Behavior Research Methods & Instrumentation
1983, 15 (5), 530-535

Computing lag sequential statistics: The ELAG program

ROGER BAKEMAN
Georgia State University, Atlanta, Georgia

ELAG, a FORTRAN program that computes lag sequential statistics from event sequence data, is described, and its use is demonstrated. ELAG lets the user specify positive or negative lags of any length, whether codes may follow themselves or not, and whether z scores are to be computed as suggested by Allison and Liker (1982) or by Sackett (1980). Data sequences for individual subjects need not be continuous but can be broken into separate observations. Summary statistics for different groups of subjects may be printed. In addition, the initial data may be modified. The user may specify that an instance of one of a group of codes or of a particular two-code sequence is to be replaced with a new code before ELAG proceeds with its computations.

ELAG is a FORTRAN program that computes lag sequential statistics. First developed by Sackett (1979, 1980), these statistics and their uses have also been described by Bakeman (1978), Bakeman and Dabbs (1976), and Gottman and Bakeman (1979). Computer programs for computing these statistics have been developed by Deni (1977), Dodd, Bakeman, Loeber, and Wilson (1981), and Sackett, Holm, Crowley, and Henkins (1979). Compared with these other programs, ELAG offers both advantages and disadvantages, as detailed below. The description of ELAG presented here assumes that the reader is familiar with the lag sequential approach.

GENERAL CAPABILITIES

Event Sequence Data

ELAG is designed to analyze "event sequence" data. I assume that the user represents behavior as a sequence of coded events, or episodes, or "states" and that the duration of these events is not important. For example, consider the following mutually exclusive codes that could exhaustively describe a young child's "play state": (1) U = unoccupied, doing "nothing" alone; (2) S = solitary, involved in toys alone; (3) T = together, involved with other children. This is an extrememly simple-minded coding scheme, but it is used here for illustrative purposes. Given this coding scheme, a segment of data might look like this:

...UTUTST...USUSTU...

I would like to thank Gene Sackett, who first encouraged me to think about sequential analyses, those researchers who used and commented on earlier versions of this program, and Bruce Dorval, who encouraged me to add the sequential code capability and whose comments helped me improve an earlier draft of this paper. Requests for reprints should be sent to Roger Bakeman, Department of Psychology, Georgia State University, Atlanta, Georgia 30303.

(For readability, letters are used to represent codes here; in fact, ELAG requires that codes be represented numerically, so the above example would be: ... 131323 ... 121231....)

If duration matters to a user, ELAG requires that it be represented explicitly, as though the user had coded each successive time interval instead of each successive event. In that case, the first part of the segment of data given above might look like this:

...UUUTTTTUTTSSSSSTT...

If the user specifies that events may follow themselves (see below), ELAG computes statistics correctly for a case like this. However, users who care about duration and whose events typically last for many time intervals may find data preparation for other programs more convenient than for ELAG.

Lags

Lags may be either positive or negative and may be of any length, limited only by the user's data. (Described here is version 3.0 of ELAG; earlier versions did not have these capabilities.) For each lag specified, ELAG computes joint frequencies, transitional probabilities, and z scores (comparing each transitional probability with its expected value) for all possible pairs of codes. Thus, ELAG does not produce individual lagged probability profiles for particular criterion codes, as some programs do, but it does produce, in one run, information from which all possible profiles can be constructed.

z-Score Computation

Recently, Allison and Liker (1982) criticized Sackett's (1979) z-score computation. At issue is whether expected values are theoretically or empirically derived, but the effect is that the Sackett computation is more conservative than the Allison and Liker one. Recogniz-

ing that different users may decide this matter differently, ELAG lets the user specify which computation is to be used.

ELAG also lets the user specify whether his or her codes may follow themselves or not. If codes are mutually exclusive and exhaustive, and if successive events are being coded, then, by definition, codes cannot follow themselves. At lag one (plus or minus), this can affect the z-score computation quite substantially, no matter whether Sackett (1979) or Allison and Liker's (1982) computation is used. Both computations involve a simple probability for the lagged code, but as Sackett (but not Allison and Liker) emphasized, this probability is estimated quite differently if the criterion code can or cannot occur at lag one. Formulas for these different computations are given below.

