Running head: ANALYSIS OF BEHAVIORAL STREAMS

Analysis of Behavioral Streams

Roger Bakeman and Deborah F. Deckner

Georgia State University

Vicenç Quera

University of Barcelona

Chapter to appear in: Teti, D. M. (Ed.), *Handbook of Research Methods in Developmental Psychology*. Oxford, UK: Blackwell Publishers.

Analysis of Behavioral Streams

Now that you, the reader, have found your way to this page, we wonder, what do you want us to be, what, if it is in our power, should we be: chronicler, sage, or seer? The word handbook appears in the title of this volume, so this might be your first stop, shopping for a complete overview, an encyclopedic chronicle of methods appropriate to the analysis of behavioral streams. Research methods also appears in the title, so you might expect some sage and methodical selection, a cutting presentation of the latest, ground-breaking methods. Still, seer may be the most challenging, and it is the path we plan to try. If our mindreading is successful, we should be able to see what brought you here and guess what you hope to find.

Our assumptions are few. Like us, you are interested in developmental psychology. You are a researcher, either seasoned or novice, who suspects that there is something to be gained from a more systematic analysis of behavior as it unfolds in time. Perhaps you are a graduate student whose advisor has sent you to this chapter, or you are advanced in rank but want to consider some new approaches; perhaps you are part of a larger study or research team who has been asked to figure out whether adding an observational component would be worth while. You may be adept with other forms of measurement, including structured interviews and standardized tests, but have not yet had much experience with the technical matters and techniques the coding and analysis of behavioral streams requires.

If our assumptions are right, you are looking for a map, and that is what we hope to provide. In this chapter, we take a narrative approach. To keep matters concrete and relatively simple, our examples are drawn primarily from two sources, a recently-published article (Lavelli & Poli, 1998) and the research in which we are currently involved with Lauren Adamson. We consider, step by step, how a research project concerned with observing and analyzing streams of behaviors itself unfolds. In so doing, we explicate in some detail many of the tools and techniques required for such an endeavor (e.g., the GSEQ computer program; Bakeman & Quera, 1995a). Not all of our examples may apply directly to your work, but our intent in providing specific details is to help you visualize better how work of this sort proceeds. (For further discussion of observational studies, see Bakeman & Gottman, 1997, and Bakeman, 2000a.)

This chapter is divided into seven sections representing seven steps. The division is somewhat arbitrary but none-the-less allows readers who are more familiar with some issues to skim or skip those sections while focusing more on others. The overall, brief summary of this chapter might be stated as follows. The systematic observation of behavior and its subsequent analysis is useful when investigators wish to capture aspects of relatively naturally occurring behavior as it unfolds in time. It is especially useful when process or dynamic aspects of that behavior are under study. It is often used for, but is not limited to, nonverbal behavior or organisms who are nonverbal (i.e., human infants and non-humans). At root, systematic observation is a measurement tool. Like standardized tests, it is another way of applying procedures to behavior to produce scores. The measuring instruments include a human component (i.e., trained observers assigning codes to events) and, as a result, matters of reliability are of considerable importance. Moreover, the data initially collected can be quite voluminous, thus data reduction and the derivation of indices from initial data are key considerations. Once data are reduced, any number of standard statistical techniques might be invoked. In that sense, this chapter is not an alternative to other chapters in Section IV of this book that focus on analytic techniques, but could serve as a precursor to them. Similarly, the chapters on developmental designs in Section I of this volume might be considered as precursors to this chapter. In sum, in this chapter we focus primarily on concepts and tools that aid in extracting scores from observed streams of behavior; such scores then serve subsequent analyses.

Define Basic Questions and Concepts

Perhaps the most oft repeated research advice is, begin with a good question, which includes clear understanding and definitions of the concepts that the questions reference.

For example, a selective list of some of the concepts that have animated research in developmental psychology over the past few decades might include attachment, temperament, and synchrony and reciprocity in social interaction. As a first example, Lavelli and Poli (1998), who emphasized the importance of newborn feeding as the first instance of social interaction, asked whether patterns of mother-infant interaction during and after breast-feeding differed from those during and after bottle-feeding, and whether any such differences changed as infants aged from 3 days to 3 months. These investigators conceptualized early mother-infant interaction as representing the first occurrence of mother-infant dialogue, emphasized its synchrony, and focused on such behaviors, among others, as mutual touches, gazes, smiles, and vocalizations.

As a second example, in earlier work with Adamson (e.g., Adamson & Bakeman, 1984, 1991; Bakeman & Adamson, 1984), joint attention was a central concept, usually defined as a state of attention shared between a mother (or any caregiver; we often use the word mother in its generic sense) and her infant, during which both shared their attention to a common object or event. Our concern with joint attention, and the way we conceptualized it, led us to develop our first joint attention coding scheme, which we describe shortly. In our current work (e.g., Adamson, Bakeman, Deckner, & Dunbar, 2000), we are focusing on the way language begins to infuse and transform joint attention just after infancy. By the end of infancy, a typically developing child usually has mastered the rudiments of conversation. We view this as an astonishing feat that entails, among other skills, the ability to engage with a social partner about a topic of mutual interest using a wealth of communicative acts including language. We term this feat *symbol-infused joint engagement*, and our concern with its development has led us to observe slightly older children (in the second and third years of life as opposed to the first and second years) and to modify our first coding scheme, again in ways we describe shortly.

Notions of joint attention or early mother-infant interaction as dialog are, of course, only two of a universe of concepts that might drive research. Questions for the reader include, what concepts underlie your research, how are they defined, and, perhaps most critically for present purposes, do they reference behavior that can be captured on a recording medium such as videotape? On playback, could you point to the screen and say, there, that is what I am talking about. If so, systematic observational methods might be useful to you.

Develop Coding Schemes Tied to Questions and Concepts

As mentioned earlier, we view coding schemes as the thermometers or rulers (i.e., the measuring instruments) of systematic observation, albeit ones that include a human component. Coding schemes are guided by the concepts that undergird the research and define the behaviors of interest in sufficient detail that observers could say with some assuredness, yes, they just saw an instance of a particular behavior defined by the scheme. Codes are often grouped into sets that, within the set, are mutually exclusive and exhaustive, that is to say, only one code applies to any thing coded (mutually exclusive) and some code applies to every thing coded (exhaustive); for example, we might code mothers as looking at their infant or not looking at their infants. Recording and analyzing codes defined and grouped into mutually exclusive and exhaustive sets often turns out to be quite useful, which is why such sets are frequently encountered. The term *coding scheme* can refer to a single set of mutually exclusive and exhaustive codes, or collectively to several sets of such codes, or to a group of codes, whether or not mutually exclusive and exhaustive, that are applied during one pass through a video tape (e.g., one pass might code a mother's behavior and another might code her infant's behavior).

Although we are getting ahead of our story a bit, measurement occurs when a particular code is assigned to a particular entity in a systematic way. Usually that entity is an event of some sort (e.g., mother gazing at infant), although certainly codes can be assigned to objects as well (Is it a red ball or a blue one?). For now, let us use the term *event* generically for the thing coded, although later we mention that time units could be coded as well. A useful distinction that we and others make (e.g., Altmann, 1974; Sackett,

1978) is between *momentary* and *duration events*. A momentary event may or may not last just a moment, but, by definition, its duration is not of interest to us and so we record only its occurrence, and perhaps its time of onset. Momentary codes can be thought of as answers to yes/no questions: Did the behavior of interest occur or not. By contrast, the amount of time devoted to duration events is of interest, and so we note how long they last, typically recording times of onset as well as times of offset.

One way of thinking about duration events is both so common and so useful that it goes by its own name, and so we talk about coding *behavioral states*. The usual assumption is that states reflect some underlying organization, so that an organism at any given time necessarily must be in one of a set of mutually exclusive and exhaustive states. One example would be the infant states of quiet alert, fussy, crying, rapid-eye movement sleep, and deep sleep (Wolff, 1966). Another would be the play states of unoccupied, onlooker, and solitary, parallel, associative, and cooperative play applied to preschoolers by Parten (1932). Not surprisingly, these distinctions apply to codes as well and so, just as we speak of momentary, duration, and state behaviors, so too we speak of momentary, duration, and state codes.

The coding schemes used by Lavelli and Poli (1998) illustrate these notions nicely. As mentioned earlier, their coding schemes were motivated by a desire to explore differences in mother-infant interaction during breast- as compared to bottle-feeding. They coded four kinds of behavior: infant state, infant’s feeding behavior, maternal behavior, and mother-infant interaction, which we present here in slightly simplified form. Following Wolff (1966) and Brazelton (1973), they defined five mutually exclusive and exhaustive states for infant state: sleeping, drowsy, alert, fussing, and crying. Another set of codes defined feeding behavior: Coders noted whether or not a feeding episode was ongoing and if so whether the infant was sucking or not. Five additional codes were used to describe behavior when the infant was not actively sucking including a cannot-observe code. Such a can’t-see off-task code is often used to make a set exhaustive. Two sets of codes were used for maternal behavior. The first coded whether or not the mother was gazing at her infants. The second coded whether the mother was using tactile, auditory, or both tactile and auditory stimulation. Defining such combination codes (both tactile and auditory stimulation) is a common way to make codes mutually exclusive and, in this case, adding a fourth code (none of the above) would make the set exhaustive as well. Finally, Lavelli and Poli coded four dyadic, interactive behaviors: mutual touch, mutual gaze, mutual smile, and mutual vocalization whose occurrence or frequency, but not duration, was of interest to them; thus we would regard them as momentary behaviors and not states.

In our own case, when we first began thinking about joint attention, the notion of state seemed to fit. The ages of the infants we observed (6-18 months) fell between those of Wolff and Parten. Moreover, some of Parten's codes seemed to apply. True, we were observing infants with their mothers whereas Parten observed preschoolers aged not quite 2 to almost 5 with each other but, like Parten, we wanted to characterize the child's engagement. After a bit of pilot work, we defined six codes. The first three came from Parten: *unengaged*, *onlooking*, and *objects* (solitary play with toys). The fourth, engagement with a *person*, usually the mother (no objects involved), was simply something afforded by the situation. The last two codes followed directly from the way we thought about joint attention. During infancy, the emergence of joint engagement appears not as a singular achievement but as a series of accomplishments. Broadly, the typical developmental progression begins with the two-to-three month old establishing ways to interact with a social partner and then rapidly transforms as the four-to-five month old becomes fascinated with handling objects. At this point, mothers may join the infant in exploring objects and in that sense the objects become shared. But this *supported joint engagement* does not require active joint attention to both object and mother from the infant. It is not until later in the first year and the beginning of the second that infants begin to coordinate attention to their mother and to an object close at hand, an accomplishment which we term *coordinated joint engagement*.

The description of the development of joint attention in the last paragraph is a paraphrase from a recent talk (Adamson et al., 2000); it both describes the six codes we used and summarizes what we learned from our research using those codes. When we were developing the coding scheme, we would not have been able to express ourselves quite this way; in fact, in our earlier work we used the term, passive joint, instead of supported joint (e.g., Bakeman & Adamson, 1984), but changed the term once we realized that infants were hardly passive during what we now call supported joint engagement. We report this bit of history to make a couple of points. First, coding schemes need to be informed by theory and incorporate theoretically important distinctions, such as the distinction between supported and coordinated joint engagement. When we made this distinction initially, we hypothesized a developmental progression, which now, years later, we can confirm. Second, coding schemes often evolve and change, sometimes due to experience (e.g., some theoretically compelling behavior occurs rarely or cannot be coded reliably) or, as here, due to reflection as the eye and mind are tutored by examples of the coded behavior.

The coding scheme that is featured prominently in our current research represents a straightforward evolution of the one just described. It reflects our concern with how symbols, especially linguistic ones, begin to infuse the joint engagement of older infants and young children. After a bit of pilot work—always a necessary step when developing any coding scheme—we became convinced that adding five codes to our existing scheme would capture the phenomenon of symbol-infused joint engagement that we wished to study. In addition to the simple codes of object and person engagement, we added a code for symbol engagement; and in parallel with object and person engagement, we added two new codes for symbol-infused object-engagement and symbol-infused person engagement. These three codes seemed necessary to complete the logical possibilities, but we expected that they would occur fairly infrequently. Of considerable more theoretical interest to us were the two new codes for symbol-infused supported joint engagement and symbol-infused coordinated joint engagement.

