and software distribution program whereby the central computing authority purchases large site licenses for software at a volume discount, and resells portions to the university's various colleges or departments. Other universities, such as the University of Washington, prefer not to broker software licenses from the central authority, choosing instead to encourage various departments to "sponsor" a site license, brokering the sub-licensing at this level. As expected, this is likely to result in topic-specific license sponsoring. For example, the Geography Department sponsors GIS software, and the Statistics Department sponsors statistical software, etc.

Normally, the difficult decision therefore isn't whether to purchase the software. Instead, the question turns quickly to "How much can we afford?" When working through budget figures to answer this question, it's important to pay attention to some key hidden values that go along with a decision to license SAS software.

HIDDEN VALUE: TRAINING

Unlike nearly any other software vendor, SAS Institute provides a wealth of support for academic institutions—to such an extent that, upon close inspection, one ought to conclude that price really isn't a factor. In the years that I've supported SAS software, under two very different academic license programs, for two very different universities, my balance sheet for supporting SAS software has always been in the black. The challenge, however, is in convincing faculty, deans, and provosts of this easy fact. Anyone who's spent very much time working for a university is familiar with the comment, "That's different money." All universities run on money supplied from various funding sources: tuition, private endowments, state supplements, federal research grants, etc., and most of these sources carry with them certain restrictions on how the money can be spent. It's important to be cognizant of the budget process, and it's absolutely imperative to ensure that everything about a software license is taken into account. In certain circles, this is referred to as "total cost of ownership".

Many university researchers may not, on their own, consider much else beyond the purchase price for a software product they require to accomplish their research goals. Researchers rarely are financial analysts or business executives. It is up to the research computing support personnel to provide the researcher with an analysis of needs, relative costs, and benefits. Usually a detailed email with figures totaled and compared is sufficient. No researcher or

technical supervisor in my experience has ever passed-over such an email note. Research-minded people respect analytic reports.

Under nearly any of their academic licensing programs throughout the years, SAS Institute has provided some means for attending training sessions, either at considerably discounted fees, or with the course fee waived altogether. Universities, deans, and faculty all respect education and training—and many grant proposals contain provisions for necessary training to provide research support personnel with the skills they require to accomplish the goals of the research. This discounted or free training from SAS Institute has a value. The market value of each SAS training course is posted as the public or academic discounted fee for the course. It behooves you to ensure that your supervisor or supervising faculty member knows what this value is. At an academic discount price of less than $250 per day in the United States, SAS training is by far one of the most affordable of any computing-related training course offerings available anywhere. Most computer-related training courses of similar or lesser quality or utility approach or exceed $1000 per day—even for academic audiences.

And occasionally, SAS Institute may sweeten the deal. In one case last year, I encountered a graduate student who was faced with a complicated research problem involving mixed models. Unfortunately, this wasn't one of those problems that is solved as a quick fix with a snippet of code that does the trick; it would be an ongoing and evolving process that would require in-depth knowledge of mixed models and the programming solutions to go along with a series of different problems. My approach to assisting this student was to make *her* the expert. Taking advantage of a "customer appreciation" offer SAS had recently extended to academic sites, we sent her to a three-day SAS public training course on mixed models. No charge.

Unless you're located in a city in which SAS maintains a training center, you'll be challenged with travel costs, despite whatever discount you may have to attend a SAS training course. Unfortunately, this is a cost that must be supported by your department. If your department is willing to support travel to a SUGI conference, don't forget to take advantage of SAS training courses offered before and after each conference! Making a presentation on what you learned at SUGI/SAS training often is enough to convince a supervisor to fund your travel. Another way to generate funds for travel to SUGI or SAS training in another city is to broker a large SAS license for your university or college. There's a considerable discount on volume licenses, and it's not unreasonable to budget enough into your license distribution service fee to generate a small amount of overhead to pay your travel and training expenses (More about that later). If you do this, be sure to offer local technical support and/or free SAS training sessions for your university, college, or department! This can be based on what you learned at SUGI or in your training course(s).

