Behavior Research Methods, Instruments, & Computers
1985, 17(2), 203-205

Electronic surveys

GARY PERLMAN
Wang Institute of Graduate Studies, Tyngsboro, Massachusetts

A programming language and computer system for the design, administration, and distribution of surveys are described. A special-purpose programming language allows the concise definition of a survey. The definition is passed to a program generator that creates a program for presenting questions and gathering answers. Conventions for distribution and collection of surveys are discussed. The system reduces many of the problems associated with conducting a survey.

Surveys are a cost-effective method of data collection. Recent developments and widespread availability of computer networking to researchers and potential respondents make "electronic" surveys even more promising. Compared with more traditional forms of administering surveys, such as mailed paper or phone interview surveys, electronic surveys offer several advantages. (1) Electronic surveys can be constructed by form entry on a terminal or by using a special-purpose programming language. (2) Electronic surveys can be distributed quickly via phone lines or over high-speed networks. (3) Instructions can be provided by an electronic survey system, and it is possible to test respondent understanding of instructions before the survey is begun. (4) Survey data can be collected using form entry or some other mode of computer interaction. (5) Input data can be validated while being collected, and this may result in a larger percentage of correct surveys. (6) Survey data can be returned to the surveyor by the same means that was used for distribution. (7) Finally, data collected using an electronic survey system are already in a medium convenient for data analysis, and error-prone, expensive, and boring hand entry can be avoided.

In this paper, I report on an electronic survey system that makes possible the distribution, presentation, and collection of surveys. Although operating system dependencies make the current implementation most useful on the UNIX operating system, parts should be portable to microcomputers with C language compilers (Kernighan & Ritchie, 1978). The methods discussed are simple enough to be implemented from scratch should researchers find that necessary.

Electronic Mail

This section provides a brief introduction to electronic mail on UNIX. Electronic mail is based on the application of well-established technology to situations in which users are connected either by being on the same time-sharing system or by being on systems connected by some networking capability. High-speed networks are relatively new and still expensive, but it is possible today to have two computers that each cost a few hundred dollars talk to each other; all that is needed is a modem and a phone line.

UNIX, the system I usually use for research, has several mechanisms for electronic mail. The simplest is one in which I want to send mail to someone on the same machine. To mail information to me, people on my machine type

mail perlman

and type in a message. That message is placed in a special file that only I can read, my mail spool file. When I read my mail, I look at that file and I know who sent me messages, and when. To send mail to me from another machine, people would type something like

mail wanginst!perlman.

I work on a machine named *wanginst*, and other machines know how to send mail to it. The peculiar syntax, using the exclamation point, is arbitrary, and different networks use different syntax. Other machines know my machine's phone number and have a modem with some software for sending mail. A more advanced form of electronic mail transfer uses high-speed networks with transmission speeds many times the standard 120 characters per second

The technology for electronic mail is simple. Modems are necessary for cheap intermachine communication. File transfers are necessary for intramachine transfers. Programs that watch for incoming mail to inform recipients, *daemons* in the UNIX vernacular, are also simple, because they depend only on alarm clocks, which are built in to most systems. The main requirement is a willingness to expend CPU cycles and money for sending, receiving, and sorting mail, bulletin boards, mass mailing lists, and so on.

The author's mailing address is: School of Information Technology, Wang Institute of Graduate Studies, Tyngsboro, MA 01879.

AN ELECTRONIC SURVEY ADMINISTRATION SYSTEM

In following sections, I provide details about how electronic survey systems help in various stages of administering a survey: (1) construction, (2) distribution, (3) presentation, and (4) collection.

Construction

Computers can aid survey construction. A special-purpose survey-question description language that allows the concise and precise description of a survey is described below. The language has a simple grammer: Each question type has a variable to be set, a short prompt (question) to be presented to a person, and some more detailed help, if needed. This is shown abstractly as

question = variable + prompt + help

Several types of questions commonly seen in surveys are recognized in the question language, some drawn from Lindgren (1973). All have the parameters listed above, and some have additional parameters, specific to the question type, that can be specified to override default values. Additional parameters are listed after each question type. In the question language they are specified in a "name=value" format to indicate that they are specific to a particular question. The question types and specific parameters (in parentheses), followed by examples, are listed below.

(1) **Thurstone (min, max).**: Rate on a scale of *min* to *max*. . . .

(2) **Likert (left, center, right, modifier).**:
Do you agree with the statement. . . . .
modifier left center modifier right

(3) **Semdiff (left, right).**: Are you more *left* or *right*.

(4) **True/False.**: True or false. . . .

(5) **Multi (choices, . . .).**: Choose one of the following values.

(6) **Value (range).**: Supply a value that is within the specified *range*.

(7) **Question.**: These are questions with no expected format for answers.

Table 1 contains some questions using the question language. Each question description begins with the type of question expected, followed by the variable name and prompt. In the examples, parameter-name=value pairs indicate that no extra help is provided for any of the questions. Table 2 shows the approximate output format of the questions.

