Seven types of data from computer-controlled experiments

RUSSELL M. CHURCH
Brown University, Providence, Rhode Island 02912

Seven types of data often need to be accessed from a computer-controlled experiment. Some of them involve only a single word and must be available within a fraction of a second; others may involve hundreds of thousands of words, but need not be available until a few hours after an experimental session. A time-sharing program for on-line control of psychological experiments should have the facilities for dealing with all seven types of data.

An on-line computer in an animal laboratory serves two major functions: it administers the experimental procedure, and it records data. Although it is difficult to imagine a psychologist performing an experiment without collecting any data, an investigator often finds himself with inadequate information. Even if the information is recorded, it sometimes becomes available to the investigator too late to be of any value; sometimes the data are available only in summary form, and the original information can no longer be recovered. An on-line computer has the potential of providing data quickly, and it has the potential of storing vast amounts of data for subsequent off-line analysis.

In our applications, we find it useful to have access to seven types of data. Although I will use our applications for illustrative purposes, many investigators undoubtedly have similar requirements.

The computer we are using is a PDP-12/30 that consists of 8K core memory, a clock, two LINCtape units, an oscilloscope, analog channels, and a Teletype. The computer has a I/O bus which is used for most of our connections to the experimental equipment. Inputs include switches (from levers, chains, and doors covering foodwells), photocells, and force transducers. Outputs include light and noise signals, pellet dispensers, sources of electric shock, and motors to extend and retract levers.

Time-sharing is essential for the efficient operation of a computer in our laboratory. At the present time, six independent experiments on aversive behavior of rats are being conducted in 14 experimental chambers by investigators who find it convenient to work on overlapping schedules. Each E starts his experiments at different times, and the sessions are of variable lengths. While the program is running, an investigator is able to monitor the performance of his animals and his equipment, and he can alter the treatment being administered to any of his Ss.

The system program we are using is TIICIE, a timed interpretive system for interactive control of independent experiments written by Howard L. Dyckman. The program and some of its applications in controlling experiments have been described elsewhere (Dyckman & Church, 1972). Here I will describe three kinds of data that the TIICIE program permits us to access (feedback, signals, and monitoring) before proceeding with four additional access needs.

FEEDBACK CONTROL

Some data are required for automatic feedback control of on-line experiments. Such data require very little space, but speed is essential. For example, to control the response rate of a rat at some relatively constant level, the severity of the electric shock may be adjusted as a function of the response rate. The severity may be changed by varying either the duration or intensity of a punishment. The control of duration is quite simple. If the animal's response rate exceeds some criterion, the duration of a punishment is increased slightly; if it is below the criterion, the duration is decreased a small amount. The control of intensity is somewhat more complex, since there is an additional feedback loop via the analog channels to verify that the intended intensity is in fact being delivered. Since this use of data is closely related to considerations of experimental control rather than of data recording, I will not dwell upon it. The point is, however, that some data are needed immediately. (In our normal operation, this means within 50 msec.)

SIGNALS TO THE EXPERIMENTER

The second kind of data consists of signals to alert the E when there is something for him to do. Obviously, the E should be signaled when the experimental session has ended. In our system, the user's program simply has an instruction (SIGEND). When it is encountered, a message is printed on the Teletype: JOB X END, where X is the number of the job which has ended.

In addition, the E may wish to be alerted if there is an equipment malfunction or if the animal is not behaving in its usual manner. The user's program can include signal instructions to print a single Teletype character or a character string.

MONITORING THE SUBJECT AND EQUIPMENT

The signal instructions report situations whenever they occur. In addition, the E may sometimes wish to observe the current performance of a given S or the equipment in a particular experimental chamber. The third kind of data serves this monitoring function. To monitor the inputs from the experimental chamber or the outputs from the E chamber, the user's program has such instructions as TYPEI; V, where V is the ASCII code of a particular letter. Whenever the program encounters this instruction (and the job is being monitored), the appropriate letter is printed on the

Teletype. Other monitoring instructions permit typing of a character string, subroutine calls, transfer of control to a location, acceptance of Teletype input, etc. In practice, we have not made extensive use of the Teletype monitoring facilities because the space available for user's program (1K) is not unlimited and most of the critical information is also available on a panel of lights.