In the preceding paragraphs, we have given a few examples of coding schemes and, in our own case, have tried to show how our codes evolved from and reflected theoretical considerations of interest to us. We have provided the names of codes but, in the interest of brevity, have not given more extended definitions. Be assured, however, that investigators who used these coding schemes labored to develop coding manuals that contained clear definitions, objective rules, and helpful examples for their coders. Other examples of coding schemes are given in Bakeman and Gottman (1997) and Bakeman (2000a), and countless others can be found in the literature. Probably the best guide for you, as an individual researcher, is to ask, what coding schemes do others researching phenomena similar to those of interest to you use; do you find the codes they use natural and compelling and clearly organized; can you see clear links between research question, underlying concepts, and codes? If your answers to these questions are affirmative, chances are good that such coding schemes, or some adaptation of them, might serve you well.

Record and Code Behavioral Streams

Once design decisions are made (see Section I of this volume) so that you know who you will observe, and how often and in what context you will observe them, and once you have settled on a coding scheme or schemes, then a series of practical and technical questions arise related to how coders will be trained and how they will record their decisions. We assume you have decided to embark on systematic observation of behavioral streams and now need to put together the human, mechanical, and electronic components of your measuring apparatus.

Your choices are many, but most likely the sort of equipment you deploy will reflect what other researchers you know use and have found satisfactory, the amount of resources you have to devote to the project, not least of which is the amount of time you have to spend, any unusual demands of the context in which you observe, and your own idiosyncratic preferences. It is possible that you will observe behavior live, in real time. In such cases, your choices for recording behavior include, in order of increasing technical complexity: simple pencil and paper; an audio recorder in which you speak codes for later transcription; some sort of hand-held electronic device into which you key codes (several are commercially available, e.g., the Psion); or a hand-held or lap-top computer that you have had programmed for your particular purpose (Emerson, Reeves, & Felce, 2000).

However, it is more likely that you will record segments of the behavior of interest for later coding. Video (i.e., a visual and sound recording) has the merit of being subject to multiple viewings, in both real time and slow motion. Recordings can be stockpiled for later training of coders; they can be viewed by different observers on different occasions, each time coding different aspects of the behavior; and they can even be viewed by the same observer on different occasions to assess observer reliability over time (more on this later). They permit reflection (and, literally, review) in a way live observation does not. Moreover, recording devices (primarily analog tape-based at the moment, i.e., 2000, but increasingly digital and either tape- or disk-based) are relatively inexpensive, easy to use, and generally reliable. Perhaps the most common argument against devices such as video cameras is that they are invasive, somehow changing the behavior of the persons observed. Increasingly this concern does not seem warranted, perhaps because video cameras and similar devices are becoming such a common feature in our daily environments, perhaps because researchers often obscure them behind one-way mirrors or smoked-glass enclosures, but also because human beings involved in their own behavior, for example mothers with their infants, seem to rapidly habituate to the presence of recording devices.

What will you do about time? Perhaps the key decision when coding behavior live or recording behavior for subsequent coding concerns whether and how time will be preserved. For some research questions, it is possible that noting only the order in which events occurred without the times they occurred may be sufficient (e.g., he said A, she said B, he said C, she said D, etc.), but almost always preserving time enriches the data and provides more analytic possibilities (e.g., he spent 20% of the time saying A). Examples of behavior coded without time information become more common as one reads older literature, but we believe the reason for this is simply technical: As electronic recording devices have become common, researchers have found it easier to do what most have always wanted to do, which is preserve time as part of the record.

One older method for recording behavioral streams is sufficiently common, especially in older literature, that it deserves special mention. At the same time, it represents a method that we think is rarely recommended today, given the alternatives now available. Technically, it does not preserve time, although it does allow information about rates and durations to be estimated. It is usually called zero-one or partial-interval or simply time sampling (e.g., Altmann, 1974). Typically, rows on a paper recording form represent successive intervals (e.g., each could represent 15-second intervals; see Konner, 1976). Columns represent particular behaviors and the observer notes with a tick mark the behaviors that occurred within each interval. The intent of the method was to provide approximate estimates of both frequency (how often specific behaviors occurred) and duration (what percentage or proportion of time behaviors occurred) information, without imposing an undue burden on observers, in an era before readily available recording devices automatically preserved time. Thus the method was a compromise with desire, reflecting the technology of the time, and even then its shortcomings were clearly understood (e.g. Sackett, 1978). Still, momentary time-sampling (e.g., recording the posture of a sleeping person at the moment when it has been exactly 15 minutes since the last recording) may still prove useful; when states typically last for some time, such point samples can provide adequate estimates of proportions of time spent in various states.

The alternative to interval recording (i.e., time-sampling) is sometimes termed continuous recording, but might more accurately be termed continuous observation with transition-activated recording (Martin & Bateson, 19xx) because observers remain continuously alert but record only when a transition occurs, that is, when one state ends and another begins, or when a momentary behavior occurs. The coarseness of the scale, which renders estimates of frequencies and percentages approximate, is the central problem of interval recording. If behaviors were time-sampled every second, for example, the problem lessens to the point of disappearance. This statement simply illustrates that the boundary between continuous and interval recording is not distinct. Continuous recording is necessarily discrete, in the sense that times are always recorded to some specified precision, often to the nearest second or tenth of a second. For example, if a coder reported that a baby started crying in second 10 and stopped in second 25, understanding that *stopped* means crying through second 24 inclusive but not second 25, then we could tick every second, 10 through 24, for baby crying. In this case, no matter whether interval (1-second intervals) or continuous recording (to the nearest second) was used, the resulting data would be indistinguishable.

Cast in this light, precision seems a more important matter than interval versus continuous recording. The precision or time unit used should be reasonable for the behaviors of interest. For example it seems reasonable to measure infant crying in seconds, but not in milliseconds, on the one hand, and not in terms of the proportion of 30-second intervals that contained crying, on the other. Historically, we believe, researchers who used interval recording with relatively coarse intervals wanted greater precision but were not able to do so easily. Now reasonable precision is readily available, and so seldom is time-sampling the first choice for a recording method.

Instead, almost always, you will be able to record the onsets (i.e., the time of onset) of momentary behaviors and the onsets and offsets of duration behaviors economically. Organizing duration behaviors into sets of mutually exclusive and exhaustive code permits further efficiency; because the onset of one behavior in the set necessarily implies the offset of the previous behavior, only onsets of behaviors in a set need be recorded. For this reason we recommend that, when feasible, codes be organized into sets of mutually exclusive and exhaustive codes, as the examples from our work and from Lavelli and Poli (1998) illustrate.

How should time be recorded? When coding live using pencil and paper, time displayed on a stop watch could be written down, although, especially when the behavior of interest is fast moving and requires attention, this can unduly burden observers and distract from observation, thereby compromising data quality. However, almost any hand-held electronic device, including lap-top computers, will record time automatically when a key is pressed. Thus the record preserved in the device consists of a series of events and the times they occurred, or more literally, of symbols representing keys and the times they were pressed. When behavior is recorded with the devices available today (e.g., camcorders), a time code can almost always be incorporated in the video signal. This could be a visual time code, which is displayed on the screen when the tape is played back, or it could be an electronic code, which has an important advantage: On play back the time code can be read and stored in a data file automatically, freeing the observer of the need to key it in. This code is often called a VITC time code because it is stored in the vertical interval of the video signal, or a SMPTE code because it follows conventions defined by the Society of Motion Picture and Television Engineers.

Represent the Initial Data

If a no-tech approach to coding relies only on pencil and paper and the naked eye, and if a high-tech approach connects computers and video recordings, then a relatively low-tech approach to coding video material might assume video recording but rely on a visual time code and would require observers to record not just codes but also the time they occurred. Ultimately, we assume, data will be processed by computer so observers viewing video could use pencil and paper for their initial records and then enter the data in computer files later or key their observations directly into a computer as they worked, whichever they find more efficient. Such a system retains all the advantages that accrue to coding previously recorded material (as opposed to live coding) and is attractive when budgets are constraining because it does not require directly connecting video playback devices with computers. It places some burden on observers, however, and can be error prone as is any system that requires repetitive entry of numeric information and attention to multiple tasks.

When feasible, a more high-tech approach has advantages, and a number of systems are available. Such systems combine video recordings, either analogue or digital, and computers in ways that serve to automate coding. Four examples are the systems developed by The James Long Company (Long, 1996); the PROCODER system developed by Jon Tapp and Tedra Walden (Tapp & Walden, 1993); The Observer developed by Lucas Noldus (Noldus et al., 2000; for current information see http://www.noldus.com); and the ObsWin program developed by Neil Martin, Chris Oliver, and Scott Hall (Martin et al., 1999; for current information see http://psgsuni.bham.ac.uk/obswin/obswin2.htm). Each of these four offer a range of different capabilities. Moreover, as with technical matters generally, their products change, and so any statements in a chapter such as this run the risk of becoming dated. Further, The Observer in particular offers a number of real time capabilities, and all four offer data analytic possibilities, but here our comments are confined primarily to the capabilities these programs/systems have for coding previously recorded material. For a review of computer systems for recording observational data see Kahng & Iwata (2000).

Speaking somewhat generically, computer-based coding systems permit researchers to define the codes they use and their attributes. Are they momentary or duration codes? Are duration codes organized in mutually exclusive and exhaustive sets? Then coders can view previously recorded information, backwards and forwards, in real time or slow motion as they decide how the material should be coded. When they decide a code should be assigned (e.g., the baby just began crying), coders can either press a key or keys or else use the computer mouse to click that code on screen. The computer program then organizes the codes and their associated times into computer files. Coders can edit those files if, on reflection and reviewing, they think their initial coding should be changed. With appropriate equipment and software, the playback of a video recording, can be controlled with mouse clicks on the screen. Researchers can even create a list of specific episodes to be played back, either for demonstration purposes or to permit other coders to recode them. In sum, such systems tend to the clerical tasks, freeing coders to focus on their primary task, which is making decisions as to how behavior should be coded. Their only disadvantage is the amount of money the required equipment and software can cost, and the amount of time it can take to learn to use that equipment and software effectively. As you might expect, the larger the coding task, the more cost effective these initial investments become.

No matter how you record and format your data initially, at some point, sooner rather than later, those data need to be arranged into files suitable for preliminary computer processing. Probably you will also need to reduce those initial data before you conduct subsequent analyses with statistical packages such as SPSS or SAS. We have defined standard conventions for representing or formatting sequential data, which we call SDIS or Sequential Data Interchange Standard (Bakeman & Quera, 1995a; Quera & Bakeman, 2000). Our intent was to reduce work and facilitate sharing. As more investigators use the same standard then they can share the analytic tools they develop; each lab need not spend time reinventing its own wheels. Files containing data that follow SDIS conventions are designed to be relatively intuitive, clear to read, and easy to use. Representing your data with these standard formats has an additional advantage: You can then use a program we have written to analysis SDIS formatted data that has considerable capability and flexibility. We call this program the Generalized Sequential Querier or GSEQ (Bakeman & Quera, 1995a; for current information see http://www.gsu.edu/~psyrab/sg.htm or http://www.ub.es/comporta/sg.htm).

Possible scenarios for representing your data include: You record data using pencil and paper and then enter it into computer files as SDIS data, or you collect data using an electronic device, which produces lists of codes and their time of occurrence, and then reformat this initial data according to SDIS conventions. Programs that reformat such data are relatively easy to write; for example, we have written programs that produce SDIS data files from data files produced by the James Long Company system and by The Observer (Bakeman, 2000b; Bakeman & Quera, 2000). Or, you collect data with ObsWin (Martin, Oliver, & Hall, 1999), which has the capability to output data already in SDIS format. There is another possibility. Most data collection programs also have data analytic capabilities; this is true to varying extents for all three systems we have mentioned. If you find these capabilities sufficient for your purposes, there is no need to represent your data in SDIS format. We are somewhat biased, of course, but if you intend to pursue a variety of analytic options, we think it is worth your while to learn more about the capabilities and flexibility of GSEQ. Certainly one strength of GSEQ is the ease with which a variety of sequential indices can be defined and computed and then output for subsequent processing by packages such as SPSS or SAS, as we demonstrate later in this chapter.