Groups, Cases, and Observations

ELAG allows the user's data to be organized in several different ways. If a "case" is the user's basic "unit of analysis" (as in SPSS) and an "observation" is an uninterrupted sequence of codes derived from observing that case (which typically would be a specific individual or a particular dyad), then the simplest way of organizing the data would be (1) as a single uninterrupted observation of one case. But ELAG also lets the user organize data (2) as several observations of one case and (3) as single or several observations of several cases. Furthermore, all cases may be assigned to one or more groups. If cases are assigned to groups, then summary statistics are printed for each group. For example, Bakeman and Brownlee (1980) observed two kinds (groups) of children, preterm and full-term. There were 41 children (cases) in all, and each child was observed on several different days (observations).

Recoding

Finally, ELAG allows for some redefining (or "recoding") of the user's data. Both "synonymous" and "sequential" recodes are permitted. An advantage of synonymous recoding is that it allows the user to analyze a given data stream using more molar definitions than those used when coding the behavior stream initially (see Suomi, 1979). As an example, consider the coding scheme and the data segment given above. We might argue that "solitary" and "together" are both instances of O = "occupied" (a new code), and for some analyses we might choose to lump them together, that is, to make them synonymous. Then we would have just two codes (since Ss and Ts would be recorded to Os) and the data segment given above would look like this (after recoding):

...UOUO...UOUOU...

If the user had specified that codes could follow themselves, this would have been ...UOUOOO...UOUOOU ... instead. However, when the user specifies that codes cannot follow themselves, and when adjacent events are recoded into the same new code, they are merged into just one new event, as the example above demonstrates. (The purpose of this example is to demonstrate synonymous recoding; in practice it would make little sense to recode mutually exclusive events in a way that leaves only two codes defined.)

We might also choose to regard every "solitary" followed by a "together" as an instance of an object-to-person transition. If we used the letter Q to indicate "solitary-to-together" sequences, we would then have four codes (the original three plus Q), and the data segment given above would look like this (after recoding):

...UTUTQ...USUQU...

(Again, this example is chosen to demonstrate sequential recoding, not to make substantive sense.) An advantage of such sequential recoding is that it allows the user to investigate what are, in effect, three-event sequences. For example, given the recode just described, we could ask whether, after a "solitary" followed by "together" (code Q), another "solitary" or an "unoccupied" episode is more likely.

PROGRAM SPECIFICS

ELAG is written in FORTRAN IV and should run with few, if any, modifications on any computer that supports a FORTRAN system. If space is a problem, DIMENSION statements can be changed to suit an individual user's needs. The program makes reference to five files: the control, the data, the print, the punch, and the error message files. The FORTRAN unit numbers for these files are set to 10, 11, 6, 12, and 6, respectively, in a DATA statement, which means that the user can easily change these assignments. Most program restrictions mentioned below can be changed simply by changing DIMENSION and DATA statements, although some would require changes to input and output lists and FORMAT statements as well. (Readers may obtain a printed listing of the program by sending the author a stamped self-addressed envelope. A machine-readable copy of the FORTRAN source will be supplied if the reader sends the author a stamped self-addressed envelope containing a computer tape and a $5.00 check made out to Georgia State University. The returned tape will contain a file of fixed-length 80-character records, 10 to a block, written in EBCDIC at 1600 bpi. An ASCII version can be made available if the user is unable to use or transform EBCDIC.)

DATA FILE

Each case (or "subject") may require several lines in the data file. ("Line" is used here in the computer sense of "record"; it might be a line in a disk file or a punched

card.) The first line contains identifying information for the case; the first 12 characters of this line are used to label the printed output. The following lines contain the data per se, that is, the coded events, in sequence. If the user does not specify how many codes each line contains and provide a format for reading those lines, ELAG assumes 20 codes per line read with a 20I4) format.

Codes must be numeric, positive, and no bigger than 999. Any code with a value of zero is ignored. (This can be useful when a previously prepared data file requires editing.) In addition to the "regular" codes, the user must also specify an "end-case" code and may specify "end-group" and "end-observation" codes as well. End-observation codes may be embedded within the sequence of codes for a case, separating different observations. Each case must end with either an end-case or an end-group code. If a case ends with an end-group code, ELAG assumes that the case is the last in its group and prints out summary statistics for the group. If any line follows the line containing the end-case or end-group code, it must be a line containing identifying information for the next case.

