Behavior Research Methods
2009, 41 (1), 137-147
doi:10.3758/BRM.41.1.137

Observer agreement for timed-event sequential data: A comparison of time-based and event-based algorithms

Roger Bakeman
Georgia State University, Atlanta, Georgia

Vicenç Quera
University of Barcelona, Barcelona, Spain

and

Augusto Gnisci
Second University of Naples, Caserta, Italy

Observer agreement is often regarded as the *sine qua non* of observational research. Cohen's κ is a widely used index and is appropriate when discrete entities—such as a turn-of-talk or a demarcated time interval—are presented to pairs of observers to code. κ-like statistics and agreement matrices are also used for the timed-event sequential data produced when observers first segment and then code events detected in the stream of behavior, noting onset and offset times. Such κs are of two kinds: time-based and event-based. Available for download is a computer program (OASTES; Observer Agreement for Simulated Timed Event Sequences) that simulates the coding of observers of a stated accuracy and then computes agreement statistics for two time-based κs (with and without tolerance) and three event-based κs (one implemented in The Observer, one in INTERACT, and one in GSEQ). On the basis of simulation results presented here, and due to the somewhat different information provided by each, the reporting of both a time-based and an event-based κ is recommended.

Investigators who use systematic observation to measure various aspects of behavior are rightly concerned with observer agreement. If the records of 2 observers recorded independently do not agree, then the accuracy of any scores derived from these records is dubious, and we conclude that modification of the coding scheme, further observer training, or both, are required. On the other hand, when observers' records substantially agree, we infer that our observers are adequately trained and that scores derived from those records will be reliable. In sum, observer agreement is regarded as a *sine qua non* of observational research and measurement (Bakeman & Gottman, 1997). However, reflecting various recording methods and different models of observer decision making, a variety of algorithms exist for assessing observer agreement, some of which are more firmly based on well-known statistical models than others.

Reflecting the development of computer and digital technology, a single data recording (or data logging) approach is becoming increasingly standard (Jansen, Wiertz, Meyer, & Noldus, 2003). Working with digital multimedia (video and sound) recordings displayed on computer monitors, observers depress keys to note onsets of events (live observation or video tapes are other possibilities). Offsets may also be explicitly logged, or they could be inferred from the onset of a later coded event in the same mutually exclusive and exhaustive (ME&E) set. With such instrumentation, continuously alert observers (continuous sampling) log data in a way that allows frequency, duration, co-occurrence, and contingency information to be derived later that is limited only by the precision with which time is recorded (which is either 0.033 . . . or 0.04 of a second, when working with US NTSC or European PAL video, respectively). Two commercially available programs that effect such data logging are Mangold International's INTERACT (www.mangold-international.com) and Noldus Information Technology's The Observer (www.noldus.com); a third is the James Long Company program (www.jameslong.com). (Hereafter, following these companies' practices, we refer to the two programs as INTERACT and The Observer, but use lowercase otherwise—e.g., the Interact algorithm.)

The present article uses computer simulation to compare five algorithms for assessing observer agreement given *timed-event sequential data* (TSD; Bakeman & Quera, 1995)—that is, continuously sampled, time-logged observational data of the sort just described. The five are time-unit κ, time-unit κ with tolerance, the Observer al-

R. Bakeman, bakeman@gsu.edu

gorithm, the Interact algorithm, and the Generalized Sequential Querier (GSEQ) dynamic programming (DP) algorithm, respectively. The first and second algorithms are implemented in GSEQ (Version 4.2 and earlier; Bakeman & Quera, 1995), the first and third in The Observer Version 5.0 (Noldus Information Technology, 2003), and the first in INTERACT (Mangold, 2006). The fourth is implemented in Version 8.4.4 of INTERACT, and the fifth is implemented in both Version 5.0 of GSEQ and the simulation program described in the present article. The GSEQ DP algorithm is an extension of a dynamic programming algorithm we developed previously (Quera, Bakeman, & Gnisci, 2007) for *event sequential data* (ESD; only sequence but no times recorded). In the next section—partly as a way to introduce common concepts and terminology—we will comment on different types of observational data and how they can be represented, and we will describe one standard model of observer decision making and observer agreement.

Templates for Event, Interval, and Timed-Event Sequential Data

Imagine that we want to define a universal template for recording and representing discrete microcoded observational data. We assume that a set or sets of ME&E codes have been defined (discrete, nominal-scale measurement) and that these codes are intended to be assigned to events as they unfold over time (microcoded), as opposed to, for example, periodically sampling something like heart rate (interval-scale measurement), or rating from 1 to 7 (ordinal-scale measurement) something like positive emotional tone over a relatively lengthy observational session (macrocoded). Defining a standard template for representing observational data is useful because it clarifies analytic possibilities and facilitates subsequent analysis. A simple grid would work. Each row would represent a different code. Columns would represent either successive events (a code-event grid) or successive time intervals (a code-interval grid), and data recording could consist simply of checking cells in this grid.

For the moment, imagine that either the events or the time intervals to be coded are defined for the coders prior to coding. For example, transcripts have been prepared, and observers are asked to code successive turns of talk—perhaps on several dimensions (e.g., Adamson & Bakeman, 2006)—or, observers are asked to note whether any of several mother and infant behaviors occurred in successive 15-sec intervals (see, e.g., Bakeman, Adamson, Konner, & Barr, 1990). In these two examples, observers were presented with discrete entities to code. In particular, they were not asked to segment (i.e., unitize) the stream of behavior into coding units before assigning codes to those units; the coding units were, in effect, prepackaged. In cases like these, when two observers independently code the same sequence of (predefined) events or intervals, the resulting agreement–disagreement tallies clearly fit the observer decision-making model assumed by Cohen's κ (1960), which characterizes agreement with respect to a set of ME&E codes while correcting for chance agreement, and is probably the most frequently used statistic of observer agreement. The decision-making model is straightforward: Pairs of observers are presented with a discrete entity to code; each makes an independent judgment (selects from one of k codes), and a tally is entered in the $k \times k$ agreement matrix. Observers make decisions when presented with a discrete entity, and the number of tallies represents the number of paired-coding decisions made.

What Bakeman and Quera (1995) termed *interval sequential data* (code-interval grid) fit κ's requirement for discrete units, but ESD (code-event grid) satisfies this requirement only when events are presented to coders as discrete units. Frequently, however, this is not the case: Observers are asked to first segment the stream of behavior into events (i.e., detect the seams between events) and then code the segments. When two observers are asked to code the same material, because of errors of omission and commission as well as simple disagreement, usually the records produced contain different numbers of events. And, exactly how they align is not always obvious (absent a master record giving the "true" state of affairs). This is a classic problem in observational research (Bakeman & Gottman, 1997), although recently we (Quera et al., 2007) proposed a solution for ESD based on algorithms originally designed for aligning nucleotide sequences; more on this later.