HIDDEN VALUE: UNLIMITED TECHNICAL SUPPORT

All SAS licenses come with unlimited technical support. Period. It appears they won't compromise on this one. What this means to you as a licensee of SAS software is that you'll never be left with a problem and nowhere to turn. Most software vendors offer limited technical support and/or an option for technical support for an additional fee. With tight research budgets, this often means that effective technical support is not readily available. Furthermore, many software vendors allow only one or two designees from each university to submit technical support questions. This can be an obstacle to a programmer in an academic department who, more often than not, does not know who the university's designee is before embarking on a series of phone calls to the campus help desk. Although SAS encourages university's to provide in-house technical support when possible—and most universities indeed prefer to provide their own technical support anyway—SAS allows anyone covered by a SAS license to contact technical support directly. Write this down and post it on your monitor:

http://support.sas.com/

In my experience, the quality of responses from SAS technical support is prompt and effective. Personally, I prefer to use the web-based method for submitting technical support questions (Select "Technical Support" and then "Submit a Problem" from the above mentioned web page). They've done a good job of standing-by the bold statement that's been at the top of their problem submission web page for years, "NOTE: Any problem or question submitted via this web form will receive a response within 24 hours or less, with the exception of holidays and weekends." Some folks, however, may prefer direct email or the telephone, both of which are available as well (Select "Technical Support" and then "Explore Services" from the above mentioned web page). I can recall only one instance where a response didn't come within 24 hours (it was 30 hours); it was a complicated question involving AIX, DB2, and SAS/Access®—and the response was a phone call from one of the SAS developers who replicated my exact operating system and software environment and correctly determined that my version of DB2 needed to be updated to perform the task I was attempting. I didn't mind the extra 6 hours so much.

The SAS support web site also includes a wealth of knowledge in the form of links to official FAQs, a large sample SAS code library, access to the full-text of dozens of SAS print manuals in HTML format, and other links. Browse around. In addition to the official treasure trove of technical support and information, many unofficial sources of technical support and information about SAS, SAS programming, and using SAS software products abound on the

Internet. I've collected links to what I consider are the most useful official and unofficial SAS technical support and information sources at:

http://www.dcsteven.com/sas/

Google (http://www.google.com/) is an obvious place to go to search the Internet for an instant solution to a SAS problem; here's a tip: There is no such word as "PROC" in the English language; however, this word often is found in the text of SAS solutions posted on the Web. If you include "PROC" as a keyword in your Google search, this can help to narrow the search results to SAS information.

Perhaps the best unofficial place to find instant suggestions for solving a SAS problem is the SAS-L email list/news group. SAS-L is a place where people post SAS questions and answers. Some of the most respected SAS programmers in the world are frequent contributors to this list. Often, it's not necessary to post a new question—someone may have already asked the same question you're struggling with. In that case, all you need to do is to search the archives of SAS-L for postings that may have already addressed your question. A web form for searching the SAS-L archives is posted at the above mentioned web site.

HIDDEN VALUE: ENTERPRISE-LEVEL SOFTWARE

One mistake that researchers often make is to compare SAS software to "other statistical software." This is perhaps the most difficult myth to overcome. Although my work environment necessitates that I work with a variety of application software products, I personally prefer researchers use SAS wherever a choice is available—for the simple reason that I know I can do whatever I need to accomplish using SAS… somehow. And don't forget the unlimited technical support, the training opportunities, and the one thing that may actually rival SAS technical support: the SAS-L email list (see above). All these solutions are available because the software components included in the SAS Academic Computing Offer are serious enterprise-level products used by Fortune 500 companies for huge mission-critical processes.

SAS files can be read by "other statistical software" products, so this is not a concern. Researchers using ArcGIS and the SAS Bridge for ESRI can work seamlessly with SAS files. Thos who need to work simultaneously with multiple data files can do so using SAS software as easily as using a single data set. And the Academic Computing Offer includes SAS products enabling powerful SQL language, parallel processing, the ability to make use of multi-processor machines, and client-server processing.

For a single researcher, working without the support of a research programmer, any of the popular statistical software products probably is as good as the other. The difference with SAS has to do with the ability of the research programmer to learn and exploit the enterprise-level features it has to offer.

To make a difference, you'll need to unlock these very powerful and sophisticated features of SAS software. The Achilles' heel of SAS Institute is that the real power of their software is dependent upon highly skilled and creative SAS programmers. You are not just a point-and-click automaton, and it's incumbent upon you to educate your supervisors or faculty of this fact. Left alone, researchers and academic supervisors are not likely to understand the difference between the undergraduate computer hack and a professional research programmer. It's your responsibility to deliver the message and to develop the programming that is available with SAS software to take the research computing process to another level.

HIDDEN VALUE: IT DOESN'T HAVE TO COST THAT MUCH!