There is no limit to the number of cases or the number of groups and no limit to the number of events within a case. The only limitation is that there may be no more than 1,000 events for each observation, and this can be changed by changing a DIMENSION and DATA statement. However, each different kind of event (or code) that appears in the data must be defined in the control file, on either a "codes" or a "recodes" card (see below).

As an example, consider the data segment given above. If the two sequences separated by ellipsis were separate observations, if the end-observation code were 8 and the end-case code were 9, and if the (20I4) default format were used, then the data file would consist of the following two lines:

```
*DEMO CASE* ....,....,....,....,....,....,
  1  3  1  3  2  3  8  1  2  1  2  3  1  9
```

This data file is used for the examples presented below. Again, it is useful for exposition only; in practice, these statistics should never be based on so few data.

CONTROL FILE

The control file consists of (1) a title card (using "card" in the sense of "record"), (2) a lags card, (3) an options card, (4) an optional format card, (5) a codes card, (6) an end-codes card, (7) a code labels card, and (8) optional recode cards. The contents of each card are detailed in Table 1.

Whether recodes are specified or not, if the last card in the control file is blank, then ELAG will read and process another set of instructions from the control file (see example below). This is useful if the user wants to examine the same data file with different recodes or different options.

PRINT FILE

Unless printing is suppressed, for each case ELAG prints (1) the number of events and number of observations for the case, (2) the simple (lag zero) frequencies and probabilities for each code, and (3) the joint frequencies, transitional probabilities, and z scores for each pair of codes at each lag. Rows represent the criterion or "given" event; columns represent the lagged or "target" event. Thus, joint frequencies indicate the number of times the target event occurred at a particular lag before (negative lags) or after (positive lags) the given event, whereas transitional probabilities indicate how likely it was that the target event would occur at the designated lag before or after the given event. For readability, probabilities are multiplied by 100; thus 23.8 is actually .238.

The z scores indicate whether the target event occurred more often (positive z score) or less often (negative z score) relative to the given event than would be expected, given the occurrence of the target event overall. They are computed as follows. Using Sackett's (1980) formula,

$$Z(g,t) = \frac{F(g,t) - P(t)F(g)}{\sqrt{P(t)F(g)[1 - P(t)]}};$$

and using Allison and Liker's (1982) formula,

$$Z(g,t) = \frac{F(g,t) - P(t)F(g)}{\sqrt{P(t)F(g)[1 - P(t)][1 - P(g)]}}.$$

F(g,t) is the frequency with which the target event, t, occurred at a particular lag before or after the given event, g. P(t) is the probability for the target event, F(g) is the frequency for the given event, and P(g) is the probability for the given event.

If events may follow themselves, or at lags other than plus or minus one, P(t) = F(t)/N. However, if events may not follow themselves–which means that the given event cannot occur at lags plus or minus one–then at those lags, P(t) = F(t)/[N – F(g)]. As the examples below demonstrate, if events may not follow themselves but P(t) is not corrected at lag one as indicated, not only the value but even the sign of the z score can be affected. In the formulas above, N is the number of pairs of events at a particular lag. Its value depends on the lag and the number of observations but will always be less than the total number of events. (For this reason, the frequencies and probabilities that enter into the z-score computation are computed separately for each lag, not estimated from the lag-zero values.)

If a case ends with an end-group code, then summary statistics are printed for all cases since the beginning or since the last end-group code encountered. These are