For the TSD with which this article is concerned, alignment is not a problem—at least when events are located on a common time line. We term the universal template for representing such data a *code-time grid*. Each column represents a time unit as defined by the precision of data recording. For example, if times are recorded or rounded to the nearest second, each column represents a second. This reflects the fact that time—although continuous in theory—is discrete in practice, with discrete units defined by the precision used. TSD are not recorded this way, of course. Observers do not make checks in the cells of a code-time grid; instead, the data-as-recorded typically consist of event onset times (and offset times in some cases) recorded to a given precision. However, the data-as-represented for subsequent analysis can be conceptualized as a code-time grid (as, e.g., by the GSEQ program; Bakeman & Quera, 1995), and doing so facilitates subsequent analysis (and the writing of general purpose computer programs).

The present article emphasizes observer agreement with respect to the data collected. When scores are derived from such data (e.g., summary scores, such as relative frequency, or proportion of time for a given code or measures of contingency, such as Yule's Q or the log odds ratio) and subsequently analyzed, their reliability could be gauged with standard psychometric methods (e.g., an interclass correlation coefficient; Suen, 1988). However, both before and during data collection, investigators are concerned with observer training and credentialing (i.e., meeting an accuracy criterion), and for these purposes, methods for characterizing observer agreement for the data collected are essential. In the next section, we will describe five algorithms for quantifying observer agreement for such data, given TSD.

Observer Agreement Algorithms for TSD

The five algorithms described in the present article all enter tallies in either a $k \times k$ or a $k + 1 \times k + 1$ agreement matrix, and all produce a summary statistic using a standard κ computation (with adjustments for structural 0 if needed). However, the observer decision-making models on which these algorithms are based differ from the classic Cohen model described earlier. For the classic model, the number of observer decisions is the same as the number of tallies in the agreement matrix; as described subsequently, such one-to-one correspondence cannot be assumed for the other models. None satisfy the assumption of independent tallies required by the classic Cohen's κ; in particular, the standard error formula for Cohen's κ, even though it is rarely used, would not be appropriate for these other κs. Still, the magnitude of the κ produced, along with the agreement matrix itself, is a useful tool for observer training, and a κ value is often cited by investigators as being indicative of acceptable observer agreement. Thus—and this is the motivation for the present article—investigators should understand differences among these various κs.

Time-unit κs with and without tolerance. The code-time grid suggests one straightforward approach to computing observer agreement for TSD: The time units can be tallied, and what we call a time-unit κ ($\kappa_{\text{time-unit}}$) can be computed (Bakeman & Quera, 1995). The agreement matrix is constructed with rows and columns labeled with the codes of the ME&E set under consideration, as usual. Then, successive columns of the linked code-time grids for the 2 observers are examined. For each column, a tally is added to the cell at the intersection of the row defined by the first observer's code and at the column defined by the second observer's code. Thus, the total number of tallies is the number of time units (e.g., 300, if seconds were the unit and 5 min were coded), and, as usual, good agreement is indicated by a preponderance of tallies on the upper-left to lower-right diagonal.

The procedure just described seems formally identical to that used with a code-event or a code-interval grid, but with a key difference: When, for example, intervals are externally defined (e.g., Konner's [1978] click in the ear at 15-sec intervals), observers are aware of making a coding decision for each successive interval. In contrast, when recording TSD, observers are continuously looking for the seams between events, but how often they are making decisions is arguable—even unknowable. One decision per seam seems too few; the observers are continuously alert. However, one per time unit—as assumed by the $\kappa_{\text{time-unit}}$ procedure just described—seems too many. Moreover, the number of tallies is affected by the precision of the time unit chosen, although multiplying all cells in an agreement matrix by the same factor does not affect the value of κ (Bakeman & Gottman, 1997), and investigators almost always focus on the magnitude of κ (which is unaffected by the number of time units) and not its statistical significance (which is). Several programs compute $\kappa_{\text{time-unit}}$ (e.g., GSEQ, INTERACT, The Observer). Jansen et al. (2003) referred to it as "labeling instances of behavior with a duration-based comparison" (p. 395), where *duration-based* indicates time, but the fact remains that the time units tallied are arbitrary in a way that the event or interval units described earlier are not.

One variant of $\kappa_{\text{time-unit}}$ we call *time-unit* κ *with tolerance* ($\kappa_{\text{tolerance}}$). Exact second-by-second agreement can seem overly stringent (Hollenbeck, 1978). Imagine, for example, given a time unit of 1 sec, we define a tolerance of 2 sec. We then examine each successive time unit for the first observer and tally an agreement if there is a match with any time unit for the second observer that falls within the stated tolerance (e.g., a 2-sec tolerance defines a 5-sec window, two units before and two after the current time unit). The effect is to move some tallies of the agreement matrix from off-diagonal to on-diagonal cells, thereby giving credit for near misses and increasing the magnitude of κ ($\kappa_{\text{tolerance}}$ is implemented in GSEQ). On the basis of our simulations, we know that the value of $\kappa_{\text{tolerance}}$ varies slightly, depending on which of the two observers is regarded as the first; hence, for the present article, we computed $\kappa_{\text{tolerance}}$ as the mean of its two possible values.

In sum, it seems almost certain that the observer decision model assumed by time-based κs overestimates—perhaps greatly—the number of decisions the observers made, but in fact there is no way to know the true number of observer decisions.

Event-based κs. A second approach to computing observer agreement for TSD focuses on events and not on time units; hence, the tallies in the agreement matrix are considerably fewer in number than they are with a time-unit κ. The underlying assumption is that instead of making decisions continuously, moment by moment, observers are making decisions only when behavior changes. Thus, whereas time-unit algorithms almost certainly overestimate, event-based algorithms almost certainly underestimate the "true" number of observer decisions. However, in both cases, the actual number of observer decisions is unknowable, which, as was noted earlier, is not the case for the classic Cohen model.

With event-based algorithms, before tallies can be placed in an agreement matrix, the events in the two observers' timed-event sequential records need to be linked. Doing so presents problems similar to those encountered when attempting to link two event sequential records, as was discussed earlier. In the present articles we describe three algorithms that attempt such a linking. One, a refinement of an algorithm described by Haccou and Meelis (1992), is implemented in Version 5 of The Observer. Jansen et al. (2003) referred to it as "labeling instances of behavior with a frequency-based comparison" (p. 395), where *frequency-based* indicates events. A second is a similar algorithm implemented in Version 8.4.4 of INTERACT (P. T. Mangold, personal communication, November 14, 2007; C. Spies, personal communication, February 1, 2008), and the third is an extension to the solution for event sequences cited earlier (Quera et al., 2007), which is implemented in Version 5.0 of GSEQ. Later, we will describe results of simulations comparing all five algorithms—the two that are time-unit based and the three that are event based—and we will present figures showing how the three event-based algorithms link events for a simple example. First, however, we will describe each of the three event-based algorithms.

