A perspective on social science data management

Stone, Philip K.

Philip J. Stone

A Perspective on Social Science Data Management

This essay makes the following arguments:

1. The available social science „software packages" only service two of a dozen data formats needed for social science research; the lack of appropriate software has inhibited social science development.
2. The large scale efforts on „database management systems" (DBMS) including the work of the CODASYL Programming Language Committee, are only marginally relevant and do not address the bulk of social science needs.
3. The social sciences would be greatly aided by an effort to delineate their needs more clearly and spearhead appropriate developments.

Current software. Most social scientists think of statistical software in terms of the statistical tests and processing options offered. Does a package offer repeated measures analysis of variance? What are its missing data options? Questions like these are usually uppermost in a reader's mind when scanning a new manual for a software package.

Another issue, however, revolves around what kinds of data formats the statistical package can process. While most statistical packages offer procedures for adding a new statistical test (by calling a user-written FORTRAN routine or the like), they are quite unable to accommodate new kinds of data structures.

Veterans of social science data processing will immediately associate the „data format" problem with the myriad of ways that numbers tended to be represented in early computers. The habit of punching multiple responses in one card column for counter-sorter tabulations led to the necessity of binary column options in early software. Such software also had to accommodate to different kinds of keypunch and paper tape formats, with basic incompatibilities between the magnetic tape formats of different manufacturers. As the representation of numbers became more standardized, new problems arose in the standardization of procedure for labeling data storage volumes (including disks and tapes) and labeling the files stored on those volumes.

The „data format" problems considered here assume that issues of how the numbers are represented and how the files are labeled are solved (even though such standards seem to be frustratingly slow in coming). It further assumes that each data file can be described by a „codebook file" that tells the user what each piece

of information stands for and tells the statistical package how it is stored in the file. The user can thus refer to each variable by name. If the data base manager decides to change the form in which the information is represented (e. g. a switch from decimal to binary), the change can simply be entered in the codebook file and remain invisible to the user. (However, the social science user is usually not as „independent" from data specifications as data base specialists might suggest. For example, the social scientist needs to know the precision with which quantitative information is stored in planning statistical analyses.)

The „data format" issues treated here instead focus on the fact that social science data need not come in a neat organization that can be efficiently represented by a „rectangular array". A „rectangular array" is one with a fixed number of rows and columns. „Efficient use" means that many of the cells are not empty or repeating information elsewhere in the array.

Existing social science software services two kinds of rectangular data representation. In one format („cross-sectional") the rows are respondents (such as in a survey research questionnaire) and the columns are the responses to the questions asked. Both the rows and columns are unordered. The respondents may be in the order in which they were surveyed, but this has no bearing on the statistical analysis. In one survey, the respondent's age may be in columns 22–23 and a code for the birthplace in columns 49–50 while in another survey these two variables may be stored elsewhere. The codebook file may directly inform the software that respondents' ages are recorded in columns 22–23, thus making the location no longer a user concern.

A second kind of social science rectangular format, associated with econometrics, has variables represented in different rows and points of time represented by the columns. The variables are unordered, like the variables in the previous format. However, the columns are ordered, with successive columns representing successive years, quarterly periods, months, or whatever is the unit of time. A variety of time series procedures can be carried out on the ordered dimension; for example, one time period measure can be lagged against another. If data for a cell is missing, one may interpolate an estimate from the adjacent time period information.

These two kinds of rectangular formats have other differences. Often the survey rectangular array may be much larger than an econometrics rectangular array. A cross-sectional study may have 500 pieces of information on 20 000 people, or an array of 10 000 000 cells. An econometric data base may have 500 measures on 200 time periods, or 100 000 cells. The difference is more a matter of scale than logic, but it has logical implications for what kinds of data storage strategies are appropriate. When the National Bureau of Economics Research adapted their time series package (called TROLL) to cross-sectional data, it turned out that the cross-sectional facility, while offering attractive processing features, was not suitable for large files. (Conversely, SPSS, the most used statistical package for cross-sectional formats, now includes a „lag" option, but this is quite primitive compared to the intensive analytic offerings of econometric packages.)

