Behavior Research Methods, Instruments, & Computers
1992, 24 (4), 554-559

— PROGRAM ABSTRACTS/ALGORITHMS —

SDIS: A sequential data interchange standard

ROGER BAKEMAN
Georgia State University, Atlanta, Georgia

and

VICENÇ QUERA
University of Barcelona, Barcelona, Spain

Few general-purpose computer programs are available that analyze sequential categorical data. If there were a sequential data interchange standard—a standard way of representing sequential data—then it would be more attractive to write general-purpose computer programs for such data. Moreover, interlaboratory sharing would be facilitated. The present paper defines such a standard, called the sequential data interchange standard, or SDIS. Both the SDIS data language and a parsing program for data that follow SDIS conventions are described. The parsing program will be made available to researchers who wish to develop analysis programs for sequential data.

Few general-purpose computer programs are available that analyze sequential categorical data (Bakeman & Gottman, 1986). Many researchers have developed their own programs, but often these are so tailored to a particular laboratory's work that they are not useful to others. Moreover, different programs make quite different assumptions about the form of the input data. Nonetheless several programs have been developed for the analysis of sequential data (see, e.g., Arundale, 1984; Bakeman, 1983; Deni, 1977; Dodd, Bakeman, Loeber, & Wilson, 1981; Gardner, 1990; Kienapple, 1987; Quera & Estany, 1984; Sackett, Holm, Crowley, & Henkins, 1979; Schlundt, 1982; Symons, Wright, & Moran, 1988; Yoder & Tapp, 1990).

If there were a standard way of representing sequential data, and if individuals who write programs were to adhere to this standard, then laboratories could more easily share programs. Moreover, if many laboratories used the same format for their sequential data, then it would be more attractive to write general-purpose computer programs for such data.

The present paper represents an attempt to define such a standard, which we call the *sequential data interchange standard*, or SDIS. The intent is to develop natural and easy-to-use forms—forms that will be easy to enter into computer files and, at the same time, will make it easy to represent aspects of the data that are important to researchers. Currently, we are developing a general-purpose computer program that will analyze data expressed using SDIS conventions (Bakeman & Quera, 1992). Our hope is that others will see the utility of this standard and will develop additional analysis programs for SDIS data. To encourage such development, we have developed a parser for SDIS data and will make it available to other researchers.

Three sections follow. The first introduces the basic conventions of the SDIS data language, the second describes more specialized features, and the third briefly describes the parser program and its availability.

The SDIS Data Language: Basic Conventions

General. Information may be entered anywhere on a line; there are no fixed columns. Elements of the data language are separated by special characters (punctuation such as commas and semicolons) and/or by one or more blanks, tabs, or returns (the end of line generated by the enter key).

Codes. Codes are formed from numbers, letters, or the underscore character, alone or in combination. Examples are *A*, *16*, *K5*, *Sit*, and *MOM_8*. Symbols other than the underscore are not allowed. Uppercase and lowercase letters are not treated the same, thus *NURSE* is not the same code as *Nurse* or *nurse*.

Times. Times consist of one, two, or three numbers separated by either colons or periods. This definition generates seven possibilities, although only the first four are likely to be used. If *u* represents any number and *v* a number in the range 0 to 59, the possible forms are:

1. *u* (e.g., 8, 21, or 6034)
2. *u:v* (e.g., 8:21)
3. *u:v:v* (e.g., 20:02:36)
4. *u.u* (e.g., 7.981)
5. *u:v.u*
6. *u.u.u*
7. *u.u:v*

Any number after a colon is assumed to be 1/60 of the preceding number. Thus, 8:21 could be 8 hours 21 minutes or 8 minutes 21 seconds, whereas 20:02:36 (or 20:2:36) could be 20 hours, 2 minutes, 36 seconds. A number 60 or greater after a colon is an error.

