7. A FRAMEWORK FOR THE EVALUATION OF SOFTWARE FOR TEACHING STATISTICAL CONCEPTS

Robert C. delMas
The University of Minnesota

INTRODUCTION

As an instructor of introductory statistics, I have developed several computer applications aimed at facilitating students' development of statistical concepts. The software design has been influenced by my experience in teaching statistics; the knowledge of learning, problem solving, and cognition that I have gathered as a cognitive psychologist; and my experience with computers as both a computer user and a computer programmer. My teaching experience has left me with the impression that many students find it particularly difficult to gain a rich understanding of some statistical concepts, such as the central limit theorem. The ideas in cognitive psychology and learning that I gravitate toward hold that students develop concepts by actively testing ideas (i.e., hypotheses) based on conjectures and implications formulated as they try to make meaning out of their experiences.

One piece of software that I have developed is called *Sampling Distributions*. The software is probably best described as a simulation in that it allows students to change settings and parameters in order to test implications of the central limit theorem. *Sampling Distributions* allows a student to manipulate the shape, and, consequently, the parameters of a parent population (see Figure 1). In the Population window, a student can select from a set of predefined population shapes or create their own population distribution. The shape of the population distribution is changed by using up and down arrows that "push" the curve up or down at various points along the number line. As the distribution's shape changes, *Sampling Distributions* calculates parameters such as the population mean, median, and standard deviation and displays the numerical values. Once the student is satisfied with the look of the population, he/she can draw samples.

The Sampling Distributions window (see Figure 2) allows a student to draw samples of any size and to designate how many samples will be drawn. The program calculates the mean and median for each sample and presents graphic displays of the distributions of the means and medians, that build up rapidly in real time, one sample mean at a time. Summary statistics are calculated and presented for the sampling distributions: the means of the sample means standard deviations and the standard deviation of the sample means. The sampling distribution statistics can then be compared to the population parameters and to theoretical values such as $\frac{\sigma}{\sqrt{n}}$ where σ is the standard deviation and n is the sample size.