Table 1
Control File

Card	Column	Contents
1	1-80	Any identifying information desired. It is printed at the beginning of the output.
2	1-4, 5-8, ..., 37-40	The lags desired. These may be any positive or negative number. No more than 10 different lags can be specified at any one time.
3	1-4	The number of codes that each line in the data file contains (may not exceed 80). If positive, the next card must contain a FORTRAN format for the data specifying this many "I" fields. If zero (or negative), ELAG reads 20 codes per line with a (20I4) format.
	5	If blank, z scores are computed according to Sackett, and if not blank, according to Allison and Liker.
	6	If blank, codes may not follow themselves; if not blank, codes may follow themselves. This affects both recoding and z-score computation at lag one, as described above.
	7	If blank, statistics for each case are printed; if not blank, case statistic printing is suppressed.
	8	If not blank, z scores are written to the punch file for subsequent processing by other programs.
	9	If not blank, transitional probabilities are written to the punch file.
	10	If not blank, joint frequencies are written to the punch file.
4	1-80	If columns 1-4 of the third card contain a positive number, then the fourth card contains a FORTRAN format for reading the data. This must begin with a left parenthesis, and end with a right parenthesis, and indicate only integer ("I") fields.
5	1-4, 5-8, ..., 77-80	The user's codes (except for end codes). Twenty are allowed. Their order is not important, but codes must be positive and no bigger than 999. Joint frequencies, transitional probabilities, and z scores are computed for all possible pairs of codes defined on this card. This is done for all lags specified on the lags card.
6	1-4	The user's end-case code.
	5-8	The user's end-group code, if any.
	9-12	The user's end-observation code, if any.
7	1-4, 5-8, ..., 77-80	Labels for the user's codes. They correspond to the numeric codes given on the codes card and are used to label output.
8 ...		(Recode cards, if any, follow the labels card. The format for both synonymous and sequential recodes is given below.)
	1-4	The "new" code or the code to which "old" codes are recoded. (Note–This code must be defined on the codes card.)
	5-8, 9-12, ..., 77-80	The "old" codes for synonymous recodes. When any of these codes occur in the data, ELAG treats it as though the new code had occurred. For example, if the user wanted each instance of old code 2 ("solitary" in the example above) and old code 3 ("together") to be regarded as an instance of a new code 4 ("occupied"), then the recode card would contain a 4 in column 4, a 2 in column 8, and a 3 in column 12. (Note–Old codes for synonymous recodes may not be defined on the codes card.)
	5-8, 9-12	If only two "old" codes are given and if both are prefixed with a minus sign, which is the way a sequential recode is indicated, then whenever the first old code (the code in columns 5-8) is followed immediately by the second old code (the code in columns 9-12) anywhere in the data, ELAG substitutes the new code for the indicated sequence of the two old codes. For example, if the user wanted a new code 4 (Q in the example above) to replace each "solitary-together" sequence (old code 2 followed by old code 3), then the recode card would contain a 4 in column 4, a −2 in columns 7-8, and a −3 in columns 11-12. (Note–Old codes for sequential recodes must be defined on the codes card. Two different sequences may not have the same second old code.)

(1) mean joint frequencies, (2) mean transitional probabilities, (3) mean z scores (the sum of the individual z scores is divided by the square root of the number of cases, not by the number of cases; see Cochran, 1954), and (4) the number of z scores for each transitional probability that were above, or below, their expected value. This is useful because a deviation from a 50-50 split can be tested with a sign test (see Bakeman & Brownlee, 1980). (Note, however, that some z scores may equal their expected values and therefore would not be counted as either above or below.)

PUNCH FILE

The user may request that z scores, probabilities, and/or frequencies be written to a file. They would then be available for processing with other computer programs; for example, z scores might be used as scores for analysis of variance. If any "punched" output is requested, the output for each case is preceded by a line that contains the case's identifying information; the number of events, the number of observations, and the case sequence number. The format for this card is (3A4,3I4). Frequencies are written with a (12F6.0) format, and probabilities and z scores are written with a (12F6.2) format. (Recall that probabilities are multiplied by 100.)

EXAMPLES

An example of a control file and ELAG output is given in the Appendix.

REFERENCES

ALLISON, P. D., & LIKER, J. K. Analyzing sequential categorical data on dyadic interaction: A comment on Gottman. *Psychological Bulletin*, 1982, **91**, 393-403.

BAKEMAN, R. Untangling streams of behavior: Sequential analyses of observational data. In G. P. Sackett (Ed.), *Observing behavior* (Vol. 2) *Data collection and analysis methods*. Baltimore: University Park Press, 1978.

BAKEMAN, R., & BROWNLEE, J. R. The strategic use of parallel play: A sequential analysis. *Child Development*, 1980, **51**, 873-878.