First, we would like to provide some sense of what SDIS data files look like. Five data types are defined, which we believe accommodates most current practice. The first and simplest we call Event Sequential Data (ESD), which consist of a single stream of codes recorded without time information. Here is what the first few lines of an ESD file might look like:

```
Event sleep drowsy alert fuss cry  * sex (female male);
<infant #1> drowsy sleep drowsy fuss cry fuss alert … (female)/
```

The first line declares the data type and, optionally, lists permissible codes; the next line here lists the sequence of codes for a participant labeled “infant #1,” who is female. A second type we term multi-event sequential data (MSD). It is used when an event is categorized by more than one dimension, but again without time information. For example, bids for a mother’s attention might be categorized by antecedent (coded as A, B, or C), form (coded as F, G, H), and result (coded as R, S, T). Here is an example:

```
Multievent (A B C) (F G H) (R S T)
<child #1> A G R. A H R. C F S.  … /
```

Sets of mutually exclusive and exhaustive codes are declared by enclosing them in parentheses and episodes are separated with periods. In this case, the codes were single letters, but codes in SDIS can be letter, digits, or a combination of letters and digits, 1-16 characters in length (and can be case-sensitive). A third type is Interval Sequential Data (ISD). It looks like MSD, except that commas are used to separate intervals. It is designed to accommodate investigators working with interval data (e.g., Bakeman, Adamson, Konner, & Barr (1990), although earlier we characterized this method of data recording as seldom recommended today.

The final two types include time information and, as you might guess from our earlier comments, are the two we find most useful. The first is State Sequential Data (SSD), which assumes that your codes are organized into one or more sets of mutually exclusive and exhaustive codes. To give you a sense of the entire process, here is a segment of a data file produced by the James Long Company system (other electronic recording systems produce similar data):

```
00:41:29:08 be, S125
00:41:29:08 un
00:41:29:16 sj
00:42:01:19 ob
00:43:26:02 un
00:43:30:19 sj
00:43:45:15 cj
00:45:18:00 ob
00:46:34:19 un
00:46:45:28 en, V1 help
```

Time was represented here as hh:mm:ss:ff, where *hh* represents hours, *mm* minutes, *ss* seconds, and *ff* frames (for standard video equipment, frames vary from 1-30 per minute). The codes, used in our first studies of joint engagement, were *un* for unoccupied and *lo* for onlooking; *ob* for object, *pe* for person (i.e., mother), *sj* for supported joint, and *cj* for coordinated joint engagement; and *be* and *en* to signal the beginning and end of the observation session. The same data, represented as SSD, are:

```
State (un lo ob pe sj cj);
* Visit (V1 V2 V3 V4 V5)  Scene (turns music help want);
<S125>  ,0:41:29  un,0:41:29  sj,0:41:30  ob,0:42:02  un,0:43:26  sj,0:43:31
cj,0:43:46  ob,0:45:18  un,0:46:35  ,0:46:46  (V1,help)/
```

Here only one set of mutually exclusive and exhaustive state codes was defined and time was represented as hh:mm:ss, rounded to the nearest second (our conversion program asks whether we want to round to the nearest second or nearest tenth of a second). Times could have been represented without hours, which in this case were all zero (several different time formats are possible). In addition two conditions were defined, Visit and Scene, which could take the values indicated. Following SDIS conventions, times are preceded by commas, and codes are followed by the times they began. Because these are state codes, the onset of a new state signals the offset of the preceding state. If there had been more than one set of mutually exclusive and exhaustive codes, onset times for codes in the second set would be separated by an ampersand from times for the first set.

The last data type, Timed-Event Sequential Data (TSD), is the most general, and can be viewed as an elaboration of SSD; any data expressed as SSD could be expressed as TSD, although a few more keystrokes would be required. For example, imagine that we defined *ix* as a momentary code for infant affective expression and that two occurred during the first episode of supported joint and one during the second episode of supported joint engagement. Then the TSD representation for the data just presented would be:

```
Timed (un lo ob pe sj cj)  ix;
* Visit (V1 V2 V3 V4 V5) Scene (turns music help want);
<S125>  ,0:41:29  un,0:41:29-  sj,0:41:30-  ob,0:42:02-  un,0:43:26-  sj,0:43:31-
cj,0:43:46-  ob,0:45:18-  sj,0:45:30-  ob,0:45:53-  sj,0:46:30-  un,0:46:35-  &
ix,0:41:37  ix,0:41:45  ix,0:43:36  ,0:46:46  (V1,help)/
```

In TSD, times for duration codes are indicated as *code,ontime-offtime,* but if no time occurs after the hyphen, it defaults to the next time encountered in the file. Here, for convenience, we have segregated all the momentary codes to a second stream, after the ampersand, but TSD conventions allow us to integrate them into one stream as follows:

```
Timed (un lo ob pe sj cj)  ix;
* Visit (V1 V2 V3 V4 V5) Scene (turns music help want);
<S125>  ,0:41:29  un,0:41:29-  sj,0:41:30-0:42:02  ix,0:41:37  ix,0:41:45
ob,0:42:02-  un,0:43:26-  sj,0:43:31-0:43:46  ix,0:43:36  cj,0:43:46-  ob,0:45:18-
sj,0:45:30-  ob,0:45:53-  sj,0:46:30-  un,0:46:35-0:46:46  (V1,help)/
```

One final example: Lavelli and Poli (1998) used Noldus' The Observer (Noldus et al., 2000). Imagine that their infant state codes were *sleep, drowsy, alert, fuss,* and *cry*; that their mother looking at infant codes were *look* and *nolook*; and that they recorded times to the nearest second. A segment of one of their Observer Data Files could look like this:

```
{start}
0 sleep
0 nolook
4 look
7 fuss
10 nolook
12 cry
14 look
…
29 {end}
```

Then the corresponding SDIS data file would look like this:

```
Timed (sleep drowsy alert fuss cry) (look nolook)
<subject id> 0, 0,sleep- 7,fuss- 12,cry- …
& 0,nolook- 4,look- 10,nolook-, 14,look- … ,29/
```

These examples do not exhaust all of the possibilities SDIS provides, but they should suffice to give you a reasonable sense of how your own data might be represented. This is an important step. Once the data you collect initially have been organized into computer files, then your first descriptive analyses can commence using all the power and accuracy that computers provide. However, first another important matter requires your attention: You need to convince yourself and others that those data were coded reliably.

Establish Observer Reliability

The accuracy of any measuring device needs to be established before weight can be given to the data collected with the particular device. For the sort of observational systems we are describing, the measuring device consists of trained human observers applying a coding scheme or schemes to streams of behavior, often video recorded. Thus the careful training of observers, and establishing their reliability, is an important part of the enterprise. Typically, observers are asked to make categorical distinctions; not surprisingly, then, the most common statistic used to establish inter-observer reliability is Cohen’s kappa (1960), a coefficient of agreement for categorical (i.e., nominal) scales. If a standard exists, which is assumed true, then observers can be compared with it. Otherwise, the usual assumption is, if two observers agree when independently coding the same material, then their data should accurately reflect the observed behavior, and hence be reliable. Either way, Cohen’s kappa is useful both when training observers and later, when reporting their reliability in published reports. Moreover, it corrects for chance agreement and thus is much preferred to the percentage agreement statistics sometimes seen, especially in older literature.

Kappa. Kappa is an index that characterizes agreement in applying a set of mutually exclusive and exhaustive codes. A value of 1 indicates perfect agreement. Fleiss (1981) characterized values over .75 as excellent, between .60 and .75 as good, and between .40 and .60 as fair. Values less than .60 are thus problematic; negative values are possible and would indicate systematic disagreement (or perhaps date entry errors). Kappa does not yield values for individual codes although useful information concerning sources of disagreement can be gleaned from the agreement matrix on which kappa is based. As an example, imagine that observers were asked to code infant states using Lavelli and Poli’s (1998) codes. Imagine further that boundaries between states had already been identified so that coders approached their task with no ambiguity as to the thing (or unit) that they were coding. Speaking generally, let *k* represent the number of codes in the scheme, in this case five. Then agreement can be tallied in a *k* × *k* matrix, which is often called an agreement matrix, or a confusion matrix (do you see the glass half full or half empty?) or, simply, a kappa table. Rows represent the first coder, columns the second, and both are labeled identically with the *k* codes. When observers primarily agree, most tallies fall on the upper-left to lower-right diagonal. Off-diagonal cells indicate disagreements (e.g., what the first observer coded *sleep* the second observer coded *drowsy*), and noting off-diagonal cells with many tallies provides useful feedback when training observers.

For the infant state example, the agreement matrix might look like this:

Coder A	Coder B					
	Sleep	Drowsy	Alert	Fuss	Cry	Total
Sleep	12	2	0	0	0	14
Drowsy	2	7	1	1	0	11
Alert	0	0	15	0	0	15
Fuss	0	0	2	8	0	10
Cry	0	0	0	4	5	9
Total	14	9	18	13	5	59

Fifty-nine infant states were coded by two observers with what appears to be reasonably good agreement. For example, cry was never confused with sleep or drowsy or alert, and sleep was never confused with alert or fuss. The coders agreed 47 times (80%) and disagreed 12 times. Symmetric disagreements indicate that coders are equally confused whereas asymmetric disagreements indicate that coders may have different thresholds, thus usually asymmetric disagreements are of greater concern. In this case, the coders disagreed about sleep versus drowsy four times, but twice Coder B coded a state sleep when Coder A coded it drowsy, and twice Coder A coded a state sleep when Coder B coded it drowsy, which is perfectly symmetric. In contrast, four times Coder B coded a state fuss when coder A coded it cry, but Coder A never codes a state cry that Coder B coded fuss. This suggests that Coder B has a higher threshold for what constitutes a cry and that the two coders need to become better calibrated through additional training.

The computation of kappa is fairly straightforward, although computer programs are available (e.g., the ComKappa program, which can be downloaded from http://www.gsu.edu/psychology/bakeman; Robinson & Bakeman, 1998). Let x_{ij} indicate a cell of the matrix. Plus signs indicate summation, so x_{i+} indicates the total for the ith row and x_{++} indicates the total number of tallies in the matrix (in this case, $x_{++} = 59$). Then

$$P_{obs} = \frac{\sum_{i=1}^{k} x_{ii}}{x_{++}}$$

represents the proportion of agreement actually observed (.80 in this case), and

$$P_{exp} = \frac{\sum_{i=1}^{k} x_{+i} x_{i+}}{x_{++}^{2}}$$

represents the proportion of agreement expected due to chance (.21). Then

$$\kappa = \frac{P_{obs} - P_{exp}}{1 - P_{exp}} \qquad (1)$$

indicates how kappa is computed, and also how the value of kappa corrects for chance agreement. For our present example kappa is .74, which according to Fleiss (1981) is good bordering on excellent.

Weighted kappa. When codes are roughly ordinal or for other reasons investigators may wish to regard some disagreements as more serious than others, Cohen (1968) has specified a way of weighting disagreements. Three $k \times k$ matrices are involved: one for observed frequencies, one for expected frequencies, and one for weights. Let x_{ij}, m_{ij}, and w_{ij} represent elements from these three matrices respectively; then $m_{ij} = (x_{+j} \times x_{i+}) \div x_{++}$, and the w_{ij} indicate how seriously we choose to regard various disagreements. Usually the diagonal elements of the weight matrix are 0, indicating agreement (i.e., $w_{ii} = 0$ for $i = 1$ through k); cells just off the diagonal are 1, indicating some disagreement; cells further off the diagonal are 2, indicating more serious disagreement, and so forth. No matter what weights are assigned

$$\kappa_{wt} = 1 - \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij} x_{ij}}{\sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij} m_{ij}} \qquad (2)$$

indicates how weighted kappa is computed. If all off-diagonal elements are 1 and all diagonal elements are 0, then κ as defined earlier and κ_{wt} yield identical results. For the present example, if the weights were defined as follows:

Coder A	Coder B				
	Sleep	Drowsy	Alert	Fuss	Cry
Sleep	0	1	2	3	4
Drowsy	1	0	1	2	3
Alert	2	1	0	1	2
Fuss	3	2	1	0	1
Cry	4	3	2	1	0

Then the value of weighted kappa would be .85, whereas the unweighted value of kappa was .74. Whenever weighted kappa is used, however, a clear rationale for the weights assigned is essential and should be articulated. The weights in the example matrix just given are appropriate when the categories are clearly ordinal; to the extent that these categories (i.e., sleep, drowsy, etc.) are not viewed as ordinal, other weights, or no weights (i.e., all disagreements weighted 1), might arguably be more appropriate.