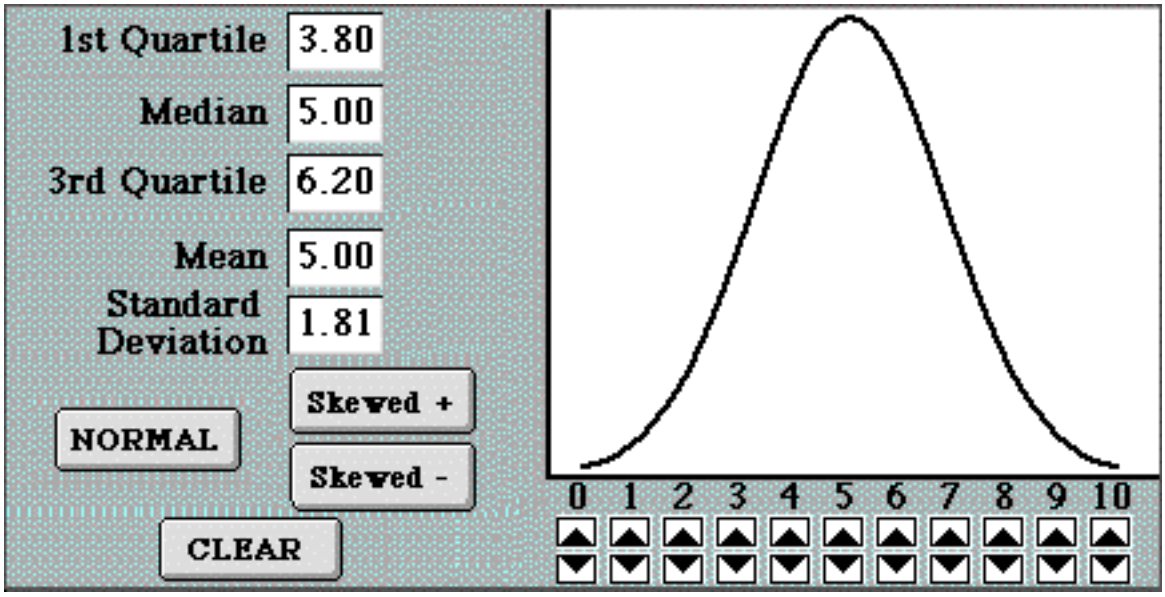

Figure 1: The population window of the *Sampling Distributions* program

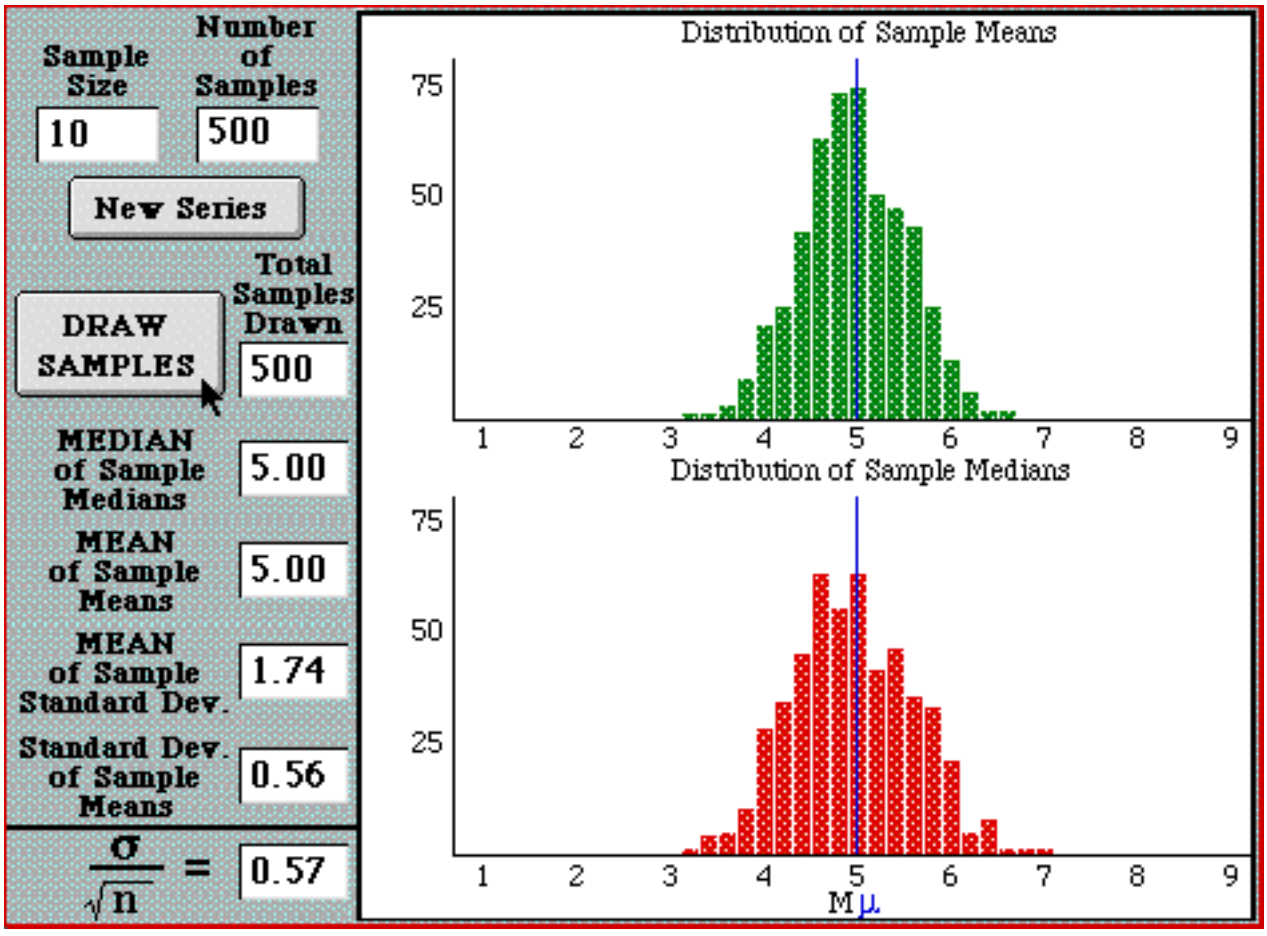

Figure 2: The sampling distributions window

I have revised *Sampling Distributions* often. It started as a HyperCard stack, was transformed into a stand-alone Pascal application for the Macintosh, and since then has gone through about four major revisions. The changes and revisions have been influenced by several sources of feedback: my personal beliefs about how students learn as outlined above, comments from colleagues who have used the program in their classrooms, what appears to work or not to work as I have observed students using *Sampling Distributions*, and the written comments of students who have offered evaluations and "wish lists" of features. These have been the informal sources of evaluation and assessment I have relied on to improve the program.

Now that the program has developed to a stage that I believe is fairly complete, I find myself wanting to address more substantive questions of evaluation and assessment. The questions fall into two areas. The first question is one of value or quality: To what extent does the *Sampling Distributions* program display the characteristics of a "good" piece of educational software? The second question is one of effect or impact: To what extent does the *Sampling Distributions* program facilitate students' development of concepts related to the central limit theorem? The first question requires a set of criteria or characteristics against which the *Sampling Distributions* program can be compared. The second question requires a rationale to guide the design of measures and outcomes that will provide meaningful feedback about the state of students' concepts as a result of having interacted with the program. The remainder of this paper will attempt to address the first question by providing a rationale for a set of characteristics that define "good" educational software. The *Sampling Distributions* program will then be compared against the set of characteristics.

DEFINING UNDERSTANDING

Software Goes to School: Teaching for Understanding with New Technologies (Perkins, Schwartz, West, & Wiske, 1995) provides the means for developing a list of software characteristics. The book consists of 15 chapters by numerous authors. Many of the authors have engaged in the systematic exploration of how technology can help students learn with understanding in science, mathematics, and computing. The various chapters present definitions and frameworks that address what it means to learn with understanding, ideas of what constitutes software that facilitates understanding, and examples of software programs that incorporate these ideas.

At the heart of the book is the conviction that only software that facilitates conceptual understanding will be discussed. The authors recognize that a definition of understanding is needed in order to provide a meaningful discussion of how software should be designed to promote understanding. Several authors offer perspectives on what it means to learn with understanding or to behave in ways that reflect a deep understanding of a concept. Nickerson (1995) offers a list of abilities that might constitute a demonstration of understanding. An individual demonstrates understanding when he or she can:

- Explain a result or observation to the satisfaction of an expert.
- Apply knowledge appropriately in various contexts.
- Produce appropriate qualitative representations for a concept.
- Make appropriate analogies.
- Effectively repair malfunctions (of ideas, predictions, models).
- Accurately predict the effects of change in structure or process.

Although this list provides an intuitive guide for determining when understanding is demonstrated, it does not provide an operational definition or a reliable means for identifying understanding. For the most part, determining whether or not a student's learning has led to understanding is left to the judgment of an expert (i.e., the instructor).

A similar list is delineated in a chapter by Perkins, Crismond, Simmons, and Unger (1995). A student demonstrates understanding when he/she provides an *explanation* that:

- Provides examples.
- Highlights critical features of concepts and theories.
- Demonstrates generalization or application of concepts to a new context through the revision and extension of ideas.
- Provides evidence of a relational knowledge structure constructed of a complex web of cause and effect relationships.

Although this list goes a step further than the Nickerson (1995) list by identifying a vehicle for observing understanding (explanation), judgment on the part of the instructor (or a body of experts) is still required to identify appropriate examples, a list of critical features, appropriate generalizations, and, perhaps, the nature of the knowledge structure. As Nickerson (1995) points out, it is difficult to define understanding without being circular. This is pointed out again in a chapter by Goldenberg (1995) in which he discusses how computer simulations can be used to conduct research aimed to develop an understanding of understanding. In other words, software designed to facilitate understanding is used to gain insight into the nature of understanding, which presents a somewhat circular arrangement.