There are two major ways in which the rectangular arrays are processed. One is

to pass all the data through the computer in order to compute the answers. All the information in one row (that is, all the information for one subject) is read in and the information in the columns needed for the analysis is selected out for statistical computations. This process is the usual strategy used by Data-Text, SPSS, OSIRIS and other well known cross-sectional format packages. The second strategy is to have the user declare what variables are going to be used beforehand and select them out for storage in computer memory or a high-speed access device. Thus, in any statistical analysis, only the subset of variables specified beforehand are passed through the computer. For cross-sectional formats, this is the strategy used by IMPRESS, the Dartmouth interactive system. A survey may have 500 variables, but the user on a particular investigation may only have need for 10 of these for an analysis. The prior selection thus results in a 50:1 reduction in the amount of information passed through the computer. This second strategy is used by most packages for analysis of time series formats. An econometric data base may have hundreds of variables stored in it, but an analysis may only require some five or ten of them at one time. By making a prior declaration, the computer brings the variables from background storage up to foreground, for repeated examination in the analysis.

Other data formats. Having said that the social science software packages service two major formats, we will now propose that the social sciences need to have facilities for considering ten other kinds of format. These additional ten formats are only for non-textual data. If one wants to include formats for textual data (a topic dear to this writer's heart) the number of formats greatly expands.

In 1975, the *SPSS Newsletter* asked readers to tell what kinds of data applications they had for which SPSS was inadequate. A couple of hundred responses were received and the folks at SPSS Inc. kindly provided a xerox of the replies. What were the major missing capabilities perceived by the users?

By far, the most cited one was the inability to handle hierarchical files and the most cited application of this was to census information. If a file, for example, is organized by region, town, tract, street, block, building, family unit, and persons, it is extremely wasteful to repeat all of the above information for each person. A hierarchy of levels means that all of the information at a higher lever is applicable to each of the units at lower levels until the next time information at the higher level is changed.

A hierarchical file can be organized in several ways. If all the analyses are to be at the lowest level (e. g. the person), then one file may be created, with each record identified as to its level. The package would use the higher information to fill out a profile and process a unit each time a record is encountered at the lowest level. The profile for a person would be the person record, plus all the current higher levels previously encountered. If, however, many of the analyses were to be at the level of the family, or the level of the housing structure, then it may be efficient to have the higher level information on separate files. A housing study can then be processed

without going through all the information about persons. Each record needs a marker indicating when it is the last record at that level for a unit. Thus, the record for the last family member may have a marker indicating the next family record should be read while the last family in a building may have marker indicating that the next building record should be read, etc. Although it is not difficult to prepare procedures to read such hierarchical information, the lack of standardization has made social science software suppliers hesitant to invade this domain. The U.S. census bureau has such hierarchical procedures, but they are not common to most cross-sectional software packages, except in some very rudimentary ordering of major and minor files.

A second common inadequacy (although much less cited in the SPSS survey than the first) concerns situations where there is a string of data. For example, a multiple response to a question such as „what do you like about the current presidential candidate" may produce a few replies from some respondents and many replies from others. Most existing packages allow the respondent to set up a fixed number of columns (corresponding to the *most* replies received) and to say a response occurs if it is coded for any of these columns. Often this is done by an „or" specification in which the user says that the response exists if it is coded in columns X or Y or Z, etc. Some packages such as CROSSTABS allow for a string option where the response is recorded if a code occurs in string S, where string S has previously been defined as a list of columns. The string option thus saves repetition of all of the „or" specifications.

There are two problems with this rudimentary string capability. One is that the storage is wasteful if the short strings representing the few replies of most subjects have to be dummied out until they are the length of the longest response string. If numerous multiple response questions appear in any one survey, then the normalization of the data into a rectangular array may involve numerous blank cells.

The second problem is that the packages do not provide ways of handling order within the string. Let us say that the string is a job history, with each unit in the string one of a series of jobs, and the jobs are ordered in terms of recenness. A question regarding how many people held such and such a job before they held another kind of job is difficult to search with most packages. People responding to the SPSS Newsletter cited a variety of such history files, including school history, police record history, credit record history, medical history, geographical and housing history (one record for each move made), etc. In each case, the data is ordered by time, but there are a varying number of units per respondent.

In contrast to our neat rectangular array, we call this kind of data „ragged". Unless we normalize with blanks into a large rectangle, there will be a varying amount of data for each subject. Each subject may have one or more strings associated with the record and each string may be of varying length, producing a varying length record overall.

Occasionally, there is a situation where there is but one long string for each respondent, but it dominates the record. One example is the time-budget study, where each record consists of some basic information about the respondent's age,

sex, family status, work status, education, housing, etc., followed by a string where each unit is an „event" and the events cover a span of an hour, a day, a week, or whatever. In this case, one may wish to consider this a form of hierarchy in which the unit is the event rather than the person, and the basic information about each person is higher order information. The ordering from one event to the next can be examined by a „lag" feature from one unit record to the next. A variety of other chronicles also take this form, such as that of a small group meeting.