Kappa max. Earlier we wrote that the maximum value for kappa is 1. However, a value of 1 can only occur when both coders distribute their codes identically, that is, when the row and column totals (or marginals) are identical. Here, both coders assigned sleep to 14 states but Coder A assigned alert to 15 states whereas Coder B assigned alert to 18 states. Thus only 15 agreements could be tallied for alert, which reduces the maximum value kappa could obtain. The more discrepant the marginals are, the more the maximum possible value of kappa is reduced. In such cases, one could argue, it makes sense to compare the value of kappa obtained, not to 1, but to the maximum value that could have been obtained if as much agreement as possible had occurred, relative to the constraints imposed by discrepant marginals. The ComKappa program (Robinson & Bakeman, 1998) computes values for *kappa max* and its formula is given in Umesh, Peterson, and Sauber (1989). For the example data used here, the value is .87. Thus one way to view the value of .74 computed for kappa is to say that it achieved 85% of its maximum value, given the way the observers distributed their codes.

For the present example, the value of kappa was reasonably respectable, but especially when codes are few and row and column totals quite discrepant, what at first appear to be only fair kappas might seem better when compared to kappa max. For example, imagine that 100 seconds were coded for mother gazing at her infant and that the agreement matrix looked like this:

Coder A	Coder B		
	Gaze	No	Total
Gaze	23	17	40
No	2	58	60
Total	25	75	100

The value of kappa would be .58, which is only fair by Fleiss' (1981) standards, but kappa max is only .67, thus the observed value of kappa was 87% of its potential maximum. Whether or not this is acceptable depends somewhat on how individual investigators view and argue the matter. For example, one might still question why the disagreements are asymmetric here, as they will be when kappa max is relatively low. Perhaps the safest conclusion is, kappa max cannot be used as a panacea for low kappas. (Highly skewed marginals in agreement matrices may present another set of problems; for a discussion of this matter see Bakeman, Quera, McArthur, and Robinson, 1997.)

Picturing observer agreement. For simplicity, our example using infant state codes assumed that boundaries betweens states had been marked previously so that coders needed only categorize states as though they were so many billiard balls to sort by color. This is a reasonable way to introduce kappa because some *thing* is assigned codes, and the number of such things is the total number of tallies entered in the table. Rarely, however, are coders presented with already segmented streams of behaviors. Instead theirs is a two-fold task: They need to both find the boundaries between states and then code the states so demarcated. A strict interpretation might be that the reliability of both these tasks should be established, but in fact rarely do coders even think of performing two separate tasks; instead, as they note onset times, the coding task seems seamless to them. And indeed, if we compute kappa based on seconds as the thing coded, then we capture in one agreement matrix both aspects of the coding task (we say *second* somewhat generically, for ease of exposition, but it could be any time unit that makes sense when coding a particular set of behaviors).

Recall that in our current research, we define eleven engagement codes: un, lo, ob, pe, sj, cj, sy, os, ps, ss, and cs for unengaged and onlooking; object, person, supported joint, and coordinated joint engagement; symbol-infused engagement alone; and symbol-infused object, person, supported joint, and coordinated joint engagement, respectively. When first training observers, and later when checking their reliability, we prepare SDIS files based on their coding. For example, here is the way one observer coded one of our contexts (i.e., conditions, in this case turn-taking) for one infant:

```
State (un lo ob pe sj cj sy os ps ss cs);
<192 V1 turns> un,0:28:55 lo,0:28:59 sj,0:29:11 cs,0:29:36 os,0:29:49 lo,0:30:07
ob,0:30:12 sj,0:30:18 os,0:31:00 sj,0:31:22 un,0:31:25 ob,0:32:34 sj,0:33:01
lo,0:34:16 ,0:34:27 /
```

and here is the way a second observer coded the same context for the same infant:

```
State (un lo ob pe sj cj sy os ps ss cs);
<192 V1 turns> un,0:28:57 lo,0:29:03 sj,0:29:09 ss,0:29:36 os,0:29:49 ob,0:29:57
lo,0:30:07 ob,0:30:12 lo,0:30:18 sj,0:30:26 ss,0:31:02 un,0:31:25 ob,0:32:34
sj,0:33:02 un,0:34:13 ,0:34:25 /
```

For brevity, these two example files contain coding for only one condition and one infant, but the files for the two observers could contain many conditions for many infants. If we had wanted to visualize the observers' coding, we could have asked the GSEQ program to plot these files. Figure 1 shows the plot that GSEQ produces, here for the first two minutes of the first observer's coding (Explore | Plot in GSEQ). From it we can easily see that the session began with a few seconds of unoccupied, moved into a few more seconds of onlooking, and thence into a period of supported joint engagement, at least as coded by the first observer.

Given two files that represent coding for the same stream of behavior as performed independently by two different observers, the question now arises, how well do they agree? Figure 2 shows the output GSEQ produces (Tools | compute Kappa in GSEQ), which consists, first, of a time-line plot showing where disagreements occurred and, second, of an agreement matrix along with the value of kappa computed. (The ampersand used to label the last row and column of the agreement matrix is used if there are any seconds for which no code was defined; in this case, there were none.) The GSEQ for Windows (GSW) kappa program permits users to define slippage, that is, an amount of time by which coders can disagree but still have their coding count as agreement. For this example, we defined slippage as plus or minus two seconds, which, in effect, counts a second as an agreement if the second coder agreed with the first within a five-second time window (the current second plus or minus two).

The time-line plot indicates each code with symbols the user chooses. Here we used the symbols u, l, o, p, s, c, Y, O, P, S, and C for un, lo, ob, pe, sj, cj, sy, os, ps, ss, and cs, respectively. The hh:mm:ss above the time line represent hours, minutes, and seconds, respectively. A hyphen underlies seconds that count as disagreements, and a period underlines seconds that would have counted as disagreements if we had not defined slippage; the pound sign simply underlies seconds that are not tallied in the agreement table because only one observer coded them. In this case, from the agreement matrix we see that three disagreements occurred for more than 10 seconds: between symbol-infused object and symbol-infused supported joint engagement (20 seconds), between symbol-infused supported joint and symbol-infused coordinated joint engagement (13 seconds), and between unengaged and onlooking (11 seconds). Further, from the time-line plot, we see exactly where these disagreements occurred, which is very helpful when we and the observers review tapes in an effort to figure out why disagreements occurred, what they might mean, and whether further training is warranted.

By way of conclusion, we would like to reiterate a point made earlier: An examination of observer reliability serves two functions, one more important at the beginning and one more important at the end of research that relies on systematic observation. When training observers, not only does kappa provide a clear criterion towards which novice observers can strive, the agreement matrix, along with the time-line plots GSEQ produces, provide very useful feedback as to what kinds of disagreements might cause kappas to be low and suggest topics for additional training. Then, throughout data collection, reasonable values for kappa assure you that observers are continuing to code reliably, or lowering values may signal the need for additional training (typically, about 15%-20% of a corpus is coded by more than one observer and observers should not know which of their work will be used to check reliability). Finally, when writing reports, once data are collected and analyzed, reporting values of kappa gives assurance to reviewers, editors, and eventually other readers that the data on which your analyses are based are sound.

Describe and Reduce Initial Data

In a previous section we talked about how the data-as-collected might be represented initially. We meant re-present literally, that is the data are presented again, to the computer, for the first round of descriptive analyses. In saying this we make two assumptions: first, that often data are collected with somewhat more detail than is ultimately useful for even preliminary analyses and so some reduction is necessary at the outset (e.g., in the process of coding we find out that some codes occur very rarely or that some distinctions are difficult to make reliably); and second, that the very first analyses we do are primarily descriptive in nature, designed to give ourselves and readers a sense of our data, whereas subsequent analyses probe our data more deeply as, for example, we address specific hypotheses. In this section we focus on those initial, descriptive analyses and emphasize the advantages of and techniques for deriving new variables from the data-as-collected. These new, derived variables may both simplify our data and better reflect our conceptual concerns. Then, in the next section we focus more on analyses that probe for patterns in the behavioral streams and emphasize ways in which variables can be combined into indices that both exploit sequential information in the data and permit answers to the research questions that motivated our investigations in the first place.

Basic descriptive statistics. Once data are collected the first step is to examine basic descriptive statistics for all our codes. For momentary codes, these include *frequencies* (how often did the observers use this code, i.e., to how many events was this code assigned), *relative frequencies* (what proportion of the events were assigned this code), and *rates* (how many times per hour, or other unit of time, did this event occur). For duration codes, these include *frequencies*, *rates*, *probabilities* (for what proportion of total time did this event occur), and *average durations* (how long, on average, did instances of this event last). Relative frequencies can be computed for duration codes but usually the proportion of time is regarded as more informative (i.e., when behavior is timed usually it is more interesting to know that code A occurred 22% of the time than to know that 18% of all events were coded A). Note, however, that frequencies, probabilities, and average durations are redundant; if frequency is low and probability is high, the average duration necessarily must be long. As a general rule, we think it makes sense to report only two of these three basic statistics for duration behaviors, but which depends on which two you think are most informative. Other statistics that can be computed for duration codes include duration (for how many seconds, or other time units, was this code assigned) and minimum and maximum lengths (what was the shortest and longest length recorded for this event). GSEQ computes other statistics as well (e.g., the average length of time between events; Version 3.7 and later), which some investigators might find useful.

Pooling. These basic statistics are easily computed with GSEQ and most other programs. When first examining our data, it makes sense to pool over all cases because initially, as we make decisions about how best to reduce our data, we are concerned with how often codes were used and not yet with how individuals varied, that is, we want to view our data whole and not yet derive scores for individuals. GSEQ allows you to pool over cases (i.e., units, which may be infants, dyads, etc.; from the examples given earlier, recall that in SDIS a unit begins with a label enclosed in angle brackets and ends with a forward slash), over variables if any were given, or both. (SDIS also permits sessions within cases and data are pooled over sessions by default; e.g., coding might be interrupted because the toddler wanders off-camera and the time before would be regarded as the first session and the time after as the second session within the case; or mother and infant might be seen three times on three days, with each day regarded as a different session within the case) .

Assume, for example, that several mother-infant dyads were observed, that the observations of each dyad constitute a separate case in the SDIS data file, that one variable indicating the infant's sex was defined, that M and P were codes for momentary behaviors, and that A, B, and C were codes for a set of mutually exclusive and exhaustive duration behaviors. Then the declaration line for the SDIS file would look like this:

```
Timed (A B C) M P * sex (male female);
```

and the GSEQ commands:

```
Pool * sex;
Simple freq relf rate (M P);
Simple freq rate prob avgd (A B C);
```

would produce basic descriptive statistics for these data, pooled over all cases (the asterisk by itself on the Pool command would mean pool over all cases, separately for males and females; the word sex on the Pool command means pool over male and female cases as well). When using GSEQ for Windows, you would not need to type in these commands: The program lets you compose them from point-and-click menus (Explore | Analyze). The benefit of proceeding this way, that is, producing basic descriptive statistics for your codes pooled over all cases (separately for each combination of levels for any variables, or pooled over levels of any variables if you so specify) is that you gain a good overview of how often the behaviors of interest to you actually occurred in your coded corpus. With this knowledge, you can now proceed with data reduction.

Basic data reduction. There are at least three good reasons to reduce your data before proceeding with further analyses, at least two of which have already been mentioned. First, you may have discovered during coding that observers could not reliably distinguish between two codes, yet when you combined those two codes into one superordinate code, the superordinate code was reliable, thus it makes sense to combine those two codes into one new one. Second, you may have discovered that some, relatively similar codes occurred fairly infrequently. If treated as separate codes, scores would be zero for some cases and quite low numbers for others, but if combined into one code, that single code would be simpler and less problematic to analyze. Third, you may have solid, theoretical reasons for combining some codes into more-encompassing variables.