Nonetheless, a common theme presented in the book (Perkins et al., 1995) is that understanding is more than just a collection of discrete pieces of information that might result from the rote memorization of terms and definitions or hours of drill and practice. In summary, the authors of *Software Goes to School* see understanding represented internally as a dynamic knowledge structure composed of complex interconnections among concepts that enables the individual to effectively anticipate events, predict outcomes, solve problems, and produce explanations, even under somewhat novel conditions. Understanding can also be thought of as more than just a state of knowledge. Nickerson (1995) suggests that understanding can also refer to the set of mental processes that produce a knowledge structure that is more adequate, functionally, than the structure previously held by a learner. Understanding is both a state and the process by which the state is produced, which again sounds somewhat circular. Perhaps it is the elusive nature of understanding that makes it so difficult to teach and for students to learn.

MODELS OF UNDERSTANDING

So far, understanding is described (1) as a knowledge structure built of complex relationships among concepts that allows an individual to produce explanations and predictions, and (2) as a set of processes that enable an individual to revise old structures and produce new ones in order to deal with new contexts and problems. A better description of the structure and processes that underlie this dualistic view of understanding is needed in order to provide recommendations for software design that will promote the development of understanding.

Several of the authors of *Software Goes to School* (Perkins et al., 1995) offer models for both the structure and process of understanding that range from the general to the specific. Carey and Smith (1995) believe that the development of understanding is best captured by a constructivist perspective. They suggest that the traditional view is that scientific understanding is developed solely by empirical observation or as the product of inductivistic activity. They argue, and provide some evidence, that both the cognitive development of understanding and the activity of scientists is better represented by a constructivist epistemology in which beliefs held by the individual guide the generation of testable predictions and the interpretation of observations. The authors provide a broad perspective regarding the type of activity that is supported by understanding and that leads to the extension or development of new understanding. However, the description does not present much detail about the processes that develop understanding or that allow an individual to act with understanding.

Perkins et al. (1995) provide a model developed by Perkins (1993) called the *Access Framework* to account for differences in peoples' ability to produce explanations. The model proposes that information resides in memory in the form of explanation structures, an entity that sounds similar to the notion of a schema or script (Rumelhart, 1980; Schank & Abelson, 1977). According to the Access Framework, some parts of the explanation structure are well-rehearsed and form a foundation, but other parts are novel, created at the moment as extensions of the structure.

The Access Framework holds that an individual needs access to four types of resources to produce effective explanations. The four types of access are:

- Access to knowledge that is relevant to a situation, context, or problem.
- Access to appropriate representational systems.
- Access to retrieval mechanisms that can recover relevant information from memory or external sources.
- Access to mechanisms (processes) that produce conjectures and elaborations in order to extend knowledge to new situations or to build new explanation structures.

Perkins et al. (1995) prescribe the types of instruction that help develop an effective explanation structure. In addition to content knowledge, instruction should develop a broad repertoire of problem-solving strategies to promote flexibility. Representations should be provided that are concrete and provide salient examples of concepts that are not familiar to the student. Learning should occur in problem-solving contexts in order to promote rich interconnections among concepts and to facilitate memory retrieval. Activities should be designed that require students to extend and test ideas so that knowledge structures are elaborated and extended. Although these prescriptions take a step toward defining software features that promote learning for understanding, the account is still lacking as a description of the structure and processes that constitute understanding.

Perkins et al. (1995) do provide some detail about two aspects of the Access Framework--the structure of knowledge and the mechanisms used to retrieve information. They rely on connectionist models of memory organization such as Anderson's (1983) ACT model of spreading activation. In the ACT model, accessing information in one part of an information structure can prime other information for retrieval given that relational links have been established through prior learning. This, however, is the extent of the detail; no clear description of either how the connections are formed or the conditions that promote the development of connections is given.

Holland, Holyoak, Nisbett, and Thagard (1987) have argued that although connectionist models of memory organization can account for a wide variety of phenomena, they have their limitations. Strong evidence for spreading activation is provided by studies of lexical priming (Collins & Loftus, 1975; Fischler, 1977; Marcel, 1983; Meyer & Schvaneveldt, 1971). However, there is much evidence suggesting that spreading activation does not occur automatically and that it does not spread throughout the entire set of concepts connected to the initially accessed information (de Groot, 1983; Thibadeau, Just, & Carpenter, 1982). Holland et al. argue, instead, that many memory and problem solving phenomena are better explained by a model in which the spread of activation is mediated by rules that vary in their strength of association and compete with each other for entry into working memory.

The goal of Holland et al. (1987) was to describe a model or framework that would begin to account for inductive reasoning. I will refer to the Holland et al. framework as the *Inductive Reasoning Model*. For Holland et al., an inductive reasoning system allows a person to:

- Organize experience in order to produce action even in unfamiliar situations.
- Identify ineffective rules for action.
- Modify or generate rules as replacements for ineffective rules.
- Refine useful rules to produce more optimal forms.
- Use metaphor and analogy to transfer information from one context to another.

This list of abilities and the Access Framework have many similarities.

I will attempt to describe the features of the Inductive Reasoning Model that are most relevant to the discussion of how students develop understanding. Not all the activity of an inductive reasoning system involves induction. According to the model, an individual is motivated to develop a knowledge structure that provides accurate predictions. As long as correct predictions can be made by the system, very little change is made to the system. Inductive processes are triggered by the failure of the system to make effective, successful predictions. This assumption is similar to the constructivist perspective of Carey and Smith (1995) mentioned above. Rules that lead to successful prediction are strengthened, and those that do not are modified or discarded. A rule's strength is directly related to its ability to compete with other rules that have their conditions satisfied. An important aspect of the Inductive Reasoning Model is that more than one rule can be selected, and a limited capacity for parallel processing is assumed so that rules with the highest "bids" can run simultaneously.

Rules are the cornerstone of the Inductive Reasoning Model. Rules are represented as condition-action pairs similar to the production rules defined by Newell (1973; Newell & Simon, 1972). The Inductive Reasoning Model proposes that information is stored in the form of production rules rather than static pieces of information connected by links. Production rules are essentionally if-then statements: If the condition is met, then the action is taken. An inductive reasoning system "decides" what action to take by identifying rules whose conditions are met, by using processing rules to determine which rules with met conditions are to be executed, and by executing the selected subset of rules. One implication is that predictions and explanations are not formed by strict pattern matching, as might be proposed by behaviorist or connectionist points of view, but rather are often constructed on the spot.