A simple number, the first number before a colon, or any number after a period is assumed to be decimal. Thus, 7.981 means 7 and 981 thousandths. Decimal numbers

We would like to thank K. Kienapple, who provided useful comments on an earlier draft, M. T. Anguera, without whom we would not have begun this work, and D. Borkman for her excellent editorial assistance. Requests for reprints may be sent to R. Bakeman, Department of Psychology, Georgia State University, Atlanta, GA 30303 (e-mail: psyrab@gsuvm1), or V. Quera, Departamento de Metodologia de las Ciencias del Comportamiento, Facultad de Psicologia, Universidad de Barcelona, 08028 Barcelona, Spain (e-mail: v.quera@ub.es).

may represent any unit the user wishes (milliseconds, seconds, hours, days, etc.), but presumably all times in a data file represent the same units.

Precision is inferred from the first time value encountered in the data. Thus, when using the *u.u* form, if any time in the file is expressed in thousandths (e.g., 7.981), the first time given in the file must indicate three digits after the decimal point (e.g., 5.000). Again, when using the *u:v* form, the first time given must be, for example, 5:00, not 5. If a simple number such as 8 or 6034 were encountered first (the *u* form), the parser would assume precision at the whole integer level.

Subjects. A forward slash (/) terminates data for a subject. *Subject* is used in the generic sense of *sampling unit*, and might refer to an individual person, an animal, a dyad, a family, and so on.

Sessions. If a subject is observed for more than one session, a semicolon separates sessions. A session consists of a sequence of codes, perhaps with associated time information, for which continuity can reasonably be assumed.

Names. Names or other identifying information, enclosed in less-than greater-than brackets (e.g., <*Subject 15*>), may be placed at the beginning of data for individual subjects and sessions.

Conditions. Subjects may be assigned to conditions as defined by a single- or multiple-factor design. If present, design information is enclosed in parentheses and placed before the slash for the last subject in a group. For example, (*1,2*) before the terminating slash indicates that all preceding subjects are associated with Level 1 of Factor A and Level 2 of Factor B.

Comments. Comments, enclosed in percent signs, may be placed anywhere in the file. Unlike names, comments need not be closed explicitly. If only one percent sign occurs on a line, then it is assumed that the comment extends to the end of the line, but not beyond.

Data types. The first word entered in an SDIS data file is either *Event*, *State*, *Timed*, or *Interval* and identifies the type of data contained in the file as event sequential data (ESD), state sequential data (SSD), timed event sequential data (TSD), or interval sequential data (ISD), respectively (Bakeman & Gottman, 1986). It is followed by an optional list of legitimate codes (an advanced feature described later) and a terminating semicolon.

Briefly, *event sequences* consist simply of coded events; duration of individual events is not of interest. *State sequences* consist of a single stream (or several parallel streams) of coded events, recorded in a way that preserves timing information for each state. The states within each stream or set are defined to be mutually exclusive and exhaustive, hence the beginning of a new state necessarily implies the end of the previous one. *Timed event sequences* allow for more complexity than the previous two data types. Codes may represent *momentary behaviors* (only frequency, and not duration, is of interest), in which case only onset times need be recorded, or codes may represent *duration behaviors*, in which case both onset and offset times would be preserved. Events need not be mutually exclusive; indeed, often the co-occurrence of various events is of interest.

Interval sequences are somewhat different. They consist of codes associated with successive time intervals. Procedures that yield interval sequences are typically inexpensive and reliable (pencil, paper, and stopwatch); hence, this data type is often used when only approximate time information is desired and more accurate recording procedures are not feasible. Procedures for recording timed event sequences, on the other hand, usually require electronic assistance (e.g., videorecorders or microcomputers with internal clocks). The increasing availability of such equipment, and the greater accuracy afforded and complexity captured, make timed event sequential data increasingly the data type of choice for sequential studies. Nonetheless, all four data types have advantages, and SDIS accommodates them.

The classification of data types used here is a matter of convenience, not logical necessity. State sequences, for example, could be regarded as logically equivalent to timed event sequences, but defining a separate data type allows for some economy in expression, as described subsequently.

Onset times. A comma preceding a time specification indicates an onset time and may be used by itself or in conjunction with a code. If *t* represents time using any of the permissible forms, then

$$,t$$

at the beginning of a session indicates the session onset time, whereas

$$c,t$$

indicates an onset time for the specified code. The *c,t* specification is used for momentary behaviors (TSD) and can be used to indicate state onset times (SSD) as well. Momentary behaviors have no offset times, whereas the offset time for a state using this specification would be the onset time for the next state indicated.