Such reduction is easily accomplished with GSEQ. At first glance, on reading Bakeman and Quera (1995a), it might seem that the Recode or Lump commands are appropriate, but for State and Timed-Event Sequential data, the Or command is probably better. When new codes are introduced in the data with the Recode and Lump commands, the old codes are removed, whereas the Or command leaves the old codes intact, where they can be referenced later if the need arises (the Or command assigns the new code to a time unit if any one or more of the old codes was already assigned to that time unit). For example, recall that the eleven codes we use to code engagement state are un, lo, ob, pe, sj, cj, sy, os, ps, ss, and cs (for unoccupied, onlooking; object, person, supported joint, and coordinated joint engagement; symbol engagement; and symbol-infused object, person, supported joint, and coordinated joint engagement). First we examined simple, basic statistics for these codes, pooled over all cases. As we expected, in our corpus engagement with symbols alone (sy) occurred very rarely as did symbol-infused object and person engagement (os, ps; these and subsequent statements are based on preliminary analyses of the first 24 mothers and infants in our study whose video tapes we had coded for engagement state). Thus we decided to create a single, symbol-infusement outside of joint engagement code that combined sy, os, and ps. For simplicity, we also decided to combine unengaged and onlooking (un, lo) because onlooking was not especially common and because the distinction between the two was not especially interesting to us,

Additionally, for theoretical reasons, we created one code that combined all joint engagement codes, whether supported or coordinated and whether or not symbol-infused (sj, cj, ss, cs); one code that represented that part of total joint engagement that was supported joint engagement (sj, ss), whether or not symbol-infused; and that part of total joint engagement that was symbol-infused (ss, cs), whether supported or coordinated. The GSEQ commands that created these new codes were:

```
File “C:\GSEQ\examples\ES24.mds”;
OR syo = sy os ps;
OR unl = un lo;
OR jnt = sj cj ss cs;
OR sup = sj ss;
OR sij = ss cs;
Save “C:\GSEQ\examples\ES24mod.mds”;
```

Earlier we had compiled the SDIS data file that contained data from these 24 mother-infant dyads (File | Open | Compile); this created an MDS or modified SDIS file (the SDIS data file is an ASCII file, easy to read and edit; the MDS file is a binary file, easier for GSEQ to process). This set of commands instructs GSEQ to open the file called ES24.mds, create five new codes, and save the results in a file called ES24mod.mds. Subsequently, we can refer to file ES24mod.mds and have available to us all of the codes in ES24.mds initially plus the five new ones created by these commands.

At this point, we would no longer want to pool over cases. Now that we have created new codes, reducing our data a bit, we want to derive and describe basic statistics for these new codes, and probably some old ones too, but consistent with usual practice, we would want to report means and variability for our sample (e.g., standard deviations and ranges) based on scores for individual cases. Later we will want to run other, more sophisticated analyses as well, and so it makes sense to learn now how to export data from GSEQ in a form that can be processed by standard statistical packages such as SPSS and SAS. For the present example, the GSEQ statements:

```
File “C:\GSEQ\examples\ES24mod.mds”;
Simple freq prob avgd (unl ob pe syo jnt sje sij);
Send "C:\GSEQ\examples\ES24fpa.dat" freq prob avgd tabs label overwrite;
```

would do two things. First, frequencies, probabilities, and average durations for the key behaviors of total unengaged (including onlooking), object and person engagement, symbol-infused outside of joint engagement, total joint engagement, total supported joint engagement, and total symbol-infused joint engagement would be computed for each case. Second, those simple statistics would be sent to a file (named here ES24fpa.dat) formatted so that they can be easily read by programs such as SPSS; in this case, each item in the file is separated with tabs, cases are identified with the label from the SDIS file (i.e., the information enclosed in angle brackets), and any other data file of the same name is overwritten.

Here is an example, derived from one 5-minute observation in our corpus. The output created by the Simple command is:

```
Variable sex, condition male
Unit #7 label 'S107V1'.
Pooling over 1 session (maximum 1 session per unit).

SIMPLE statistics

 Codes        FREQ    PROB       AVGD
         ------------------------------
 ob      |       4  0.1688      13.25 |
 pe      |       0  0.0000       0.00 |
 syo     |       2  0.0350       5.50 |
 unl     |       2  0.0637      10.00 |
 jnt     |       7  0.7293      32.71 |
 sup     |       9  0.4331      15.11 |
 sij     |       2  0.1497      23.50 |
         ------------------------------
 Totals:        26  1.5796

Total number of time units: 314
```

The first line of the data file (ES24fpa.dat in this example) created by the Send command, followed by the line that represents data from the case just shown, is:

```
Unit  →sex  →FRob  →PRob  →ADob  →FRpe  →PRpe  →ADpe  →FRsyo
→PRsyo  →ADsyo  →FRunl  →PRunl  →ADunl  →FRjnt  →PRjnt  →ADjnt
→FRsup  →PRsup  →ADsup  →FRsij  →PRsij  →Adsij

S107V1  →1  →4  →0.1688  →13.25  →0  →0.0000  →0.00  →2
→0.0350  →5.50  →2  →0.0637  →10.00  →7  →0.7293  →32.71  →9
→0.4331  →15.11  →2  →0.1497  →23.50
. . .
```

where the arrows represent tab characters. The first line contains names for the scores in the Send file; these will be understood as variable names by the program that imports the file. Some of them correspond to variables defined in the SDIS file, like sex, while others are statistics computed by GSEQ. The second line, then, gives the values for these scores for one case; an actual data file would have several such lines. This data file can then be imported easily and directly into standard statistical packages such as SPSS.

At this point you may wonder, since GSEQ computes simple statistics for each case in the SDIS data file, why does it not also compute group summary statistics, that is, means, standard deviations, and ranges for all cases (or for all cases in groups defined by a variable like sex). We (i.e., Bakeman & Quera) view it as a matter of economy and efficiency, and as a lack of desire on our part to reinvent too many wheels. Statistical packages compute means and standard deviations and much more. It does not make sense to add abilities to GSEQ that already exist elsewhere, in far more developed form. Some statistics, such as means and standard deviations, are easy enough to compute, but others, such as statistics associated with multiple and logistic regression, present considerable challenges. Thus we view GSEQ as a program that permits users to manipulate and reduce sequential data in quite flexible ways, often with an eye to computing scores and indices for individual cases. The various scores and indices are then delivered to general purpose statistical programs for subsequent, more sophisticated analyses. In the next section we elaborate on this theme further; first, however, we end this section with other, more sophisticated examples of data reduction.

Advanced data reduction. All examples of data reduction presented so far have used the Or command. This makes sense because the most common form of data reduction combines several codes into one, superordinate code, but GSEQ offers other possibilities as well, including the full complement of logical operations (i.e., And, Or, Not, Nor, and Xor). After the Or command, the And command is probably the most useful because, with it, you can identify those seconds (i.e., time units) that contain two or more named codes (the And command assigns the new code to a time unit if all of the old codes were already assigned to that time unit). For example, you might want to identify times when the mother is gazing at her infant and the infant is crying, which could be done with:

```
And GazeCry = GazeAt Cry;
```

In the next section we describe how co-occurrences can be tallied in two-dimensional tables, and how co-occurrence statistics (i.e., statistics derived from two-dimensional tables) can be computed, but sometimes it is advantageous to also form a specific co-occurrence code, as in this example, because then you can request simple statistics (with the Simple command) such as the average duration of episodes when mothers were gazing at an infant who was crying.

Some of the more interesting ways of effecting data reduction with GSEQ involve the Window command. With it, you can assign new codes to time units that are keyed to the onset and offset times of existing codes. For example, you could define a new code that included the second (or whatever time unit you use) when crying began and the following nine seconds, or a new code that included the five seconds before the onset of the infant entering the alert state, or a new code that included only the first (i.e. onset) second when that mother soothed her infant, and so forth. With such new codes, you can address questions such as, do mothers begin to sooth their infants within five seconds of the infant beginning to cry, more so than at other times, and do infants become alert within ten seconds of mothers' stimulating their infants, more so than at other times, and so forth.

Often investigators come to sequential analysis having read about lag-sequential analysis (Sackett, 1979) and, since their data include onset and offsets time for the behaviors of interest (i.e., they have state or timed-event sequential data), they attempt to apply the timed version of lag sequential analysis using one-second lags. Almost always, in our experience, the questions such investigators have are better answered by forming new codes of the sort such described (using the Window command), and then computing indices as described in the next section. We say this because typically it is not so critical whether a mother responded to her infant's cry at a lag of two versus three versus five seconds, and so forth; what matters is whether she responded within a specified window of time. For many questions, treating each second as a separate lag applies too fine-grained a scale to the phenomenon of interest.

For example, imagine that Cry and Sooth were the codes used for infant crying and soothing, respectively. Then the following commands:

```
Window OnSooth = (Sooth;
Window TenCry = (Cry+9;
Window 5B4Cry = (Cry-5,(Cry-1;
Window p5aftCry = Cry+5;
Window 2nearEnd = Cry)-2,Cry)+2;
```

would produce the new codes as shown in Figure 3. In this case, the onset of mother soothing occurred within five seconds after the offset of the one infant cry shown but not within ten seconds of the onset of infant crying. For the Window command, a left parentheses before a code indicates the onset second and a right parenthesis after a code indicates the offset second for that code, inclusive. Thus, here, the first Window command indicates the onset second for Sooth. The second Window command indicates the onset second for Cry and the following 9 seconds. The third Window command indicates the fifth second before the onset of Cry up to and including the first second before the onset of Cry. The fourth Window command indicates all seconds of Cry and the following five seconds. Finally, the fifth Window command indicates the second time unit (i.e., second) before the offset of Cry up through and including the second time unit (i.e., second) after the offset of cry. (For other examples using the Window command, see Bakeman and Quera, 1995a.)

The Window command, coupled with other GSEQ data modification commands, can be used in even more complex and useful ways (I would like to thank Michelle Becker, Eugene Buder, Frank Floyd, and Yana Markov, among others, who suggested examples like these). For example, imagine that 60 seconds of mother-child interaction were coded, but that for a study of social referencing you want to analyze behavior only during the 20 seconds after the first time a child looked to his or her mother (we have selected relatively short times here so that the figures we present can be compact; in an actual study, the length of time interaction was observed and the length of time examined after the first look would likely be longer). Imagine further that in addition to Look, the code for infant looking to mother, you have already created superordinate codes cb and mb. Code cb represents any of several child behaviors of interest; code mb is the same but for mother behaviors. You are interested in how often codes cb and mb co-occur, but only within 20 seconds after a Look.

First we would need to create a code that identifies the 20-second trials initiated by a child's gaze to the mother. The following commands assign a code of T (codes can be case-sensitive in GSEQ, so code t and T are different codes) to the 20 seconds initiated by the first time during the minute that the child looked to his or her mother:

```
Window t = (Look+19;
Window x = (t+20,(t+60;
Not X = x;
And T = t X;
And C = cb T;
And M = mb T;
```

To understand how these data modifications work, see Figure 4, which shows the effect of these commands on a small segment of example data. The first Window command assigns a code of t to the 20 seconds initiated by any look to the mother (i.e., the onset second for Look and the following 19 seconds) during the 60-second observation, but remember we only want the first such episode. Therefore, the second Window command assigns a code of x to the 40 seconds after the offset of any of these 20-second trials (i.e., from the 20th second after the onset of t up to and including the 60th second after the onset of t.; a value of 60 was used here because we knew that the sessions were not longer than 60 seconds). Thus the x code will overlay any 20-second trials after the first one and extend to the end of the session. Next, we use the Not command to assign a code of X to any second not already coded x; this will include the first 20-second trail and any time before it. Finally, we assign a code of T to any second that contains both t and X (first And command), thereby identifying the first 20-second trial; we also assign codes of C and M to child and mother behaviors occurring during the 20-second trial (second and third And commands).