There are several activities that an inductive reasoning system performs to support the development of understanding: rules that produce reliable predictions are strengthened; rules that are commonly executed together become associated; the rule system is modified through the identification of new objects and

events to which rules can be effectively applied; and new rules are produced to account for exceptions. When a rule or set of rules produces useful and effective predictions, the primary change that occurs in the inductive reasoning system is the strengthening of the rule set. When a novel object or situation is encountered, the inductive reasoning system identifies a set of rules with the highest strengths that have their conditions met. These rules are modified by adding the unique features of the novel object or situation to the conditions of the production rules.

Whether the new extended rules gain strength, become modified, or are discarded depends on their future utility. Finally, objects and events that match the conditions at higher levels in the hierarchy but produce failures will invoke general system rules to generate new rules that record the failures as exceptions.

One critical implication of the Inductive Reasoning Model is that effective prediction is dependent on an ability to encode critical features of the environment. Holland et al. (1987) propose that an inductive reasoning system has three broad responses to failure. Extreme failure encountered during the early stages of learning prompts an inductive reasoning system to find a new way to categorize the environment. This is dependent on an organism's ability to encode features of the environment that are relevant to accurate prediction. As learning progresses and successful predictions become more frequent, failure will result in the generation of rules that account for exceptions. Again, the system must encode relevant features of the environment that can be incorporated into the condition part of an exception rule. Late in learning, after the default hierarchy has become quite specialized, failure is likely to be treated as evidence of uncertainty. At this stage, failure results in the generation of rules that produce probability estimates instead of more specific predictions. In general, the success of an inductive reasoning system is dependent on its ability to detect and categorize features of the environment that evoke rules, that can be used to modify existing rules, or that become the building blocks for the generation of new rules.

Holland et al. (1987) elaborate on how bids are determined, describe the nature of system processes that might be used to modify rules and generate new ones, and provide convincing examples of how the model can account for behaviors ranging from operant conditioning to complex problem solving. These details are beyond the scope of this paper (and, at times, my full comprehension). Nonetheless, the features presented above provide some implications for conditions that support the development of understanding. I propose that the ability to provide explanations and the ability to make predictions stem from similar knowledge structures and cognitive processes, and that both of these abilities reflect understanding. The Inductive Reasoning Model suggests that understanding is facilitated when learning conditions are established that (the I prefix is used in reference to the **I**nductive Reasoning Model):

I1. Facilitate the student's encoding of relevant information by relating new entities to familiar, well-rehearsed encodings.

I2. Promote the simultaneous encoding of key features of the environment.

I3. Provide ways for the student to make and test predictions.

I4. Promote frequent generation and testing of predictions.

I5. Provide clear feedback on the outcomes of predictions.

I6. Make it clear to the student which features of the environment are to be associated with the outcome of a prediction.

I7. Promote the encoding of information that relates the variability of environmental features to specific actions.

This set of recommendations emphasizes the roles that encoding, prediction, and feedback play in the development of understanding. All three are needed to support the strengthening, revision, and generation of rules, as well as the fleshing out and extension of categories and concepts. Consistent with the definitions of understanding provided by Nickerson (1995) and Perkins et al. (1995), the Inductive Reasoning model implies that a student's ability to provide explanations and to solve problems is dependent on the development of a complex web of interrelationships among concepts (i.e., rules).

SOFTWARE FEATURES THAT FOSTER UNDERSTANDING

Several of the authors in *Software Goes to School*. (Perkins et al., 1995) prescribe conditions that promote the development of understanding. Nickerson (1995) provides five maxims for fostering understanding that place an emphasis on encoding, exploration, and prediction, which is consistent with the characteristics of an inductive reasoning system. Nickerson makes the important point that although technology does not promote understanding in and of itself, it does represent a tool that can readily incorporate the five principles listed above. Real-world models can be developed as explorable microworlds that allow students to test out assumptions, make predictions, highlight misconceptions so that they "stand out," and promote active processing by changing parameters and defining entities. Computer simulations can present dynamic representations that go beyond the modeling of static entities by making the processes that produce phenomena more concrete and observable.

A list of software features that promote the development of understanding can be developed through an integration of Nickerson's first four maxims with the learning conditions suggested by the Inductive Reasoning Model (the F prefix is used in reference to software **F**eatures). These features include:

F1. The program should start where the student is, not at the level of the instructor (I1).
The program should accommodate students' conceptual development in a domain, common misconceptions that occur, and the knowledge that students typically bring to the classroom.

F2. The program should promote learning as a constructive process in which the task is to provide guidance that facilitates exploration and discovery by
   (a) providing ways for the student to make and test predictions (I3),
   (b) promoting frequent generation and testing of predictions (I4), and
   (c) providing clear feedback on the outcomes of predictions (I5).

F3. The program should use models and representations that are familiar to the novice. Novices tend to generate concrete representations based on familiar concepts or objects from everyday life. This will facilitate the student's encoding of relevant information by relating new entities to familiar, well rehearsed encodings.

F4. Simulations should draw the student's attention to aspects of a situation or problem that can be easily dismissed or not observed under normal conditions. The program design should
   (a) promote the simultaneous encoding of key features of the environment (I2),
   (b) make it clear to the student which features of the environment are to be associated with the outcome of a prediction (I6), and
   (c) promote the encoding of information that relates the variability of environmental features to specific actions (I7).

Nickerson's (1995) fifth maxim describes features of the classroom environment that best promote understanding. Another chapter from *Software Goes to School*. (Perkins et al., 1995) by Snir, Smith, and Grosslight (1995) provides some additional recommendations for how software can be used in the classroom to promote understanding. A combination of Nickerson's fifth maxim with three recommendations made by Snir et al. provides a set of general guidelines for the incorporation of software into a course in which the intent is to develop students' conceptual understanding (the C prefix is used in reference to the **C**lassroom or **C**urriculum). These guidelines consist of:

C1. Provide a supportive environment that is rich in resources, aids exploration, creates an atmosphere in which ideas can be expressed freely, and provides encouragement when students make an effort to understand (Nickerson, 1995).