BAKEMAN, R., & DABBS, J. M., JR. Social interaction observed: Some approaches to the analysis of behavior streams. *Personality and Social Psychology Bulletin*, 1976, **2**, 335-345.

COCHRAN, W. G. Some methods for strengthening the common χ^2 tests. *Biometrics*, 1954, **10**, 417-451.

DENI, R. BASIC-PLUS programs for Sackett's lag sequential analysis. *Behavior Research Methods & Instrumentation*, 1977, **9**, 383-384.

DODD, P. W. D., BAKEMAN, R., LOEBER, R., & WILSON, S. C. JOINT and SEQU: FORTRAN routines for the analysis of observational data. *Behavior Research Methods & Instrumentation*, 1981, **13**, 686-687.

GOTTMAN, J. M., & BAKEMAN, R. The sequential analysis of observational data. In M. E. Lamb, S. J. Suomi, & G. R. Stephenson (Eds.), *Social interaction analysis: Methodological issues. Madison: University of Wisconsin Press, 1979.*

SACKETT, G. P. The lag sequential analysis of contingency and cyclicity in behavioral interaction research. In J. D. Osofsky (Ed.), *Handbook of infant development*. New York: Wiley, 1979.

SACKETT, G. P. Lag sequential analysis as a data reduction technique in social interaction research. In D. B. Sawin, R. C. Hawkins, L. O. Walker, & J. H. Penticuff (Eds.), *Exceptional infant* (Vol. 4) *Psychosocial risks in infant-environment transactions*. New York: Brunner/Mazel, 1980.

SACKETT, G. P., HOLM, R., CROWLEY, C., & HENKINS, A. A FORTRAN program for lag sequential analysis of contingency and cyclicity in behavioral interaction data. *Behavior Research Methods & Instrumentation*, 1979, **11**, 366-378.

SUOMI, S. J. Levels of analysis for interactive data collected on monkeys living in complex social groups. In M. E. Lamb, S. J. Suomi, & G. R. Stephenson (Eds.), *Social interaction analysis: Methodological issues.* Madison: University of Wisconsin Press, 1979.

APPENDIX

Control File

```
*** ELAG VERSION 3.0, EXAMPLE # 1.
   1   2
   0
   1   2   3
   9       8
UNOC SOL TOG

*** ELAG VERSION 3.0, EXAMPLE # 2.  USE A&L COMPUTATION.
   1  -1
   0A
   1   2   3
   9       8
UNOC SOL TOG

*** ELAG VERSION 3.0, EXAMPLE # 3.  ASSUME CODES CAN FOLLOW THEMSELVES.
   1
   0 S
   1   2   3
   9       8
UNOC SOL TOG

*** ELAG VERSION 3.0, EXAMPLE # 4.  DEMONSTRATE SYNONOMOUS RECODE.
   1  -2
   0
   1   4
   9       8
UNOC OCC
   4   2   3

*** ELAG VERSION 3.0, EXAMPLE # 5.  DEMONSTRATE SEQUENTIAL RECODE.
   1
   0
   1   2   3   4
   9       8
UNOC SOL TOG OPT
   4  -2  -3
```