If the bottom unit in a hierarchy is an event, then the event may take several different forms, with data for different events being different sizes. For example, if the next thing a person does is watch television, we may want to know what kind of program is watched, and who watched the program with the respondent, while if the next event is a telephone conversation, we may want to know other kinds of information. Each bottom unit could then be coded to inform the package which kind of end unit it is.

In other instances, investigators have preferred to score life histories so that medical history is separate from educational history, yet there should be ties so one is able to piece out time overlaps and time sequences. The separate histories simplify applications for secondary analysis, where an investigator interested in medical history need not pass through educational information unless it is wanted. In these instances, software is totally inadequate and life history projects in United States, Norway, and Poland, among others, have had to struggle with their own software. It might be thought that the coordination of different kinds of history strings for one respondent might be especially related to current work in data base management systems.

Partial orderings may be partitioned in larger „chunks". In a life history, we have a number of events with at least several kinds of information (what, when, where, with whom) for each event. In other kinds of research, we have fewer „events", with more information about each one. A panel survey, in which the same people are interviewed at different points in time, can have each survey regarded as one event. In experimental research, data from each subject may be gathered at different time periods in the course of the experiment, with each repeated measure analogous to an „event".

A variety of further compounds, however, may occur. For example, the Institute for Social Research at The University of Michigan has recently employed a panel design to gather time budgets for the four seasons of the year. Each survey includes both the husband and wife in the same family. Thus, the data consists of eight varying length strings (the time budgets), each paired (husband end wife on the same day) at one sample day for each of the four seasons of the year. One can imagine analyses made of this data set that would involve searching across strings of couples at different times of the year for comparisons.

Another form of chunking is where groupings are made but the groupings themselves are unordered. A cross-national comparative design may have the subjects grouped by country. In experimental work, subjects may be grouped by treatment. Grouping of subjects may be combined with ordering of variables. A cross-national

study of time budgets, for example, may group subjects by country. An experimental design may have repeated measures for each subject, with subjects grouped by treatment. Such groupings may be considered a rudimentary form of hierarchy, with the subjects in each group implicitly sharing common characteristics, even if it is no more than a code indicating treatment type.

At this point, we have expanded our perspective from:

current software:

a) rectangular, unordered both dimensions,
b) rectangular, ordered on one dimension, unordered on other,

to various combinations of

needed software:

a) rectangular
b) ragged
c) ungrouped
d) grouped
e) grouped hierarchy
f) unordered
g) ordered on one dimension
h) partially ordered (partitioned chunks or embedded strings)

Order versus ordering. In most data analysis applications, any order on the data is from time sequences. Another aspect of order is the rearranging of data so as to reveal patterns.

A mechanical ordering procedure was once used to produce Guttman scalings. As described in *The American Soldier*, a box was created with slats, in which each slat represented the responses of the subject, and the columns were arranged so as to give first priority to the item getting the most yeas, the next most yeas, etc. The slats were then rearranged so as to group most of the yea marks on one corner of the rectangle.

Modern computer techniques can be used to rearrange both whole rows and whole columns at a time so as to make order apparent where there was little to be seen. If all responses are either zero or one, the ordering can be either to maximize areas of zeros or maximize areas of ones. H. White, R. Breiger and associates have developed techniques for rearranging matrices so as to group clusters. Suppose the rows are persons citing and the columns are persons cited. Initially, they may be unordered. If they are rearranged to form clusters, we may find (as, for example, Breiger does for journal citations) a group that cites each other, a set of asymetrical intersections in which the group cited does not reciprocate, and a set of intersections where people do not cite each other.

Thus, we should distinguish data sets having order over time from the mechanical ordering process to uncover underlying relationships. The ordering process links file handling to measurement theory. One advantage of the mechanical ordering is that the unexpected case, that is the case that is the case that appears where not expected once ordering takes place, is readily salient.

Graphs and list structures. If a group is relatively small, then the combinations of persons choosing and persons chosen can easily be represented by a rectangle. A larger information structure may not lend itself to such rectangular convenience. For example, if the membership of all corporation boards are listed, one may wish to produce an index of interlock between different boards, and cluster the corporations in terms of the connectedness of their board membership. This structure again requires different management software, such as represented by the „Baron" program by Levine for representing interlocks.