C2. The curriculum should not rely completely on computer simulations. Physical activities need to be integrated with computer simulations to establish that the knowledge gained from simulations is applicable to real world phenomena (Snir et al., 1995).

C3. Instructional materials and activities designed around computer simulations must emphasize the interplay among verbal, pictorial, and conceptual representations. Snir et al. (1995) have observed that most students will not explore multiple representations on their own accord and require prompting and guidance.

C4. Students need to be provided with explicit examples of how models are built and interpreted in order to guide their understanding (Snir et al., 1995). Although we are guaranteed that students will always attempt to come to some understanding of what they experience, there is no guarantee that students will develop appropriate and acceptable models even when interacting with the best-designed simulations.

EVALUATING A SOFTWARE PROGRAM

As stated in the introduction, my intent for detailing a list of software features and guidelines was to evaluate the merits of the software that I developed, *Sampling Distributions*. How well does the *Sampling Distributions* program fare when compared to the recommended software features outlined above? For example, does the program meet students at their current level of understanding (F1)? This is a difficult question to answer. The program is used in the second half of the term, once students have had several weeks of experience with descriptive statistics and frequency distributions. I assume that the concept of a frequency distribution, continuous distribution, standard types of distributions, and various descriptive statistics such as the mean, median, standard deviation, and interquartile range are familiar given that they form a major part of the content and activities presented in the first half of the course. The *Sampling Distributions* program uses these concepts to present information related to the central limit theorem. In this respect, the program appears to start where the student is and uses representations that should be familiar to the student.

Does the program reflect the emphasis placed on encoding (F4)? There are many ways in which *Sampling Distributions* has been designed to facilitate the encoding of key features as well as the association of related features and ideas. For example, the program reinforces the association between labels for distributions and the shape of the distributions as the student selects one of the standard forms by

clicking a button in the Population window. *Sampling Distributions* also supports the presentation of multiple representations simultaneously so that concepts from different representational systems become associated (F4c). A student can, if prompted, observe how the various population parameters change as the shape of the distribution is manipulated. For example, students can be instructed to start with a normal distribution and then incrementally increase the frequency at the high end of the distribution by clicking the up arrow for the value 10. Students are asked to record the population parameters after each incremental increase. A comparison of the change in both the mean and the median provides a way for students to observe directly that extreme values have a greater effect on the mean than the median. This is an example of the type of guidance that Snir et al. (1995) suggest is needed to make appropriate use of simulations in the classroom. This also provides an example of how experience with the *Sampling Distributions* program can help students create associations between actions and outcomes, a necessary step for building understanding that is implied by the Inductive Reasoning Model (F4b).

Encoding enhancements are also present in the Sampling Distributions window. Students observe the creation of the sampling distribution, one sample mean at a time, so that they have a visual, concrete record of the process (F3). Verbal and visual referents are presented together (the label "Distribution of Sample Means" is placed above the frequency distribution) to promote association. The graphs provide visual representations of center and spread, and the sampling distribution statistics provide symbolic counterparts (F2c and F4a). Again, as pointed out by Snir et al., the students cannot be assumed to make these connections automatically, so activities are designed to highlight the correspondence between the visual representation and the sampling distribution statistics.

For example, students can be asked to identify which statistic corresponds to the spread of the sampling distribution of sample means--the typical choices are the mean of sample standard deviations or the standard deviation of sample means (F4b). Some students inevitably select the former, which can prompt a discussion of which is the correct selection and why. This can lead to focusing on the theoretical value of $\frac{\sigma}{\sqrt{n}}$ and its relationship to the sampling distribution. I also have students position the Population and Sampling Distributions windows so that they can see both the population parameters and the sampling distribution statistics, then ask them to identify the correspondences between the two sets of values (F4a). Deeper understanding can be promoted by asking students to provide explanations of how two values are related (e.g., that the mean of sample means provides an estimate for the population mean or is expected to equal the population mean).

The *Sampling Distributions* program can facilitate guided exploration and discovery by allowing students to change the shape of the population or the size of the samples drawn and then run a simulation by randomly drawing a large number of samples (F2a). Students are provided with a sheet that asks them to start with a normally distributed population and to draw 500 random samples. A recording sheet is provided that prompts the students to change the sample size in increments from n = 5 to n = 100 (F2b). The sheet provides spaces for the student to record the sampling distribution statistics that result and to describe the shape, spread, and center of the two sampling distributions (F2c). After completing the last run for n = 100, several questions are presented to the students to promote the development of explanation structures: What is the relationship between sample size and the spread of the sampling distributions? Which distribution tends to have the larger spread, the one for sample means or the one for sample medians? At what sample sizes do each of the sampling distribution statistics begin to stabilize (not change significantly as the sample size is increased)? Did the sampling distribution statistics provide good, accurate estimates of

the population parameters? Overall, did the sampling distribution statistics behave in accordance with the central limit theorem?

Some of these questions might prompt students to conduct additional runs (F2b). For example, a student may conduct several runs at n = 5, several runs at n = 10, and several runs at n = 25 to see if fluctuations in the sampling distribution statistics are larger for one sample size than for another. This type of activity helps students develop specificity in their understanding (e.g., there is a bit of variation in the standard deviation of sample means for sample sizes below 15 or 20, but the statistic becomes more consistent with larger samples) and to form generalities (e.g., regardless of the sample size, the mean of the sample means is always very close to the population mean).

Once a student completes the prescribed simulations, he/she is free to explore (F2a and F2b). The activity suggests that students create different population distributions by selecting one of the preset distributions provided by the program or by creating distributions of their own design. I provide illustrations of a bimodal and a trimodal distribution as possibilities (see Figures 3 and 4, respectively). Students typically need 10-15 minutes to go through the prescribed simulations. They typically spend 30-45 minutes exploring non-normal population distributions using additional recording sheets provided with the activity.