STATUS REPORTS ABOUT THE SYSTEMS PROGRAM

The investigator should be able to get answers to various questions about the programs in operation. For example, he should be able to find out: (a) Which jobs are currently in operation? (b) What is the current setting of some constant? (c) Where on tape will the next data be stored? (d) How much space is available in the buffer containing instructions to execute in the future? (e) How much of the time is the program in the wait loop? (f) How much time has the program been running? Answers to such questions are printed on the Teletype whenever requested by an investigator.

OBSERVATION OF EMERGING RESULTS

The E should be able to browse through the results of an experiment as they are emerging. For example, he should be able to observe the contents of some counter or timer while an experiment is in progress. In the user's program, there may be such instructions as INCR; R 55, which will increment Register 55 whenever encountered. To observe the contents of Register 55 of a given job, say Job 10, the operator simply types J 10 EM 55 and the computer prints a message, e.g., 10 55 233, to show that 233 responses have been made.

SUMMARY DATA AT THE END OF A SESSION

When a particular S's session has ended, the E should be able to obtain the summary data at once. In our system, the summary data are normally available in 64 registers dedicated to a given job. The registers may contain counts, times, or values of measures. If the contents of a large number of registers must be printed at the end of a session, we generally make the request by means of the paper tape reader rather than the Teletype keyboard.

The summarized data in these registers are quite useful, but they are often not sufficient for detailed analysis of an experiment. The major disadvantage is that they are already summarized, and they cannot be decomposed and reanalyzed in some other way. We may have recorded the number of responses during an entire session but now wish to exclude the observations during the first 15 min; we may have recorded a histogram in 2-sec intervals and now wish to examine 1-sec intervals; we may have a record of the number of responses and the number of punishments received but now wish to examine the temporal relationships between them; we may have a record of the mean latency but now wish to look at the median or the geometric mean. All of these decisions must be made in advance of the testing session. Once the original data are crunched, they cannot be recaptured.

DETAILED DATA AT THE END OF AN EXPERIMENTAL DAY

It is far more important to have detailed data at the end of an experimental day than to have summary data at the end of a session. Since there is insufficient space in core for all the information we wish to analyze, we have been putting most of our data on LINCtape. This is done with a double buffer and in no-pause mode, so that neither data nor experimental control is lost when the data are actually being written.

The command in the user's program is simply: DATA. Whenever the program encounters the DATA instruction in any user's program, it puts a four-word entry in a tape buffer. When a tape buffer is filled (or a special instruction is encountered), the information in the tape buffer is stored on LINCtape.

The four-word entries on tape are in a standard form. The first word identifies the S, and the remaining three words are the data. The information can be of any type. Single- or double-precision numbers may be used to represent times, counts, measures of force, shock intensity, etc. Alternatively, bit configurations may be used to represent the occurrence of particular events. There is also a standard end code which includes a S identification.

Of course, you can imagine that the final tape of unsorted data is not yet in a form which a given E would wish to use. A general interactive Assembly-language program has been written to select and order the data from the experimental tape. Normally, an investigator requests that all information pertaining to his Ss be rewritten on his own tape (ordered by S), and this becomes his primary source of data.

The data are then in excellent form for analysis with FOCAL-12 programs. This extension of the standard FOCAL language is excellent for interactive data analysis because it includes instructions to access and create data files on magnetic tape and instructions to control the oscilloscope display. The major problem with FOCAL-12 for our applications is that it deals easily with numerical values, but only awkwardly with bit configurations. Some of our information, e.g., S identification and codes for current conditions, is more easily considered as bit configurations than as numerical values. Therefore, a bit function was added to our version of FOCAL-12 to provide a rapid means for recognizing Ss and conditions. In other respects, the version that we are using is fairly standard: To create more space for the user's program, the extended functions (other than log and exponent) have been omitted and some formatting changes have been made, e.g., omission of spaces between commands and variables. The random number has been replaced by the

one recommended by Siegel et al (1971). FOCAL programs are then written to analyze the sorted data in the primary data files on magnetic tape. The results may be stored in files on magnetic tape, printed on the Teletype, or displayed on the oscilloscope.