Output

```
*** ELAG VERSION 3.0, EXAMPLE # 1.
PROGRAM ELAG:  LAGGED EVENT ANALYZER, VER 3.0 (MAY 1983).
R. BAKEMAN, DEPT OF PSYC, GA STATE UNIV, ATLANTA GA 30303.

  NCODE   NPERL  ..OPTS  ENDCAS  ENDGRP  ENDOBS
     3      20               9       0       8

 IFMT=(20I4)

 CODES=   1   2   3
 LABEL=UNOC SOL TOG


*DEMO CASE*    NEVENTS=  12.0    NOBS=  2.0    NCASE=  1.

LAG  0         UNOC   SOL   TOG
       FREQ     5.0   3.0   4.0
       PROB    41.7  25.0  33.3


LAG  1         UNOC   SOL   TOG
 GIV=  FREQ     0.0   2.0   2.0
 UNOC  PROB     0.0  50.0  50.0
       ZSCO    0.00  0.29 -0.29

LAG  1         UNOC   SOL   TOG
 GIV=  FREQ     1.0   0.0   2.0
  SOL  PROB    33.3   0.0  66.7
       ZSCO   -0.33  0.00  0.33

LAG  1         UNOC   SOL   TOG
 GIV=  FREQ     2.0   1.0   0.0
  TOG  PROB    66.7  33.3   0.0
       ZSCO    0.58 -0.58  0.00


LAG  2         UNOC   SOL   TOG
 GIV=  FREQ     2.0   1.0   1.0
 UNOC  PROB    50.0  25.0  25.0
       ZSCO    0.52  0.00 -0.52

LAG  2         UNOC   SOL   TOG
 GIV=  FREQ     1.0   1.0   0.0
  SOL  PROB    50.0  50.0   0.0
       ZSCO    0.37  0.82 -1.10

LAG  2         UNOC   SOL   TOG
 GIV=  FREQ     0.0   0.0   2.0
  TOG  PROB     0.0   0.0 100.0
       ZSCO   -1.10 -0.82  1.83

*** ELAG VERSION 3.0, EXAMPLE # 2.  USE A&L COMPUTATION.
PROGRAM ELAG:  LAGGED EVENT ANALYZER, VER 3.0 (MAY 1983).
R. BAKEMAN, DEPT OF PSYC, GA STATE UNIV, ATLANTA GA 30303.

  NCODE   NPERL  ..OPTS  ENDCAS  ENDGRP  ENDOBS
     3      20  A            9       0       8

 IFMT=(20I4)

 CODES=   1   2   3
 LABEL=UNOC SOL TOG


*DEMO CASE*    NEVENTS=  12.0    NOBS=  2.0    NCASE=  1.

LAG  0         UNOC   SOL   TOG
       FREQ     5.0   3.0   4.0
       PROB    41.7  25.0  33.3


LAG  1         UNOC   SOL   TOG
 GIV=  FREQ     0.0   2.0   2.0
 UNOC  PROB     0.0  50.0  50.0
       ZSCO    0.00  0.37 -0.37

LAG  1         UNOC   SOL   TOG
 GIV=  FREQ     1.0   0.0   2.0
  SOL  PROB    33.3   0.0  66.7
       ZSCO   -0.40  0.00  0.40

LAG  1         UNOC   SOL   TOG
 GIV=  FREQ     2.0   1.0   0.0
  TOG  PROB    66.7  33.3   0.0
       ZSCO    0.69 -0.69  0.00


LAG -1         UNOC   SOL   TOG
 GIV=  FREQ     0.0   1.0   2.0
 UNOC  PROB     0.0  33.3  66.7
       ZSCO    0.00 -0.69  0.69

LAG -1         UNOC   SOL   TOG
 GIV=  FREQ     2.0   0.0   1.0
  SOL  PROB    66.7   0.0  33.3
       ZSCO    0.40  0.00 -0.40

LAG -1         UNOC   SOL   TOG
 GIV=  FREQ     2.0   2.0   0.0
  TOG  PROB    50.0  50.0   0.0
       ZSCO   -0.37  0.37  0.00

*** ELAG VERSION 3.0, EXAMPLE # 3.  ASSUME CODES CAN FOLLOW THEMSELVES.
PROGRAM ELAG:  LAGGED EVENT ANALYZER, VER 3.0 (MAY 1983).
R. BAKEMAN, DEPT OF PSYC, GA STATE UNIV, ATLANTA GA 30303.

  NCODE   NPERL  ..OPTS  ENDCAS  ENDGRP  ENDOBS
     3      20  S            9       0       8

 IFMT=(20I4)

 CODES=   1   2   3
 LABEL=UNOC SOL TOG
```

```
    *DEMO CASE*     NEVENTS=  12.0    NOBS=  2.0    NCASE=  1.

    LAG  0         UNOC    SOL    TOG
           FREQ     5.0    3.0    4.0
           PROB    41.7   25.0   33.3


