Psychological Review
2006, Vol. 113, No. 1, 57–83

0033-295X/06/$12.00 DOI: 10.1037/0033-295X.113.1.57

Global Model Analysis by Parameter Space Partitioning

Mark A. Pitt, Woojae Kim, Daniel J. Navarro, and Jay I. Myung
Ohio State University

To model behavior, scientists need to know how models behave. This means learning what other behaviors a model can produce besides the one generated by participants in an experiment. This is a difficult problem because of the complexity of psychological models (e.g., their many parameters) and because the behavioral precision of models (e.g., interval-scale performance) often mismatches their testable precision in experiments, where qualitative, ordinal predictions are the norm. *Parameter space partitioning* is a solution that evaluates model performance at a qualitative level. There exists a partition on the model's parameter space that divides it into regions that correspond to each data pattern. Three application examples demonstrate its potential and versatility for studying the global behavior of psychological models.

Keywords: model comparison, model complexity, MCMC, connectionist modeling

The experimental method of scientific inquiry has proved to work quite well in psychology. Its unique blend of methodological control and statistical inference are effective for testing qualitative (i.e., ordinal) predictions derived from theories of behavior. Data collection and dissemination have become very efficient, so much so that far more may be known about a behavioral phenomenon than is reflected in its corresponding theory.

That our knowledge extends beyond the reach of theory may be a sign of productive science and underscores the fact that theories are broad conceptualizations about behavior that cannot be expected to explain the minutia in data. Cognitive modeling is a research tool that can act as a counterforce to slow and fill this explanatory gap. It compensates for a theory's limitations of precision in data synthesis, description, and prediction. Whether the model is an implementation of an existing theory or a neurally inspired environment in which to study processing (e.g., information integration, representational specificity, probabilistic learning), models are rich sources of ideas and information on how to think about perception, cognition, and action. The pros and cons of various implementations can be evaluated. Inconsistencies and hidden assumptions can come to light during model creation and evaluation. In short, the modeler is forced to confront the complexity of what is being modeled and, in the process, can gain insight into the relationship between variables and the functionality of the model (see Shiffrin & Nobel, 1998, for a personal account of this process).

Of course, the virtues of modeling are accompanied by vices. One of the more serious, often leveled against connectionist models (Dawson & Shamanski, 1994; McCloskey, 1991) but by no means restricted to them, is that model behavior can be mysterious and difficult to understand, which can defeat the purpose of modeling. A model should not be as (or more) complex than the data being described. Rather, models should offer a simpler and tractable description.

The preceding observations are not so much a comment about models or modeling per se but a comment about the need for methods for analyzing model behavior. The computational power of cognitive models requires correspondingly sophisticated tools to study them. The wide variety of models in the discipline makes the need for universal tools all the more pressing and challenging. In this article, we introduce a versatile model analysis and comparison method. We begin by situating it in relation to existing methods.

Methods of Model Analysis

Model analysis methods can be differentiated along two dimensions, whether they measure a model's local or global behavior, and whether model behavior is evaluated quantitatively or qualitatively. The position of many analysis methods along these, largely independent, dimensions is shown schematically in Figure 1. In the following discussion, quantitative methods are reviewed before qualitative ones.

Quantitative Model Analysis

Most quantitative evaluations of models are local. That is, they consider the behavior of the model only at its best fitting parameter values. In contrast, global methods aim to elucidate a model's behavior across the full range of its parameter values. Not surprisingly, the two approaches are complementary in what they tell us about model behavior.

Local Quantitative Methods

The most common form of local model analysis is data fitting, in which a model is tested by measuring how closely it approximates (i.e., fits) data generated in an experiment. Because data are a reflection of the psychological process under study, a good fit to

Mark A. Pitt, Woojae Kim, Daniel J. Navarro, and Jay I. Myung, Department of Psychology, Ohio State University.

Daniel J. Navarro is now at the Department of Psychology, University of Adelaide, South Australia.

This work was supported by Research Grant R01-MH57472 from the National Institute of Mental Health, National Institutes of Health. Daniel J. Navarro was also supported by a grant from the Office of Research, Ohio State University.

Correspondence concerning this article should be addressed to Mark A. Pitt, Department of Psychology, 1885 Neil Avenue, Ohio State University, Columbus, OH 43210-1222. E-mail: pitt.2@osu.edu

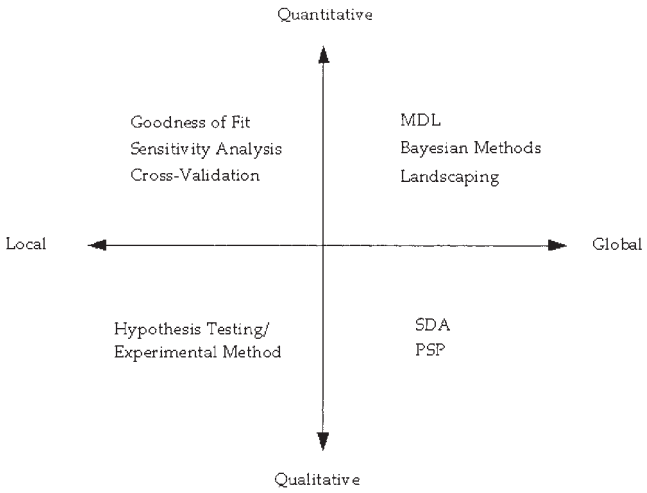


Figure 1. The locations of methods of model analysis and comparison in a two-dimensional space, defined by the degree to which the method evaluates quantitative versus qualitative model performance (vertical axis) and whether the method focuses on local or global model behavior (horizontal axis). MDL = minimum description length; SDA = signed difference analysis; PSP = parameter space partitioning.

the data is a necessary condition a model must satisfy to be taken seriously. A good fit determines how well a model passes the sufficiency test of mimicking human performance. It is especially useful in the early stages of model development as a quick and easy check on sufficiency. Quantitative measures of *goodness-of-fit* include percentage of variance accounted for, root mean squared deviation, and maximum likelihood. Although a good fit makes a model a member of the class of possible contenders, the size of this class will depend on the amount and variety of evidence already accumulated. As more data are collected about an underlying cognitive process, the number of viable models will diminish. If two models that one is comparing fit all of the data similarly well, other analysis methods are needed to choose between them.

Another local method that is useful for probing model behavior more deeply than goodness-of-fit is a *sensitivity analysis,* in which a model's parameters are varied around its best fitting values to learn how robust model behavior is to slight variations of those parameters. If a good fit reflects a fundamental property of the model, then this behavior should be stable across reasonable parameter variation. Another reason a model should satisfy this criterion is that human data are noisy. A model should not be so sensitive that its behavior changes noticeably when noise is encountered. *Cross-validation*, in which a model is fit to the second of two data sets using the best fitting parameter values from fitting the first data set, is a fit-based approach to quantifying this sensitivity (Browne, 2000; Stone, 1974).

Global Quantitative Methods

A drawback of local analysis methods is the very fact that they are local. Each fit provides a view of the model's behavior at a particular point in its parameter space but does not provide any information about how it behaves at other parameter settings. This can be particularly disconcerting if the behavior of the model is sensible only at a few settings. Furthermore, relying on purely local methods leaves one with a few snapshots of model performance that are difficult to piece together into a comprehensive understanding of the model. The task of comparing two models is even more arduous with local methods.

For these reasons, researchers have begun developing global analysis techniques. They are intended to augment local methods, not replace them. From a global perspective, the goal is to learn something about the full range of behaviors that a model exhibits. By doing so, we can gain a deeper understanding of the model and how it compares to competing models. Two of the most popular quantitative global methods are *Bayesian methods* (e.g., Kass & Raftery, 1995; Myung & Pitt, 1997) and *minimum description length* (MDL; e.g., Grünwald, 1998; Grünwald, Myung, & Pitt, 2005; Pitt, Myung, & Zhang, 2002; Rissanen, 1996, 2001). In both, the focus is on predictions made by the model at all of its parameter values. This global perspective yields a natural measure of a model's a priori data-fitting potential (i.e., the model's flexibility or complexity; Myung, 2000). Although both methods are statistically rigorous, technical requirements currently limit their application to the diverse range of models in psychology.

More recently, Navarro and colleagues (Kim, Navarro, Pitt, & Myung, 2004; Navarro, Pitt, & Myung, 2004; see also Wagenmakers, Ratcliff, Gomez, & Iverson, 2004) introduced a global quantitative method, called *landscaping,* that was developed with an eye toward increased versatility and informativeness about model discriminability. The essence of the technique involves determining how well two models fit each other's data. What is being measured is the extent to which two models mimic one another. The approach is attractive as landscapes are relatively easy to create, requiring only a comparison of fits to data sets. In addition, landscapes can be used to assess the informativeness of data in distinguishing pairs of models by inspecting where within these landscapes experimental data are located. In short, the landscape provides a global perspective from which to understand the relationship between two models and their fits to data. However, like MDL and Bayesian methods, landscaping requires that the models make quantitative predictions. A method free from this restriction would have much wider applicability.

Qualitative Methods

Implicit in the use of quantitative methods of model analysis is that the goal of modeling is to approximate closely empirical data. Although there are good reasons for doing so, one must be careful to avoid modeling the quantitative minutiae of a data set while missing theoretically important qualitative properties or trends in the data. In the words of Box (1976), "since all models are wrong the scientist must be alert to what is importantly wrong. It is inappropriate to be concerned about mice when there are tigers abroad" (p. 792). Thus, it is important not to lose sight of qualitative behavior. In this regard, model evaluation at the qualitative level cannot only be informative, but sometimes more appropriate. It is certainly the most common in the psychological sciences.

Hypothesis Testing as a Local Qualitative Method

Hypothesis testing, as used in the experimental method of psychological inquiry, is, in essence, a local qualitative method of

model analysis. Hypotheses generated in experimental settings are qualitative predictions about an ordinal pattern of data across conditions. One might, for instance, observe a preference reversal under some experimental conditions that only one of two models is able to reproduce. A correct prediction is generally held to be strong evidence in favor of the winning model and not without reason (Platt, 1964). Only in rare circumstances will a model predict the exact (interval) quantitative differences among conditions. As such, most psychological models are often intended to be illustrative of some underlying process rather than a precise description of its inner workings. Accordingly, the failure of a model to reproduce the "fine grain" of a data set is not necessarily fatal to the model. It may merely indicate that minor changes are required. On the other hand, if a model cannot capture the gross qualitative pattern of the data, something is seriously amiss. As with all local methods of model analysis, hypothesis testing is most informative when the qualitative pattern in the data can be captured by only one model.

Global Qualitative Methods

Hypothesis testing is the workhorse of model evaluation in much of psychology, but like its quantitative counterpart, goodness-of-fit, its focus on local behavior is a limitation. We gain only a glimpse of what the model can do. It would be useful to know how many of the other qualitative patterns in that same experiment the model could elicit. A model that can produce any logically possible pattern is no more impressive than one that fails to produce the empirical pattern (Roberts & Pashler, 2000). In other words, there is the implicit problem pertaining to what we might call "qualitative complexity or flexibility." The solution is the same as with quantitative methods such as MDL: Study the qualitative behavior of the model across a broad range of parameter values.

Dunn and James's (2003) *signed difference analysis* is an example of global qualitative model analysis. One seeks to identify the set of signed differences (i.e., pluses or minuses) permitted between two data vectors when a model's parameter values are varied across the entire parameter space. It is a simple means of deriving testable ordinal predictions from models in which the functional relationship between task performance measurements (e.g., dependent variables) and underlying constructs (i.e., model parameters) is assumed to be monotonic and otherwise unspecified. This method of global analysis is probably most useful in the early stages of cognitive modeling where models are defined primarily in terms of their qualitative predictions and no or few assumptions are made about other details of the underlying process. As models become more refined, with elaborate relationships between dependent variables and model parameters, a more sophisticated method is needed.

The purpose of the current article is to introduce a highly informative method of global qualitative model analysis, *parameter space partitioning* (PSP). It involves doing exactly what the name implies: A model's parameter space is literally partitioned into regions that correspond to qualitatively different data patterns that the model could generate in an experiment. Study of these regions can reveal a great deal about the model and its behavior, as we show in three application examples. In particular, one can learn how representative human performance is of the model as well as the characteristics of other behaviors the model exhibits.