REFERENCES

Dyckman, H. L., & Church, R. M. The TIICIE system for interactive control of independent experiments. *Computers in the psychology laboratory*. Vol. II. Maynard, Mass: Digital Equipment Corp., 1972.

Siegel, W., Whittle, K., & Siegel, J. Generating random numbers with Focal. No. 12-61, August 25, 1971, DECUS Program Library.

Use of a notation system for digital control and recording*

ARTHUR G. SNAPPER

Western Michigan University, Kalamazoo, Michigan 49001

This paper describes (1) a notation system for digital control and recording, (2) some improvements to SKED, and (3) future developments and uses of minicomputers.

The main import of this presentation is an appeal to users to develop a standard-process control language that would serve our basic control and recording purposes. The standard language should incorporate provision for adding special-purpose machine language patches for implementing specific requirements, but it should also be as complete as possible. The first section of the paper describes a notation system that could serve as the basis of this language. The second section describes some improvements to SKED, a program for the 4K PDP-8 computer that is based on state notation. And the third section includes a projection of additional tasks a minicomputer could be expected to perform for the research scientist.

THE NOTATIONAL SYSTEM: STATE GRAPHS

The notational language has been designed to characterize any sequential system in terms of inputs, outputs, and states. Although originally developed to aid in the design of computers and other sequential switching networks (Moore, 1956), the state graph notation is easily adapted to completely describe and display the critical features of both behavioral procedures and recording schemes (Snapper, Knapp, & Kushner, 1970). The notation can facilitate communication among scientists concerning the rather complex procedures used in modern psychology. It can be easily demonstrated that the verbal descriptions of reinforcement contingencies typically found in procedure sections of the current literature are often incomplete or imprecise, thus making it difficult, if not impossible, to replicate procedures in exact detail. Although the missing features of the procedures may often be inconsequential, occasionally the descriptions are so incomplete that the main import of the experiment may be impossible to evaluate without extensive correspondence with the author.

To substantiate this rather strong statement, I would like to report the experience of one of my colleagues (Dr. Jack Michael), who recently conducted a seminar on the experiments to be found in a single issue of *The Journal of the Experimental Analysis of Behavior*. He and his students attempted to notate the procedures of each study from the details of the procedure sections. In many cases, he found at least some procedural ambiguities that forced him to guess at the exact details of the study, and in a few cases he was unable to reconstruct even the critical portions of the experiment from the written description.

I would like to make the case that extensive use of the state notation system would help to eliminate communication difficulties of this sort. The notation makes behavioral procedures and recording arrangements explicit. It is complete in what it attempts to do, that is to formalize statements of the rules by which stimuli of an experiment are modified by time, by responses, or by combinations of both events. Any and all behavioral procedures can be described unambiguously in terms of the notation. Furthermore, the formal representation of the contingencies helps to emphasize critical stages of the procedure with a resulting reduction in the probability of omission of crucial procedural details.

The preceding discussion is a prologue to the rest of this presentation. The notation system has a second purpose: it can serve as a computer language for the control and recording of behavioral experiments. This function of the notation system follows from the fact that it is unambiguous and complete. It is clear that computers can be programmed to translate a formal notational language into the binary code that controls the computer in achieving the desired outcome. In this way, the computer can be used to simplify its own programming.

For example, the well-known FORTRAN language is an unambiguous, complete notation for most mathematical operations. Computers are routinely programmed (usually by the manufacturer) to translate FORTRAN notation into the specific binary format used by that computer to perform the mathematical steps of the program. In this way, computer programming is greatly simplified. Those who use the

*This work was supported in part by Research Scientist Development Award K2-MH-70483 from the National Institute of Mental Health.