    LAG  1         UNOC    SOL    TOG
     GIV=  FREQ     0.0    2.0    2.0
     UNOC  PROB     0.0   50.0   50.0
           ZSCO   -1.31   0.87   0.41

    LAG  1         UNOC    SOL    TOG
     GIV=  FREQ     1.0    0.0    2.0
      SOL  PROB    33.3    0.0   66.7
           ZSCO    0.13  -1.13   0.94

    LAG  1         UNOC    SOL    TOG
     GIV=  FREQ     2.0    1.0    0.0
      TOG  PROB    66.7   33.3    0.0
           ZSCO    1.39   0.13  -1.41
```

```
*** ELAG VERSION 3.0, EXAMPLE # 5.  DEMONSTRATE SEQUENTIAL RECODE.
PROGRAM ELAG:  LAGGED EVENT ANALYZER, VER 3.0 (MAY 1983).
R. BAKEMAN, DEPT OF PSYC, GA STATE UNIV, ATLANTA GA 30303.

  NCODE    NPERL  ..OPTS  ENDCAS  ENDGRP  ENDOBS
     4       20                9       0       8

  IFMT=(2014)

 CODES=   1   2   3   4
 LABEL=UNOC SOL TOG OPT

    NEW....OLD CODES...(RECODE OLD TO NEW).
      4       -2  -3     SEQUENTIAL RECODE


*DEMO CASE*     NEVENTS=  10.0    NOBS=  2.0    NCASE=  1.

LAG  0         UNOC    SOL    TOG    OPT
         FREQ    5.0    1.0    2.0    2.0
         PROB   50.0   10.0   20.0   20.0


LAG  1         UNOC    SOL    TOG    OPT
 GIV=  FREQ     0.0    1.0    2.0    1.0
 UNOC  PROB     0.0   25.0   50.0   25.0
       ZSCO     0.00   0.25   0.41  -0.61

LAG  1         UNOC    SOL    TOG    OPT
 GIV=  FREQ     1.0    0.0    0.0    0.0
  SOL  PROB   100.0    0.0    0.0    0.0
       ZSCO     1.15   0.00  -0.63  -0.63

LAG  1         UNOC    SOL    TOG    OPT
 GIV=  FREQ     1.0    0.0    0.0    1.0
  TOG  PROB    50.0    0.0    0.0   50.0
       ZSCO     0.00  -0.63   0.00   0.50

LAG  1         UNOC    SOL    TOG    OPT
 GIV=  FREQ     1.0    0.0    0.0    0.0
  OPT  PROB   100.0    0.0    0.0    0.0
       ZSCO     1.00  -0.45  -0.71   0.00
```

```
*** ELAG VERSION 3.0, EXAMPLE # 4.  DEMONSTRATE SYNONOMOUS RECODE.
PROGRAM ELAG:  LAGGED EVENT ANALYZER, VER 3.0 (MAY 1983).
R. BAKEMAN, DEPT OF PSYC, GA STATE UNIV, ATLANTA GA 30303.

  NCODE    NPERL  ..OPTS  ENDCAS  ENDGRP  ENDOBS
     2       20                9       0       8

  IFMT=(2014)

 CODES=   1   4
 LABEL=UNOC OCC

    NEW....OLD CODES...(RECODE OLD TO NEW).
      4       2   3


*DEMO CASE*     NEVENTS=   9.0    NOBS=  2.0    NCASE=  1.

LAG  0         UNOC    OCC
         FREQ    5.0    4.0
         PROB   55.6   44.4


LAG  1         UNOC    OCC
 GIV=  FREQ     0.0    4.0
 UNOC  PROB     0.0  100.0
       ZSCO     0.00   0.00

LAG  1         UNOC    OCC
 GIV=  FREQ     3.0    0.0
  OCC  PROB   100.0    0.0
       ZSCO     0.00   0.00


LAG -2         UNOC    OCC
 GIV=  FREQ     3.0    0.0
 UNOC  PROB   100.0    0.0
       ZSCO     1.41  -1.41

LAG -2         UNOC    OCC
 GIV=  FREQ     0.0    2.0
  OCC  PROB     0.0  100.0
       ZSCO    -1.73   1.73
```

Note—The control file (listed at the beginning of the Appendix) used with the data file described in the text will produce the output listed in the subsequent parts of the Appendix.

(Manuscript received June 29, 1983;
revision accepted for publication August 30, 1983.)