Because the focus of PSP is on qualitative differences in performance, the method is more widely applicable than its quantitative counterparts such as MDL and Bayesian methods (see Figure 1), which are confined to models that can generate probability distributions (e.g., the generalized context model of Nosofsky, 1986; the fuzzy logical model of perception [FLMP] of Oden & Massaro, 1978). PSP cannot only be applied to this class of models but to others as well (e.g., connectionist), making it possible to compare models across classes, a highly desirable feature given the diversity of models in the discipline. The popularity and complexity of connectionist models make them an ideal class in which to demonstrate the potential of PSP for studying model behavior.

Parameter Space Partitioning: A Global Qualitative Method

A simple example illustrates the gist of PSP. Consider a visual word recognition experiment in which participants are asked to categorize stimuli as words or nonwords. The mean response time to words is then measured as the dependent variable across three experimental conditions A, B, and C. In this situation, it may be reasonable to claim that the important theoretical property is the ordinal relationship (fastest to slowest) across conditions. In this case, there are 13 possible orderings (including equalities) that can be observed across the three conditions (e.g., $A > B > C$, $A > B = C$, $B > C = A$, etc.). Each of these orderings defines a *qualitative data pattern*. Suppose further that mean participant performance yielded the pattern $B > C > A$.

Now consider two hypothetical models, M_1 and M_2, of word recognition, each with two parameters. Using PSP, we can answer the following questions about the relationship between the models and the empirical data generated in the experiment: How many of the 13 data patterns can each model produce? What part of the parameter space includes the empirical pattern? How much of the space is occupied by the empirical pattern? What data patterns are found in nearby regions as well as the rest of the parameter space?

Figure 2 shows the parameter space of each model partitioned into the data patterns it can generate. Model M_1 produces three patterns, one of which is the empirical pattern. Note how it is central to model performance, occupying the largest portion of the parameter space. Even though the model generates two other patterns, they are smaller and differ only minimally from the empirical pattern. In contrast, M_2 produces nine of the 13 patterns. Although one is the empirical pattern, M_2's performance is not impressive because it can mimic almost any pattern that could possibly have been observed in the experiment. Indeed, the fact that M_2 can mimic human performance seems almost incidental. Not only does the empirical pattern occupy a small region of the parameter space but larger regions are produced by patterns that are not human-like (e.g., $C > A > B$).

As this example illustrates, PSP analyses can be of two types, one focused on the number of patterns and the other on the size of the region occupied by a pattern. The goal of the "counting analysis" is to find and count all of the data patterns that a model could potentially generate by varying its parameter values. When appropriate, one may also perform a "volume analysis," in which the volumes of the regions in parameter space that are occupied by

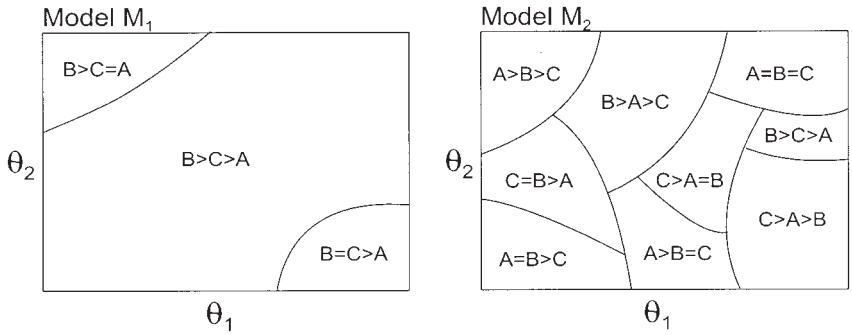


Figure 2. The partitioned parameter space of two hypothetical two-parameter models in an experiment with three conditions (A, B, C). M_1 generates only 3 of the possible patterns. M_2 generates 9, indicating that it is more complex. See text for further details.

each data pattern are compared, to gain a more thorough understanding of model behavior. Using this information, we can estimate the proportion of the parameter space that is taken up by each pattern and then use these quantities to identify those that are most and least representative. For instance, if a pattern can be produced only within a tiny region of a model's parameter space, then one would have grounds to conclude that it is irrelevant to the model's overall performance.

The idea of looking for patterns across the entire parameter space is not new. For example, most recently Johansen and Palmeri (2002) used a grid search to look for predictions made by categorization models. What is new about PSP is the approach taken to solving the search problem, which overcomes the limitations of grid search. With PSP, high-dimensional models can be partitioned with good accuracy in a reasonable amount of time. This capability makes it possible to gain a great deal of insight into the behavior of cognitive models, as we demonstrate below. The method cannot only be used to learn about the behavior of a single model but also to compare models, develop models, and use the analyses to guide future experimentation.

Implementing Parameter Space Partitioning

Implementation of PSP requires solving two nontrivial problems. One is how to define a data pattern, and the other is how to devise an efficient search algorithm to find the data patterns. Each set of a model's parameter values generates a model output but not every model output is a distinct data pattern. How many qualitatively "different" outputs can a model produce? Answering this question, which is what PSP enables us to do, depends critically on how a data pattern is defined. This is something which will vary from experiment to experiment and indeed may vary within an experiment depending on what a researcher wants to learn. Although there is no general solution, the scientist usually knows what patterns should be found in order to support or falsify a model. Nevertheless, it is a good idea to try out a couple of different pattern definitions and to perform sensitivity analyses to ascertain whether and to what extent conclusions obtained under one definition hold across others. An example of this process is presented later in the article (in the section comparing TRACE and Merge). Most of the time, ordinal predictions are being tested, so a "natural" definition of a data pattern is the ordinal relationship of model outputs, such as $A > B$ and $A = B$. However, if one is interested in ordinal patterns, a critical issue regards how to define "equality." In many situations it may seem appropriate that a quantitative prediction of $A = 1.99$ and $B = 2$ be treated as the qualitative pattern $A = B$. "Equivalence" must be defined a priori. How this is done will depend on the models and the experimental design, but the basic idea is no different than that used in null hypothesis significance testing. Quantitative criteria are adopted that define equivalence and nonequivalence (i.e., what constitutes one data pattern vs. another). These can be more or less stringent, as we demonstrate in the TRACE and Merge examples. Another example of how equivalence can be defined is presented in the first application, which focuses on the ALCOVE model of category learning.

The Search Algorithm

Once a data pattern is defined, the next challenge is to find all patterns a model can simulate. The set of all data patterns forms a partition on the parameter space, in the sense that each point in the space can correspond to only one pattern, however improbable. Once we have a definition of a data pattern, we have in effect created an unknown partition on the parameter space. Accordingly, the problem to be solved is to search a multidimensional parameter space in such a way that we visit each part of the partition at least once. As the number of parameters in a model increases, the space becomes higher dimensional, and the search problem can become very hard. The consequence of this is that brute force search methods to find all regions, such as grid search or a random search procedure like simple Monte Carlo (SMC) will not work or will take far too long to succeed. Markov chain Monte Carlo (MCMC; Gilks, Richardson, & Spiegelhalter, 1996; Robert & Casella, 1999) is a much more sophisticated sampling method that we incorporated into an algorithm that efficiently finds all regions.

Application of the PSP algorithm begins with a starting set of parameter values at which the model can generate a valid data pattern (i.e., one that satisfies the definition of a data pattern). This initial set can be supplied by the modeler or from an exploratory run using SMC. Given the parameter set and the corresponding data pattern generated by the model, the algorithm samples nearby points in the parameter space to map the region that defines the data pattern. MCMC is used to approximate quickly and accurately the shape of this region. The process then begins anew by sampling a nearby point just outside of this region.

Figure 3 illustrates how the algorithm works in the space of a two-parameter model (see Appendix A for a fuller discussion). The process begins with the initial parameter set serving as the current point in the parameter space (filled point in Panel a). A *candidate* sample point (shaded point) is drawn from a small, predefined region, called a jumping distribution, centered at the current point. The model is then run with the candidate parameter values, and its output is evaluated to determine whether the data pattern is the same as that generated by the initial point. If so, the candidate point is accepted as the next point from which another candidate point is drawn. If the new candidate point does not yield the same data pattern as the initial one, it is rejected as belonging to the current region. Another jump from the initial point is attempted, accepting those points that yield the same data pattern. The sequence of all accepted points, including repetitions of the same point, recorded across all trials is called the *Markov chain* corresponding to the current data pattern. In the language of MCMC, this implementation is called the Metropolis–Hastings algorithm with a uniform target distribution defined over a given region (Robert & Casella, 1999, ch. 6). As such, the theory of MCMC guarantees that the sample of accepted points will eventually be distributed uniformly over the entire region. This feature of MCMC allows us to estimate the volume occupied by the region, regardless of its size (Panel b). A region's volume is estimated with a multidimensional ellipsoid whose shape and size are computed using the estimated mean vector and covariance matrix of the MCMC sample of the region (see Appendix B for details).

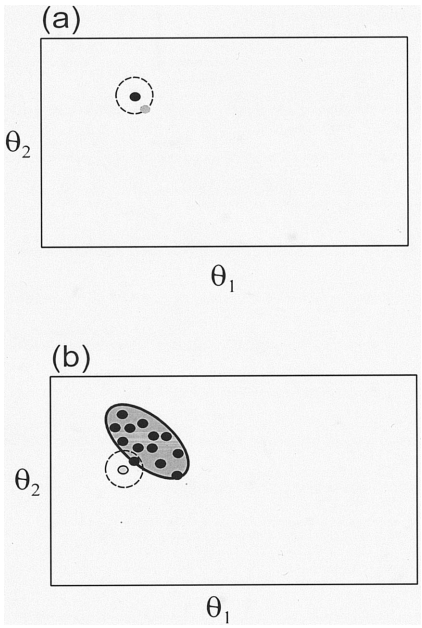


Figure 3. Illustrations of two initial steps in the parameter space partitioning algorithm. Panel (a) shows the selection of a point (black circle) in the parameter space and its accompanying jumping distribution (dashed circle). The shaded circle represents a sample from within the distribution. Panel (b) shows a region in parameter space that the algorithm has mapped out as generating the same data pattern (enclosed area filled with black dots). The point outside the region (shaded circle), along with a jumping distribution, denotes the start of a new search process to map out another region in the parameter space.

Every rejected point, which must be outside the current region, is checked to see whether it generates a new valid data pattern. If so, a new Markov chain corresponding to the newly discovered pattern is started to define the new region (shaded point in Panel b). In effect, accepted points are used to shift the jumping distribution around inside the current region to map it completely, whereas rejected ones are used to initiate new MCMC search processes to map new regions. Over time, as many search processes as there are unique data patterns will be run. A more extensive and detailed discussion of the algorithm is provided in Appendix A.

Algorithm Evaluation

We tested the accuracy of the PSP algorithm by measuring its ability to find all of the data patterns defined for a particular model. The difficulty of the search problem was varied by manipulating the definition of a pattern (i.e., number of patterns) and the number of parameters in the model. The extent of both (see Table 1) was deliberately made large to make the test challenging. The efficiency of the algorithm was measured by comparing its performance to SMC (random search).

The model was a hypercube whose dimensionality d (i.e., number of parameters) was 5, 10, or 15. To illustrate the evaluation method, a two-dimensional model ($d = 2$) is depicted in Figure 4 that contains 20 data regions (outlined in boldface) that the algorithm had to find. Note that a large portion of the space does not produce any valid data patterns. Exactly what constitutes an invalid pattern will vary across models and experimental setups, but this area roughly corresponds to those patterns that the model can produce but do not meet criteria established by the researcher to be considered in the analysis. The modeling contexts in which PSP is demonstrated below provide two very different examples of how a data pattern is defined.

In Figure 4, note that the sizes of the valid data regions vary a great deal. Some are elicited by a wide range of parameter values whereas others can be produced only by a small range of values. This contrast grows exponentially as the dimensionality of the model increases and was purposefully introduced into the test to make the search difficult and approximate the complexities (i.e., nonlinearities) in cognitive models. Ten independent runs of each search method were carried out to assess the reliability of algorithm performance.

Figure 5 shows performance of the PSP algorithm for a search problem in which there were 100 regions embedded in a 10-dimensional hypercube ($d = 10$). The PSP algorithm found all regions and did so in 9 minutes. SMC found only about 23 patterns in 9 minutes and, given its sluggish performance, it seems doubtful that SMC would find all of them in anything close to a reasonable amount of time.