A list processing system represents data as a series of lists. A list may be composed of items, much like a grocery list, or it may also contain the names of other lists. The system can answer questions like: Is X on list A or any list named by list A? Which lists contain X? Are X and Y on a common list or any lists connected by a higher order list? How many such connections exist between X and Y in the entire data domain?

A central feature of list processing languages is the use of pointers. Each item in a list, in addition to containing a pointer to the contents of that item, also contains a pointer to the next item on the list. Once one enters a list, one can use the pointers to find one's way to successive items on the list. If a list is symmetrical, then each item contains pointers both to the previous item, as well as the next item. Thus, one can find one's way back up the list as well. In terms of the computer, a pointer is nothing more than an offset address, relative to the workspace available for list processing.

The development of list processing and graph procedures has been more extensive in cognitive information processing, especially as represented at Carnegie Mellon University by Simon, Newell, and others. Its application in sociology has been rare and in some cases it was used where it was not necessary. List languages have their costs, as do most data base management systems, so one must generally consider what features are indeed necessary. If a rectangle or a simple hierarchy will suffice, then it is by all means to be preferred, especially if the data base is large.

Summary: The various capabilities we have cited may form odd combinations and they do not summarize in a simple table. However, some of the most important thus are as follows:

- Rectangular
  1) – both dimensions unordered
  2) – one dimension ordered
  3) – one (or more) dimensions grouped
  4) – ordering on two or more dimensions

- Hierarchical
  5) – units unordered
  6) – units at one or more levels ordered
  7) – varying end units

– Ragged (varying size units)
  8) – single string for each unit
  9) – multiple embedded strings
  10) – ordered versus unordered strings

– Networks or graphs
  11) – structures
  12) – processes

Both hierarchies and ragged data forms may logically be an economy of storage for information that can be normalized into a rectangular array by data repetition or padding with blanks. Both ordered data, and the capacity to order data, involves different data referencing and data processing capabilities than packages that limit themselves to unordered data. Networking or graph procedures get into elaborate (and time consuming) pointer systems that may be necessary for some data forms where a matrix would be extremely sparse. At present, the available social science packages focus on the first two capabilities in the summary list.

Role of Data Base Management. Much of the work on commercial data base systems is oriented towards inventory or client list problems. One may have a large inventory of parts and want to order it both in terms of supplier, warehouse location, and object use. Clients may be ordered by location, salesman, speciality, etc. If there is considerable repetition, then one may substitute a code for a complete entry and use a table lookup to get the full entry, or one may turn to a hierarchy so that entries are grouped by their sharing of common information.

If, however, a hierarchy is used, then it is difficult to access information that crosscuts the hierarchy. If parts are organized by object use, then it is difficult to retrieve those that come from the same location other than by searching the entire file. Much of data base management is concerned with threading various indexing schemes through the same data base, either by the use of pointers in relational arrays, hierarchies, or networks. Major emphasis is given to designing such structures to facilitate adding, deleting, updating, and retrieving.

Most data base management systems are for files that are continuously undergoing change, like a current inventory and client list, so that there is a willingness to pay a price for abilities to continually make updates and other changes. In contrast, most social science problems are more set. One may think of a social indicator project, for example, as a matter of repeated updating, such as National Opinion Research Center annual General Survey. But this is updating of a different kind. Once the 1974 survey is completed and edited, for example, it remains constant. The 1975 and 1976 additions will be tacked on as additional units in the data base, but the 1974 unit will not be subject to repeated change. If the social indicator is based

on panel techniques, a simple respondent identifier may tie one unit to the next. Past units then are relatively fixed; not to be molested unless a coding error is discovered.

With little emphasis on updating, social science data entry and storage problems are less difficult. Data entry, both for raw data and codebooks, is straightforward, with routine entry and editing procedures being quite satisfactory. Once the data is checked, it can then be made more compact for final storage, including the use of bit strings and binary representations instead of digital storage. The documentation of this compacting is made part of the machine readable codebook accompanying each data set.

Although we have emphasized the range of social science data formats, the bulk of day to day processing, of course, will continue to be in rectangular format and handled by conventional software. Two level hierarchies may well be handled by social science software such as the Norwegian DDPP system; the „group" option within this software allows for higher level information (e. g. households) to be separately stored, but automatically cross-referenced, with the lower level data (e. g. respondents). Existing data base management systems, such as the IBM *information Management System's* hierarchical approach built to CODASYL specifications, may well be useful for some social science applications. More recent developments of relational file management strategies, using the data management algebra and calculus developed by E. F. Codd, may also prove useful.