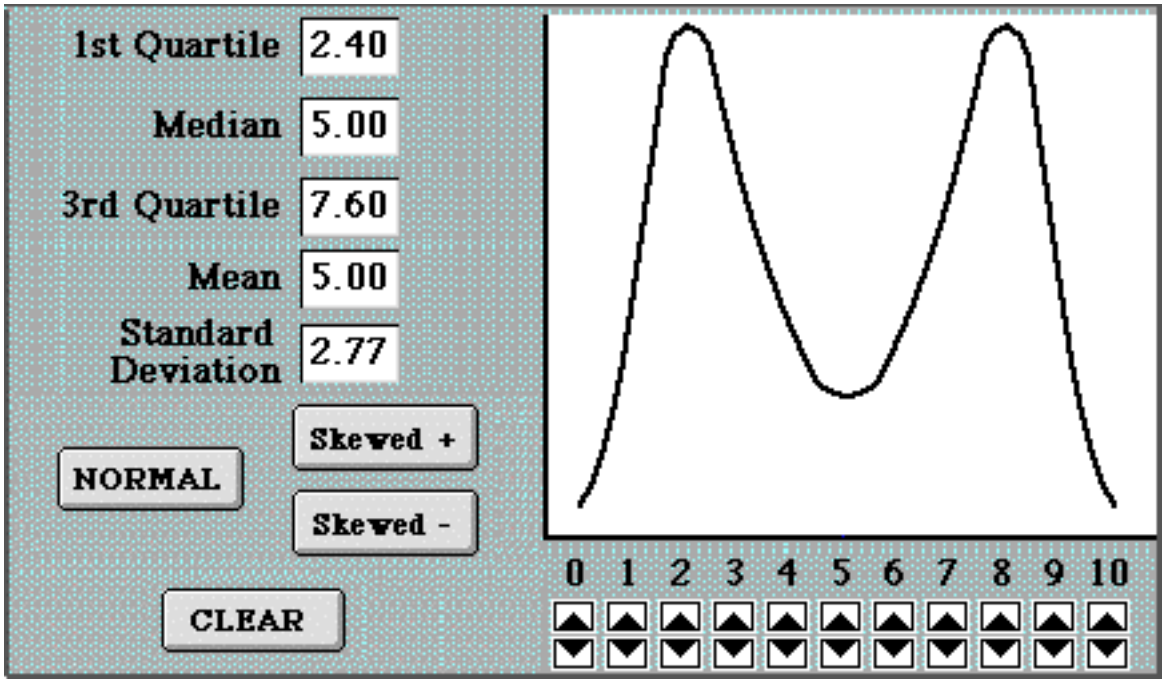

Figure 3: Creating a bimodal distribution in the population window

I stop the activity after an hour's time and engage the class in a discussion of what they observed. Classes come to the consensus that with large enough samples the sampling distribution of sample means will tend to be normal in shape regardless of the shape of the population. The use of the program to test out predictions helps students to draw an abstract generalization from concrete experience and observation (F3). I am sure that a physical apparatus could be constructed that would provide students with a similar experience, but I believe it would not allow students to explore as many possibilities in the same amount of

time and with the same amount of feedback on the relationship between visual and symbolic representations of the sampling distributions.

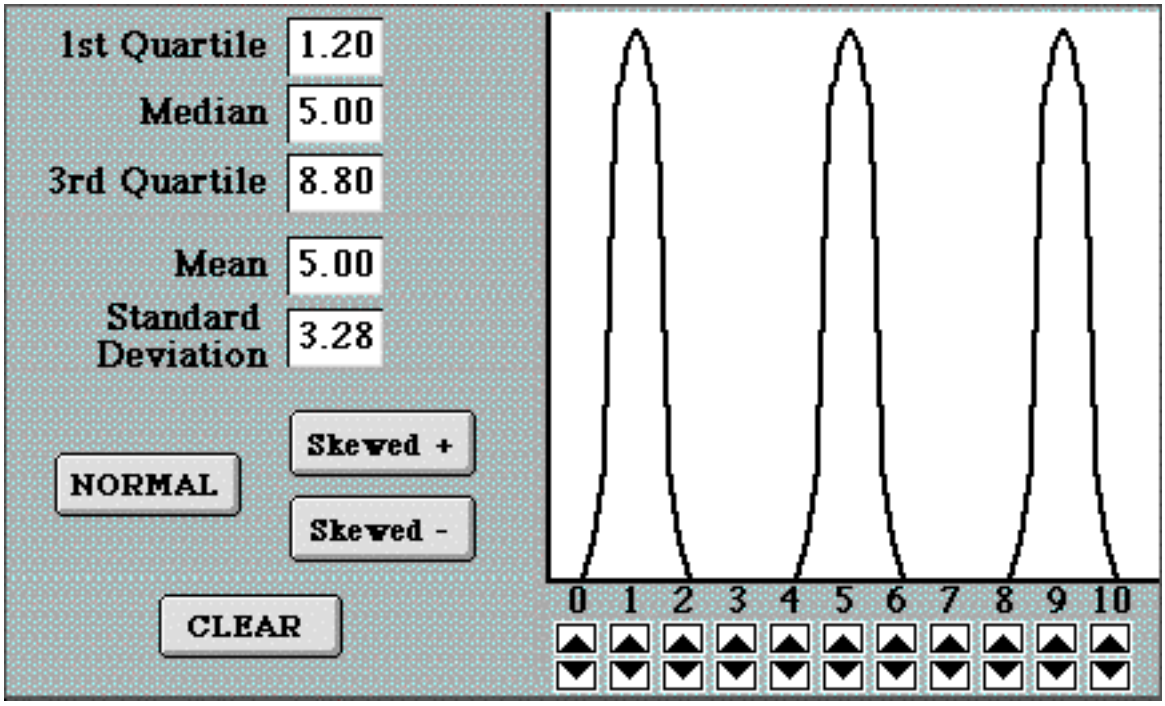

Figure 4: Creating a trimodal distribution in the population window

FUTURE DEVELOPMENTS

My conclusion is that the *Sampling Distributions* program does exhibit many features that should promote conceptual development and understanding. Although it is easy to see the features that are included in a piece of software, it is more difficult to take an objective look and identify features that are missing. A review of the models of understanding presented earlier has helped me identify enhancements that may improve the effectiveness of the program. For example, lines similar to those used in the Sampling Distributions window can be included in the Population window to identify the population mean and median. This should facilitate the formation of associations between the visual placement of the symbolic values in the graph of the Population Window (F1) as well as support an understanding of the relationship between the population parameters and the sampling distributions (F4). Another idea is to provide a button that superimposes the shape of a normal distribution over the bar graphs for the sampling distributions, based on the theoretical parameters of the Central Limit Theorem (F2). This would facilitate students' judgment of whether or not the sampling distributions are normal in shape and match the implications of the central limit theorem.