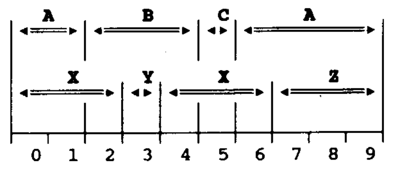
State durations. An equals sign preceding a time indicates duration (for SSD files only). Thus,

$$c=t$$

indicates that state *c* lasted the amount of time indicated by *t*.

Multiple streams. When more than one set of mutually exclusive and exhaustive codes is applied to a session (SSD files only), the different streams are separated with an ampersand (&). With the *c=t* form, the ampersand signals that the next specification should begin at the session onset time so that the next stream can overlay the previous one. With the *c,t* form, a state onset time equal to or less than the previous state's onset time is usually regarded as an error. The ampersand suspends this rule for the next specification, again allowing the next stream to overlay the previous one. This and other SSD specifications are shown in Figure 1.

The ampersand is also permitted with TSD files. Momentary and duration behaviors are allowed to over-



```
State  <subject #1>  A,0  B,2  C,5
A,6  ,10  &  X=3  Y=1  X=3  Z=3;
... ;  %last session% ... /
<#2> ... ; ... /  <#3> ... /
<#4> ... (1,1) / ...
```

Figure 1. An example of state sequential data. Two streams are shown: the first includes codes *A*, *B*, and *C*, and the second includes codes *X*, *Y*, and *Z*. The design includes two factors. These 4 subjects are assigned to Level 1 of Factor A and Level 1 of Factor B.

lap, which is an advantage of timed event sequences. Still, codes must be entered sequentially (because this is an effective way to catch data entry errors); therefore, any onset time that is less than the onset time for the previous code is usually regarded as an error. An ampersand preceding a specification allows times to be entered out of sequence. Thus,

A,4:32 & B,4:27-4:51

would not be regarded as an error.

Offset times. Session offset times are indicated the same way as session onset times, with a comma followed by the time. In addition, offset times for codes are indicated by a minus sign or hyphen preceding the time specification. Thus,

$c,t_1\text{-}t_2$

indicates that the code beginning at Time 1 lasted up to (but not including) Time 2. This specification is used for duration behaviors (TSD files only). If no time specification follows the hyphen

$c,t_1\text{-}$

then the offset time is assumed to be the onset time for the next code given. In effect, this specification renders state sequential data files unnecessary since SSD can be converted to TSD by adding a hyphen to all state onset times. However, for users whose data consist solely of states, adding hyphens to all onset times complicates data entry and increases the size of the file; we have therefore preserved SSD as a separate type. Moreover, SSD, and only SSD, allows the $c{=}t$ form.

Inclusive offset times. By default, duration is computed *exclusive* of stop time. Thus, the duration for a session that started at 8 and ended at 43 would be 35 units, and the duration for a code that started at 1:56 and ended at 2:02 would be 6 units (1:56, 1:57, 1:58, 1:59, 2:00, and 2:01). A right parenthesis after the time specification indicates that duration should include and not exclude the stop time. For example,

,8 . . . codes . . . ,43);

indicates a session that started at 8, ended at 43 inclusive, and so lasted 36 units, whereas

A,1:56-2:02)

indicates that Code *A* lasted 7 time units.

Timed context codes. Some behaviors may last a long time, and so it may be convenient to indicate onset and offset times at separate points in the data file (TSD files only). This is especially useful for contextual or situational information that spans an entire session. A plus or minus sign preceding the time specification is used to indicate an onset or offset time, respectively. Thus,

$c,+t_1$

and

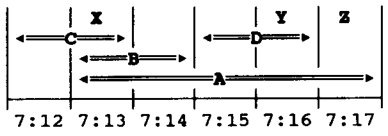
$c,\text{-}t_2$

or

$c,\text{-}t_2)$

indicate that code *c* began at Time 1 and lasted up to Time 2, exclusive or inclusive, respectively. If no matching offset time is encountered before the end-of-session semicolon (or the end-of-subject slash for the subject's last session), the offset time for such codes becomes the session offset time; thus, context codes turned on at the beginning of a session are automatically turned off at the end. Similarly, if there is no earlier matching onset time for an offset specification, the onset time for such codes becomes the session onset time. These and other TSD specifications are shown in Figure 2.