As you can see, the data modification commands of GSEQ can be treated almost as a primitive programming language, allowing you to create new codes with considerable flexibility. Here is another, final example. Image that you have coded fuss during mother-infant interaction, but as you examine your data, you realize that the average duration of fuss was typically just a few seconds and that, moreover, occurrences of fuss often tended to cluster near other occurrences of fuss. You decide that you want to create a new code representing fussy episodes, which you define as a string of fusses separated by no more than 5 seconds. In effect, you want to create a new code, which we will call Fussbout, by filling in gaps between individual fusses that are 5 seconds or less. The following GSEQ commands would accomplish this task:

```
Window after1 =fuss)+1,fuss)+1;
Window before5 =(fuss-5,(fuss-1;
And Startgap = after1 before5;
Window gap5 = Startgap+4;
Xor XStart = after1 Startgap;
Window exclude=  XStart+4;
Not include = exclude;
And Gap = gap5 include;
Or Fussbout = fuss Gap;
```

Figure 5 shows the effect of these commands on a small segment of example data. The first Window command assigns a code, after1, to the second immediately following any fussing event (i.e., from the 1st second after the offset of fuss up to and including the same second). If we had said only:

```
Window after1 =fuss)+1;
```

two seconds would be coded after1, both the offset second for fuss and the one following. The second Window command assigns a code, before5, to the five seconds preceding any fussing event (i.e., from the 5th second before the onset of fuss up to and including the 1st second before the onset of fuss). The first And command assigns a code of Startgap to any second that was coded both after1 and before 5; such seconds signal the possible start of the 5-second gaps we want to fill in. Then the third Window command fills in those gaps by assigning a code of gap5 to any series of five seconds beginning with a code of Startgap. However, as in the stringent test case presented in Figure 5, if one fuss event began less than 5 seconds from the end of another fuss event, gap5 codes might overrun the end of those seconds we intend to code Fussbout. To prevent this possibility, The Xor command assigns a code of XStart to any second that was coded after1 but not before5 (the Xor, or exclusive or, command assigns the new code to a time unit if one and only one of the old codes were already assigned to that time unit). Next, the fourth Window command assigns a code of exclude to such seconds and the following four seconds and the Not command assigns a code of include to any seconds not coded exclude. Then the second And command assigns a code of Gap to any seconds coded gap5 and include; these represent gaps we want to fill in, but also may co-occur with seconds already coded fuss. Therefore, finally, the Or command assigns a code of Fussbout to any seconds already coded fuss or Gap or both, which was the goal of this exercise.

You may not have reasons to derive new codes quite as complex as those exemplified in the previous few paragraphs. Our intent in presenting these examples was to suggest possibilities, including some you might not have thought of before, and to demonstrate that data reduction need not end with combining a few codes, especially if a powerful and flexible data reduction and modification program such as GSEQ is available. A thoughtful and sophisticated approach to data reduction and modification permits you to craft variables that are faithful to your research ideas. Such variables can themselves be combined into precisely targeted indices, which is the topic of the next section.

Describe and Analyze Patterns in Behavioral Streams

In the previous section we suggested that, at least for the time-based data we have been discussing, thinking in terms of lag-sequential analysis where lags are associated with time units such as seconds, applies too fine-grained a scale to be very useful (cf. Bakeman & Quera, 1995b). A more productive approach, we believe, involves developing targeted time-based indices in ways we demonstrate shortly. Such indices can be computed for individual participants (dyads, couples, etc.) and so can serve as scores in subsequent analyses that rely on any of the conventional techniques available in the standard statistical packages (e.g., analyses of variance, including repeated measured; multiple regression; structural equation modeling, etc.). Moreover, sequentially-based indices can be combined in a data set with other sorts of scores (e.g., age, sex, and other background information; scores on various self-report scales; etc.). From this point of view, sequential analysis and the analysis of behavioral streams generally is not a separate statistical domain. Instead, as we noted earlier, it is a measurement approach, one of many that may contribute to a research endeavor.

For an example, let us return to the Lavelli and Poli (1998) study we mentioned earlier. Recall that they coded infant state (sleeping, drowsy, alert, fussing, and crying). These codes were regarded as mutually exclusive and exhaustive, so that one (and only one) was coded for every second of the observation. They also coded when mothers were stimulating their infants with touch (tactile), with any sort of vocalization or verbalization (auditory), or both. They used the strategy of defining a combination code (both tactile and auditory) to create a mutually exclusive and exhaustive set (tactile, auditory, both, neither), but they could as easily have coded tactile (yes | no) and auditory (yes | no) stimulation separately, and then have created a combination code with the GSEQ Or command if they wished. Alternatively, given their four codes, it would also be easy enough with GSEQ’s data modification commands to create a data set that contained separate auditory (auditory or both coded) and tactile (tactile or both coded) stimulation codes. For the present example, then, we will assume a data set in which infant state, maternal tactile stimulation, and maternal auditory stimulation is coded.

Many research questions can assume the form, did Behavior B begin within a specified time window relative to Behavior A. For example, we might ask, did Behavior B begin within 10 seconds after Behavior A began, or within 15 seconds after Behavior A ended, or during a period of time that included when Behavior A was occurring and also 5 seconds after it ended. For the present example, assume that we want to know whether maternal stimulation appeared to result in infants becoming alert. More specifically, we want to know whether infants shifted to an alert state either during maternal auditory stimulation or within 5 seconds of its offset; likewise for tactile stimulation. Note, the answer could be yes for both, no for both, or yes for only one of these two forms of stimulation.

We would begin by creating two new codes. One would note whether or not an alert state began in each second, and the other would note whether or not each second was already coded for maternal stimulation or was within five seconds of the offset of such stimulation. After creating these new codes, we would ask GSEQ to tally them in appropriate 2×2 tables and compute appropriate statistics. The following GSEQ commands would create these codes, produce the required tables, and also produce an export file that could be imported into a statistical package such as SPSS:

```
File “C:\GSEQ\examples\LPex.mds”;
Window OnAlert = (Alert;
Window MTS5 = Tactile+5;
Window MAS5 = Auditory+5;
Stats jntf yulq phi odds lnor;
Target OnAlert &;
Given MAS5 &;
Target OnAlert &;
Given MTS5 &;
Export "C:\GSEQ\LPex.dat" jntf yulq phi odds lnor tabs label append;
```

The new codes are OnAlert, which identifies onset seconds for the alert state, MTS5, which identifies seconds coded for maternal tactile stimulation or five seconds thereafter, and MAS5, which identifies seconds coded for maternal auditory stimulation or five seconds thereafter (Tactile+5 means all seconds coded for Tactile and the five following seconds; likewise Auditor~~t~~y+5). The Stats command requests that various statistics be computed for the tables that follow. In this case they are joint frequencies, Yule’s Q, the phi coefficient, the odds ratio, and the log odds, all of which are defined shortly (other statistics are also available; see Bakeman & Quera, 1995a). The Target and Given commands request that two 2×2 tables be produced. The columns for both tables are labeled the same: In the first column, seconds coded for the onset of alert (OnAlert) are tallied, whereas all other seconds

(&, which here means, and everything else) are tallied in the second column. Similarly, in the first row for the first table, seconds coded for auditory stimulation or five seconds thereafter are tallied (MAS5), whereas all other seconds (&) are tallied in the second row. The second table is the same but for seconds coded for tactile stimulation or five seconds thereafter (MTS5). Finally, the Export commands asks GSEQ to write a file that contains these statistics, separating items with tabs, using subject labels from the SDS file (i.e., the info enclosed in angle brackets), and appending to any existing file with the name LPex.dat.

Imagine that the tables produced by GSEQ for a single 1-month old breast-fed infant and her mother had the following tallies, first for auditory stimulation:

Mother	Infant		
	OnAlert	&	Total
MAS5	20	460	480
&	20	1300	1320
Total	40	1760	1800

and, second for tactile stimulation:

Mother	Infant		
	OnAlert	&	Total
MVS5	10	530	540
&	30	1230	1260
Total	40	1760	1800

These manufactured data assume that mothers spent approximately 27% and 30% of the time engaging in auditory and tactile stimulation, respectively, and that there were 40 episodes (hence 40 onsets) of an infant alert state during a 30-minute feeding observation for this particular mother and infant.

As a descriptive matter we note that the probability of an infant becoming alert in any one second overall is .0222 (40/1800), but that the conditional probability of any infant becoming alert during seconds coded for maternal auditory stimulation (or 5 seconds thereafter) is somewhat greater, .0417 (20/480) ; the comparable number for tactile stimulation is less, .0185 (10/540). Thus the probability of OnAlert is greater than its baseline for auditory stimulation, but less than baseline for tactile stimulation. On its surface, this suggests that, for this infant at least, auditory but not tactile stimulation promoted an alert state.

However, what should we do next? Our claim is, most of the time most investigators who have time-based sequential data will be able to define 2×2 tables like the ones exemplified here, one for each of several constructs of interest. This requires a bit of forethought, of course, but forethought is the hallmark of reasonable, and reasoned, research. Once such constructs are defined, once the appropriate new codes have been added to the sequential data file (usually using GSEQ's Window command), and once tallies have been entered into those tables, what then? We need an index that summarizes, separately for each participant, the information in each table. Values of this index are then entered into a data file, along with other information about each participant, and then the data file is analyzed using whatever statistical techniques (e.g., multiple regression, structural equation modeling, etc.) you find appropriate.

Odds ratio. The question remains, what should that index be? There are several possibilities and it is worthwhile having some familiarity with them. Perhaps the most straightforward descriptively is the odds ratio, a statistic that is better know in epidemiology than psychology. Imagine that we label the cells of a 2×2 table as follows:

a	*b*
c	*d*

Then, to use our example data, the odds of an alert state beginning during a second coded for auditory stimulation are 20:460 or 1 to .043 or 23 to 1, whereas the odds of an alert state beginning in seconds not coded for auditory stimulation are 20:1300 or 1 to .015 or 65 to 1. That is, the odds of an alert state beginning are about 2.83 times greater with auditory stimulation than without it, which is the ratio of the two odds (i.e., the odds ratio). Formally, the odds ratio is defined as

$$\text{odds ratio} = \frac{a/b}{c/d} \qquad (3)$$

(where *a*, *b*, *c*, and *d* refer to observed frequencies for the cells of a 2×2 table as noted earlier; notation varies, but for definitions in terms of population parameters, see Bishop, Fienberg, & Holland, 1975; and Wickens, 1993). Multiplying numerator and divisor by *d/c*, this can also be expressed as

$$\text{odds ratio} = \frac{ad}{bc}\,. \qquad (4)$$

Equation 4 is more common although Equation 3 reflects the name and renders the concept more faithfully. To continue with our example, the odds ratio for maternal tactile stimulation is 0.774, which means that the odds of an alert state beginning during tactile stimulation is actually less than at other times.

The odds ratio can assume values from 0 (the odds for the first row are vanishingly small compared to the second row), to 1 (the odds for the two rows are the same), to infinity (the odds for the second row are vanishingly small compared to the first row). Especially when the odds ratio is greater than 1 (and it can always be made ≥ 1 by swapping rows), it has the merit, lacking in many indices, of a simple and concrete interpretation. Here, for example, because human minds seem to grasp numbers one or greater better than fractions less than one, we might better say that the odds of an alert state beginning are 1.29 times greater without tactile stimulation than with it (i.e., 1/0.774 = 1.29), as say that the odds of an alert state are 0.774 times greater with than without tactile stimulation. However, we would reverse rows in this way only if we wanted to highlight one particular table; if our purpose were to compare several tables, all with similar row and column headings but from different participants, we would of course want them all organized the same.

As useful as the odds ratio may be descriptively, due to its odd distribution (from 0 to 1 to infinity), it is not the best choice as a score for subsequent analyses. A better choice is the natural logarithm (ln) of the odds ratio, which is estimated as

$$\text{log odds ratio} = \ln\left(\frac{ad}{bc}\right), \tag{5}$$

and which extends from minus to plus infinity and equals 0 when there is no effect (i.e., when the odds in the two rows are the same). However, Equation 5 estimates are biased (Wickens, 1993). An estimate with less bias, which is also well defined when one of the cells is zero (recall that the log of zero is undefined), is obtained by adding ½ to each cell,

$$\text{log odds ratio} = \ln\frac{(a+½)(d+½)}{(c+½)(b+½)} \tag{6}$$

(Gart & Zwiefel, 1967; cited in Wickens, 1993, Equation 8). Thus, when describing the pattern captured by a 2×2 table, we think the odds ratio has much to recommend it. However, when subjecting a 2×2 table summary score to subsequent parametric analyses, such as *t* tests, analyses of variance, multiple regression, and the like, we recommend the estimate of the log odds ratio, computed per Equation 6.

For the present example, the log odds for auditory stimulation is 1.04 and for tactile simulation is –0.23. Although these numbers are logarithms (in this case exponents for *e*), for which most of us have little intuitive feel, these numbers do give the sense that the facilitating effect of auditory stimulation is greater than the inhibiting effect of tactile stimulation. However, their purpose is to serve as scores in subsequent parametric analyses. Here we have given two values derived from two 2×2 tables from one mother-infant dyad, but Lavelli and Poli (1998) had many such dyads. The infants they studied varied in age, and some were breast fed whereas others were bottle fed. Thus, had they wanted to explore this question in greater detail, they could have computed auditory and tactile tables, such as those displayed earlier, for all their subjects, computed pairs of log odds ratios per Equation 6, and subjected those scores to a paired-samples *t* test or a mixed model analysis of variance, with type of feeding (breast vs. bottle) and possibly age as between-subjects variables and type of stimulation as the repeated measure. Likely they would have discovered what their state transition diagrams suggested, that infants generally shifted into an alert state following maternal auditory but not tactile stimulation.