Although *Sampling Distributions* has many features that should help students develop their conceptual understanding of sampling distributions, this does not necessarily mean the program is effective. This is an empirical question that requires research on whether or not students' conceptual understandings change as a result of using the software and, if so, in what ways. My colleagues (J. Garfield and B. Clothier from the University of Minnesota, and B. Chance from the University of the Pacific) and I will begin to explore the

outcomes related to students interactions with *Sampling Distributions* during the upcoming year. Our research will initially look at students' understandings and expectations for the shape, center, and spread of sampling distributions. In keeping with the guidelines for the incorporation of software into a course stated above (C1 to C4), the research will explore how understanding changes when the *Sampling Distributions* program is used in different activities under different conditions. One planned study is to compare the effects of having students explore the effects of many different sample sizes for only a few, preset population distributions with the effects of having students examine sampling distributions generated from many population distributions but over a range of fewer sample sizes. We also plan to examine differences in understanding between students who interact directly with the *Sampling Distributions* program and students who receive instruction about sampling distributions either through a primarily lecture-based format or through the demonstration of other simulations.

We plan to gather several different types of information. One source of information will consist of interviews conducted with students as they use the software, which is similar to the approach described by Goldenberg (1995). Along with the interviews, we plan to conduct pretest/posttest assessments using a graphics-based measurement instrument. Pretests will be given to students in introductory statistics courses at the point in the course when the central limit theorem is typically introduced. The posttest will be administered after students receive more detailed instruction on sampling distributions. The instruction may include the presentation of simulations or hands-on experience with sampling distributions either through experiments with physical objects or using the *Sampling Distributions* program.

An example of the graphs for one test item are given in Figure 6. After looking over the graphs, students are asked to respond to a question like the following:

Which graph represents a distribution of sample means for samples of size 4? (circle one).

Students are also presented a set of reasons and asked to select the ones that come closest to matching their reasons for their chosen graph. Some examples of the reasons are shown in Figure 5.

- ❑ I expect the sampling distribution to be shaped like a NORMAL DISTRIBUTION.
- ❑ I expect the sampling distribution to be shaped like the POPULATION.
- ❑ I expect the sampling distribution to have LESS VARIABILITY than the POPULATION.
- ❑ I expect the sampling distribution to have MORE VARIABILITY than the POPULATION.

Figure 5: Reasons for selecting a graph (in Figure 6)

For each situation, a second question is asked with respect to drawing samples of a larger size, such as:
Which graph represents a distribution of sample means for samples of size 25? (circle one).

Students are again asked to identify the reasons for their choices. Some additional reasons are included, as shown in Figure 7.

The distribution for a population of test scores is displayed below. The other five graphs labeled A to E represent possible distributions of sample means for 500 random samples drawn from the population.