The Observer algorithm. The Observer algorithm consists of five steps, or passes (Jansen et al., 2003; Noldus Information Technology, 2003, see pp. 450–452). The purpose is to link each event in each observer's record with one or more events in the other observer's record. When a pair of events is linked, the appropriate tally (either an agreement or disagreement) is added to the agreement matrix. Users may define a tolerance; our description of the algorithm in the present article assumes a code-time grid representation with a precision of 1 sec, a stated tolerance (e.g., 2 sec, which is The Observer's default), and records of 2 observers with the same start and end times. On each pass, the algorithm considers any yet unlinked events in turn (called the current event)—ordered from earliest to latest on the basis of their onset times—no matter which observer logged it. Processing stops when all events are linked.

On the first pass (perfect matches), two events are linked if an identical event in the other observer's record overlaps any of the current event's duration, and if the other observer's event is yet unlinked.

On the second pass (tolerance matches), two events are linked if the difference in onset times between the current event and an identical event in the other observer's record falls within the tolerance window (i.e., is less than or equal to the stated tolerance), and if the other observer's event is yet unlinked.

On the third pass (unlinked events within tolerance), two events are linked if the difference in onset times between the current event and an event in the other observer's record falls within the tolerance window, and if the other observer's event is yet unlinked. If multiple events fall within the tolerance window, then the first is selected.

On the fourth pass (any events within tolerance), two events are linked if the difference in onset times between the current event and an event in the other observer's record falls within the tolerance window, even if the other observer's event is already linked. If multiple events fall within the tolerance window, then the last is selected.

Finally, on the fifth pass (other events), any unlinked events left are linked to the other observer's nearest event—again, even if that event is already linked. In sum, because events may be linked to more than one event, the number of tallies in the agreement matrix—which, in theory, reflects the number of times the pair of observers made a decision—may be greater than the maximum number of events coded by either observer.

The Interact algorithm. The Interact algorithm is a modification of that of The Observer (P. T. Mangold, personal communication, November 14, 2007; C. Spies, personal communication, February 2, 2008); thus, most statements made in the previous section describing the Observer algorithm apply. The Interact algorithm makes six passes through the data; like The Observer algorithm, on each pass, it considers the onset of each unlinked event in turn, ordered from earliest to latest, on the basis of their onset times. Users define a tolerance window and a percentage overlap (*P*). Processing stops after the sixth pass, or possibly earlier if all events are linked.

On the first pass (overlaps), two events are linked if an identical event in the other observer's record overlaps *P*% of the current event's duration, even if the other observer's event is already linked. The second through fourth passes are the same as in The Observer. On the fifth pass (other events), any remaining unlinked events are linked if an event in the other observer's record overlaps *P*% of the current event's duration. On the sixth pass, any remaining unlinked events are linked to a nil event of the other observer (i.e., are regarded as omission–commission errors; see the following paragraph).

The GSEQ dynamic programming algorithm. When coders are asked to detect seams in the stream of behavior (i.e., to unitize), they may be too sensitive and will therefore make errors of commission. Or, they may not be sensitive enough and will thus make errors of omission. As a result, records from 2 observers coding the same material usually contain different numbers of events. Therefore, when comparing 2 observers, some codes in 1 observer's record may be left unlinked to codes in the other observer's record (omission errors from the point of view of the former observer, and commission errors from the point of view of the latter one), and vice versa. The Observer algorithm does not allow for errors of commission and omission; instead, it links all events—even ones that are quite distant in time from each other. As a result, The Observer algorithm likely overestimates agreement. In contrast, both the Interact and GSEQ DP algorithms allow for errors of omission and commission.

As was noted earlier, the GSEQ DP algorithm for timed-event sequences described in the present article is an extension of the one developed for event sequences (Quera et al., 2007). The algorithm described by Quera et al. for ESD is an application of the classic Needleman–Wunsch (1970) algorithm for aligning sequences of nucleotides, whereas the GSEQ DP algorithm for TSD described in the present article is based on Mannila and Ronkainen (1997), who proposed how the Needleman–Wunsch algorithm could be modified for computing similarity between two timed-event sequences, with additional modifications by us. The Needleman–Wunsch algorithm belongs to a broad class of methods known as *dynamic programming*, in which the solution for a specific subproblem can be derived from the solution for another subproblem immediately preceding it. It can be demonstrated that the method guarantees the optimal solution (i.e., it finds the alignment with the highest possible number of agreements between the sequences; Sankoff & Kruskal, 1999, p. 48) without being exhaustive; that is, it does not need to explore all possible alignments (Galisson, 2000).

The algorithm determines the optimal global alignment between two event sequences. It was adapted from sequence alignment techniques that are common in molecular biology to compare and classify nucleotide sequences (see, e.g., Sankoff & Kruskal, 1999). Given two event sequences, many different global alignments are possible. The task is to find an optimal alignment that is guided by costs that the investigator supplies for various transformations, as will be described shortly. The algorithm proceeds step by step, recording the transformations that are required to convert one sequence ($\mathbf{S_1}$) into the other ($\mathbf{S_2}$; bolding indicates a vector, as is the case here, or a matrix). The goal of the algorithm is to determine an optimal alignment—that is, the alignment that requires the minimum possible transformations and yields both the most matches and the lowest distance (defined later) between the two sequences (for details, see Quera et al., 2007). This algorithm is considerably more complex than the two just described; consequently, its description is much longer.

When aligning the sequences, the algorithm either links a code (i.e., an event) in one sequence to a code in the other (this represents an agreement if events are identical, and a disagreement otherwise), or it links an event in one sequence to a nil code, which the algorithm inserts in the other (this represents an omission–commission error) on the basis of the costs that the investigator defines (as will be discussed shortly), and that is mindful of criteria that maximize similarities between the two original sequences. The transformations result in two modified sequences of identical length, which are aligned so that successive pairs each contribute a tally to the agreement matrix. Let k be the number of unique codes in the ME&E scheme under consideration, C_i a particular code (indexed $0 \ldots k$), and $\mathbf{A}$ the $k + 1 \times k + 1$ agreement matrix (indexed $0 \ldots k$) used to tally agreements and disagreements. The first row and column, indexed 0, are used to tally events coded by 1 observer but not the other (i.e., omission–commission errors).

At each step, three transformations are possible:

1. A substitution, which can be either an agreement or a disagreement (a code from $\mathbf{S_1}$ is linked with an identical code from $\mathbf{S_2}$, or a code from $\mathbf{S_1}$ is linked with a different code from $\mathbf{S_2}$).