Table 1
Survey Description of Some Questions

```
Thurstone(happiness,How happy are you?,max=5)
Likert(support,How much do you support our candidate?,
    left=support,right=against,modifier=strongly)
Semdiff(outlook,Rate your outlook,left=gloomy,right=rosy)
Value(age,How old are you?,value > 10 AND value < 100)
Multi(answer,Choose the best answer,10, 20, 30, NOTA)
Question(name,What is your name?)
```

Table 2
Formatted Questions Based on Table 1

Rate on a scale of 1 to 5:

How happy are you? __

How much do you support our candidate?

strongly against		undecided		strongly support
−2	−1	0	1	2

Rate your outlook

gloomy 1 2 3 4 5 6 7 rosy

How old are you? __

Choose the best answer

1. 10
2. 20
3. 30
4. None of the Above

What is your name? _____

The type-specific parameters can be set for more than one question at a time. Parameters supplied at the end of a question description are effective only for the duration of that question. It is common to have several questions in a row that repeat formats. The PAR function sets global default values for any parameters. For example, to specify that the minimum and maximum values for a Thurstone scale are to be 0 and 5, respectively, the expression

PAR(min=0,max=5)

would affect all subsequent questions using those parameters.

Presentation

Based on the description of the survey, a C program is constructed to present the questions and collect responses. The C program is a series of calls to functions that handle the question formatting and response collection. Most functions present one type of question: There is one for Thurstone scales, one for Likert scales, and so on. Each makes use of a general function for present-

ing questions, providing help, and collecting responses.

Distribution and Collection

To be concrete, assume I am interested in finding out whether people find this survey system useful. I make up a series of questions and put them into a file called USEFILE. I create another file, say, ADMIN, with information needed to administer the survey. This file contains: (1) the *name* of the survey, (2) a *distribution list* for the survey (in this case, a list of machine and user names), (3) the *purpose* of the survey, used to introduce it to people, (4) an electronic mail *notice* to give more information about the survey, (5) *instructions* for people completing the survey, and (6) a *return address* for the survey.

On UNIX systems, it is possible to have some operation done at the same time every day. To prepare a machine for survey administration, I have to log on and tell it to check every day a special directory where surveys will arrive. Suppose the directory name is SURVEY and that it is in a publicly accessible place.

The survey distribution system takes my two files, USEFILE and ADMIN, and goes through several steps. (1) USEFILE is passed through a C language code generator written with the m4 macro preprocessing language (Kernighan & Ritchie, 1981). (2) The generated program makes use of a library of functions for presenting survey questions using standard CRT terminals. (3) A command script that presents instructions, administers the survey, and returns the results is constructed; it is given the name of the survey. (4) The command script and generated program are copied out to all machines on the distribution list. (5) Mail messages are sent to potential respondents telling them the name and purpose of the survey and how to participate.

DESIGN AND IMPLEMENTATION

The survey description language is the most interesting part of the system, and is a useful way of thinking about surveys. The language was designed such that all the question types have some overlap of function; this is necessarily true because, if there was no overlap, there would be no reason to have a language. Then, for each question type, special parameters particular to each were identifed. Parameter setting was arranged to be as easy as possible, with default values set to the most likely values; facilities were provided to reset these defaults. Global parameters to specify external characteristics, such as the terminal screen size, were also given modifiable default values.

For each type of question, a C function was created to handle its parameters and display format. These functions made calls to low-level functions that corresponded to the parts of questions common to all question types: variables, prompts, and help. This helped ensure standard interaction between the electronic survey and the respondent.

CONCLUDING REMARKS

Availability

The software described in this paper is in the public domain and is available from the author. People with access to a C compiler for microcomputers or any non-UNIX system should be able to use many of the library functions. There is a library function for every part of the survey-question description language. Experienced C progammers may want to bypass the macro language and implement surveys directly, but the high-level language exists so that people can avoid programming in C and concentrate on their research.

Enhancements

Several enhancements are possible, and most of the following are likely. (1) Printed paper forms might replace the network distribution system for researchers and participants who do not have access to such facilities. (2) An electronic survey to create electronic surveys should exist, but does not yet. (3) Reaction times are easy to obtain with computer-controlled surveys, and timing facilities are certain to be incorporated in future versions. (4) Templates to allow redefining the format of questions would be useful. (5) Often, the relevance of sections of a survey depends on the answer to a key question. Facilities should be added to make the presentation of a question depend on some criterion so that each question would have a precondition as well as a postcondition.

Limitations

Experience. There has not been much experience with the electronic survey system, because of its recent development. My hope is that with use and feedback the system will evolve to meet many needs of the research community.

Resources. The usefulness of the system becomes limited when some of the parts are missing. (1) Respondents need to have access to a computer or printed versions of surveys, and hand entry of data would have to be used. (2) If networking capabilities are missing, the surveys will have to be set up on each machine by hand. (3) If the m4 macro preprocessor is not available, the surveys will have to be written in C.

Confidentiality. With many time-sharing computer systems, there is a question of confidentiality of respondents. With electronic surveys, it is possible to obtain the user name, time of log-in, and machine name. Any one of these data can be used to identify respondents. One solution is to provide guest respondent log-ins on such systems.

REFERENCES

KERNIGHAN, B. W., & RITCHIE, D. M. (1978). *The C programming language*. Englewood Cliffs, NJ: Prentice-Hall.

KERNIGHAN, B. W., & RITCHIE, D. M. (1981). *The m4 macro preprocessor* (Technical Memorandum). Murray Hill, NJ: Bell Laboratories.

LINDGREN, H. C. (1973). *An introduction to social psychology* (3rd ed.). New York: Wiley.