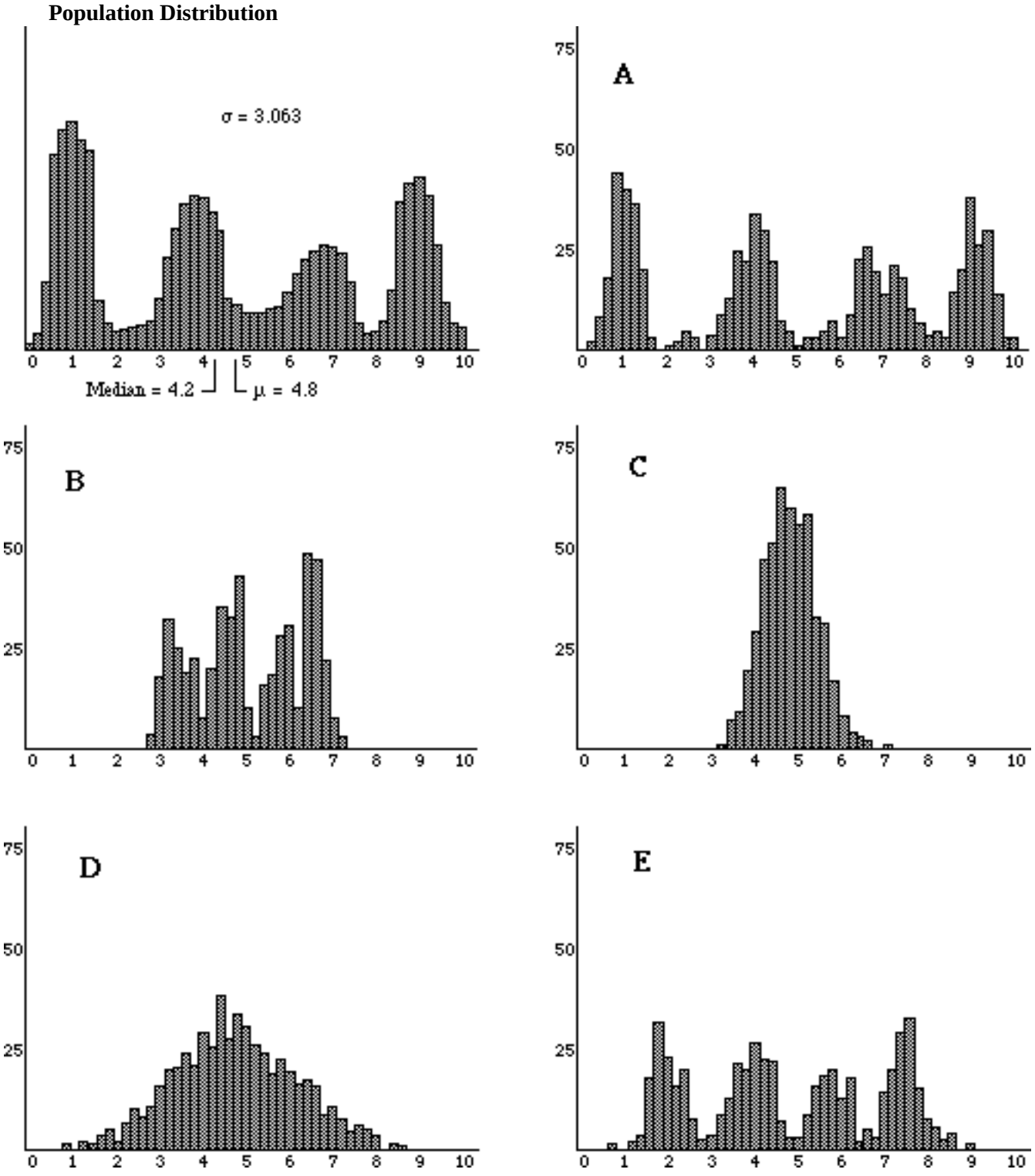

Figure 6: Example of a test item from the Sampling Distributions Reasoning Test

- ❑ I expect the second sampling distribution to have MORE VARIABILITY than the first.
- ❑ I expect the second sampling distribution to have LESS VARIABILITY than the first.
- ❑ I expect the second sampling distribution to look MORE like the POPULATION than the first.
- ❑ I expect the second sampling distribution to look LESS like the POPULATION than the first.
- ❑ I expect the second distribution to look MORE like a NORMAL population than the first.
- ❑ I expect the second distribution to look LESS like a NORMAL population than the first.

Figure 7: Reasons listed for selecting a second distribution

What do we hope to accomplish with this line of research? Information from the interviews will hopefully provide insights into what students do and do not attend to as they use the software, whether or not prompts are needed to direct students' attention to various aspects of the program, the development of students' thinking as they interact with the program, and features of the program that could be enhanced, improved, or added to make the characteristics of sampling distributions more evident. Students responses to the items on the pretest should provide us with a better understanding of what students understand and do not understand about sampling distributions prior to receiving more detailed instruction or experience with sampling distributions. Posttest results will help us determine the effects of different types of instruction on students' understanding and to make comparisons among different approaches to teaching students about sampling distributions. Just as attempts to define understanding tends to lead us in circles, I expect that our investigations will produce a continuous cycle in which increases in our understanding of how and what students learn about sampling distributions lead to further revisions of the software and research methods, as well as to a better understanding of how to teach this complex and rich topic in statistics.

REFERENCES

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge, MA: Harvard University Press.

Carey, S., & Smith, C. (1995). Understanding the nature of scientific knowledge. In D. N. Perkins, J. L. Schwartz, M. M. West, & M. S. Wiske (Eds.), *Software goes to school: Teaching for understanding with new technologies*. New York: Oxford University Press.

Collins, A. M., & Loftus, E. F. (1975). A spreading-activation theory of semantic processing. *Psychological Review*, *82*, 407-428.

de Groot, A. M. B. (1983). The range of automatic spreading activation in word priming. *Journal of Verbal Learning and Verbal Behavior, 22*, 417-436.

Fischler, I. (1977). Semantic facilitation without association in a lexical decision task. *Memory & Cognition, 5*, 335-339.

Goldenberg, E. P. (1995). Multiple representations: A vehicle for understanding understanding. In D. N. Perkins, J. L. Schwartz, M. M. West, & M. S. Wiske (Eds.), *Software goes to school: Teaching for understanding with new technologies.* New York: Oxford University Press.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. (1987). *Induction: Processes of inference, learning, and discovery*. Cambridge, MA: The MIT Press.

Marcel, A. J. (1983). Conscious and unconscious perception: Experiments on visual masking and word recognition. *Cognitive Psychology, 15*, 197-237.

Meyer, D. E., & Schvaneveldt, R. W. (1971). Facilitation in recognizing pairs of words: Evidence of a dependence between retrieval operations. *Journal of Experimental Psychology, 90*, 227-334.

Newell, A. (1973). Production systems: Models of control structures. In E. G. Chase (Ed.), *Visual information processing*. New York: Academic Press.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Nickerson, R. S. (1995). Can technology help teach for understanding? In D. N. Perkins, J. L. Schwartz, M. M. West, & M. S. Wiske (Eds.), *Software goes to school: Teaching for understanding with new technologies*. New York: Oxford University Press.

Perkins, D. N. (1993). Person plus: A distributed view of thinking and learning. In G. Salomon (Ed.), *Distributed cognitions.* New York: Cambridge University Press.

Perkins, D. N., Crismond, D., Simmons, R., & Unger, C. (1995). Inside understanding. In D. N. Perkins, J. L. Schwartz, M. M. West, & M. S. Wiske (Eds.), *Software goes to school: Teaching for understanding with new technologies*. New York: Oxford University Press.

Perkins, D. N., Schwartz, J. L., West, M. M., & Wiske, M. S. (1995). *Software goes to school: Teaching for understanding with new technologies*. New York: Oxford University Press.

Rumelhart, D. E. (1980). On evaluating story grammars. *Cognitive Science, 4*, 313-316.

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals and understanding: An inquiry into human knowledge structures.* Hillsdale, NJ: Erlbaum.

Snir, J., Smith, C., & Grosslight, L. (1995). Conceptually enhanced simulations: A computer tool for science teaching. In D. N. Perkins, J. L. Schwartz, M. M. West, & M. S. Wiske (Eds.), *Software goes to school: Teaching for understanding with new technologies*. New York: Oxford University Press.

Thibideau, R., Just, M. A., & Carpenter, P. A. (1982). A model of the time course and content of reading. *Cognitive Science, 6*, 157-203.