2. A deletion (a code from $\mathbf{S_1}$ is linked with a nil code from $\mathbf{S_2}$; i.e., a hyphen is inserted in $\mathbf{S_2}$).

3. An insertion (a nil code from $\mathbf{S_1}$ is linked with a code from $\mathbf{S_2}$; i.e., a hyphen is inserted in $\mathbf{S_1}$. Hyphens indicate the nil code and are tallied in the first row or column of the agreement matrix, depending on whether they are inserted in $\mathbf{S_1}$ or $\mathbf{S_2}$, respectively).

From the point of view of the first observer, a deletion is an error of omission, and an insertion is an error of commission on the part of the second observer. Note that the resulting agreement matrix $\mathbf{A}$ contains a logical (or structural) 0 at cell (0,0) because linked nil codes (i.e., simultaneous omissions) are not possible, according to the algorithm. As a consequence, the expected frequencies required by the κ computation cannot be estimated with the usual closed-form formula for κ, but require an iterative proportional fitting (IPF) algorithm instead (see, e.g., Bakeman & Robinson, 1994).

The algorithm considers pairs of codes in turn. As pairs of codes are being aligned, a measure of the distance between the two sequences up to that point is computed. At each step of the dynamic programming algorithm, the transformation selected (out of the three possible transformations) is the one that causes the smallest increment in distance; consequently, two thirds of all possible alignments up to the two codes being checked are discarded, and at the next step, a new optimal alignment is obtained that incorporates the alignment that was obtained at the previous one. Consequently, the algorithm need not exhaustively consider all possible alignments, but only a small fraction of them.

Let m and n be the number of codes in the initial $\mathbf{S_1}$ and $\mathbf{S_2}$. The algorithm proceeds by filling in three $m \times n$ matrices, guided by a fourth $k + 1 \times k + 1$ matrix. The distance matrix ($\mathbf{D}$) accumulates generalized Levenshtein distances; the length matrix ($\mathbf{L}$) accumulates common subsequent lengths (i.e., number of matched codes); and the pointer matrix ($\mathbf{P}$) indicates the transformations used to find the optimal global alignment (again, for details, see Quera et al., 2007). The transformation selected at each step is guided by the distance matrix and the cost or weight matrix ($\mathbf{W}$). Confronted with an apparent pairing of C_i in $\mathbf{S_1}$ and C_j in $\mathbf{S_2}$, a substitution transformation could be selected (first observer coded C_i but the second C_j), or a deletion and an insertion (second observer missed C_i, but the first missed C_j). The transformation selected depends on $\mathbf{W}$, as defined by the investigator, and on the accumulated distance between the sequences up to the two codes currently being checked. $\mathbf{W}$ is indexed $0 \ldots k$, with rows and columns $1 \ldots k$ indicating codes $1 \ldots k$, respectively. Column 0 indicates deletion and Row 0 insertion costs (omissions and commissions); thus, w_{20} is the cost of deleting C_2, and w_{02} is the cost of inserting C_2. Otherwise, diagonal elements (w_{ii}) represent an identity transformation (an agreement; its costs are always 0; i.e., when agreement exists, an identity transformation is always selected), and off-diagonal elements represent disagreements; thus, w_{ij} indicates the cost of substituting C_i with C_j.

For event sequences, Quera et al. (2007) recommended setting agreement, disagreement, and omission and commission costs to 0, 1, and 2, respectively, thus giving omission–commission errors twice the weight of disagreements, reasoning that this best reflects what investigators expect of observer agreement. (In contrast, if costs for disagreements were more than twice the cost of omission–commissions costs, a substitution transformation would never be selected; i.e., all apparent disagreements would be resolved with insertions and deletions, which seems unrealistic.) The contribution of Mannila and Ronkainen (1997) was to suggest that in order to align timed event sequences, the occurrence or onset times of the events can be taken into account when substitution costs are calculated. In addition, we define a tolerance. Discrepancies between onset times that are less than or equal to the tolerance are considered perfect matches (i.e., their cost is 0), otherwise, they are assigned weights that are directly proportional to the discrepancies.

In sum, for the GSEQ DP algorithm, the cost matrix is defined dynamically and depends on onset times. Let $r = 1 \ldots m$ and $c = 1 \ldots n$; then, s_{1r} and s_{2c} identify a pair of codes in $\mathbf{S_1}$ and $\mathbf{S_2}$, respectively. Let h be a tolerance window. Then, the cost of substituting s_{1r} with s_{2c} is set to

$$\begin{array}{ll} w(s_{1r},s_{2c}) = V\cdot(|t_{1r} - t_{2c}| - h) & \text{if } |t_{1r} - t_{2c}| > h \\ w(s_{1r},s_{2c}) = 0 & \text{if } |t_{1r} - t_{2c}| \leq h, \end{array}$$

where t_{1r} and t_{2c} are the onset times of events s_{1r} and s_{2c} and V is a constant, which should be $V \leq \min[w(s_{1r},0) +$

$w(0,s_{2c})$] for all codes (otherwise, a deletion plus an insertion would always be better than a substitution). We can use the same cost for applying a tolerance window both when $s_{1r} = s_{2c}$ and when $s_{1r} \neq s_{2c}$; that way, substitution of one code for a different one is possible only if the difference between their onset times is less than the tolerance. We propose setting insertion and deletion (indel) costs to 1 and V to 2. For example, if we set $h = 5$, then a substitution is preferable over an indel if the difference between the onset times is less than or equal to 5, and an indel is preferable if it is greater than 5. For other details as to how the **D**, **L**, and **P** matrices are filled in, how Levenshtein distance is defined, and how the final alignment is determined by a backward trace through the **P** matrix, see Quera et al. (2007).

We made one additional modification to the GSEQ DP algorithm. Imagine that the first observer recorded Code A for 20 sec and then Code B for 20 sec, whereas the second observer recorded Code B for the entire 40 sec. Left unmodified, the dynamic programming algorithm would tally this as two disagreements (assuming $h < 20$): a substitution that increments cell a_{12} in the agreement matrix, followed by a deletion that increments cell a_{20} (first observer coded a B that the second observer did not). In fact, an examination of the timeline suggests a different scenario. True, when the first observer began by coding A and the second by B, they disagreed. But, after 20 sec, the first observer "decided" to code B, and the second "decided" to continue coding B; this is an agreement that time-unit κ tally would capture, and, it seems reasonable to think, one that other algorithms should as well. Consequently—during the backward trace, and before tallying a potential deletion or insertion—in a manner similar to Interact's Pass 1, the GSEQ DP algorithm asks whether an identical event in the other observer's record overlaps P% of the proposed insertion or deletion event's duration. If so, an agreement is tallied instead of an omission or commission disagreement. In such cases, the GSEQ DP algorithm may link one event to two others.