Interval duration. Interval sequences (ISD) are quite different from event, state, and timed event sequences. Rather than recording events and the times they occur,



```
State  <1st subject>  C,7:12-7:14
X,7:13  A,+7:13  B,7:13-
D,7:15-7:16)  Y,7:16  Z,7:17
% end of session %  ;
...  % end of 1st case %  /  ...
```

Figure 2. An example of timed event sequential data. Duration codes are *A*, *B*, *C*, and *D*; momentary codes are *X*, *Y*, and *Z*. Session onset and offset times are 7:12 and 7:18 by default.

the presence of events is recorded at specified time intervals, and so the recording interval or duration becomes an essential part of the definition of an ISD file (for additional discussion of what can become a complex topic, see Altmann, 1974; Quera, 1990; Suen & Ary, 1989). The interval duration follows an equals sign (like a state duration). This specification appears immediately after the word *Interval* at the start of an ISD file. For example,

Interval =10 . . .

indicates that an interval duration of 10 time units is in effect for this particular ISD file.

Intervals. Commas indicate interval boundaries (only for ISD). For example,

A B, A, A C, , X;

places codes *A* and *B* in Interval 1, *A* in Interval 2, *A* and *C* in Interval 3, no codes in Interval 4, and *X* in Interval 5. The same code or codes, or lack of codes, may characterize several successive intervals, in which case the codes need not be repeated. Instead, the number of identical intervals can be indicated with an asterisk followed by a number, which signifies that all codes in the present interval characterize not just the present interval but the number of intervals instead. For example,

*A X *3, B, *2, Y;*

and

A X, A X, A X, B, , , Y;

are equivalent. No other codes can follow the **n* specification. If present, it must be the last piece of information for the interval. It applies to all codes in the interval *except* context codes (see next paragraph).

Interval context codes. Some codes may characterize all or most of the intervals in a session and so, rather than entering those codes in all of the relevant intervals, it may be more convenient to indicate an onset and an offset interval. Similar to the conventions used for timed context codes, a plus or minus sign following a code indicates onset or offset, respectively. Thus,

c+

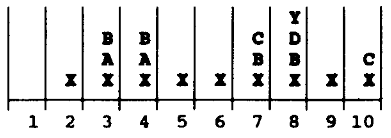
and

c-

or

c-)

indicate that *c* is coded for all intervals beginning with the interval in which the *c+* appears and continuing up to, but not including, the interval in which the *c-* appears (exclusive form), or up to and including the interval in which the *c-)* appears (inclusive form); *c)* may be used as a shorter version of *c-)*. Any *c+* specification that lacks a matching *c-* is assumed to last until the end-of-session semicolon (or the end-of-subject slash for the subject's last session); likewise, any *c-* specification that lacks a matching *c+* one is assumed to begin at the start of the session. These and other ISD specifications are shown in Figure 3.



```
Interval  <id = 02651>
% a.m. observation session
,  X+,  A B *2,  *2,  B C,
B D Y,  ,  C;  ...
...  % end of 1st case %  /  ...
```

Figure 3. An example of interval sequential data. *X* is a context code, which is turned off automatically at the end of session.

The SDIS Data Language: Advanced Features

Legitimate codes. By default, any code encountered in the data file is regarded as legitimate, which means that if *Jump* were intended, *jump* or *Jupm* would be regarded as a new code instead of a data entry error. However, a list of legitimate codes can be entered in the data file immediately after the *Event*, *State*, *Timed*, or *Interval* declaration that normally begins the file and before the terminating semicolon. If present, this list may span more than one line. Codes that constitute a mutually exclusive and exhaustive set may be enclosed in parentheses. Any codes encountered in the data that are not on the list will generate an error message.