Table 1 summarizes results from the complete test. The mean proportion of patterns found is listed in each cell. Results are clear and consistent. The PSP algorithm almost always found all of the patterns, whereas SMC failed to do so in every condition. Most noteworthy is the success of the PSP algorithm in the toughest situation when there were 15 parameters and 500 data patterns. Its near perfect success suggests it is likely to perform admirably in other testing situations. (How the current version of the algorithm performs when regions are not contiguous is discussed in Appen-

Table 1
Search Efficiency of the Parameter Space Partitioning (PSP) Algorithm and Simple Monte Carlo (SMC) as a Function of the Number of Parameters and Data Patterns

No. of parameters	No. of patterns to find: 20 PSP	20 SMC	100 PSP	100 SMC	500 PSP	500 SMC
5	1.00	0.85	1.00	0.78	1.00	0.87
10	1.00	0.32	1.00	0.23	0.99	0.24
15	1.00	0.09	1.00	0.02	0.99	0.03

Note. Shown in each cell is the mean proportion of patterns found based on 10 independent runs.

dix A, which contains a test of the accuracy of volume estimation.) In the remainder of this article, we describe its applications to analyzing the behavior of a single model and to comparing design differences between two models.

Application 1: Evaluating the Qualitative Performance of ALCOVE

In standard model-fitting analyses, a model's ability to fit (or simulate) the data is taken as evidence that it approximates the underlying cognitive process. In a PSP analysis, the definition of *fit* is relaxed to be a qualitative, ordinal relation on the same scale as the experimental predictions themselves. Model performance is then evaluated by determining how many of the possible orderings of conditions it can produce and how central the empirical pattern is among them (see Figure 2). In this section, we examined the behavior of ALCOVE (Kruschke, 1992), a highly successful exemplar-based account of human category learning in the context of the seminal Shepard, Hovland, and Jenkins (1961) experiment. Whereas there are some category learning effects that it does not capture without extension or modification (e.g., Kruschke & Erikson, 1995; Lee & Navarro, 2002), ALCOVE remains a simple and powerful account of a broad range of phenomena.

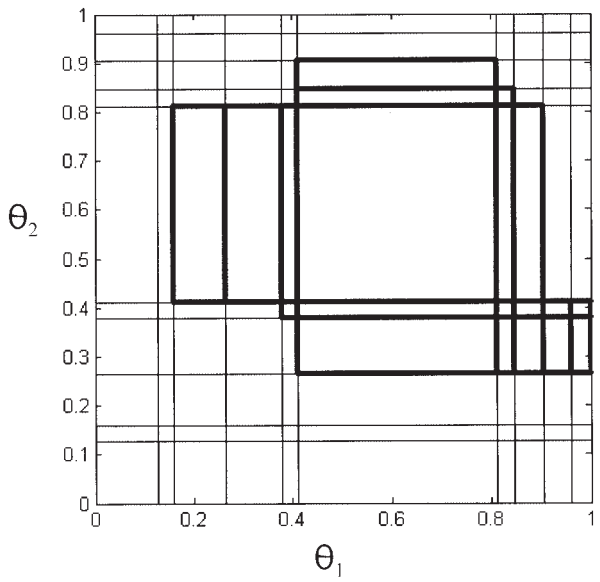


Figure 4. Illustration of the two-parameter model used to test the parameter space partitioning algorithm. The algorithm had to find the 20 areas outlined in bold (valid patterns). The regions outside of this area correspond to invalid data patterns. They should be viewed as distracting portions of the parameter space and have the effect of increasing the difficulty of the search problem.

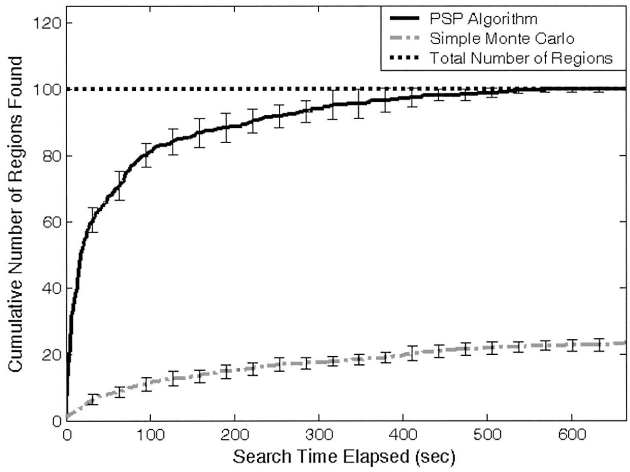


Figure 5. Search efficiency of the parameter space partitioning (PSP) algorithm compared with simple Monte Carlo (random search). A total of 100 patterns had to be found in a 10-parameter model.

Background to the Analysis

The ALCOVE Model

In some category learning experiments, participants are shown a sequence of stimuli, each of which possesses some unknown category label. The task is to learn which labels go with which stimuli, using the feedback provided after responses are made. ALCOVE solves this problem in the following way (for a detailed description, see Kruschke, 1992). When stimulus i is presented to ALCOVE, its similarity to each of the previously stored exemplars, s_{ij}, is calculated. Following Shepard (1987), similarity is assumed to decay exponentially (with a width parameter c) as a function of the attention-weighted city-block distance between the two stimuli in an appropriate psychological space. After estimating these similarities, ALCOVE forms response strengths for each of the possible categories. These are calculated using associative weights maintained between each of the stimuli and the categories. The probability of choosing the kth category follows the choice rule (Luce, 1963) with parameter ϕ.

Having produced probabilities for each of the various possible categorization responses, ALCOVE is provided with feedback from an external source. This takes the form of a "humble teacher" vector, in which learning is required only in cases where the wrong response was made. Two learning rules are then applied, both derived by seeking to minimize the sum-squared error between the response strengths and the teaching vector, with a simple gradient descent approach to optimization. Using these rules, ALCOVE updates the associative weights (with parameter η_w for the learning

rate) and the attention weights (with a learning rate parameter η_a) prior to observing the next stimulus.

The Shepard, Hovland, and Jenkins (1961) Task

In an experiment, Shepard et al. (1961) studied human performance in a category learning task involving eight stimuli divided evenly between two categories. The stimuli were generated by varying exhaustively three binary dimensions such as color (black vs. white), size (small vs. large), and shape (square vs. triangle). These authors observed that if these dimensions are regarded as interchangeable, there are only six possible category structures across the stimulus set, illustrated in Figure 6a. This means, for example, that the category structure that divides all squares into one category, and all triangles into the other, is regarded as equivalent to the category structure that divides small shapes from large ones, as shown in the lower right.

Empirically, Shepard et al. (1961) found robust differences in the way in which each of the six fundamental category structures was learned. In particular, by measuring the mean number of errors made by participants in learning each type of category structure, they found that Type I was learned more easily than Type II, which in turn was learned more easily than Types III, IV, and V (which all had similar error measures) and that Type VI was the most difficult to learn. More recently, Nosofsky, Gluck, Palmeri, McKinley, and Glauthier (1994) replicated Shepard et al.'s (1961) task with many more participants, and reported detailed information relating to the learning curves. Figure 6b shows the mean proportion of errors for each category type. Consistent with the conclusions originally drawn by Shepard et al. (1961), it is generally held that the theoretically important qualitative trend in these data is the finding that there is a natural ordering on these curves, namely that I < II < (III, IV, V) < VI. This kind of pattern is called a weak order, as the possibility of ties is allowed.

The psychological importance of this weak order structure is substantial. Suppose we had two models of the category learning process, M_1 and M_2, of roughly equal complexity. M_1 provides a reasonably good quantitative fit to the data, by assuming that all types are learned at the same rate, and closely approximates the average of the six empirical curves. In contrast, M_2 reproduces the ordering I < II < (III, IV, V) < VI but learns far too slowly and, as a result, fits the data much worse than M_1. Since the models are of equivalent complexity, a classical model selection analysis would prefer model M_1. However, while clearly both models have some flaws, most psychologists would prefer M_2, because it captures the *theoretically relevant property* (i.e., ordinal relationship among the six curves) of the data. This discrepancy arises because quantitative model selection methods like MDL tend to assume that all properties of the data are equally relevant. In many psychological applications, this is not the case. In what follows, we assume that the weak order structure is the important theoretical property of the empirical data and use the PSP method to ask how effectively ALCOVE captures this structure.

The PSP Analysis

As with any parameterized model, ALCOVE makes different predictions at different parameter values. When applied to the Shepard et al. (1961) task, ALCOVE will sometimes produce curves that have the same qualitative ordering as the empirical

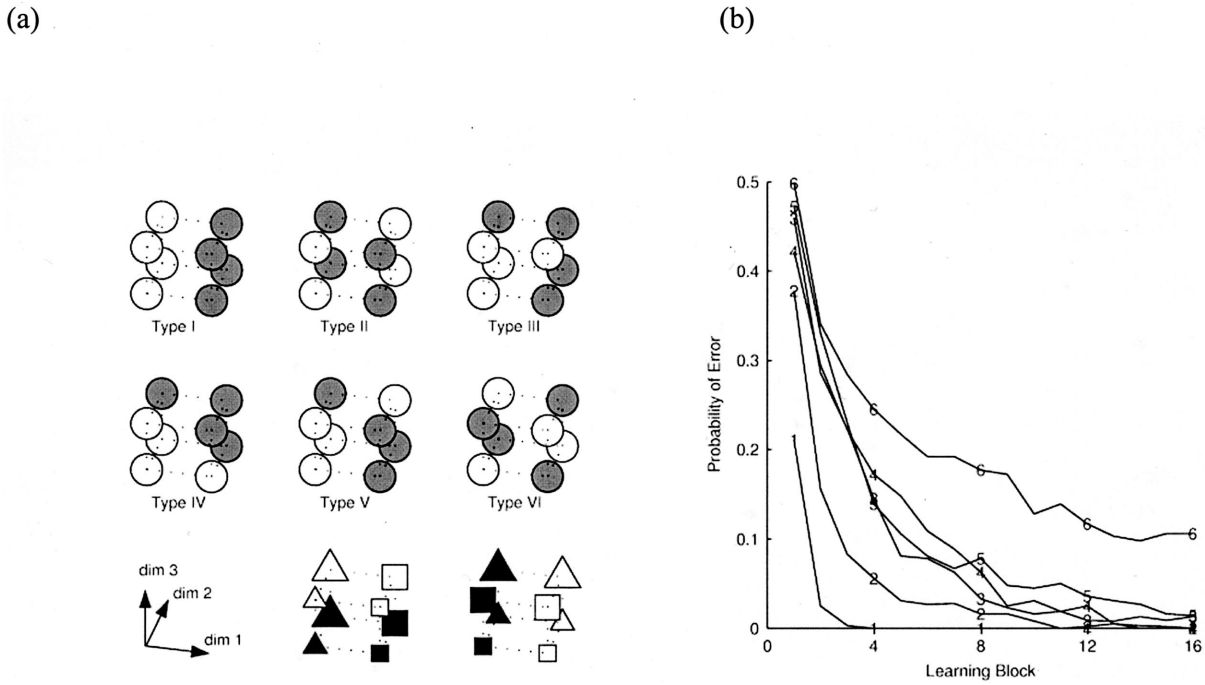


Figure 6. The Shepard et al. (1961) category types (a), and the learning curves for those types found by Nosofsky et al. (1994) in their replication (b). For visual clarity, the curves in Panel b are labeled with arabic numerals rather than with the more conventional roman numerals. dim = dimensions.

data, but at other times they will look quite different. It would be nice to know something about the other orders that ALCOVE can produce, as it seems that we might learn something about the model itself. A PSP analysis can provide such information. Using the "weak order" definition of a pattern of curves, there are 4,683 different data patterns that could possibly be observed in the experimental task. One would hope that ALCOVE generates only a small proportion of these and that the extra patterns it does produce are interpretable in terms of human performance.

Preliminaries

At this stage, we have only an intuitive definition of a data pattern. However, to perform PSP analyses, one must define a formal method of associating a set of learning curves with a particular pattern. The weak order that defines a data pattern in the ALCOVE example can be decomposed into a set of equivalence relations and a strong order on the equivalence classes. The strong order part is easy: We simply rank the classes in terms of mean learning rate. The equivalence relations are more difficult to derive. How does one justify the equivalence of III, IV, and V when mean learning rate may be similar but not identical? We solved the problem by relying on a clustering procedure to partition the six curves into a natural ordering.

The procedure we used was a variant on the clustering technique introduced by Kontkanen, Myllymäki, Buntine, Rissanen, and Tirri (2005). The essence of the technique is to view a clustering solution as a probabilistic model for the data. In the current application, the likelihood function for the data takes the form of a mixture of binomials, with a single multivariate binomial for each cluster. The clustering procedure now reduces to a statistical inference problem, which is solved by choosing the set of clusters that optimizes a minimum description length statistic. The six learning curves reported by Nosofsky et al. (1994) are averaged across 40 participants over the first 16 blocks, consisting of 16 stimulus presentations each. Each data point is thus pooled across $40 \times 16 = 640$ trials. Using this technique, it is possible to infer that I $<$ II $<$ (III, IV, V) $<$ VI is indeed the natural structure for these data (Navarro & Lee, 2005).