METHOD

The OASTES Simulation Program

To compare the five algorithms just described, we developed a computer program that models the behavior of two independent coders. The program, which we call OASTES (for Observer Agreement for Simulated Timed Event Sequences), was programmed in Pascal (using Borland Delphi Professional Version 7.0). As will be described shortly, various characteristics can be specified. For each unique combination of characteristics, OASTES generates one or more records of events for a hypothetical session. These are called "master records" because they contain a presumed "correct" sequence of events and their durations. For each master record, the coding behavior of one to three pairs of independent observers is simulated. The user specifies the number of pairs (may be from 1 to 3) and a percentage accuracy for the observers in each pair (may be from 50% to 100%). For each pair, with each representing a different level of accuracy, κs per the five algorithms are computed. For each combination of characteristics, as many master records are generated as the user requests (the number of replications may vary from 1 to 10,000). Then, for stability, values of κ for each algorithm are averaged over the number of replications specified.

Procedure for Generating Sessions to Code

When generating master sessions, five characteristics can be specified: the number of codes (k) in the ME&E set under consideration, the variability of their relative frequencies and durations (low, medium, or high), the mean duration of coded events, the mean variability of that duration, and the length of the session (time units are assumed to be seconds). The variable k can vary from 2 to 20—a range in which the number of codes in ME&E sets used by researchers typically fall.

Codes are rarely equiprobable with equivalent mean durations; thus, we defined three sets of circumstances that are intended to reflect the kind of variability encountered in practice. For each value of k, we defined three possible patterns: low, medium, and high variability. Let C_i be a code (where $i = 1 \ldots k$) and R_i its relative frequency—that is, the probability that an event will be coded C_i. These probabilities are used when generating master records (and simulating observers). Define $R_0 = 1/k$; if codes were equiprobable, then all $R_i = R_0$. For low, medium, and high variability, respectively, we set $R_1 = 0.75R_0$, $0.50R_0$, and $0.25R_0$; and $R_k = 1.25R_0$, $1.50R_0$, and $1.75R_0$. We let other R_i assume appropriate intermediate values. For example, for $k = 5$ the probabilities used to generate sessions for the five codes varied from .15 to .25, .10 to .30, and .05 to .35 for low, medium, and high variability, respectively.

When generating data for a session, OASTES selects successive events using a probabilistic random process. The user specifies whether a selected event can be the same as the previous one (*cannot* is the default); otherwise, the probabilities used to select events are the relative frequencies (R_is, as was defined in the previous paragraph) associated with the current variability level and value of k. Each event's duration is likewise determined using a probabilistic random process, which will be described in the following paragraph. By default, the base mean duration (M_0) is set to 20 sec (it can vary from 10 to 100), but the mean duration for each code (M_i) varies depending on the variability level and value of k. Specifically, $M_i = M_0/R_ik$. Per this definition, less probable codes have longer average durations; thus, other things being equal, about the same number of events will be generated for sessions of different variability levels. Consequently—and this is why we defined variability in this way—results for sessions of different variability levels can be directly compared. For example, for $k = 5$ and $M_0 = 20$, the mean durations used to generate sessions for the five codes vary from 26.7 to 16.0, 40.0 to 13.3, and 80 to 11.4 sec for low, medium, and high variability, respectively.

The duration for each event is randomly selected from a normal distribution whose mean is the M_i for the selected code, as was defined in the previous paragraph, and whose standard deviation is set by default to $.20M_i$ (e.g., if $M_i = 20$, then $S_i = 4$; it can vary from 0% to 50%). Any durations that are less than 3 are set to 3 sec, a minimum time that is often used for coding events (see, e.g., Adamson, Bakeman, & Deckner, 2004). OASTES stops generating data for a session when the accumulated time exceeds a specified total duration (can vary from 60 to 3,600 sec; given a mean event duration of 20 sec, about 45 events will be generated for a 900-sec, or 15-min, session).

Procedure for Simulating Observer Coding Behavior

Accuracy (A) is the key parameter when simulating an observer coding a session, but the parameters defined in the previous section are also used (the R_i, M_i, and S_i for the value of k and level of variability for the session being coded). OASTES generates an observer's record by selecting successive events and their durations from the master record using a random probabilistic process; the user specifies whether or not a selected event can be the same as the previous one (*cannot* is the default), and coding stops when the accumulated time exceeds that of the master record.

Successive events are selected by first noting the concurrent code in the master record (defined as the code for the time unit or column in the master record corresponding to the first yet uncoded column in the observer's code-time grid, or the next code in the master record if the current one ends within 3 sec). OASTES gives the observer an A% chance of selecting the concurrent code in the master record. If it is not selected, then the probabilities used to select events are the R_i-defined probabilities for the master records' variability levels and values of k.

The duration for each event is randomly selected from a normal distribution whose mean is the time remaining before a new code begins in the master record, and whose standard deviation is smaller for more accurate observers, with the additional constraint that no duration can be less than 3 sec. Specifically, the standard deviation for the selected code is multiplied by $\sqrt{1 - A}$; this factor was based on a pilot study that showed it gave more reasonable results than did, for example, $1 - A$. We reasoned that accurate observers would be likely to detect an end to the current event in the master record, but that how closely they detected it would be affected by their accuracy.

RESULTS

We ran the simulation program specifying three values of k (5, 10, and 15), three levels of variability (low, medium, and high), three levels of observer accuracy (75%, 85%, and 95%), a base mean duration of 20 sec with an *SD* of 4 sec, a total duration of 900 sec (15 min), and a tolerance of ±2 sec for $\kappa_{\text{tolerance}}$ and of 5 sec for The Observer, Interact, and GSEQ DP algorithms (reasoning that this provided an equivalent window as the ±2 sec for time-unit κ). Overlap was 80% for the Interact and GSEQ DP algorithms.

Simulation Results

Agreement statistics, averaged over 1,000 simulations, for $k = 5$, 10, and 15, are shown in Figure 1. For observers who are 75%, 85%, and 95% accurate individually, we would expect their joint accuracy to be the product: 56%, 72%, and 90%. The average percentage agreements for time intervals over the circumstances simulated were 55%, 70%, and 87%. Thus, from a time-based perspective at least, the simulation program delivered about the percentage agreement expected. Moreover, average values of κ over the circumstances simulated for $\kappa_{\text{time-unit}}$ and $\kappa_{\text{tolerance}}$, and The Observer, Interact, and GSEQ DP algorithms ranged from .47 to .86, .52 to 91, .52 to .91, .44 to .90, and .43 to .88, respectively, for 75% to 95% observer accuracy. Thus, over all algorithms, values of κ were generally in the range that might be expected for the accuracies simulated.