Interval sampling. Different sampling strategies can be used to collect interval sequences (ISD), and these strategies can have implications for a variety of computations. Three types of sampling are defined. (1) *Momentary* (also called instantaneous): as with a snapshot, the observer records the state of affairs as of a single, essentially instantaneous point in time. (2) *Partial interval* (also called zero-one): the observer notes whether or not specified behaviors occurred at some point during the interval. Behaviors that occurred only once or several times are coded as present. (3) *Whole interval*: the observer notes whether or not specified behaviors occurred during the entire interval. Behaviors that did not occur or did not occupy the entire interval are coded as absent.

Sampling strategy must be specified if the user plans to use analysis programs that take it into account. The default is momentary. For example, *Interval =10* indicates an interval duration of 10 time units with momentary sampling. Partial and whole interval sampling are indicated with a *single quote* and a *double quote* after the interval duration, respectively.

Observing and recording subintervals (ISD only). Sometimes observers using partial or whole interval sampling, especially those who rely on pencil-and-paper recording procedures, may divide the interval into an observing period and a recording period. If this is the case, and if analysis programs that take this distinction into account are used, then the subintervals must be specified. Like an undivided interval duration, durations for the observing and recording intervals follow an equals sign, except that the subintervals are separated by a period. Thus,

$$=x.y$$

indicates an observing subinterval of x time units and a recording subinterval of y time units. The duration for the entire interval is the sum of the durations for the two subintervals.

Absolute session onset times. Normally, all times in a session are regarded as relative to the session onset time, thus if the session onset time is 8:00, 8:03 occurs at Time Unit 4, 3 units after the onset time. Under some circumstances, it may be convenient if onset and offset times for codes are relative to zero, while session onset and offset times reflect, perhaps, the actual time of day. This is accomplished with an asterisk after the session onset time ($,t^*$). Except for the session offset time, all other times would be relative to an assumed session onset time of zero.

Defaults for session onset and offset times. Strictly speaking, defaults for session onset and offset times are not part of the SDIS language. Still, they are usage conventions worth stating.

Onset and offset times are optional for ESD files. If rates are to be computed later, they would need to be provided, or they might be given for documentation purposes. No defaults are defined.

For SSD files, defaults depend on the form. If the $c=t$ form is used, the session onset time defaults to zero and the offset time is determined from the accumulated state durations. Explicit onset and offset times might be provided if the user wanted to document the actual time of day the observations occurred. If the c,t form is used, the session onset time defaults to the onset for the first state; normally, an explicit onset time would not be given. Usually, an explicit offset time would be provided; if none were, it would default to one greater than the onset time for the last state.

Onset and offset times are usually provided for TSD files. If no onset time is given, it defaults to the onset time for the first code. Similarly, if no offset time is given, it defaults to one greater than the offset time for the last code.

SDIS Parser

A parsing program for data files that follow SDIS conventions has been developed. This program checks SDIS data files for syntactical correctness and translates them into a format more suitable for subsequent analysis programs. Translation is optional; the user can just check data files without translating them. The program also produces a summary file.

The current version of the SDIS parser (Version 1.0) converts data into MADAP+ format. This is the sequential format used by the analysis programs included in the (forthcoming) MADAP+ package, which represents an upgrade of MADAP (Micro Analytic Data Analysis Package; Kienapple, 1987). Future versions of the parser will convert SDIS data into other formats as well. In MADAP+ format, every row represents a behavioral change; columns represent time of transition, behaviors that were occurring before the transition (they are coded as bits in a word), and design conditions, subject, and session that correspond to that transition time. The parser is written in Turbo C Version 2.0 and runs on any XT- or AT-compatible computer with MS-DOS Version 2.0 or greater, preferably with a hard disk.

Availability. The program and source code are available from the authors. To receive a copy of the parsing program, mail a formatted floppy disk (IBM compatible, 3.5 or 5.25, preferably double-density, DD) with a self-addressed return mailer, stamped if within the same country, to either author. Enclose a note stating that you will use the parser only for noncommercial purposes, that you will not give copies to others, and that you understand that it is not guaranteed free of error.

REFERENCES

Altmann, J. (1974). Observational study of behaviour: Sampling methods. *Behaviour*, **49**, 227-267.