If you're lucky, your university has a centralized software site-licensing program that offers fair pricing. Many software vendors, including SAS Institute, offer what amounts to a volume discount for large licenses. In the case of the new Academic Computing Offer (ACO) from SAS, the marginal cost of increasing the size of an ACO license drops dramatically with larger licenses. As I understand it, the overhead cost that SAS incurs to administer each academic SAS "site" is considerable, and these costs are included in the licensing fee for each site. Larger licenses do not incur much in the way of additional administrative costs, so the marginal increase in price doesn't need to include much in the way of administrative overhead. Recognizing this, some universities broker a large SAS license: 300 "work units", for example, and then sell it off in smaller chunks to academic departments within the university. The price per work unit for a 300 work unit license that is shared among 12 academic departments is considerably less than the price per work unit for 12 departments each licensing their own 25 work unit license directly from SAS Institute.

If your university does not offer such a program—or if the program that your university offers does not fit your licensing needs, it can be worth the effort to identify other departments on your campus with similar SAS licensing interests. A consortium of independent departments within a university certainly may pitch-in funds to purchase a single SAS site license collectively, and share this larger license at a lower price per work unit. It's been done before, and SAS Institute is familiar with such arrangements. All that's really required is for one department to act as the

contact, and for each of the departments to coordinate how they're going to work-out billing arrangements internally. Your SAS account representative should be able to help with suggestions about how other universities or departments have approached licensing options. Keep the lines of communication open with your SAS account representative. Be sure to let him/her know what your needs and goals are, and be persistent and open-minded in discovering a mutually acceptable solution. I'm pretty sure SAS Institute is just as interested in getting you to use their software as you may be in getting the software licenses you need for your department.

And there's a hidden value within this hidden value: When you collaborate with other departments to share a SAS license, you'll likely discover other SAS programmers in those departments with whom you can share programming problems and ideas.

YOU'RE LICENSING SAS SOFTWARE, NOW WHAT?

Supporting SAS software in a university setting is a challenge. Most folks find it difficult to get started with, as compared to "other statistical software." Researchers are impatient. The illusion is that if it's easy to get started with, then you'll save time. In the long-run, this couldn't be farther than the truth. I can't count the times I've assisted a researcher in starting over from the beginning because s/he took the easiest initial route. Such "bail-outs" invariably involve the highest levels of stress, both for the researcher and the programmer, they occasionally mean that some research objectives must be re-evaluated downward, and usually at a cost of time and resources. What this means to the research computing support professional is that you must provide a continuing education program.

The first defense in this war to assist research at the university is to provide regular training courses. Nearly any department supporting a computing staff will accommodate any number of scheduled research computing seminars. These will most likely be attended by grad students and other support staff. Faculty researchers have little flexibility in their schedules to attend scheduled research computing seminars. Oftentimes, this is because they will not recognize the relative importance of such seminars vis-à-vis their teaching, administrative, and research agendas. However, once researchers approach an impasse with their research computing, they often will find an immediate need for research computing support. Nothing pleases university faculty more than an "on-demand" SAS research computing seminar.

ON-DEMAND SAS RESEARCH COMPUTING SEMINARS AND HOW TO LEVERAGE SAS INSTITUTE OFFERS

Although graduate students and research computing staff often will make arrangements in their calendar to attend a scheduled SAS programming seminar, research faculty usually will prefer that you make arrangements in your calendar to provide a SAS programming seminar that fits their schedule. And this is reasonable. You support the faculty, not the other way around! What this means, is that you will need to develop SAS programming "modules" that you can draw-upon to create an impromptu seminar on demand. Fortunately, SAS Institute will help you with doing this!!

SAS Institute currently provides university educators with training materials culled from their excellent time-tested SAS training courses. This is done free of charge and requires only that you agree to use the materials exclusively for courses or seminars sponsored by your university for enrolled students, faculty, and academic staff. It's not available for external use. Here's another web site you should have in your bookmarks:

http://www.sas.com/highered/

In my experience, students and faculty are much more comfortable using software that is (1) available, and (2) for which training and technical support is readily available. Nothing beats one-on-one consulting, but you'll be more efficient if you can work with several students at a time. Use your imagination to determine what works best in your work environment. Sometimes just asking around about training interests and arranging something informally works well, in other environments, distributing an email survey to determine what SAS seminar topics are needed. One novel approach is to post a list of topics you can teach and what times you're available each week for an impromptu seminar. Once someone requests a seminar topic and time, announce it to the rest of the students, faculty, or staff you support. Accommodate those who you support; don't make them accommodate you.