Before adapting any system, the user does well to examine the speed and capabilities of the system, making certain it is neither unnecessarily costly or unnecessarily constraining. So far, the experience with attempting to use commercial data base management packages on complex social science data has been frustrating, with users often ending up writing their own programs. This situation is especially true where data comes in many forms and often involve linking varying levels and intensities of reporting. Large scale evaluation research projects, including the areas of negative income tax, health care delivery, and housing programs, are particularly notable for such problems.

One system that may prove useful for a wide range of applications is the SAS statistical package, developed by the SAS Institute in Raleigh, North Caroline. This system separates data statements from procedure statements. The data section employs a pointer system, with the ability to read data in either a stream or user edited mode. The data section can contain GO-TO transfers so as to provide the flexibility of a programming language. The same input file may be referenced in different statements, each time advancing the pointer (or backing it up) by a controlled amount. The data section may be used to combine selected data from several input files; its features include extensive merging options. Data records may be of fixed or variable length.

The result of the data section of SAS is an intermediate file, much like the intermediate files of SPSS, DATA-TEXT or OSIRIS. In fact, SAS can read the intermediate files of these other software systems. The statistical procedure section may be then called, offering a wide range of advanced statistical procedures, plus the op-

tion to access the BMDP statistical package subroutines. Recording and regrouping is accomplished by „sort" or „matrix" procedures.

In future work, the fundamental distinction between data handling and statistical analysis procedure stages may be more carefully observed. Although we have emphasized the hierarchical, ragged character of many kinds of social science data, many advanced statistical analyses require cleaned up rectangles (such as correlation matrices) or triangles (such as symmetrical distance measures) as input. The statistical procedure section should take care to hold intermediate matrices (such as cross products) so as to not have to recalculate them each time a new statistic is used.

We may envision various data base management systems as serving the data stage, well interfaced to turn data over to analysis procedures. A suitable overseer procedure may be needed to supervise the passage of control, or it may just be that the systems „know about" each other. This knowledge might seem only natural; however, there has now been more than a decade of fuzziness, with the most used statistical software systems not being able to read each other's intermediate files.

Is a fundamental separation of stages, with many interfaces, appropriate to a „user oriented" software? I think so, providing it is within reasonable limits. Special care is needed that the data management side is both flexible and natural to the user. In my opinion, an extended APL (to handle trees and other nonrectangular structures) may be natural to a minority of social scientists, but its economy of notation does not come naturally to the community at large. On the other extreme, a COBOL type of pseudo-English is not needed either. Again, SAS has offered a suitable, but highly flexible, intermediate.

Several examples help to point out what has user acceptance. The IBM statistical subroutines, for example, do not find high popularity among social scientists. Part of the reason is that they lack many needed built in features, such as missing data options. The other reason is that they assume that the users write much of their data handling programming. Similarly, „coherent programming", which allowed the user flexible power in putting together building blocks, again was too complex to win large audiences. On the other hand, a simple language like *Utility Coder* offer remarkable power in data handling, and is easily interfaced to a statistical analysis mate such as *Crosstabs*.

The level of effort in social science software programming remains small compared to many commercial centers. Just one division of the American Telephone Company, New England Telephone, has over 400 programmers. Some companies probably have a larger level of effort than all university based social science programming put together. The social sciences can no longer accept the lore of how a few faculty and assistants built systems in the 1950s. As Brook's book, *The Mythical Man-Month*, well describes, this lore tends to lead to mythical estimates and even the belief by old timers, that they can do it again in today's settings. As Brooks shows, the costs of coordinating and documenting increasingly complex systems multiplies almost exponentially. With limited financial resources, the social sciences thus do well to watch the commercial ventures closely, as well as developments by government agencies such as Census and Social Security.

In order that social scientists can understand their own needs better, we need to go beyond the SPSS user survey and carefully examine the different kinds of analysis problems that arise. How then can the spectrum of problems be best partitioned? Which tasks are adequately covered and where is more development most urgently needed? In some cases, such partitioning may bring together researchers from quite different fields who were previously unaware that they had data analysis formats in common.

Once the many disciplines of the social sciences, including economics, demography, geography, sociology, history, psychology, political science, evaluation research, education, survey research, group dynamics, and anthropology have been integrated into a suitable analysis framework, our future steps may reflect considerably more wisdom. By identifying commonalities, it may be that enough different groups show a shared data handling problem so that a larger systems endeavor is more appropriate than any one group could justify. The commonalities may help some groups understand better what their data problems are and the range of options open to them. A reference point can be established for future planning.