Generally, $\kappa_{\text{tolerance}}$ and The Observer κs tended to be higher, $\kappa_{\text{time-unit}}$ and GSEQ DP values lower, and Interact κs intermediate: Mean values over the 27 circumstances simulated for $\kappa_{\text{time-unit}}$ and $\kappa_{\text{tolerance}}$, and The Observer, Interact, and GSEQ DP algorithms, were .66, .72, .72, .68, and .65, respectively. With 95% accuracy, differences among the algorithms were relatively muted (.85–.90). With 85% accuracy, differences were greater (.64–.71), and with 75% accuracy, differences were greater still (.45–.55). Over all algorithms, the number of codes (k) had little effect. Variability had little effect on time-based κs, but for the event-based algorithms, higher variability was associated with lower values of κ (see Figure 1).

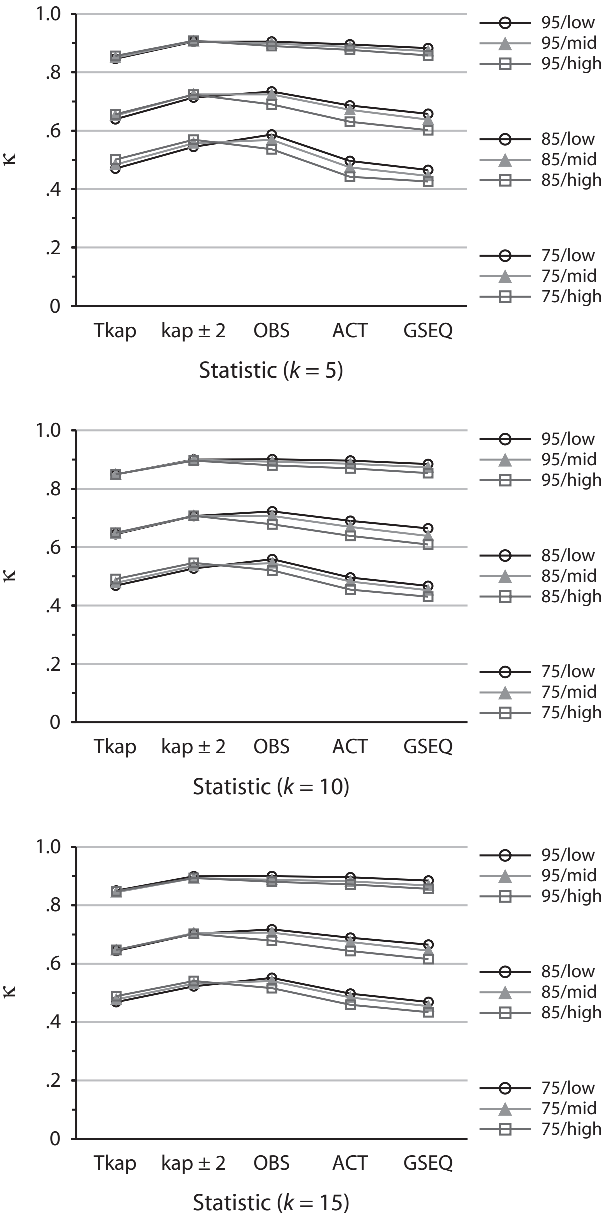

Figure 1. Values for time-unit κ and time-unit κ with 2-sec tolerance, as computed per The Observer, Interact, and GSEQ dynamic programming algorithms for $k = 5$, 10, and 15 codes when observer accuracy is 75%, 85%, and 95%, and when the variability of code frequency and duration is low, moderate, and high.

Finally, we simulated the results when adjacent codes both could and could not be equal (an option in OASTES), but doing so generated little difference in results. Over the circumstances simulated, time-based κs were similar and event-based κs were slightly higher, when adjacent codes could not be the same (.011 for The Observer, .004 for the Interact and GSEQ DP algorithms).

An Example

To illustrate the five agreement algorithms for TSD, we prepared a κ table (i.e., an agreement or confusion matrix) and computed κ per each algorithm, given the data shown in Table 1. For simplicity, the example assumes five codes and 2 observers who independently coded a 5-min

Table 1
Agreement Records for Two Observers for a 5-min Example Session

Observer 1					Observer 2				
#	Code	Onset	Offset	Dura	#	Code	Onset	Offset	Dura
1	A	1	4	4	1	D	1	14	14
2	D	5	12	8	2	E	15	25	11
3	E	13	31	19	3	D	26	41	16
4	D	32	44	13	4	B	42	68	27
5	B	45	63	19	5	C	69	76	8
6	C	64	82	19	6	A	77	112	36
7	E	83	92	10	7	C	113	116	4
8	C	93	109	17	8	A	117	157	41
9	E	110	123	14	9	C	158	174	17
10	C	124	146	23	10	B	175	201	27
11	D	147	157	11	11	D	202	215	14
12	B	158	198	41	12	C	216	238	23
13	D	199	217	19	13	A	239	274	36
14	C	218	236	19	14	E	275	290	16
15	A	237	284	48	15	A	291	300	10
16	C	285	300	16					

Note—Onset, offset, and duration (dura) times are in seconds. Offset times are inclusive.

(300-sec) session. We deliberately selected example data that represent less than stellar agreement; doing so better demonstrates how the algorithms work (the example data were selected from pairs of observer records generated by the OASTES program with $k = 5$, moderate variability of code frequencies and durations, and observer accuracy of 75%). The tolerance for $\kappa_{\text{tolerance}}$ was 2 sec. For The Observer, Interact, and GSEQ DP algorithms, it was 5 sec (for the reasons given earlier), and the overlap for Interact and GSEQ DP was 80%. IPF was used to compute expected values for Interact and GSEQ DP κs because of the structural 0 at cell (0,0).

Consistent with the simulation results, the value for $\kappa_{\text{time-unit}}$ was .08 less than the value for $\kappa_{\text{tolerance}}$. As shown in Figure 2 (top), given a 2-sec tolerance, 20 of the 300 1-sec tallies moved to the diagonal, giving a value of .45 for κ with tolerance, as compared with .37 without. Still, the cells with the largest tallies in the κ tables changed little, if any. This indicated, for example, that the first observer coded at least 54 sec as Code C when the second coded them as Code A.