Additionally, we need to be able to associate a set of predicted learning curves with a data pattern. A set of response probabilities is first discretized to the same resolution as the empirical data, by finding the expected values for the data, given by np, where n is the sample size and p is the average response probability predicted for some category type across any given block of trials, and then rounding to the nearest integer. Although the rounding error is a nuisance, it is negligible for $n = 640$. The discretized curves are then mapped onto a qualitative data pattern by using the same clustering technique used to classify the empirical data.

For the PSP analysis of ALCOVE, we constrained the parameter vectors $(c, \phi, \eta_w, \eta_a)$ to lie between (0, 0, 0, 0) and (20, 6, 0.2, 0.2) and disallowed any parameter combination that did not produce monotonic curves. These ranges were deliberately chosen to be quite broad with respect to the kinds of parameter values that are typically observed in experimental contexts, and nonmonotonic curves seem highly inappropriate in a simple learning task. In other words, the chosen constraints reflect those provided by empirical data and by theory.

A technical complication is introduced by the fact that ALCOVE's predictions are slightly dependent on the order in which stimuli are observed. Each stimulus has a different effect on ALCOVE, so variations in order of presentation produce slight perturbations in the response curves. However, even these minor perturbations can violate the continuity assumptions that underlie the PSP algorithm. In order to deal with these order effects, we chose 20 random stimulus orders and ran the PSP algorithm 10 times for each stimulus order, yielding a total of 200 runs. As it turns out, the important properties of ALCOVE appear to be invariant under stimulus reordering but some unimportant properties are not.

How Many Data Patterns Can ALCOVE Produce?

After running the PSP algorithm 200 times, we observed that each run produced a different number of patterns, ranging from a minimum of 32 to a maximum of 122. Although this range is substantial, it reflects an inherent variability in ALCOVE more so than the PSP algorithm itself. The mean number of patterns recovered for a particular stimulus order ranged from 46.5 to 102.8, whereas the range in the number of patterns recovered within an order was minimal: The smallest range was a mere 7 patterns, and the largest was 36. Moreover, there was an important amount of redundancy across the 200 runs, with 17 patterns being found on every occasion, which included the empirical pattern I $<$ II $<$ (III, IV, V) $<$ VI. We refer to these 17 patterns as *universal* patterns, and the other 183 patterns as *particular* patterns.

Compared with the set of all 4,683 possible data patterns, even the largest count of $17 + 183 = 200$ patterns encompassed by ALCOVE is quite a small number. In a sense, this is quite a success for the model, because the empirical pattern, rather than being a random ordering, suddenly looks far less unlikely if we assume humans do something rather ALCOVE-like. Even in the scenario where we allow all 200 recovered patterns to be treated as a genuine ALCOVE prediction, the empirical pattern is one pattern in 200, rising from the much less satisfying base rate of 1 in 4,683. If the substantive predictions are restricted to the set of universal patterns, the empirical pattern is now 1 in 17. Either way, ALCOVE provides a reasonably good qualitative account of these data.

This initial analysis demonstrates that ALCOVE passes a basic sufficiency test in that it can account for the qualitative structure of the observed data without "going overboard" and producing every possible pattern. Of course, as psychologists, we are interested in more than just how many patterns ALCOVE produces. For example, it would be useful to know what kinds of category-type orderings are generally preserved across all 200 data patterns. One crude method for determining this is to find the average position (i.e., its rank among the six curves) of each category type across all patterns. This is illustrated in Figure 7, which plots the mean rank for each of the six types across all patterns for both universal and particular patterns. Rank increases or stays flat as the index of the type increases. The main difference between the two types of patterns is that the universals do not really distinguish between Types II, III, and IV, whereas the particulars do not distinguish between IV and V. Of importance, on average, rank does not decrease as index increases. This is encouraging because both empirically and algebraically, the difficulty of the task either

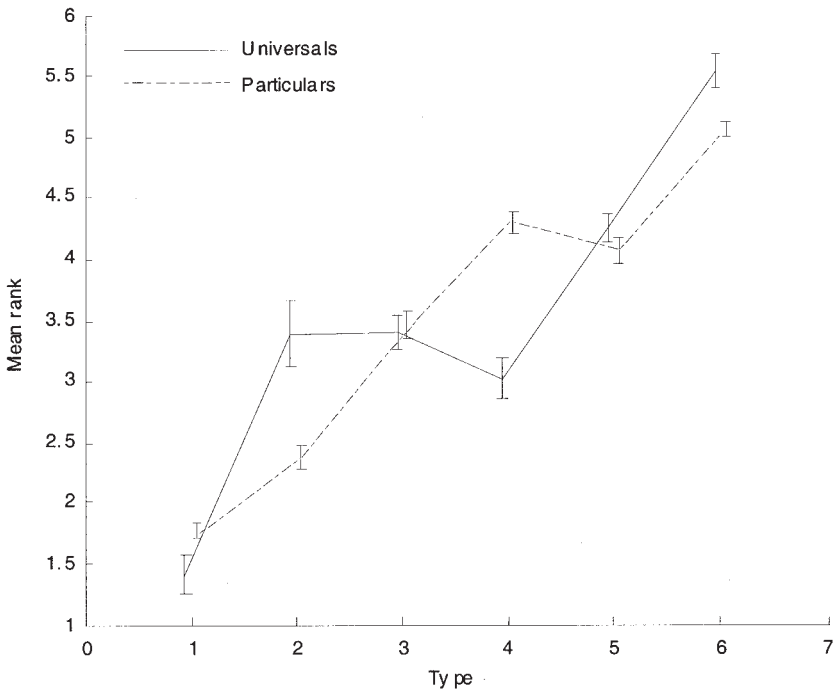


Figure 7. Means and standard errors for the rank of each category type in the data patterns recovered by parameter space partitioning. The rankings for universals do not differ strongly from the rankings of the particulars.

increases or stays constant as the index increases (see Feldman, 2000). This analysis demonstrates that on average, the set of 200 patterns that make up ALCOVE's entire set of qualitative predictions roughly preserve an important property of the empirical data: a monotonic increase in learning difficulty across category type.

Which Patterns Matter?

The finding from the preceding counting analysis that some patterns are frequent (universals) and others rare (particulars) suggests that some may be more representative of model behavior than others. However, the frequency with which a pattern is found is an unsatisfying definition of its importance to a model's behavior, as it confounds the model's properties with the robustness of the search algorithm. This naturally calls for a volume analysis, in which the proportion of the parameter space that is occupied by each pattern is estimated.

Although a model's parameter space is a potentially important source of information about a model, two things must be kept in mind when wanting to distinguish the major patterns that are responsible for most of ALCOVE's behavior from the minor patterns that are irrelevant to the model. The first is that a volume analysis is meaningful only if one is willing to ascribe meaning to the model's parameters themselves by linking them to psychological constructs. In the case of ALCOVE, such connections have been well motivated (see Kruschke, 1993). Arguably, the nature of the exemplar theory on which ALCOVE is based, and the manner in which the model captures Kruschke's (1993) "three principles," provide exactly this kind of justification. The second issue pertains to the generality of the results. A volume analysis is specific to a particular parameterization of the model (i.e., how the parameters are defined and combined in the model equation). If the model is reparameterized, the volume analysis could change. In this regard, a volume analysis can draw attention to the appropriateness of a model's parameterizations, such as parameter ranges and their interdependence.

As it turns out, for ALCOVE there is a strong relationship between the size (i.e., volume) of the region occupied by a pattern and the frequency with which it was discovered across the 200 runs. In Figure 8, the log of the average volume for each of the 200 patterns discovered is plotted against the frequency with which they were discovered across the 200 runs of the algorithm. Patterns on the far left are the ultimate particulars, having been discovered only once, whereas patterns on the far right are the universals, having been discovered on every occasion. Noting that the scale on the *y*-axis is logarithmic, we observe that the universals are by far the largest patterns. The empirical pattern, indicated by the circle, is one of the larger patterns, and is a universal.

The data in Figure 8 allow us to refine our answer to the question, "To what extent does ALCOVE *predict* the empirical pattern?" The fact that the empirical pattern is among the 17 universals is encouraging, as is the regularity in Figure 7. However, by considering the size of the various regions, we can take this analysis a step further. We could, for instance, exclude all patterns that do not reach some minimum average size. This approach is illustrated by the horizontal threshold shown in Figure 8: Only patterns that occupy more than 1% of the parameter space on average lie above this line. This is a pretty stringent test given that the parameter space is four-dimensional. Only seven patterns

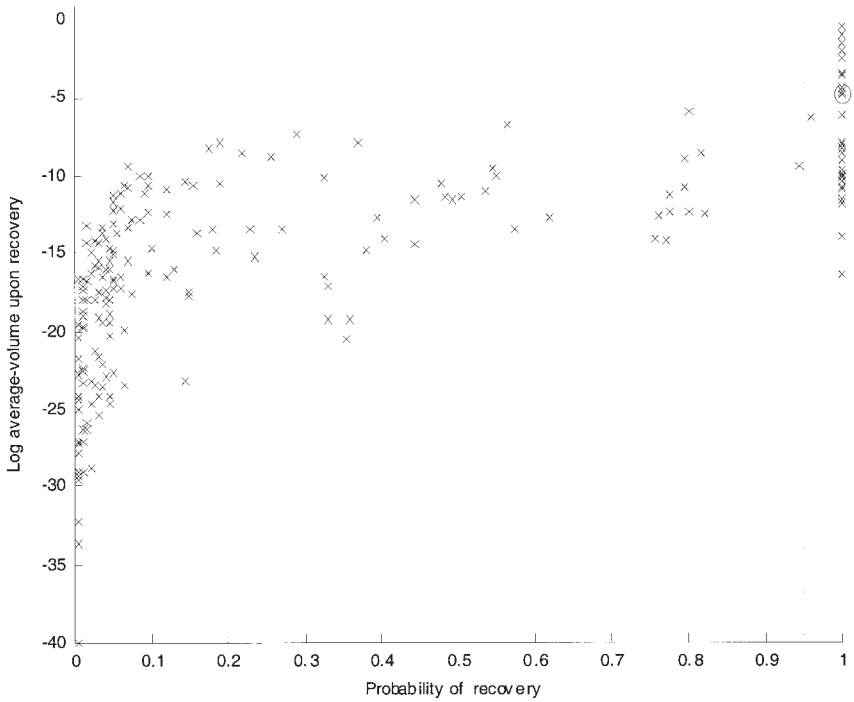


Figure 8. Scatterplot of the log average-volume for each data pattern against the frequency with which the pattern was recovered. Note that since volume is shown on a logarithmic scale, a few universals (major patterns) occupy the vast majority of the parameter space. The empirical pattern is indicated by the hollow circle in the upper right corner; the dashed horizontal line designates the 1% volume threshold; and the dashed vertical line designates the 95% recovery threshold.

exceeded it, and the empirical pattern is not one of them, occupying 0.52% of the space.

If we next examine the universal patterns that occupy as much as or more space than the empirical pattern, some general properties of ALCOVE's behavior emerge. Table 2 lists these 11 patterns, ranked by volume from largest to smallest. (The remaining patterns—universals plus particulars—are so small that together they occupy less than 1% of the total volume.) Looking down the table, it is clear that Types III and IV are always (11 of 11) predicted to be learned at about the same rate, and Type V is also about the same (9 of 11). Type VI, on the other hand, is most often learned slower than III, IV, and V (7 of 11). Type I is usually (8 of 11) faster than Types III, IV, V, and VI, as is Type II (7 of 11). Thus, not only is the empirically observed pattern I < II < (III, IV, V) < VI among the universals, but many of the largest patterns preserve the pairwise relations found in the empirical data. They are, in short, "close" to the empirical pattern. The exception to this claim regards the relationship between Types I and II. Their ordering is ambiguous. It might be I < II (4 of 11), I = II (4 of 11), or even II < I (3 of 11). In this case, ALCOVE does not make a strong qualitative prediction.

Table 2
The Eleven Largest Data Patterns Predicted by ALCOVE, Shown in Descending Rank Order of Region Size (Largest to Smallest)

Pattern No.	Ranking I II III IV V VI	% of volume
1	I = II = III = IV = V = VI	39
2	I = II < III = IV = V = VI	25
3	II < I < III = IV = V = VI	15
4	II = I = III = IV = V = VI	9.5
5	I < II = III = IV = V < VI	5.4
6	I < II < III = IV < V < VI	2.1
7	I = II = III = IV = V < VI	1.7
8	I < II = III = IV < V < VI	0.84
9	I < II < III = IV = V < VI	0.63
10	II < I = III = IV = V < VI	0.54
11 (empirical)	I < II < III = IV = V < VI	0.52

Note. All patterns are universal.

