Introduction

For more than a decade, the Department of Biostatistics has taught a one semester service course (BIOS 111, Introduction to Statistical Computing and Data Management) designed to teach graduate students in the School of Public Health sufficient computing skills for the completion of their degree requirements. The course has no prerequisites, and is taken primarily by first semester graduate students with no prior exposure to computers. The contents of the course, described in detail below, include: an introduction to computers, using the facilities of the operating system (currently MVS JCL and TSO), an introduction to data processing concepts and techniques (using SAS) and an introduction to statistical computing (again using SAS). A major problem in organizing such a course is the complex network of interrelationships among the concepts to be learned in a short period of time. For example, running the simplest SAS job requires the student to have a basic conceptual understanding of computer hardware, an operating system, and data sets and data representation, as well as SAS itself. The optimal depth of coverage and ordering of these topics is not obvious.

Although the course is popular (70-100 students three semesters a year) and generally well received, we have never been completely satisfied with any of the orders of presentation of the material that we have tried. This paper will present four of the approaches that we have tried over the years. These include: teaching the operating system control programs (JCL, TSO) before, after, and in the middle of teaching SAS; teaching SAS programming and data set manipulation before, after, and in the middle of teaching statistical computing; teaching data input; and teaching statistical computing before, after, and in the middle of the other topics. The successes and failures encountered using these alternatives will be discussed.

The "Logical" Sequence.

Figure 1 lists the topics covered in BIOS 111, in what might be called their logical sequence. That is, this is the order in which the material would be presented in order to cover each topic only once and to have everything fully defined before it is used.

The list begins with a description of computer hardware, and definition of basic terms. Next the concept of binary arithmetic and types of data representation (EBCDIC, floating point, etc.) are covered. Discussion of programming proceeds from machine language through assembler to higher level languages. Next the concept of a data set is presented and organization of data sets on cards, tape and disk are discussed.

In the second section, the Operating System is discussed. After an introduction to its purpose and functions, the two system control programs used at UNC (MVS JCL and TSO) are presented. This section concludes with a discussion of several commonly used utility programs for creating, copying, and deleting data sets, for mapping tapes and disks, and for managing collections of data sets.

The third section covers data management using SAS, and is divided into three subsections. First an introduction to basic SAS concepts is presented (DATA and PROC steps, the structure of a SAS data set, syntax rules, etc.). Second the DATA step is covered completely. This begins with creating SAS data sets from raw data, starting with instream data (using the DATA, INPUT, OUTPUT, RETURN and CARDS statements) followed by data from external files. The DATA step programming language (the assignment, IF/THEN/ELSE, DO/END, and ARRAY statements) is taught next. This is followed by a discussion of transforming an existing SAS data set using the BSET, DROP, KEEP, OUTPUT, DELETE, LENGTH, REShape, and subsetting IF statements. At this point, SAS data libraries and permanently saving SAS data sets are presented. Next, SAS data set management is covered (interleaving, concatenating, merging and updating; the use of IN, END, FIRST, and LAST variables, etc.). This is followed by using the FILE and PUT statements in the DATA step to create external files and custom reports. The third part of this section deals with the PROC step. Utility procedures such as PRINT, CONTENTS, SORT, COPY, and DATASETS are covered first. The semester winds up with the MEANS, UNIVARIATE, SUMMARY, FREQ, CHART, and PLOT procedures for descriptive statistics.

While this order may be "logical," it is a disaster from an educational point of view. The worst (i.e., complex and arbitrary) language, JCL, is presented before the best (i.e., elegant and logical) language, SAS. Within SAS, the worst (most complex) topics—data input and transformations—are presented before the best (most motivating and useful) topics, descriptive statistics and data management. Note that interesting homework cannot be assigned until topic 11, about two-thirds of the way through the semester. Of course, this may not be much of a problem since after the last day to drop a course (near the end of topic 3, JCL) there may be no students left to complain about boring homeworks.

The Expanding Spiral.

The second approach, presented in Figure 2 in what we call the "expanding spiral," Variants
of this approach are probably the most common method of organizing this material. In this approach, we begin with basic computer concepts, as before, but then defer data sets and data representation and teach the minimal JCL necessary to run a SAS job using instream data and WORK SAS data sets. Next we present a minimal set of DATA step statements (DATA, LIST INPUT, CARDS) to let students create a WORK data set from instream cards, with no transformations needed. Next we introduce simple SAS PROCs (PRINT, PLOT, MEANS), with few or no options. Then we switch back and cover data sets and data representation. Next it's back to JCL and the DD statement to define data sets to the operating system. Then back to SAS to cover reading and writing external files (INFILE, FILE, column and formatted INPUT, PUT, INFORMAT, FORMAT) and programming. Next back to the operating system to cover utility programs, and so on, switching around and around among the topics covering each in increasing complexity on each revolution.

This approach has the advantage that each topic starts out simple and can be made to seem consistent. In addition, it allows interesting topics and homeworks to appear earlier. However, it has two major disadvantages that lead us to search for a different approach. First, until the last loop through the spiral, everything you tell the students is a lie (or more charitably the truth but not the whole truth). Indeed is it precisely these half-7 (quarter-7 tenth-7) truths that allow JCL and raw data input in SAS to appear simple and consistent. Such an approach becomes confusing the second and third times students have to generalize rules they are just beginning to assimilate. The second problem lies in the frequent switches among topics. Considerable time is lost reminding students "where we were when last we saw this topic," and more importantly students have difficulty distinguishing among the topics. The confusion engendered by this approach is exemplified by students who try to SET raw data or INPUT a SAS data set. Syntax such as:

```
//DATA ONE;
```

has also been observed.