The agreement and disagreement linkages effected by each of the three event-based algorithms are shown graphically in Figure 3. As shown in Figures 2 and 3, The Observer found 9 agreements and 8 disagreements; comparable numbers for Interact and GSEQ DP were 8 and 11, and 9 and 10, respectively. The Observer and GSEQ DP found the same 9 agreements. Unlike the Observer and GSEQ DP, Interact did not link the first observer's Code D at 32–44 sec with the second's Code D at 26–41 sec inclusive (hereafter Code $D_{1,4}$ and Code $D_{2,3}$, where the first subscript indicates observer and the second, serial position), because neither overlapped the other 80% or more. Interact left these two events unlinked until Pass 6; it also left Code $E_{2,14}$ unlinked until Pass 6, whereas The Observer linked it to Code $C_{1,16}$ (a Code C–E disagreement), and GSEQ DP regarded it as a commission–omission error (the second observer coded it, but the first observer did not). For other events as well (see Figure 3), GSEQ DP indicated omission–commission errors that both The Observer and Interact regarded as disagreements, although they sometimes differed as to what the disagreements were. To our knowledge, only the INTERACT program prepares linkage figures like those shown in Figure 3.

In sum, the GSEQ DP κ table (Figure 2, bottom) indicates possible omission–commission errors, which are absent from The Observer table and differ from those in the Interact table, but which can be useful when training observers. On the other hand, both The Observer and GSEQ DP found an agreement that Interact—with an 80% overlap criterion—regarded as two disagreements. Figure 3 illustrates differences in how the three event-based algorithms linked, or did not link, events for the data in Table 1—differences that are reflected in the Figure 2 κ tables. In Figure 3, a solid line connecting two events indicates agreement; a dotted line connecting two events indicates disagreement; a dotted line connected to the top of the figure indicates an event coded by Observer 1 but missed by Observer 2; and a dotted line connected to the bottom of the figure indicates an event coded by Observer 2 but missed by Observer 1.

DISCUSSION

In an earlier article, and in the context of event sequential data, Bakeman, Quera, McArthur, and Robinson (1997) argued that no one value of κ can be regarded as universally acceptable. The present article supports that earlier conclusion for the context of TSD. For example, bigger values of κ are not, for that reason, necessarily better. Still, given TSD, and the need to train observers and provide them with useful feedback, which of these algorithms should an investigator favor? Of the two time-based algorithms, we prefer $\kappa_{\text{tolerance}}$, because we think it reasonable for most behaviors of interest to behavioral investigators not to count minor errors of timing on the order of just a few seconds; moreover, the event-based al-

Observer 2's Codes

Observer 1's Codes	A	B	C	D	E
A	36	0	2	4	10
B	0	43	17	0	0
C	56	5	27	0	6
D	11	6	2	32	0
E	20	0	4	8	11

$\kappa_{\text{time-unit}} = .37$ 300

Observer 2's Codes

Observer 1's Codes	A	B	C	D	E
A	40	0	0	4	8
B	0	45	15	0	0
C	54	3	31	0	6
D	11	2	0	38	0
E	20	0	4	4	15

$\kappa_{\text{tolerance}} = .45$ 300

Observer 2's Codes

Observer 1's Codes	A	B	C	D	E
A	1	0	0	1	0
B	0	2	0	0	0
C	3	0	2	0	1
D	0	0	1	3	0
E	1	0	1	0	1

The Observer $\kappa = .41$ 17

Observer 2's Codes

Observer 1's Codes	nil	A	B	C	D	E
nil	–	0	0	0	1	1
A	0	1	0	0	1	0
B	0	0	2	1	0	0
C	0	3	0	2	0	0
D	1	1	0	0	2	0
E	0	1	0	1	0	1

Interact $\kappa = .31$ 19

Observer 2's Codes

Observer 1's Codes	nil	A	B	C	D	E
nil	–	1	0	0	0	1
A	1	1	0	0	0	0
B	0	0	2	1	0	0
C	2	1	0	2	0	0
D	1	0	0	0	3	0
E	0	1	0	1	0	1

GSEQ DP $\kappa = .39$ 19

Figure 2. Agreement matrices and κ values for the timed-event sequential data in Table 1 per the five observer agreement algorithms.

gorithms all include some sort of tolerance. Additionally, eliminating such errors from the agreement matrix leaves those disagreements that are arguably more serious, and that can profitably serve as a basis for further observer training.

Of the three event-based algorithms, we believe the GSEQ dynamic programming algorithm is more accurate. The Observer algorithm does not allow for errors of omission and commission. It regards two events, even when quite distant in time, as a single disagreement instead of two separate errors—one of commission and one of omission—and so is likely to overestimate agreement. Thus, it is not surprising that The Observer algorithm produced higher κ values than did either the Interact or the GSEQ DP algorithm in our simulations. We should stress, however, that our results are limited to the circumstances we simulated, and other values for the simulation parameters could give different results. We set simulation parameters to values that made sense to us, but individual circumstances vary, which is why we are making OASTES available to investigators who wish to consider parameter values other than those we chose for investigation.

In addition to providing relatively conservative values, we think the GSEQ DP algorithm has another advantage. The Needleman–Wunsch algorithm, on which the GSEQ DP algorithm is based, is conceptually sophisticated and has a firm basis in the literature. Its conceptual sophistication is reflected in an apparent paradox: Although it required considerably more prose to describe, its programming required fewer lines of computer code.

One dilemma remains. A time-unit κ (with a tally for each time unit) likely overestimates how often observers are making decisions, whereas event-based algorithms (with a tally for each agreement and disagreement, and perhaps each omission and commission) likely underestimate the number of decisions observers make. Sometimes (perhaps often) observers decide that an event is continuing and not changing to another event; such agreements