Summary

These analyses both confirm and extend our knowledge of ALCOVE. It should comfort category learning researchers to learn that ALCOVE captures the I < II < (III, IV, V) < VI ordering in a robust manner (it is a universal, not a particular) and that the various pairwise relations that the ordering implies are almost always satisfied. On the other hand, some might be worried by the volume occupied by this single pattern (0.52%), wishing it were larger, and thus more central to model behavior. Despite its smallish size, the volume analysis as a whole shows that the empirical pattern is in fact "central" to the model, in the sense that the dominant alternative patterns are quite similar to the empirical one.

"Distant" patterns (i.e., violations of weak orderings) are rarely if ever generated by ALCOVE; this information is very useful to know in model evaluation. In this regard, the added understanding provided by this global analysis of ALCOVE increases one's confidence in claiming that the model can "account" for the data precisely because we know the range of behaviors it exhibits in this experimental setting.

The PSP analysis also revealed some unexpected behaviors. It is somewhat surprising that it is even possible for ALCOVE to predict II $<$ I. It may be that this happens only with very odd choices of parameters (i.e., ones that are computationally feasible but psychologically bizarre) and is certainly something that would be interesting to explore in future work.

Application 2: Comparing the Architectures of Merge and TRACE

In addition to learning about the behavior of a single model, PSP analyses can inform us about the behavioral consequences of design differences between models. In this and the next example, PSP is applied to two localist connectionist models of speech perception, TRACE (McClelland & Elman, 1986) and Merge (Norris, McQueen, & Cutler, 2000). We compared them in two experimental settings, one intended to bring out architectural differences and the other differences in weighting bottom-up (sensory) information.

Background to the Analysis

Architectural Differences of TRACE and Merge

The two models are illustrated schematically in Figure 9. They are similar in many ways. Both have a phonemic input stage and a word stage. There are excitatory connections from the phoneme input to the word stage and inhibitory connections within the word stage. They differ in how prior knowledge is combined with phonemic input to yield a phonemic decision (percept). In TRACE, word (prior) knowledge can directly affect sensory processing of phonemes. This is represented by direct excitatory connections from the word stage back down to the phoneme stage. Also note that in TRACE the phoneme stage performs double duty, also serving as a phoneme decision stage. In Merge, these two duties are purposefully separated into two distinct stages to prevent word information from affecting perceptual processing of phonemes. Instead, lexical knowledge affects phoneme identification via excitatory connections from the word to the phoneme decision stage. In contrast to the direct interaction between phoneme and word levels in TRACE, these two sources of information are integrated at a later decision stage in Merge.

Although not visible in the diagrams in Figure 9, the models differ in another important way. In keeping with the belief that bottom-up (sensory) information takes priority in guiding perception early in processing, activation of a phoneme decision node in Merge must be initiated by phoneme input *before* excitatory lexical activation can affect phoneme decision making. TRACE contains no such constraint.

The goal of our investigations was to assess the impact of these two design differences on global model behavior. Norris et al. (2000) proposed Merge as an alternative to TRACE because they felt that the evidence from the experimental literature did not warrant interaction (i.e., direct word-to-phoneme feedback). The adequacy of the Merge architecture was demonstrated in simulation tests in which it performed just as well as, if not slightly better than, TRACE in reproducing key experimental findings. PSP analyses of the models' behaviors were performed in two of these experimental settings. In the first, the subcategorical mismatch study by Marslen-Wilson and Warren (1994), the consequences of splitting phoneme processing into separate input and decision stages, was evaluated. In the second, the indirect inhibition experiment of Frauenfelder, Segui, and Dijkstra (1990), the contribution of the bottom-up priority rule to model performance, was examined.

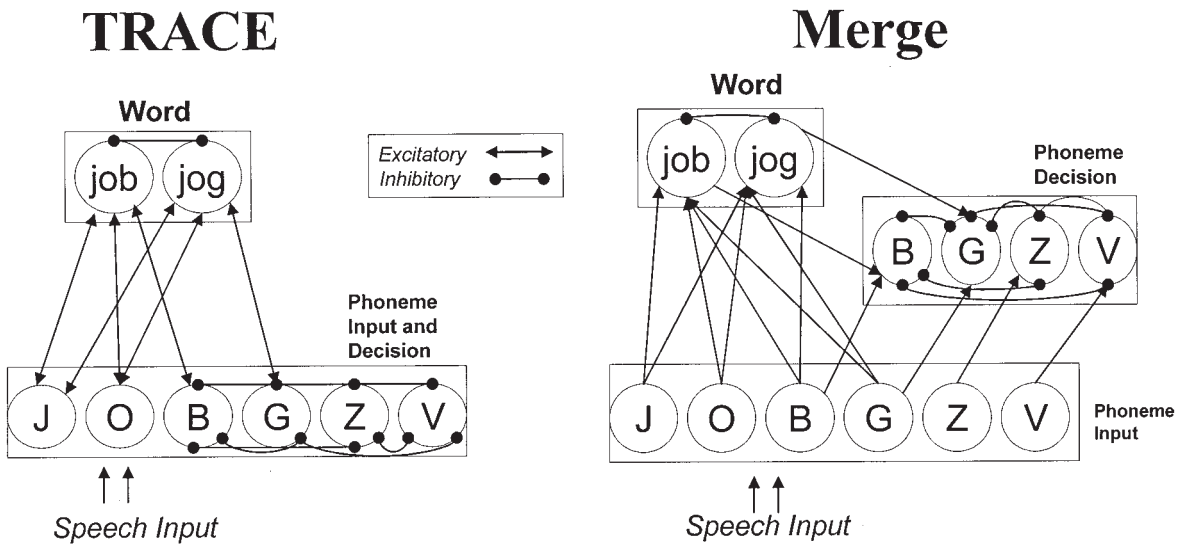


Figure 9. Schematic diagrams of TRACE and Merge.

In order to compare the two models, it was necessary to equate them on all nonessential dimensions (e.g., size of lexicon) to ensure that differences in performance were attributable only to design differences (e.g., connectivity, number of parameters). In essence, we wanted to compare the fundamental structural and functional properties that define the models. We did this by first implementing the version of Merge described in Norris et al. (2000), and then making the necessary changes to Merge to turn it into TRACE. In the end, TRACE required four fewer parameters than MERGE. The names of the parameters, along with other model details, are provided in Appendix C.

The Subcategorical Mismatch Experiment

The first comparison of the two models was performed in the context of the subcategorical mismatch experiment of Marslen-Wilson and Warren (1994, Experiment 1; McQueen, Norris, & Cutler, 1999, Experiment 3). The setup is attractive because of the large number of conditions and response alternatives, which together permitted detailed analyses of model behavior at both the phonemic and lexical levels. In the experiment, listeners heard one-syllable utterances and then had to classify them as words or nonwords (lexical-decision task) or categorize the final phoneme (phonemic decision task). The stimuli were made by appending a phoneme (e.g., /b/ or /z/) that was excised from the end of a word (e.g., *job*) or nonword (e.g., *joz*) to three preceding contexts, to yield six experimental conditions (listed in Table 3). The first context was a new token of those same items but with the final consonant removed (e.g., *jo*), to create cross-spliced versions of *job* and *joz*. The second consisted of equivalent stretches of speech from a word that differed only in the final consonant (e.g., *jo* from *jog* in both cases). The third was the same as the second except that the initial parts were excised from two nonwords (e.g., *jo* from *jod* and *jo* from *jov*).

Because cues to phoneme identity overlap in time (because of coarticulation in speech), a consequence of cross-splicing is that cues to the identity of the final consonant will conflict when the first word ends in a consonant different from the second. For example, *jo* from *jog* contains cues to /g/ at the end of the vowel, which will be present in the resulting stimulus when combined with the /b/ from *job* (Condition 2 in Table 3).

Marslen-Wilson and Warren (1994) were interested in how such stimuli affect phoneme and lexical processing. As the results in Table 3 show (taken from McQueen et al., 1999), in the lexical decision task, reaction times slowed when listeners heard cross-spliced stimuli, but responding was not affected by the source of the conflicting cues (i.e., equivalent RTs in Conditions 2 and 3). Phoneme categorization, in contrast, was sensitive to the subtle variation in phonetic detail, but only when the stimulus itself formed a nonword (e.g., *joz*; Conditions 5 and 6). Notably, the use of cross-spliced stimuli nullifies the bottom-up priority rule in Merge, which otherwise might have contributed to any differences (see Norris et al., 2000, for details). To the extent that differences are found across models, they are probably a result of their different architectures.

The Parameter Space Partitioning Analysis

The subcategorical mismatch experiment contains two dependent measures of performance, classification decisions and the speed with which those decision were made (response time). We treated classification performance as a qualitative pattern in the PSP analysis. We then performed a separate investigation of the RT predictions.

Preliminaries

Classification in these two models is usually defined in terms of the activation state of the network when specific decision criteria are met, such as a phoneme node exceeding an activation threshold. Because there are multiple conditions in the subcategorical mismatch experiment, a data pattern is really a profile of classifications across these conditions. In this experiment there are six conditions, with a phoneme and lexical response in each, for a total of 12 categorization responses that together yield a single data pattern. With four possible phoneme responses and three possible lexical responses, there were a total of 2,985,984 ($4^6 \times 3^6$) patterns. Of interest is how many and which of these patterns TRACE and Merge produce.

We applied two classification decision rules. The effect of these rules is that they establish the necessary mapping between the continuous space of network states to the discrete space of data

Table 3
Design of the Subcategorical Mismatch Simulation in Norris, McQueen, and Cutler (2000)

Condition	Example	Phonemic categorization	Lexical decision	Phonemic categorization				Lexical decision		
Word				/b/	/g/	/z/	/v/	'job'	'jog'	nonword
Matching cues	job+job	668	340	√				√		
Mismatching cues from word	jog+job	804	478	√				√		
Mismatching cues from nonword	jov+job	802	470	√				√		
Nonword				/b/	/g/	/z/	/v/	'job'	'jog'	nonword
Matching cues	joz+joz	706	NA			√				√
Mismatching cues from word	jog+joz	821	NA			√				√
Mismatching cues from nonword	jov+joz	794	NA			√				√

Note. Underlined segments denote the sections of the utterances that were excised and then combined. The condition names describe whether the cues to the final consonant in the first word matched those of the second word. Reaction time data are from McQueen et al (1999). Response sets in each task in the simulations are shown at the right. Check marks denote listeners' dominant response. NA = not applicable.

patterns, making it possible to associate each data pattern with a region in parameter space. Because any one rule could yield a distorted view of model performance, the use of two rules enabled us to assess the generality of the results. In addition, we found that some model properties that are not evident with one criterion emerged when performance was compared across rules. The first rule, labeled *weak threshold,* was a fixed activation threshold, with values of 0.4 for phoneme nodes and 0.2 for lexical nodes. It is the same rule used by Norris et al. (2000) and was adopted to maintain continuity across studies.

The second rule, called the *stringent threshold,* required classification to be more decisive, by requiring the activation level of competing nodes to be significantly lower than the winning node. Two thresholds were used, the higher of which was the lower bound for the chosen node and the lower of which was the upper bound for the nearest competitor. These values were 0.45 and 0.25 for phoneme classification and 0.25 and 0.15 for lexical decision. Two other constraints were also enforced as part of the stringent threshold. There had to be a minimum difference in activation between the winning node and its closest competitor of 0.3 for phoneme classification and 0.15 for lexical decision. Finally, for nonword responses in lexical decision, the difference in activation between the two lexical items, *jog* and *job,* could not be more than 0.1.

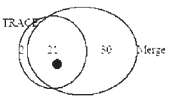
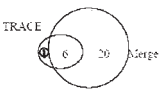
The models were designed as depicted in Figure 9. The only lexical nodes were *job* and *jog.* All necessary phoneme nodes were included, with /b/, /z/, /g/, /v/ being of primary interest. The PSP algorithm was run for both models and both decision rules. To obtain a data pattern, all six stimuli were fed into the model, and both phoneme and lexical classification responses assessed. The algorithm and models were run in Matlab on a Pentium IV computer. The time required to find all patterns varied greatly, taking as little as 22 min (TRACE, stringent threshold) and as long as 24 hr (Merge, stringent threshold). The consistency of results was ascertained using five multiple runs for each model and threshold. The averaged data are presented below.

How Many Patterns Do the Models Produce?