The JCL Sandwich.

In order to reduce the confusion caused by the expanding spiral approach while retaining the advantages of presenting simpler material first and having the "best" topics earlier, we next tried the organization shown in Figure 3, the JCL sandwich. Here we teach no JCL until after most of SAS has been covered. Standard JCL statements are given to each student with no explanation as to their purpose. After the coverage of basic computer concepts, we teach as much SAS as possible without any mention of JCL or GS data sets. All data is read instream, and no SAS data sets are saved. Then, starting on approximately the first lecture after the last day to drop a course, data sets and representation, JCL, TSO, and operating system utilities are presented in a single block. After that, SAS topics such as reading and writing external files, data set management, and using utility procedures are presented.

This approach does alleviate the confusion of frequent topic switches while keeping the simple material first and interesting topics early, but one class of "lies" is made even worse. Now we are faced with explaining raw data input in SAS without the supporting concepts of data set formats and data representation. We've found it nearly impossible to give any coherent explanation of how a SAS data set differs from cards, what the difference is between INPUT and SET, and what character versus numeric variables are without this background.

Teaching SAS Backwards.

Recently, we came to what now seems an obvious solution to the basic flaw of the JCL sandwich. Since the problem lies in distinguishing SAS from non-SAS data sets, simply defer any mention of non-SAS data sets until the end of the course. Once this occurred to us, other reorganizations became obvious. Start with the most exciting part of the course, creating charts, plots, analyses and reports using SAS, and have the "best" topic.s earlier, we next tried the organization shown in Figure 3, the JCL sandwich. Here we teach no JCL until after most of SAS has been covered. Standard JCL statements are given to each student with no explanation as to their purpose. After the coverage of basic computer concepts, we teach as much SAS as possible without any mention of JCL or GS data sets. All data is read instream, and no SAS data sets are saved. Then, starting on approximately the first lecture after the last day to drop a course, data sets and representation, JCL, TSO, and operating system utilities are presented in a single block. After that, SAS topics such as reading and writing external files, data set management, and using utility procedures are presented.

This approach does alleviate the confusion of frequent topic switches while keeping the simple material first and interesting topics early, but one class of "lies" is made even worse. Now we are faced with explaining raw data input in SAS without the supporting concepts of data set formats and data representation. We've found it nearly impossible to give any coherent explanation of how a SAS data set differs from cards, what the difference is between INPUT and SET, and what character versus numeric variables are without this background.

Teaching SAS Backwards.

Recently, we came to what now seems an obvious solution to the basic flaw of the JCL sandwich. Since the problem lies in distinguishing SAS from non-SAS data sets, simply defer any mention of non-SAS data sets until the end of the course. Once this occurred to us, other reorganizations became obvious. Start with the most exciting part of the course, creating charts, plots, analyses and reports using SAS procedures. In order to do this we create a SAS data library holding SAS data sets to be analysed by the student and a private SAS JCL procedure to make it available by using CLASSLIB as the first level of a SAS data set name. The SAS data sets are created by us; to the students they are logical tables stored "on the computer. The analyses lead naturally (through a little prompting) to the students wanting to transform data sets—recoding, creating derived variables, subsetting observations and so on. The students do this using the SET statement to store data sets in their own individual SAS data libraries, also defined in the cataloged procedure and accessed using a first level SAS data set name of MYLIB. From there, the transition to data set management and programming is easy and natural. By the time data sets, JCL, and so on are introduced the students are confident of their programming skills and learn JCL and TSO without panic or confusion (albeit with a healthy scorn towards JCL). Only at the end of the course do we explain how to create SAS data sets from non-SAS data sets. In a sense, we start students out with computing as it should be (within the protective womb of SAS), slap them in the face with cold cruel reality (JCL), and then discuss the bridge from one to the other.

Summary.

Although we never expect to find the "perfect" organization of this material, we feel that our experiences do support four principals in organizing the material in a course such as this:

1. Avoid frequent switches among topics.
2. Don't assume that the best order for tutorial purposes is the "natural" order—either in terms of definition of
concepts or in terms of typical sequence of learning by experience.

3. Begin with the "best" topics (those that are easiest to learn and most motivating).

4. Motivate the more difficult or awkward topics in terms of end products the students sense a need for.

We feel that giving students a solid base of programming experience in a consistent, high level language like SAS before introducing them to more complex, lower level concepts like physical file formats and JCL increases student motivation and decreases "computer phobia." Saving the best for last is not a good approach to this material.

1. BASIC COMPUTER CONCEPTS
2. DATA SETS AND DATA REPRESENTATION
3. JCL
4. TSO
5. OPERATING SYSTEM UTILITY PROGRAMS
6. BASIC SAS CONCEPTS
7. CREATING SAS DATA SETS FROM INSTREAM DATA
8. READING EXTERNAL FILES
9. SAS PROGRAMMING
10. TRANSFORMING SAS DATA SETS
11. SAS DATA SET MANAGEMENT
12. WRITING EXTERNAL FILES AND CUSTOM REPORTS
13. SAS UTILITY PROCEDURES
14. SAS DESCRIPTIVE PROCEDURES

Figure 1: The "Logical" Sequence

Figure 2: The Expanding Spiral