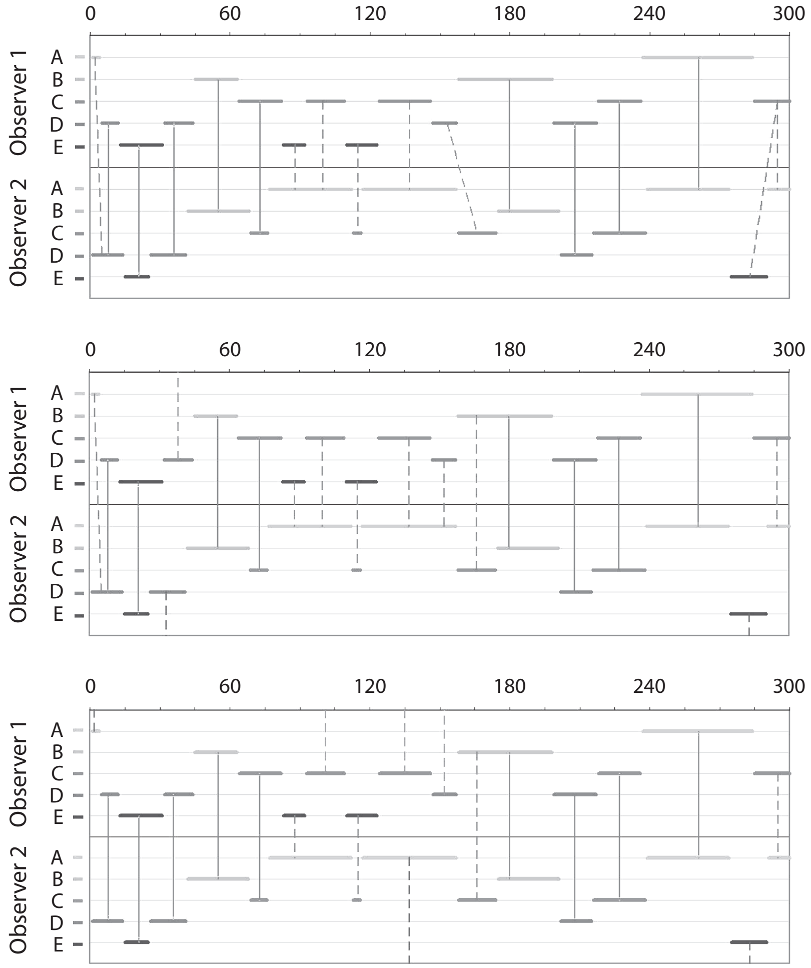

Figure 3. Agreement (solid lines) and disagreement (broken lines) linkages for The Observer (top), INTERACT (middle), and the GSEQ dynamic programming (bottom) algorithms for observer agreement, given the timed-event sequential data in Table 1.

are not counted by the event-based algorithms. Indeed, how often these private mental events occur is unknowable, although to a limited extent, the Interact and GSEQ DP algorithms (by including percentage overlaps) attempt to reflect these sorts of decisions. Still, as was noted earlier, the observer decision model for both the time-based and event-based algorithms lacks the one-to-one correspondence between observer decision and tally of the classic Cohen model.

We conclude with a simple recommendation: not either–or, but both. We recommend that investigators report values for both a time-unit κ (preferably $\kappa_{\text{tolerance}}$) and an event-based κ (preferably the GSEQ DP one, if for no other reason than its more conservative values); their range likely captures the "true" value of κ. Similarly, we recommend that investigators provide observers with agreement matrices for both a time-unit and an event-based κ. Each provides somewhat different (time-based vs. event-based) but valuable information as to how observers are disagreeing, and are thus useful in different ways as observers strive to improve their agreement.

Simulation Program Availability

Users who wish to explore values other than those reported here may download a zip file containing a description of the OASTES program and the program itself at no cost from the authors' Web sites (www.gsu.edu/~psyrab/BakemanPrograms.htm and www.ub.es/comporta/vquera). The program was written in Pascal using Borland Delphi Professional Version 7.0 and assumes a Windows 95 or later environment. OASTES lets users set various parameter values, display intermediate results, and even read in their own data, if desired.

AUTHOR NOTE

Correspondence concerning this article should be addressed to R. Bakeman, Georgia State University, Department of Psychology, P.O. Box 5010, Atlanta, GA 30302-5010 (e-mail: bakeman@gsu.edu).

REFERENCES

Adamson, L. B., & Bakeman, R. (2006). Development of displaced speech in early mother–child conversations. *Child Development*, **77**, 186-200. doi:10.1111/j.1467-8624.2006.00864.x

Adamson, L. B., Bakeman, R., & Deckner, D. F. (2004). The development of symbol-infused joint engagement. *Child Development*, **75**, 1171-1187. doi:10.1111/j.1467-8624.2004.00732.x

Bakeman, R., Adamson, L. B., Konner, M., & Barr, R. G. (1990). !Kung infancy: The social context of object exploration. *Child Development*, **61**, 794-809. doi:10.2307/1130964

Bakeman, R., & Gottman, J. M. (1997). *Observing interaction: An introduction to sequential analysis* (2nd ed.). New York: Cambridge University Press.

Bakeman, R., & Quera, V. (1995). *Analyzing interaction: Sequential analysis with SDIS and GSEQ*. New York: Cambridge University Press.

Bakeman, R., Quera, V., McArthur, D., & Robinson, B. F. (1997). Detecting sequential patterns and determining their reliability with fallible observers. *Psychological Methods*, **2**, 357-370. doi:10.1037/1082-989X.2.4.357

Bakeman, R., & Robinson, B. F. (1994). *Understanding log–linear analysis with ILOG: An interactive approach*. Hillsdale, NJ: Erlbaum.

Cohen, J. A. (1960). A coefficient of agreement for nominal scales. *Educational & Psychological Measurement*, **20**, 37-46. doi:10.1177/001316446002000104

Galisson, F. (2000, August). *Introduction to computational sequence analysis*. Tutorial presented at the Eighth International Conference on Intelligent Systems for Molecular Biology, San Diego, CA.

Haccou, P., & Meelis, E. (1992). *Statistical analysis of behavioral data: An approach based on time-structured models.* Oxford: Oxford University Press.

Hollenbeck, A. R. (1978). Problems of reliability in observational data. In G. P. Sackett (Ed.), *Observing behavior: Vol. 2. Data collection and analysis methods* (pp. 79-88). Baltimore: University Park Press.

Jansen, R. G., Wiertz, L. F., Meyer, E. S., & Noldus, L. P. J. J. (2003). Reliability analyis of observational data: Problems, solutions, and software implementation. *Behavior Research Methods, Instruments, & Computers*, **35**, 391-399.

Konner, M. J. (1978). Maternal care, infant behavior, and development among the !Kung. In R. B. Lee & I. DeVore (Eds.), *Kalahari hunter-gatherers: Studies of the !Kung Sari and their neighbors* (pp. 218-245). Cambridge, MA: Harvard University Press.

Mangold, P. (2006, September). *Getting better results in less time: When using audio/video recordings in research applications makes sense*. Paper presented at the Third Congress of the European Society on Family Relations, Darmstadt, Germany.

Mannila, H., & Ronkainen, P. (1997, May). Similarity of event sequences. In *Proceedings of the Fourth International Workshop on Temporal Representation and Reasoning* (pp. 136-139). Daytona Beach, FL.

Needleman, S. B., & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, **48**, 443-453.

Noldus Information Technology (2003). *The Observer: Professional system for collection, analysis, presentation and management of observational data* [Reference Manual, Version 5.0]. Wageningen, The Netherlands: Author.

Quera, V., Bakeman, R., & Gnisci, A. (2007). Observer agreement for event sequences: Methods and software for sequence alignment and reliability estimates. *Behavior Research Methods*, **39**, 39-49.

Sankoff, D., & Kruskal, J. (Eds.) (1999). *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison* (2nd ed.). Stanford, CA: CSLI Publications.

Suen, H. K. (1988). Agreement, reliability, accuracy, and validity: Toward a clarification. *Behavioral Assessment*, **10**, 343-366.

(Manuscript received February 25, 2008;
revision accepted for publication July 22, 2008.)