The analysis began by comparing classification (asymptotic) performance of the two models. Table 4 contains three measures that define their relationship under the weak and stringent thresholds. The second column contains Venn diagrams that depict the similarity relation between the models when measured in terms of the number of data patterns each can generate. Looking first at the weak threshold data, both models generate 21 common patterns, with TRACE producing only a few unique patterns compared with Merge (2 vs. 30). The filled dot represents the empirical pattern, which both models produced.

The nature of the overlap in the diagram indicates that TRACE is virtually nested within Merge, with 21 of its 23 patterns also being ones that Merge produces. However, Merge can produce an extra 30 patterns, suggesting that in the subcategorical mismatch design, Merge is more flexible (i.e., complex) than TRACE.[1] That said, the difference between them is not all that great, although it does not disappear under further scrutiny (see below). Perhaps most impressively, both models are highly constrained in their performance, generating fewer than 60 of the some 3 million patterns that are possible in the design.

Table 4
Classification Results From the Subcategorical Mismatch Test

Threshold	Pattern overlap	Percentage of total volume occupied by valid patterns		Percentage of valid volume occupied by common patterns	
		TRACE	Merge	TRACE	Merge
Weak		4%	8%	100%	54%
Stringent		1%	2%	92%	23%

When the stringent threshold is imposed, both models generate fewer patterns, with TRACE producing 7 and Merge 26, as indicated in the lower half of Table 4. Nevertheless, the relationship between the models remains unchanged: TRACE is almost nested within Merge. However, the one unique TRACE pattern turns out to be the empirical data, which Merge no longer generates.

Notably, the change in threshold primarily caused a drop in the number of shared patterns, indicating that the models are more distinct under the stringent threshold. This can be understood by turning one's attention to columns 3 and 4, which contain estimates of the proportion of the parameter space occupied by the valid data patterns. Although the percentages for both models is rather small, half as much of TRACE's parameter space is used compared with Merge's. Predictably, the regions shrink when the stringent threshold is applied. If one then examines how much of each valid volume is occupied by common and unique patterns, the reason for the increased distinctiveness of the models presents itself. For TRACE (column 5), this region is occupied almost entirely by the 21 common patterns (100%). Under the stringent threshold, this value drops slightly to 92%, but because there is only one unique pattern in this case, its value must be 8%, the size of the region occupied by the empirical pattern. The remaining 15 common patterns (21 – 6) occupied such tiny regions in TRACE's parameter space under the weak threshold that application of the stringent threshold eliminated them. A similar situation occurred with Merge (column 6), with 12 of the 16 common patterns (of which the empirical pattern was one) disappearing as a result of a change in threshold. Four became unique to Merge. A few patterns unique to each model also failed to satisfy the stringent threshold (2 for TRACE and 7 for Merge).

[1] Indeed, equating the number of patterns with model complexity is a natural and well-justified definition of complexity (Myung, Balasubramanian, & Pitt, 2000).

Although a change in threshold brings out differences in the models, analysis of their common patterns under both thresholds reveals an impressive degree of similarity. For example, under the weak threshold, the regions in parameter space of all 21 patterns are comparable in size across models. To measure this, we correlated the rank orderings (from smallest to largest) of the volumes in the two models. Use of the actual volume estimates themselves is inappropriate because of differences in model structure and parameterization. With $\rho = .75$, the correlation is high. For the stringent threshold it is slightly lower, $\rho = .60$, but keep in mind that there were only six data points in this analysis. These associations indicate that the overlap between models is not just nominal in terms of shared data patterns, but those regions are similar in relative size with all other common regions, making the models functionally highly similar.

Is the Variation Across Patterns Sensible?

To the extent that a model produces data patterns other than the empirical one, a mark of a good model is for its performance to degrade gracefully. In the first application of PSP, ALCOVE's performance degraded gracefully because all of its patterns looked similar to the empirical one. In the current analysis, we measured deviation by counting the number of mismatches (different decisions) between a particular pattern and the empirical one. Because a pattern consists of 12 decisions, there are a maximum of 12 possible mismatches (6 phonemic and 6 lexical). Histograms of the mismatch counts were created for both models and are shown in Figure 10 alongside a histogram showing the base rate at each mismatch distance (e.g., there are thousands of possible patterns with six mismatches, but only 12 patterns with one mismatch).

Both models show a remarkable proclivity to produce human-like data. The distributions are located near the zero endpoint (empirical pattern) with peaks between two and four mismatches. The probability is virtually zero that a random model (i.e., a random sample of patterns) would display such a low mismatch frequency. The weak threshold distributions are slightly to the right of the stringent threshold distributions, and Merge's distributions are slightly to the right of TRACE's. In both cases, this is probably a base rate effect; allowing more patterns increases the likelihood of more mismatches.[2]

An equally desirable property of a model is that patterns that mismatch across many conditions occupy a much smaller region in the parameter space than those that mismatch by only one or two conditions. That is, larger regions should correspond to patterns that are more similar to the empirical pattern. When region volume is correlated with mismatch distance, this is generally true for both models but stronger for Merge ($\rho = -0.51$) than TRACE ($\rho = -0.38$).

The mismatch distributions beg the question, What types of classification errors do the models make? To answer this, the proportion of mismatches in each of the 12 conditions was computed for each model, and these are plotted in Figure 11. The profile of mismatches across conditions again reveals more similarities than differences between the models. For the most part the models performed similarly. The only noteworthy differences are that Merge produced more phoneme misclassifications (Conditions 3 and 6), and TRACE produced a greater proportion of lexical misclassifications. In the four phoneme conditions in which errors were made, the final phoneme was created by cross-splicing two different phonemes, creating input that specified one weakly and the other strongly. The errors are a result of misclassifying the phoneme as the more weakly specified (lowercase) alternative (e.g., *g* instead of *B*). By design, Merge's bottom-up only architecture heightens its sensitivity to sensory information, which in the present simulation made it a bit more likely than TRACE to misclassify cross-spliced phonemes.

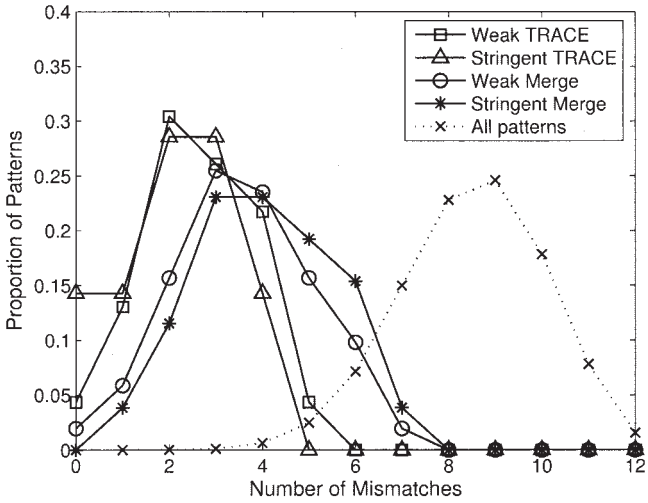


Figure 10. Similarity of all Merge and TRACE data patterns to the empirical (human) pattern measured in terms of the number of mismatches to the empirical pattern. Data from the application of the weak and stringent thresholds are shown separately. The dashed line represents the distribution of mismatches for all patterns in the experimental design.

Lexical misclassification errors in TRACE are due primarily to a bias to respond "nonword." However, a small percentage of these lexical errors were due to classifying the stimulus as *jog* (Condition 5). This also occurred in Condition 2, but much less often, with misclassifications as *jog* constituting 8% of the mismatches for TRACE and 3% for Merge.

Which Patterns Matter?

Although some misclassifications can be legitimized on many grounds (e.g., humans make such errors, perception of ambiguous stimuli will not be constant), it is important to determine whether they are characteristic behaviors of the model or idiosyncratic patterns, such as the "particulars" defined in the ALCOVE analysis. That is, it is useful to distinguish between unrepresentative and representative behaviors. To do so, we measured the volumes of all regions identified by the PSP algorithm. Recall that absolute volume estimates are not compared directly, but rather relative to the total volume of the valid regions of the model, as ratios (percentages). This step is essential because it makes it possible to compare volume estimates of equivalent regions (same data pattern) between models. The size of a region occupied by a data pattern is normalized by the total volume of each model, placing them on an equal footing. Differences in volume must therefore be

[2] Subsequent analyses in this section are restricted to the weak-threshold data, in part because of the paucity of common patterns. General conclusions do not change.

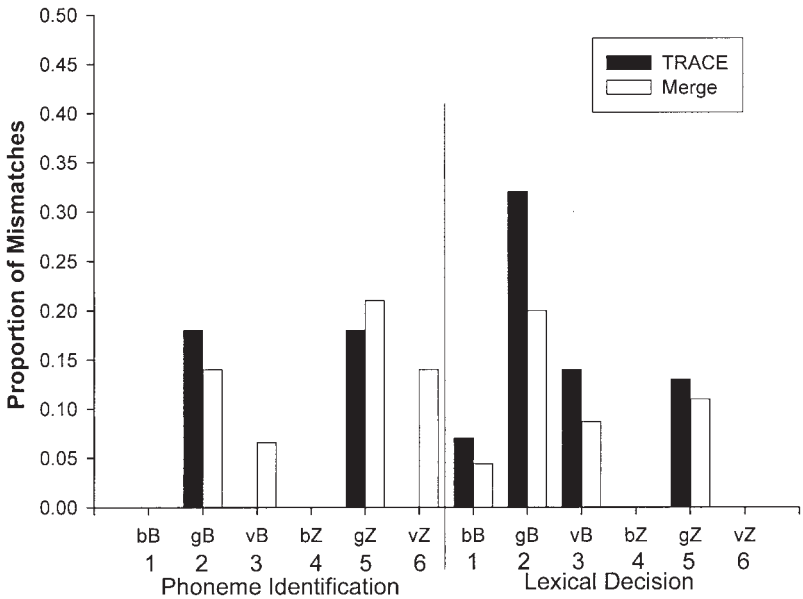


Figure 11. Proportion of mismatches by Merge and TRACE across the 12 conditions. The two-letter sequences are shorthand descriptions of the cross-spliced stimuli used in the conditions. The lowercase letter denotes the final phoneme of the first word, and the uppercase letter denotes the final phoneme of the second word.

attributable to differences in model behavior, whether it is something as simple as parameter range differences or something more fundamental, such as how the parameters are combined.

As in the ALCOVE analysis, a threshold of 1% of the valid volume was adopted to define a meaningful pattern. When this is done, many patterns turn out to be noise and the set of representative patterns is reduced to a handful. For TRACE, 21 of its patterns (2 unique and 18 common) do not meet this criterion. Four patterns, all common, dominate in volume, together accounting for 99% of the volume (range = 4.1% to 57.9%). For Merge, the set of dominant patterns is larger and is split equally between common and unique patterns. Thirty-six patterns (21 unique and 15 common) fail to reach the 1% criterion. Seven common (range = 2.0% to 27.6%) and eight unique (range = 1.4% to 21.1%) patterns do so and make up 97% of the valid volume. Even with a threshold that eliminated 75% of all patterns, the asymmetry in pattern generation between the models is still present (TRACE = 4; Merge = 15).

The volumes of these representative common patterns are graphed in Figure 12. The numerals in the legend refer to the mismatch distance of each common pattern from the empirical pattern. Most obvious is the fact that the empirical pattern is much larger in TRACE than in Merge (27% and 6.8%) and that one mismatching pattern (filled black) dominates in both models (57.9% and 27.6%). This pattern turns out to be one in which there is a bias to classify all stimuli as nonwords. As a group, the eight unique Merge patterns mismatch the empirical pattern more than the common patterns. The largest pattern occupies a region of 21.1% (five mismatches), more than twice the next largest region (8.4%). In this pattern, not only did Merge exhibit the same nonword response bias, but it also categorized cross-spliced phonemes as the competing (remnant) phoneme.

Response Time Analyses

Up to this point in the analysis we have compared only the classification behavior of the models, but the time course of processing is equally important, as experimental predictions often hinge on differences in RT between conditions. TRACE and Merge are often evaluated on their ability to classify stimuli at a rate (e.g., number of cycles) that maintains the same ordinal relations across conditions found with RTs. To assess the robustness of simulation performance, parameters can be varied slightly and additional simulations then run to ensure that the model does not violate this ordering (e.g., by producing a change in the RT pattern). In their comparison of Merge and TRACE, Norris et al. (2000) found that neither produced an RT reversal. Our test was a more exhaustive version of theirs.