Yule's Q. Another useful descriptive statistic for 2×2 tables, derived from the odds ratio, is Yule's Q. It is simply an algebraic transformation of the odds ratio designed to vary, not from 0 to infinity with 1 indicating no effect, but from –1 to +1 with 0 indicating no effect, just like the familiar Pearson product-moment correlation. For that reason you might find it more useful descriptively than the odds ratio. First, *c/d* is subtracted from the numerator so that Yule's Q is zero when *a/b* equals *c/d*. Then, *a/b* is added to the denominator so that Yule's Q is +1 when *b* or *c* or both are zero and –1 when *a* or *d* or both are zero, as follows:

$$\text{Yule's Q} = \frac{\frac{a}{b} - \frac{c}{d}}{\frac{c}{d} + \frac{a}{b}} = \frac{\frac{ad - bc}{bd}}{\frac{bc + ad}{bd}} = \frac{ad - bc}{ad + bc}. \qquad (7)$$

Yule's Q can be expressed as a monotonically increasing function of the odds ratio, thus these two indices are equivalent in the sense of rank ordering subjects the same way (Bakeman, McArthur, & Quera, 1996).

For the present example, values of Yule's Q would be .48 and –.13 for auditory and tactile stimulation, respectively. As with the log odds, facilitating effects are indicated by values above zero and inhibiting effects by values below zero. When cell *c* or *b* is zero, values of Yule's Q will be +1, and when cell *a* or *d* is zero, values will be –1. Usually these values will make sense. For example, if all 40 times that an infant shifted to an alert state occurred during auditory stimulation, cell *c* would be zero and Yule's Q would be +1. Occasionally, however, zero cells can result in anomalous values of Yule's Q (e.g., when *a* is roughly equal to *c* and *b* is roughly equal to *d*, indicating no effect, but *b* and *d* contain quite small values, if either *b* or *d* were zero, a value of either +1 or –1 would result), a possibility to which investigators should be alert. Further, if any of the marginals (i.e., row and column totals) are zero, Yule's Q cannot be computed and its values should be regarded as missing.

Phi. Another extremely common index for 2×2 tables is the phi coefficient. This is simply the familiar Pearson product-moment correlation coefficient computed using binary coded data (Cohen & Cohen, 1983; Hays, 1963). A common definition for phi is

$$\phi = \frac{z}{\sqrt{N}} \tag{8}$$

where *z* is computed for the 2×2 table and hence equals $\sqrt{\chi^2}$. Thus phi can be viewed as a *z* score corrected for sample size. In terms of the four cells, phi is

$$\phi = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} \; . \tag{9}$$

Like Yule's Q it varies from –1 to +1 with zero indicating no association. For the present example, values of phi are .08 and –.02 for auditory and tactile stimulation, respectively. However, phi can achieve its maximum or minimum values only when the marginals are equal. As a result, Cureton (1959; see also Zysno, 1997), among others, recommends correcting phi to adjust for differences between marginals. Certainly, this would be desirable whenever one wanted to compare phi coefficients from tables with differently skewed marginals. For the present example, values for corrected phi are .32 and –.17, which were computed by dividing .08 by its maximum positive value and –.02 by its minimum negative value, respectively.

The differences between Yule's Q and phi are relatively subtle (Bakeman, McArthur, & Quera, 1996), so as a practical matter it rarely matters which is used, or whether another index such as the odds ratio or log odds ratio is used instead. In all cases, however, attention should be paid to zero or low frequency cells in the 2×2 tables. Zero cells often result in computational problems. For example, if either *b* or *c* or both are zero, an odds ratio cannot be computed (divide by zero). Likewise, none of these statistics (except the log odds per Equation 6) can be computed if any of the marginals are zero. A zero in a single cell can be informative (which is why Equation 6 is useful) but a zero or even a very low marginal suggests that insufficient information is available to estimate a value for any of these statistics. Our own rule of thumb is this: If any marginal is less than 5, we do not compute statistics for that 2×2 table but instead regard their values as missing.

In sum, in order to describe patterns in behavioral streams, we recommend first defining a series of 2×2 tables that reflect the constructs and questions of your research. These 2×2 tables can be described with whichever of these statistics seems useful to you: the odd ratio, Yule's Q, or phi. However, for parametric analysis, we recommend that scores be log odds ratios, for reasons given earlier. It remains only to say that all of these statistics can be computed using GSEQ, and that we assume such scores will be integrated with other data sets containing scores from other sources, using different measurement approaches. Finally, we assume that such merged data sets will ultimately be analyzed with standard statistical programs such as SPSS, SAS, LISREL, and the like.

Summary

Throughout this chapter we have discussed methods and means of analyzing information that may lie within behavioral streams, which we have presented as a series of seven steps. We began by discussing the need to begin with clear concepts and a clear sense of the research questions that motivate our work. We noted that coding schemes, which we view as the primary measuring instruments of observational research, are informed by those concepts and then honed in practice. We also provided several examples of coding schemes that we and others have found useful and discussed how coding schemes are used to capture data from behavioral streams. Sometimes behavior is observed live, but more often it is first recorded, which permits reviewing. We emphasized that information concerning the timing and duration of behavior is highly desirable and, given current technology, easy enough to preserve. Even so, there are a variety of ways information about the timing and occurrence of behavior could be represented for subsequent analysis. We described a standard notation that we have developed (SDIS or the Sequential Data Interchange Standard; Bakeman & Quera, 1995a) and gave several examples of how sequential data would be expressed using SDIS conventions. We also noted the importance of demonstrating, to ourselves and others, that the observers on whom we rely to code the behavior of interest to us do so reliably. We emphasized Cohen’s kappa, in particular, and also discussed the usefulness of visualizing observer agreement using time plots. Next we discussed ways to first describe and then reduce the data as collected initially. We provided several examples of how data description and reduction can be effected with a computer program we have developed (GSEQ or the Generalized Sequential Querier; Bakeman & Quera, 1995a). Some of these examples were quite basic but others were more complex. They demonstrated, first, the fairly unusual flexibility of GSEQ’s data modifications capabilities, but more importantly, they suggested the importance of thoughtfully deriving scores from our data that truly reflect the concepts and questions that motivated our research in the first place. Once that is done, it a relatively straightforward matter to describe and analyze whatever patterns exist in our sequential data. The approach we emphasized requires that 2×2 tables be defined for sequences of interest, that time units (seconds, usually) be tallied into these tables, and that various indices be derived from these tables (e.g., an odds ratio to describe the data, a log odds ratio to analyze it). Finally, such indices would be merged with other data sets and this ultimate data set would then contain scores derived from both behavioral streams and other sources. It would then serve as the basis for the usual sorts of analyses, such as multiple and logistic regression, analyses of variance and covariance, and structural equation modeling, as well for other more innovative analytic techniques such as those described in other chapters in this section (i.e., Section IV) of the present volume.

References

Adamson, L. B., & Bakeman, R. (1984). Mothers' communicative actions: Changes during infancy. *Infant Behavior and Development*, *7*, 467-478.

Adamson, L. B., & Bakeman, R. (1991). The development of shared attention in infancy. In R. Vasta (Ed.), *Annals of Child Development* (Vol. 8, pp. 1-41). London: Kingsley.

Adamson, L. B., Bakeman, R., Deckner, D. F., & Dunbar, B. (2000, July). A developmental perspective on the joint attention deficit in autism. In M. Harris (Chair), *Typical and atypical pathways to symbolic communication*. Symposium presented at the biennial meeting of the International Conference on Infant Studies, Brighton, England.

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

Bakeman, R. (2000a). Behavioral Observations and Coding. In H. T. Reis & C. K. Judd (Eds.), *Handbook of research methods in social psychology* (pp. 138-159). New York: Cambridge University Press.

Bakeman, R. (2000b). VitcVert: A program for converting James Long Company data files to SDIS files (Tech. Rep. No. 16). Atlanta, GA: Georgia State University, Developmental Psychology Laboratory.

Bakeman, R., & Adamson, L. B. (1984). Coordinating attention to people and objects in mother-infant interaction. *Child Development*, *55*, 1278-1289.

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

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

Bakeman, R., McArthur, D., & Quera, V. (1996). Detecting group differences in sequential association using sampled permutations: Log odds, kappa, and phi compared. *Behavior Research Methods, Instruments, and Computers, 28*, 446-457.

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

Bakeman, R., & Quera, V. (1995b). Log-linear approaches to lag-sequential analysis when consecutive codes may and cannot repeat. *Psychological Bulletin, 118,* 272-284.

Bakeman, R., & Quera, V. (2000). OTS: A program for converting Noldus observer data files to SDIS files. *Behavior Research Methods, Instruments, and Computers, 32,* 207-212.

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.

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

Bishop, Y. M. M., Fienberg, S. R., & Holland, P. W. (1975). *Discrete multivariate analysis: Theory and practice.* Cambridge, MA: MIT Press.

Brazelton, T. B. (1973). *Neonatal behavior assessment scale*. Philadelphia: Lippincott.

Cohen, J. A. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement, 20*, 37-46.

Cohen, J. (1968). Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin*, *70,* 213-220.

Cureton, E. E. (1959). Note on Phi / Phi (max), *Psychometrika, 24 (1),* 89-91.

Emerson, E., Reeves, D.J., & Felce, D. (2000). Palmtop computers for behavioral observation research. In T. Thompson, D. Felce, & F.J. Symons (Eds.), *Behavioral observation: Technology and applications in developmental disabilities* (pp. 47-59). Baltimore, MD: Paul H. Brookes.

Fleiss, J. L. (1981). *Statistical methods for rates and proportions*. New York: Wiley.

Kahng, S.W., & Iwata, B.A. (2000). Computer systems for collecting real-time observation data. In T. Thompson, D. Felce, & F.J. Symons (Eds.), *Behavioral observation: Technology and applications in developmental disabilities* (pp. 35-45). Baltimore, MD: Paul H. Brookes.

Konner, M. J. (1976). Maternal care, infant behavior, and development among the !Kung. In R. B. DeVore (Eds.), *Kalahari hunter-gathers* (pp. 218-245). Cambridge, MA: Harvard University Press.

Lavelli, M., & Poli, M. (1998). Early mother-infant interaction during breast- and bottle-feeding. *Infant Behavior and Development, 21*, 667-684.

Long J. (1996). *Video coding system reference guide*. Caroga Lake, NY: James Long Company.

Martin, P.R., & Bateson, P.P. (1993). *Measuring Behaviour: An Introductory Guide* (2nd ed.). Cambridge, UK: Cambridge University Press

Martin, N., Oliver, C., & Hall, S. (1999). ObsWin: Observational data collection and analysis for Windows*.* *CTI Psychology Software News, 9*, 14-16.

Noldus, L. P. J. J., Trienes, R. J. H., Henriksen, A. H. M., Jansen,H., & Jansen, R. G. (2000). The Observer Video-Pro : New software for the collection, management, and presentation of time-structured data from videotapes and digital media files. *Behavior Research Methods, Instruments, and Computers, 32*, 197-206.

Parten, M. B. (1932). Social participation among preschool children. *Journal of Abnormal and Social Psychology, 27*, 243-369.

Quera, V., & Bakeman R. (2000). Quantification strategies in behavioral observation research. In Thompson, T., Felce, D., & Symons, F. J. (Eds.), *Behavioral observation: Technology and applications in developmental disabilities*. (pp. 297-315). Baltimore, MD: Paul H. Brookes.

Robinson, B. F., & Bakeman, R. (1998). ComKappa: A Windows 95 program for calculating kappa and related statistics. *Behavior Research Methods, Instruments, and Computers, 30*, 731-732.

Sackett, G. P. (1978). Measurement in observational research. In G. P. Sackett (Ed.), *Observing behavior* (Vol. 2): *Data collection and analysis methods* (pp. 25-43). Baltimore: University Park Press.