ENHANCING SAS TRAINING SESSIONS WITH AUTOMATED PROGRAMMING FEEDBACK

A weak point with traditional or impromptu SAS research programming training seminars is the ability to assign homework and provide feedback. I employ a method I originally developed for homework assigned for the Stat 480 course on SAS programming, at Penn State University. This method enables the student to submit a SAS programming exercise via the web, and receive instant detailed feedback via email. The SAS system is unique in enabling this just-in-time training/feedback system/

The traditional model for SAS programming exercises consisted of an exercise for which students would write and run a SAS program plus a set of questions to answer about their output. The SAS program, LOG, and OUTPUT would be printed and handed-in along with the answers to question set. The instructor and/or teaching assistant would grade the exercises and return them to the students at the next lab session.

This method delayed feedback on programming exercises for a number of days, offered and supported only limited interactivity to correct mistakes and receive additional feedback. It also was time and labor intensive, and allowed plenty of opportunity for uneven evaluation among students as well as mistakes (such overlooking a programming error or inefficiency). Surely such an evaluation process could be improved.

Since each exercise came with a specific data set and list of requirements, one may expect that submitted exercises all should contain a number of SAS statements in common, such as an INFILE statement of some sort and etc. Additionally, a correctly completed assignment would generate a LOG that could be expected to contain certain NOTES and/or WARNINGS. Likewise, the OUTPUT would be expected to contain certain values or key words. With so many predictable patterns, one would expect that a monkey could be trained to evaluate programming exercises… or a computer.

The EVALU8 system for Stat 480 programming exercise thus came into existence. Using standard HTML and Perl CGI programming techniques, the course web page was updated to include a secure login/authentication for students to access a homework "hand-in" site. Upon completing a programming exercise, students would save their SAS program to disk and submit the file via this web page to a server with access to SAS. To maintain security, a separate Perl program, scheduled to run every few minutes would check for and process any newly submitted SAS programming exercise files. The Perl program first screens the student's SAS code for security violations (e.g. the use of the SAS "X" command to start a shell on the server). If it passes the security check, EVALU8 proceeds with the evaluation subroutines.

Figure 1. Feedback email message from the EVALU8 program and an excerpt from the assignment.

```
Mon Nov 13 07:39:03 EST 2000

STUDENT:    Jane Student
EXERCISE:   #4
SUBMISSION: #2

MAXIMUM SCORE FOR THIS EXERCISE:     100
MAXIMUM SCORE FOR PROGRAM PORTION:    80
MAXIMUM SCORE FOR QUESTION SHEET:     20

***--> YOUR SCORE FOR THE PROGRAM PORTION:     56 out of 80

Comments:

*      Are you sure you're using your INFILE statements properly?
*      It appears that an INPUT statement isn't quite right... probably either a
missing variable or a missing or incorrectly used '$'.  Check your LOG and pay
attention to NOTEs that tell you what was read-in by the data step.
*      Missing or incorrect result(s) for PROC CORR (Q6).
*      Bad result(s) for L.E. vs. LOG(People per TV) in PROC REG (Q7).  Checklist:
Use a separate PROC REG for each MODEL.  Use only one dependent and one independent
variable in the MODEL statement (We're not doing a multiple regression here).  Make
sure you have dependent_var = independent_var in your MODEL statement.

-Dave Steven

========== THE FOLLOWING CODE WAS SUBMITTED FOR EVALUATION ==========

[Omitted for presentation]
```

Exercise Part 2: Life Expectancy, Doctors, and Television Sets

In this part, we will use the following **space-delimited** data files:

1. demog.dat.
2. tv.dat.

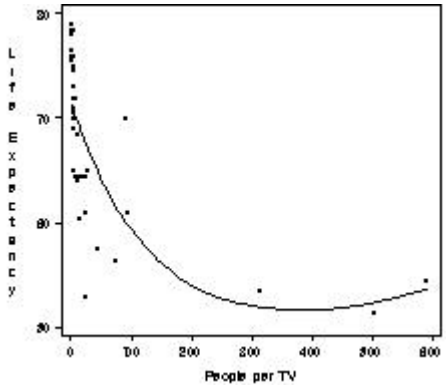
The data contained in these two files are from each of the forty largest countries in the world (according to 1990 population figures). Data are given for the country's life expectancy at birth, number of people per television set, and number of people per physician.
The first data file, demog.dat, contains the following variables:

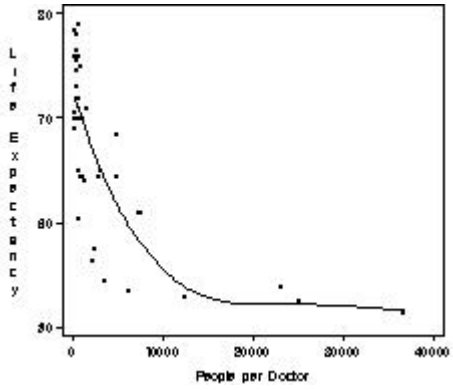
```
country  = Country
le       = Life Expectancy
lefemale = Female Life Expectancy
lemale   = Male Life Expectancy
```

The second data file, tv.dat, contains the following variables:

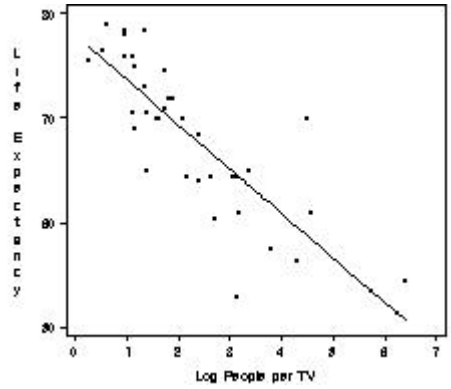
```
country  = Country
pertv    = People per TV
perdoc   = People per Doctor
```

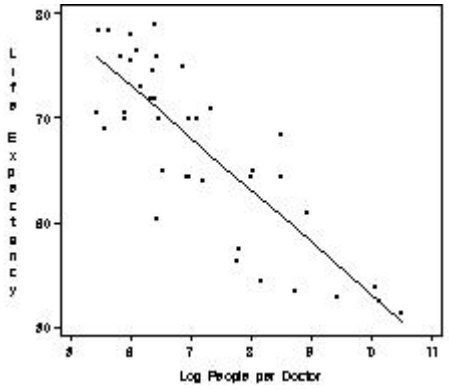
We will use PROC CORR in this part and PROC REG to generate to investigate the relationships between life expectancy vs. number of people per doctor & number of people per TV in these countries. Correlation analysis and the ordinary least squares regression analysis that is performed using PROC REG both assume linear models--they test the *linear* relationship between variables. If we look at plots of the data, however, we see that the realtionship between Life Expectancy vs. Number of People per TV and Life Expectancy vs. Number of People per Doctor both are quite non-linear:





The variables seem to have roughly a *log-linear* relationship. This means that the logarithm of one variable vs. the other would have an approximately linear relationship. We can do what is called a *log transformation* on one of the analysis variables to "linearize" the relationship. A log transformation is merely using the LOG function on one of the variables. If we look at plots of Life Expectancy vs. the LOG(Number of People per TV) and Life Expectancy vs. the LOG(Number of People per Doctor), we can see that these relationships are roughly linear:





Thus, after taking the logarithms, we can examine the linear relationship between life expectancy and the *logarithm* of the other two

Assignment (Part 2):

Write a SAS program to read-in the data. Be sure to LABEL all variables. You should use two data steps to read the data from the two data files. Then SORT the data by the identification variable, "country", and create a third data set to match-MERGE the first two datasets. Then:

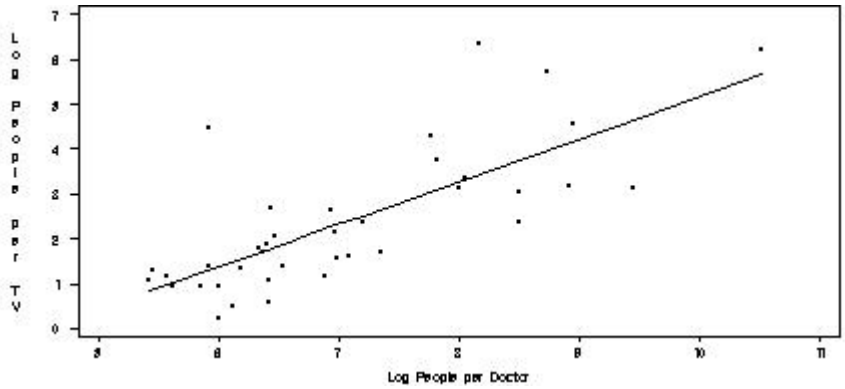
4 In the merged data set, use the LOG function to create two new variables, one for the LOG(Number of of People per TV), and another for LOG(Number of People per Doctor).