The PSP analysis defines for us the region in the parameter space of each model that corresponds to the empirical data pattern. When assessing RT performance, what we want to know is whether there are any points (i.e., parameter sets) in this region that yield invalid RT patterns, as defined by the ordinal relations among conditions in the experiment (i.e., the RT pattern in the six phonemic and three lexical conditions in Table 3). To perform this analysis, 10,000 points were sampled over the uniform distribution of the empirical region. Simulations were then run with each sample, and the cycle time at which classification occurred was measured and compared across all conditions. Violations were defined as reversals in cycle times between adjacent conditions (e.g., Condition 1 vs. Condition 2) in phoneme classification and lexical decision. Just as Norris et al. (2000) reported, we did not find a single reversal between conditions for either model.

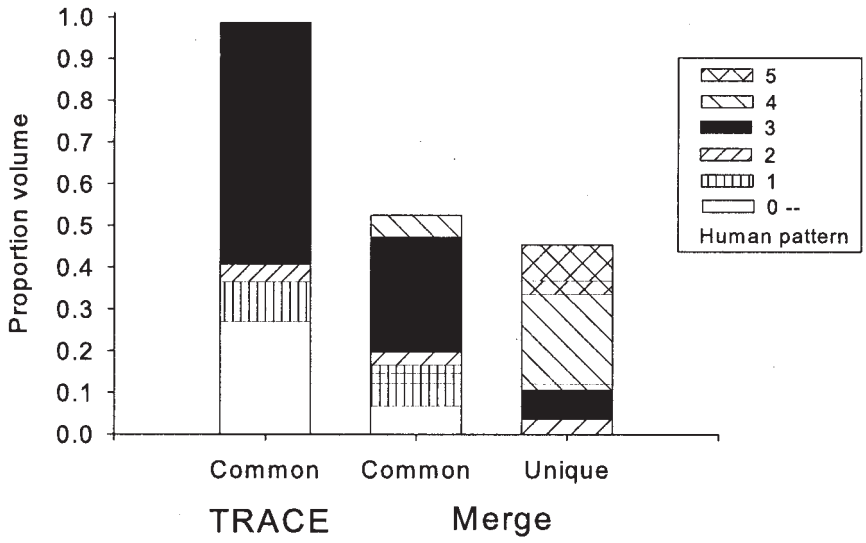


Figure 12. Comparison of Merge and TRACE volumes in the subcategorical mismatch experiment. Regions in the valid parameter space correspond to data patterns that occupy more than 1% of the volume. Each slice represents a different data pattern. Numerals in the legend specify the mismatch distance of the pattern from the empirical (human) data.

Summary

The PSP analyses reveal that the consequence of splitting the phoneme level in two is to produce a slightly more flexible model. This was found when all patterns were considered and when those most representative were considered, so the nature of this additional flexibility is of some interest. By splitting the phoneme level in two, Merge was not transformed into a different model. Rather, the model's behavior was expanded beyond that of TRACE, as the nested relationships in Table 4 show. Merge retained many of TRACE's behaviors (producing most of its patterns) plus acquired new ones. A consequence of this expansion is that the representativeness of these behaviors is considerably different across models, as shown in Figure 12. It is because these differences are quantitative and not qualitative that the models are so similar on other dimensions (e.g., frequency and types of mismatches, relative region size, response time relations between conditions). PSP is a powerful tool for addressing pointed questions about the functionality of model architectures. Another example is presented next. Other uses of PSP are discussed at the end of the article.

Application 3: The Bottom-Up Priority Rule in Merge and TRACE

Background to the Analysis

The Bottom-Up Priority Rule

This second comparison of TRACE and Merge was performed to determine how the addition of a bottom-up priority (BUP) rule in Merge distinguishes it from TRACE. Recall that in Merge, for a phoneme decision node to become activated (see Figure 9), excitation must be initiated from the phoneme input stage (i.e., evidence for the phoneme must be in the acoustic signal). In TRACE, this is not required. Initial activation via top-down connections from the word stage is possible. For example, having been presented with *jo* in *job*, excitation from the *job* node will feed back and excite the *b* node. Inhibitory connections between phoneme nodes make it possible to observe indirect word-to-phoneme inhibition, because the activated *b* node will in turn inhibit competing phoneme nodes (e.g., *g*). Thus, word nodes have the ability to excite phoneme nodes directly and inhibit them indirectly.

Of course, one can easily incorporate a BUP rule into TRACE, simply by not allowing top-down feedback to influence a phoneme node until after that node has received some bottom-up input. Similarly, the BUP rule can be removed from Merge, by allowing phoneme decision units to be activated by word layer units without first having to be activated by phonemic input units.

This observation suggests an elegant way to assess the rule's contribution to model behavior: Run PSP analyses on both models with and without the BUP rule. This amounts to the 2×2 factorial design shown in Table 5. The lower left and upper right cells represent the models as originally formulated. This comparison serves as a reference from which to understand the contribution of the priority rule and the model's structures in affecting behavior. Comparisons of results between columns (i.e., models) neutralize the effects of the priority rule. If the rule is primarily responsible for differences in model behavior, then the results for both models

Table 5
Factorial Design of the Parameter Space Partitioning Comparisons That Were Performed to Determine Whether the Differences in Bottom-Up Priority in the Two Models Is What Distinguishes Their Behaviors in the Test of Indirect Inhibition

	Model	
Bottom-up priority rule	TRACE	Merge
Yes		
No		

should be quite similar when the rule is and is not operational. If differences still remain, then structural differences are also contributing to their diverse behaviors. In short, these analyses will tell us whether Merge, without its priority rule, behaves like TRACE, and whether TRACE, with the priority rule, mimics MERGE.

The Frauenfelder et al. (1990) Data

If indirect word-to-phoneme activation is possible, then one would expect anomalous word endings to slow down responses to those endings. Frauenfelder et al. (1990, Experiment 3) tested this prediction by having listeners monitor for phonemes that occurred late in multisyllabic words and nonwords. Three conditions are of interest in the present analysis. A word condition (e.g., *habit*, with /t/ as the target phoneme) served as a lower bound on responding because lexical and phonemic information should combine to yield fast RTs. A control nonword condition (e.g., *mabil*, with /l/ as the target phoneme) served as a reference against which to measure inhibition. Because *mabil* is not a word, there should be no top-down lexical facilitation or inhibition when responding to /l/; responses should be based on sensory input alone. In the third, inhibitory condition, listeners heard nonwords such as *habil*, with *l* as the target phoneme. A slowdown in RT relative to the control nonword condition is expected if there is in fact inhibition. This is because the first part of the stimulus, *habi*, will excite *habit*, whose activation should then feed back down and excite /t/, which will then inhibit /l/.

The RT slowdown in the inhibition condition was small and not reliable despite significant effects in companion conditions and experiments. That this seemingly clear prediction of TRACE was not borne out in the data has generally been interpreted as arguing against word-to-phoneme excitation in TRACE. However, in simulations of indirect inhibition, TRACE's behavior is not cut and dry, with inhibition being more likely with longer than shorter words (Norris et al., 2000). In contrast, the BUP rule in Merge guarantees that it produces consistent performance that never yields inhibition. TRACE's variable behavior is a sign that it can generate more data patterns than Merge. If this is the case, is the absence of the priority rule the main cause, or is it also due to structural differences between the models?

The Parameter Space Partitioning Analysis

Preliminaries

The design of the indirect inhibition experiment is much simpler than the subcategorical mismatch experiment. There are only three conditions and only a single response (phonemic decision). To simulate the experiment, the combination of so few conditions and a simple model design (one lexical and two critical phoneme nodes) would yield so few potential data patterns that the analysis might not provide satisfying answers to our question. We therefore added a lexical decision response (word or nonword) to the design on the grounds that listeners would, if asked, accurately categorize each stimulus as a word or nonword. The models should perform similarly. This additional response permitted a more fine-grained analysis and comparison of the models.

Input to the models consisted of three utterances: *habit*, *mabil*, *habil*. They were selected to be moderately long (five phonemes) so that indirect inhibition would have a chance to emerge. Both models were modified from the previous test to consist of only one lexical node (*habit*) and the appropriate phoneme input/decision nodes, with /t/ and /l/ of most interest because their activation functions were used to test for inhibition. With two classification responses, each with two alternatives, and three stimulus conditions, there were a total of 64 possible data patterns ($2^3 \times 2^3$).

To examine the effect of the BUP rule on model performance, we adjusted phoneme activation parameters accordingly in each model prior to running the PSP algorithm. The same two decision rules were again used to assess the generality of results. With two decision rules and two priority rules, the algorithm was run four times on each model. The consistency of the results for each analysis was ascertained by rerunning the algorithm five times. The averaged data are presented. No more than 7 min were required to find all patterns in any run.

Classification Analyses

The classification data are shown in column 3 of Table 6, with the first row containing the comparison of the models as originally designed (TRACE without BUP and Merge with it). The Venn diagram shows that the relationship between the models is opposite that in the subcategorical mismatch experiment (see Table 4), with Merge now embedded in TRACE, and TRACE producing three times as many data patterns as Merge (12 vs. 4). Both models are again highly constrained in their behavior, producing nowhere near the 64 possible data patterns in the experimental design.

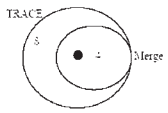
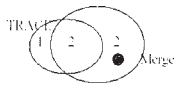
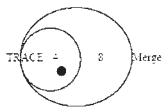
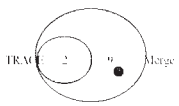
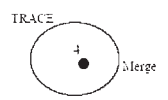
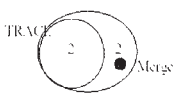
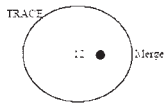
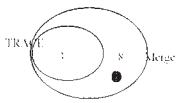
TRACE's extra flexibility in this experimental setup agrees with Norris et al.'s (2000) observation of TRACE's variable behavior in producing indirect inhibition. That this flexibility is due to the absence of a BUP rule can be seen by examining the Venn diagrams in the remaining rows. When the priority rules are swapped between models (row 2), their relationship reverses as well. TRACE shrinks from 12 to 4 patterns while Merge grows from 4 to 12, embedding TRACE in Merge.

This reversal is not merely a reversal in the number of data patterns but extends to the actual data patterns themselves. That is, even though the models trade places in terms of their relationship, the common and unique patterns remain the same. To see that this is the case, the reader is referred to the Venn diagrams in rows 3 and 4, where the priority rule is either on or off at the same time in the two models. In both situations TRACE and Merge overlap completely, producing only common patterns, four when the priority rule is on and 12 when it is off, which matches exactly what was found in rows 1 and 2. The preceding observations can be neatly summarized by noting that when the priority rule was on, the models always generated four patterns. When it was off, they generated 12.

The isomorphism of the models with and without the priority rule is surprising. Within this experimental design, their qualitative behaviors are identical in terms of the patterns they can generate and point to the priority rule itself as a primary determiner of behavioral differences. As will be seen shortly, this interchangeability also shows up in the RT analyses.

Columns 4–7 in Table 6 contain volume measurements whose relationship between models is somewhat different from that found in the subcategorical mismatch analyses. Although both are again small, TRACE's valid region of the parameter space is the same size as Merge's. This relationship changes little as a function of the priority rule. When common data patterns occupy only a portion of

Table 6
Results From the Parameter Space Partitioning (PSP) Analyses of Merge and TRACE in the Indirect Inhibition Simulation

Bottom-up Priority Rule		**Pattern overlap Weak Threshold**	**Percentage of total volume occupied by valid patterns**		**Percentage of valid volume occupied by common patterns**		**Pattern overlap Stringent Threshold**
TRACE	Merge		TRACE	Merge	TRACE	Merge	
no	yes		10%	10%	72%	100%	
yes	no		8%	9%	100%	60%	
yes	yes		8%	10%	100%	100%	
no	no		10%	9%	100%	100%	

the valid volume (rows 1 and 2), the region is larger for TRACE (72%) than Merge (60%), indicating that the unique patterns occupy a smaller region in TRACE's parameter space. This outcome foreshadows what happens under the stringent threshold.