Arundale, R. B. (1984). SAMPLE and TEST: Two FORTRAN IV programs for analysis of discrete-state, time-varying data using first-order Markov-chain techniques. *Behavior Research Methods, Instruments, & Computers*, **16**, 335-336.

Bakeman, R. (1983). Computing lag sequential statistics: The ELAG program. *Behavior Research Methods & Instruments*, **15**, 530-535.

Bakeman, R., & Gottman, J. M. (1986). *Observing interaction: An introduction to sequential analysis*. New York: Cambridge University Press.

Bakeman, R., & Quera, V. (1992). *Analyzing interaction: A general program for sequential analysis*. Manuscript in preparation.

Deni, R. (1977). BASIC-PLUS programs for Sackett's lag sequential analysis. *Behavior Research Methods & Instruments*, **9**, 383-384.

Dodd, P. W. D., Bakeman, R., Loeber, R., & Wilson, S. C. (1981). JOINT and SEQU: FORTRAN routines for the analysis of observational data. *Behavior Research Methods & Instruments*, **13**, 686-687.

Gardner, W. (1990). CONTIME: Continuous-time analysis of parallel streams of behavior. *Multivariate Behavioral Research*, **25**, 205-206.

Kienapple, K. (1987). Micro-analytic data analysis package. *Behavior Research Methods, Instruments, & Computers*, **19**, 335-337.

Quera, V. (1990). A generalized technique to estimate frequency and duration in time sampling. *Behavioral Assessment*, **12**, 409-424.

Quera, V., & Estany, E. (1984). ANSEC: A BASIC package for lag sequential analysis of observational data. *Behavior Research Methods, Instruments, & Computers*, **16**, 303-306.

Sackett, G. P., Holm, R., Crowley, C., & Henkins, A. (1979). A FORTRAN program for lag sequential analysis of contingency and cyclicity in behavioral interaction data. *Behavior Research Methods & Instruments*, **11**, 366-378.

Schlundt, D. G. (1982). Two PASCAL programs for managing observational data bases and for performing multivariate information analysis and log-linear contingency table analysis of sequential and nonsequential data. *Behavior Research Methods & Instruments*, **14**, 351-352.

Suen, H.K., & Ary, D. (1989). *Analyzing quantitative behavioral data*. Hillsdale, NJ: Erlbaum.

Symons, D. K., Wright, R. D., & Moran, G. (1988). Computing lag sequential statistics on dyadic time interval data: The TLAG program. *Behavior Research Methods, Instruments, & Computers*, **20**, 343-346.

Yoder, P. J., & Tapp, J. T. (1990). SATS: Sequential analysis of transcripts system. *Behavior Research Methods, Instruments, & Computers*, **22**, 339-343.

(Manuscript received November 1, 1991;
revision accepted for publication June 5, 1992.)

**Fifth Annual International Conference of
The Society for the Advancement of Socio-Economics (SASE)
New York City
March 26-28, 1993**

Call for Papers

The Fifth Annual International Conference of the Society for the Advancement of Socio-Economics (SASE) will be held in New York City, March 26-28, 1993. The theme of the conference is Incentives and Values as Foundations of Social Order.

John Kenneth Galbraith (Harvard Univesity) and Robert Heilbroner (New School for Social Research) are among the featured speakers.

Specific topics will include Markets and Democracy in Eastern Europe and Latin America, War and Conversion, Health Care, Ethics, Decision Making, Endogenous Growth, The Environment, Risk Taking, Micro and Macro Socio-Economics, and Gender Issues in the Workplace.

SASE is a group of academic scholars, policy makers, and business people devoted to the development of new theoretical and methodological frameworks which explain economic and more general choice behavior. Premised upon the belief that economic behavior is not an isolated, abstract phenomenon, socio-economics draws upon the disciplines of psychology, sociology, political science, philosophy, and history in an attempt to recognize the complexity of human decision-making processes, and to locate economic behavior within a philosophical, historical, institutional, and ethical context.

Those interested in presenting a paper, organizing a session, or learning more about SASE should write to 714H Gelman Library, 2130 H Street NW, Washington, DC 20052 (phone, 202-944-8167; FAX, 202-994-1639).