Sackett, G. P. (1979). The lag sequential analysis of contingency and cyclicity in behavioral interaction research. In J. D. Osofsky (Ed.), *Handbook of infant development* (1st ed., pp. 623-649). New York: Wiley.

Tapp, J., & Walden, T. (1993). PROCODER: A professional tape control coding and analysis system for behavioral research using videotape. *Behavior Research Methods, Instruments, and Computers, 25*, 53-56.

Wickens, T. D. (1989). Multiway contingency tables analysis for the social sciences. Hillsdale, NJ: Erlbaum.

Wickens, T. D. (1993). Analysis of contingency tables with between-subjects variability. Psychological Bulletin, 113, 191-204.

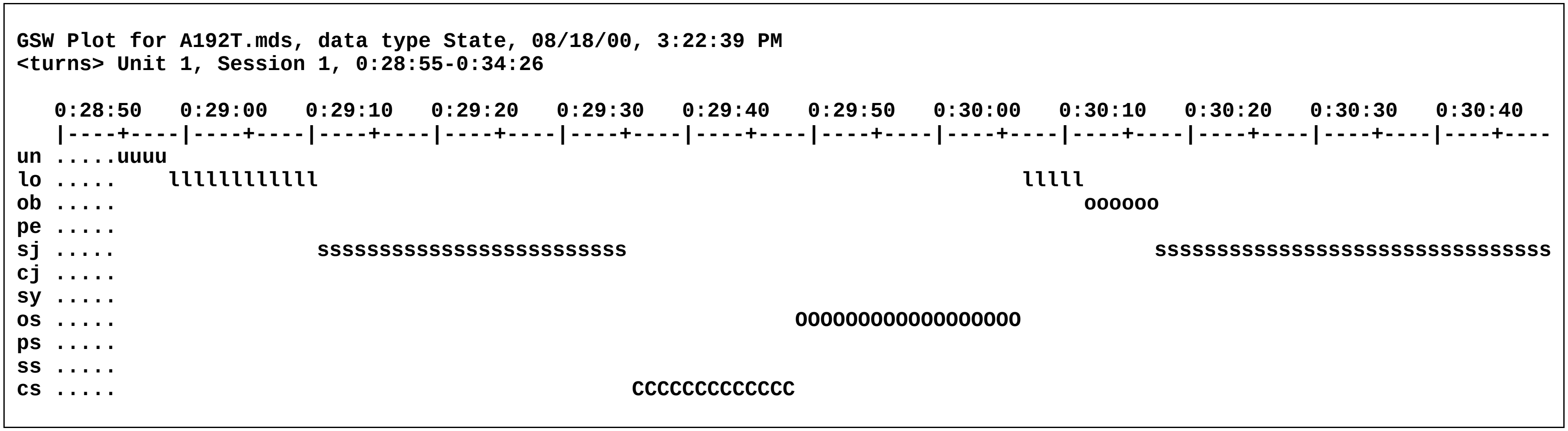
Wolff, P. (1966). The causes, controls, and organization of the neonate. *Psychological Issues, 5* (whole No. 17).

Umesh, U. N., Peterson, R. A., Sauber, M. H. (1989). Interjudge agreement and the maximum value of kappa. *Educational and Psychological Measurement*, *49*, 835-850.

Zysno, P. (1997). The modification of the phi-coefficient reducing its dependence on the marginal distributions. *Methods of Psychological Research Online, 2* (1). Available: http://www.ppm.ipn.uni-kiel.de/mpr/issue2/art5/article.html

Author Note

Correspondence concerning this article should be addressed to Roger Bakeman, Department of Psychology, Georgia State University, Atlanta, GA 30303, USA. Electronic mail may be sent via Internet to bakeman@gsu.edu.

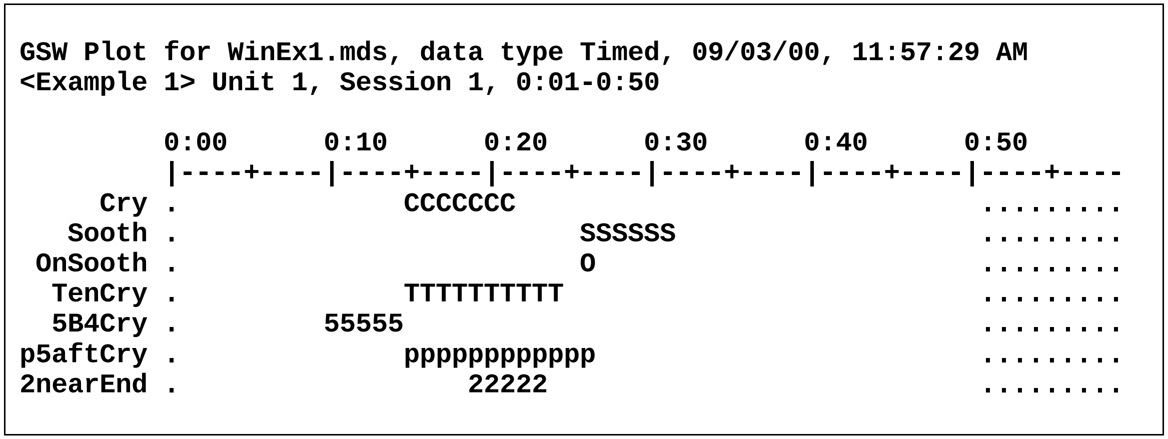


```
GSW Plot for A192T.mds, data type State, 08/18/00, 3:22:39 PM
<turns> Unit 1, Session 1, 0:28:55-0:34:26

   0:28:50   0:29:00   0:29:10   0:29:20   0:29:30   0:29:40   0:29:50   0:30:00   0:30:10   0:30:20   0:30:30   0:30:40
   |----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----
un .....uuuu
lo .....    llllllllllll                                                        lllll
ob .....                                                                             oooooo
pe .....
sj .....                sssssssssssssssssssssssss                                          ssssssssssssssssssssssssssssssss
cj .....
sy .....
os .....                                                      OOOOOOOOOOOOOOOOOO
ps .....
ss .....
cs .....                                         CCCCCCCCCCCCC
```

Figure 1. Example of a data segment using the engagement state codes described in the text. This and subsequent plots were produced with the GSEQ for Windows (GSW) plot routine (Explore | Plot).

```
GSW Agreement/Kappa, 08/18/00, 3:29:21 PM
Agreement plot:  line 1 is A192T, line 2 is B192T

0:28:50   0:29:00   0:29:10   0:29:20   0:29:30   0:29:40   0:29:50   0:30:00   0:30:10   0:30:20   0:30:30   0:30:40
|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----
     uuuullllllllllllsssssssssssssssssssssssssCCCCCCCCCCCCCOOOOOOOOOOOOOOOOOOllllloooooossssssssssssssssssssssssssssssss
       uuuuuullllllsssssssssssssssssssssssssssSSSSSSSSSSSSSOOOOOOOOoooooooooollllloooooollllllllssssssssssssssssssssssss
     ##  --..      ..                         -------------        ..--------           ------..

0:30:50   0:31:00   0:31:10   0:31:20   0:31:30   0:31:40   0:31:50   0:32:00   0:32:10   0:32:20   0:32:30   0:32:40
|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----
ssssssssssOOOOOOOOOOOOOOOOOOOOOOsssuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuoooooooooooooooo
ssssssssssssSSSSSSSSSSSSSSSSSSSSSSSuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuoooooooooooooooo
          -------------------------

0:32:50   0:33:00   0:33:10   0:33:20   0:33:30   0:33:40   0:33:50   0:34:00   0:34:10   0:34:20
|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----
ooooooooooosssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssllllllllllll
oooooooooooossssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssuuuuuuuuuuuu
           .                                                                       ..----------##


Cohen's kappa =  0.7360, agreement =  80.49%, window = +|-  2
Rows: A192T, columns: B192T

              un      lo      ob      pe      sj      cj      sy      os      ps      ss      cs       &  Totals
----------------------------------------------------------------------------------------------------------------
     un       71       0       0       0       0       0       0       0       0       0       0       0      71
     lo       11      15       0       0       0       0       0       0       0       0       0       0      26
     ob        0       0      33       0       0       0       0       0       0       0       0       0      33
     pe        0       0       0       0       0       0       0       0       0       0       0       0       0
     sj        1       6       0       0     135       0       0       0       0       3       0       0     145
     cj        0       0       0       0       0       0       0       0       0       0       0       0       0
     sy        0       0       0       0       0       0       0       0       0       0       0       0       0
     os        0       0       8       0       2       0       0      10       0      20       0       0      40
     ps        0       0       0       0       0       0       0       0       0       0       0       0       0
     ss        0       0       0       0       0       0       0       0       0       0       0       0       0
     cs        0       0       0       0       0       0       0       0       0      13       0       0      13
      &        0       0       0       0       0       0       0       0       0       0       0       0       0
----------------------------------------------------------------------------------------------------------------
 Totals       83      21      41       0     137       0       0      10       0      36       0       0     328
               u       l       o       p       s       c       Y       O       P       S       C
```

Figure 2. Example of an agreement plot and kappa calculation; conventions for the plot are described in the text. This output was produced with the GSEQ for Windows (GSW) kappa routine (Tools | compute Kappa).



```
GSW Plot for WinEx1.mds, data type Timed, 09/03/00, 11:57:29 AM
<Example 1> Unit 1, Session 1, 0:01-0:50

         0:00      0:10      0:20      0:30      0:40      0:50
         |----+----|----+----|----+----|----+----|----+----|----+----
     Cry .              CCCCCCC                             .........
   Sooth .                         SSSSSS                   .........
 OnSooth .                         O                        .........
  TenCry .              TTTTTTTTTT                          .........
  5B4Cry .         55555                                    .........
p5aftCry .              pppppppppppp                        .........
2nearEnd .                  22222                           .........
```

Figure 3. Examples of new codes produced from original codes of Cry and Sooth using the data modification commands given in the text.

```
GSW Plot for WinEx2.mds, data type Timed, 09/04/00, 10:02:29 AM
<Example 2> Unit 1, Session 1, 0:00-0:59

     0:00      0:10      0:20      0:30      0:40      0:50
     |----+----|----+----|----+----|----+----|----+----|----+----
Look      LLLL       LLLL                 LLLLL
  mb           mmmm          mmmmmm   mm           mmmmm
  cb             cccc          cc               cccc
   t      tttttttttttttttttttt+tttttttttt tttttttttttttttttttt
   x                          xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
   X XXXXXXXXXXXXXXXXXXXXXXXXX
   T      TTTTTTTTTTTTTTTTTTTT
   C             CCCC
   M           MMMM          M
```

Figure 4. Examples of new codes produced from original codes of Look, mb, and cb using the data modification commands given in the text.

The new code T identifies a 20-second trial keyed to the first look to the mother.

```
GSW Plot for WinEx3.mds, data type Timed, 09/04/00, 1:56:12 PM
<Example 3> Unit 1, Session 1, 2:01-3:14

         2:00      2:10      2:20      2:30      2:40      2:50      3:00      3:10
         |----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----
    fuss .         fffff   f f      f     ff  fffffff     ff      ff                .....
  after1 .              a   a a      a      a        a      a       a               .....
 before5 .    bbbbb   bbbbb+b  bbbbb bbbbb+bbb       bbbbb   bbbbb                  .....
Startgap .              S   S        S      S        S                              .....
    gap5 .              ggggg+ggg    ggggg  ggggg    ggggg                          .....
  XStart .                    X                             X       X               .....
 exclude .                    eeeee                         eeeee   eeeee           .....
 include .iiiiiiiiiiiiiiiiiiii     iiiiiiiiiiiiiiiiiiiiiiiii     iii     iiiiiiiiiii.....
     Gap .              GGGGGG       GGGGG  GGGGG    GGGGG                          .....
Fussbout .         FFFFFFFFFFF      FFFFFFFFFFFFFFFFFFFFFFFF      FF                .....
```

Figure 5. Example of a new code, Fussbout, created by filling in gaps of five seconds or less between occurrences of an original code, fuss. The plus sign (e.g., the bbbbb+b at 2:13) indicates the onset of a second event immediately after the offset of a first event with the same code. In this example, the Window command created overlapping events; the first began at 2:13 and ended at 2:17 inclusive, whereas the second, which initially ran from 2:15 to 2:19 inclusive is shown as beginning at 2:18, immediately after the first event ended.