5 Use PROC PRINT to print-out the combined dataset.

6 Use PROC CORR with a VAR statement and a WITH statement to generate Pearson's correlation coefficients for (1) Life Expectancy vs. LOG(Number of People per TV), and for (2) Life Expectancy vs. LOG(Number of People per Doctor).

7 Use PROC REG to generate a regression analysis with Life Expectancy as the dependent variable for (1) Life Expectancy vs. LOG(Number of People per TV), and for (2) Life Expectancy vs. LOG(Number of People per Doctor). Make sure you can determine the equations for the regression lines from your output. Note the R-square values and compare them to the plots above.

A few questions to ponder: Does it appear that we might increase life expectancy by initiating a program to distribute TV sets? Or would we be better off developing incentives to increase the number of doctors available to treat people? Do correlations "speak for themselves"? What might the following plot add to your understanding about the relationships between life expectancy, television sets, and doctors?:



The EVALU8 program begins with the highest score for the exercise and works by subtracting from that, with limits set on how much can be subtracted for each type of error. First the student's SAS code is tokenized and parsed for the inclusion of certain requirements, like the existence of a certain number of data steps or procedure names, a "$" in INFILE statements if character variables are expected, etc. Then the student's code is submitted for batch SAS processing, using a copy of the exercise data on the server. The resulting LOG file is parsed for specific NOTES, WARNINGS, and ERROR messages, and the OUTPUT is parsed for specific key words and numerical results. The consistency of SAS LOGs and default OUTPUT formats are key to the success of this method. Once the evaluation process is completed, an email note is returned to the student with a score for the program and comments on any errors or problems identified by the EVALU8 program (Figure 1).

Students may then respond to any comments by fixing their program and resubmitting it for re-evaluation. They're currently allowed to do this up to five times for each exercise. In addition to enabling a higher level of feedback and interactivity, the delay between completing a SAS program and receiving feedback has been all but eliminated, and students may participate in this process at any time of the day or night. The instructor receives a copy of all transactions and may further complement the process by sending email with additional comments. Finally, as a student works at a particular exercise, submitting and re-submitting, the instructor may learn from the process by observing how students respond to the feedback.

CONCLUSION

Supporting SAS at the university can be as simple as installing the software and looking-up answers to problems. However, to leverage the powerful features of the software effectively, you'll need to take a serious interest in budget issues and license fees, training and continuing education, advertising technical support options, and developing SAS code to facilitate the research process. You'll run into obstacles along the way, not the least of which will be in the form of resistance from those to whom you've pledge to support. The fix with the easiest initial steps is not necessarily the best one (Just as correlation does not mean causation!). If you have the patience to take small steps, you'll find that after just a few months, your efforts to build effective SAS support remains cumulative and will strengthen. Start with reducing licensing costs by sharing a license with other departments, and then offer training courses. While doing this, develop your web-based technical support resources. After a few months, you'll find students and staff seemingly coming out of the cracks, asking for more (this *always* happens!). Teach the students and staff, and they will teach their friends, helping to take some of the burden off of your shoulders. Encourage your

constituents to interact, perhaps by starting an in-house SAS Users' Group. If you get this far, you'll be well on your way.

ACKNOWLEDGMENTS

The wonderful experiences that lead to the ideas and revelations in this paper would not have happened without the unique and far-reaching opportunities presented by Dr. James Goodnight while the author was employed by Penn State University's Center for Academic Computing (recently re-organized and renamed to Information Technology Services at Penn State). Genuine concern, undying support, and amazingly creative cooperation and compromise from Sally Muller, Jerry Oglesby, Susan Walsh, and Sondra Nevitt of SAS Institute guaranteed success at every step along the way. A special thanks goes to Kevin Morooney, Senior Director of Academic Services and Emerging Technology at Penn State, for giving me the latitude to explore and expand user services support at Penn State, and to Dr. James Rosenberger, professor and Head of the Statistics Department at Penn State, for his trust and encouragement of my educational experiments in the classroom. My sincere gratitude goes to Dr. Darryl Holman, professor and Director of Computing at the Center for Studies in Demography and Ecology, for improving my quality of life by inviting me to "do it all again" in Seattle at the University of Washington, and for supporting me in that endeavor in every way imaginable.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

David C. Steven
The University of Washington
Box 353412
218D Raitt
Seattle, WA 98195
+1 206 616-6687 (Work)
+1 206 616-6687 (FAX)
dcsteven@dcsteven.com
http://www.dcsteven.com/