A quick glance down the last column of Table 6 (stringent threshold) shows the isomorphism between the models no longer holds. This can be seen most clearly in the top cell, where the embedding is the reverse of that found with the weak threshold. TRACE produced only 3 of the 12 patterns, two of which are shared by Merge. In contrast, Merge produced the same four patterns as found under the weak threshold. In rows 2–4, TRACE remains embedded in Merge, always producing the same few patterns (never the empirical one). It appears that under the stringent threshold, TRACE generated so few patterns to begin with that there was little opportunity for the priority rule to alter model behavior. In contrast, Merge behaved just as it did under the weak threshold, generating more patterns without the rule (rows 2 and 4) than with it (rows 1 and 3). Notice that what changes most down this column is the number of unique patterns generated by Merge.[3]

The reason for this asymmetric effect of threshold on the two models can be understood by examining the volume measurements under the weak threshold in Figure 13. Region sizes are shown for both models with and without the priority rule. The reader is referred to the two "No BUP" bars. Each slice of a vertical bar represents a different data pattern, except those denoted with gray shading at the top of the bar, which represents the combined area of eight regions, most of which are visible for Merge but not for TRACE. In fact, these eight bars together total less than 1% of TRACE's valid volume. These regions plus the empirical region are so small that they disappear when the stringent threshold is applied, which is the cause of the sharp drop in the number of data patterns. These same regions are much larger in Merge (only one is less than 1% of the volume), and although they shrink in size, they do not disappear under the stringent threshold.

Further insight into the effects of the priority rule on model behavior can be obtained by comparing the types of misclassification errors made by the models when the rule was on and off. As in Figure 12, each slice of a vertical bar in Figure 13 is filled with a graphic pattern that denotes the mismatch distance of the data pattern from the empirical data. In TRACE, the empirical pattern (bottom row) itself occupies a small region in the parameter space that increases slightly in size when bottom-up information is given priority. In Merge, however, not only is this region much larger, but it increases in size when the priority rule is invoked, occupying a full 27% of the valid volume. Although TRACE can produce the empirical pattern, it is clearly a more central pattern in Merge. What is more, the priority rule enhances this centrality.

Most of the mismatching patterns, especially those that occupied the largest regions, rarely veered far from the empirical pattern,

[3] The four corresponding columns of percentages for the stringent threshold were omitted from Table 6 because the values changed in the same ways as those in the analysis under the weak threshold.

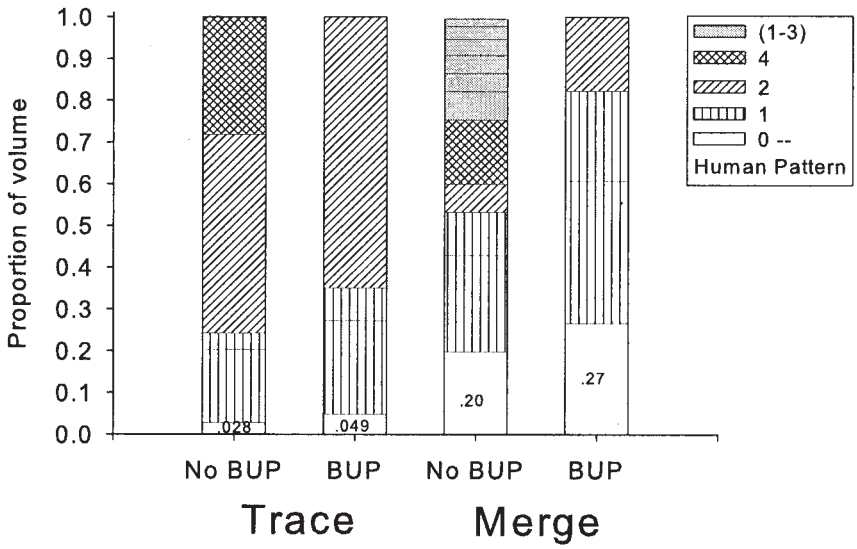


Figure 13. Comparison of Merge and TRACE volumes in the indirect inhibition experiment: Regions that correspond to data patterns of each model without the bottom-up priority rule (No BUP) and with the rule (BUP). Each bar represents a different data pattern, whose shading specifies its mismatch distance from the empirical pattern (see numerals in legend).

differing by one or two responses out of a possible six. Comparison of the bars between models shows that the same biases exhibited by the models in the subcategorical mismatch experiment are present here. A few patterns dominate the parameter space in TRACE, whereas the space is split more equitably among patterns in Merge. This is most evident in the No BUP conditions, where a pattern with two mismatches occupies almost half (47.6%) of TRACE's volume. The errors in this instance are due to lexical misclassifications in which the two nonwords (*mabit* and *habil*) were categorized as words. Merge exhibits a much weaker tendency to do this (6.7%). Instead, the pattern occupying the largest volume in Merge does just the opposite: *habit* is classified as a nonword. TRACE produces this error as well. Application of the priority rule (bars 2 and 4) increases these tendencies in both models (i.e., the regions increase in size).

Reaction Time Analyses

RT analyses were performed only on the weak threshold data because TRACE failed to generate the human pattern under the stringent threshold. The region occupied by the empirical pattern was probed for parameter sets that violated the ordering of mean participant response times across the three conditions (*habit* < *mabil* = *habil*). Model behavior with and without the priority rule was also examined.

The procedure was the same as that used in the subcategorical mismatch analysis. Samples (10,000) were drawn from the uniform distribution over the region of the empirical pattern, and the model was run with each set of parameter values. The cycle at which phoneme classification occurred was recorded across conditions. Because violations amount to a reversal in cycle classification time between adjacent conditions, we calculated the difference in cycle time of neighboring conditions, *habit* versus *mabil*, and *mabil* versus *habil*. Cycle times to *mabil* were subtracted from those in the other two conditions. Distributions of these difference scores (over all samples) are plotted in Figure 14. The top graphs contain the comparison of the models as originally formulated (TRACE without BUP, Merge with BUP). In the bottom pair of graphs, the priority rules were swapped across models. The left-hand graphs contain the *habit–mabil* comparison and the right-hand graphs the more theoretically important *habil–mabil* comparison.

In the upper left graph, both models produce the correct ordering, with phoneme classification always being faster in a word (*habit*) than nonword (*mabil*) context. What differs between the models are the shapes of the distributions, with TRACE's being sharply peaked and Merge's more diffuse. In the right graph, the effects of the priority rule are visible, with Merge always generating the correct prediction of identical classification times in the two conditions (zero RT difference, a single point) and TRACE showing indirect inhibition, with recognition times slightly longer to /l/ in *habil* than in *mabil*, hence the negative cycle difference score. Although this effect is almost always small, it is clear from the shape of the distribution that choice of parameter settings could lead to different conclusions, where on rare occasion the effect would be large.

That equivalent classification times in the *mabil* and *habil* conditions are due to the priority rule can be seen in the lower right graph, where TRACE was run with the BUP rule and Merge without it. Just as in the classification data in Table 6, their performance reverses: TRACE now produces no difference across conditions, whereas Merge displays indirect inhibition.

Finally, scanning across all four graphs, it is clear that the interactive architecture of TRACE constrains differences in classification time more so than the integrative architecture of Merge. Regardless of priority rule, both models show a tendency to produce small rather than large differences scores, but TRACE more so than Merge, given the height and location of the peaks of the distributions.

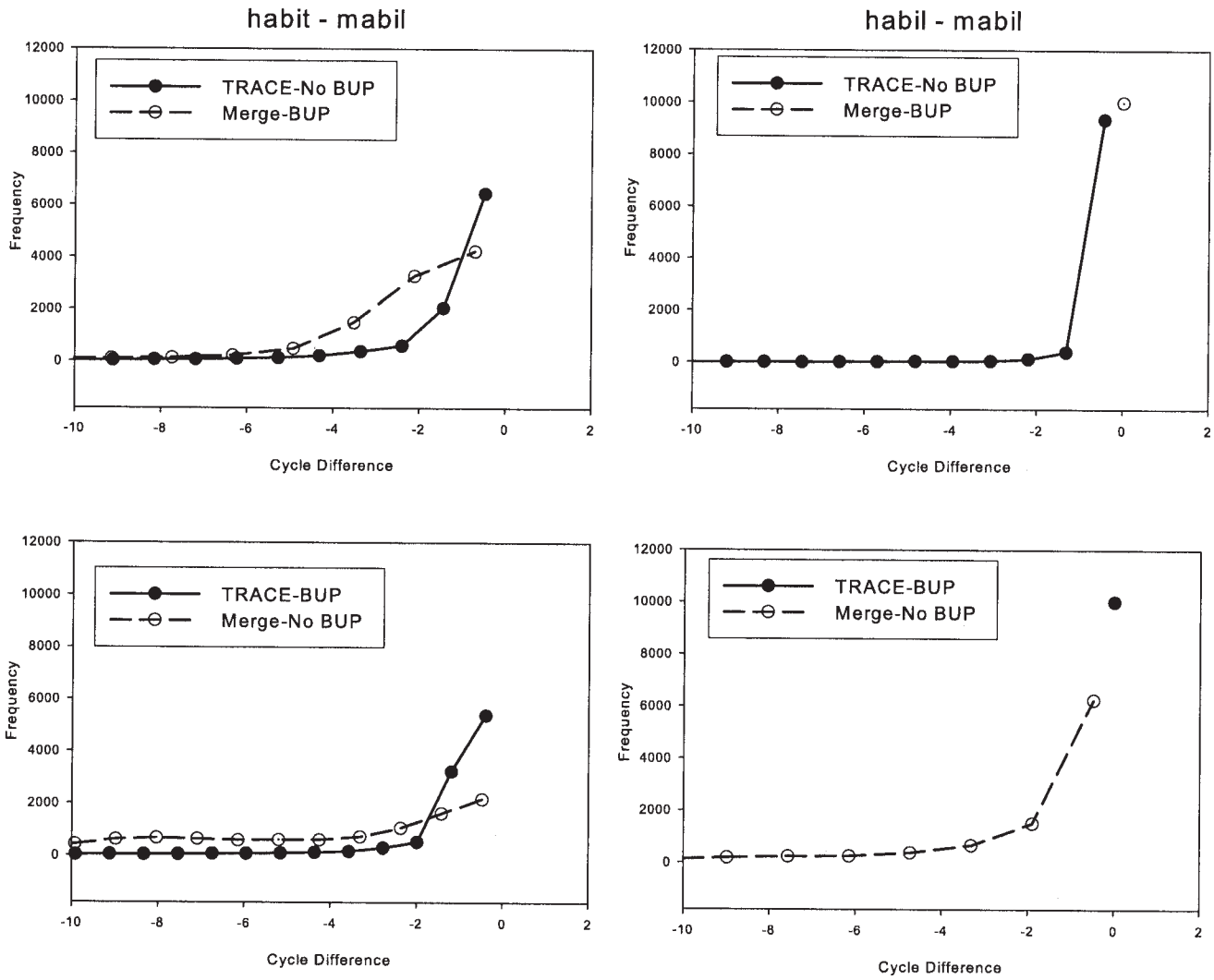


Figure 14. A comparison of model classification response times in the indirect inhibition experiment. Differences in classification times (cycles) between adjacent experimental conditions are plotted for each model, with the *mabil* versus *habit* comparison on the left, and the *mabil* versus *habil* comparison on the right. In the top graphs, the models were run as originally formulated (TRACE without the bottom-up priority rule [BUP], Merge with the rule [BUP]). In the bottom graphs, TRACE was run with the priority rule, and Merge was run without it.

Summary

This second comparison of TRACE and Merge was undertaken to learn how the BUP rule can differentiate the two models and to demonstrate another way in which PSP can be used to study model behavior. The PSP analyses under the weak and stringent threshold showed that the priority rule reduces model flexibility and confirmed its necessity for simulating the correct pattern of response times. By turning the priority rule on and off across models, we learned that they can behave identically, producing qualitatively indistinguishable data patterns. Only when the volumes of these patterns were inspected did performance differences due to model architecture emerge. Merge's integrative architecture is somewhat better suited for mimicking human behavior in this experimental design. Not only is the empirical pattern more central in Merge, but this was true regardless of whether the priority rule was in place. These findings and conclusions are possible because of the global perspective on model behavior that a PSP analysis provides.

General Discussion

Models are becoming increasingly useful tools for psychologists. To use them most productively, it is important to have a thorough understanding of their behavior. Although a host of methods have been developed with this purpose in mind (see Figure 1), their applicability is limited, in large part because of the diversity of models in psychology. To the extent that new methods are introduced, it would be most helpful to understand model behavior at a level of granularity that matches the qualitative predictions that are made when testing models experimentally.

PSP was developed to meet these goals. A model's ability to mimic human behavior is evaluated in the context of the other data patterns that it can generate in an experimental design, thereby providing a richer understanding of what it means for a model to account for an experimental result. We specifically chose connectionist models to demonstrate PSP's versatility, as there are few methods available for analyzing these widely used yet complex models. The method is not limited to this class of models, however. It can be just as easily applied to other types (e.g., algebraic), which makes comparison of models across classes a relatively straightforward undertaking. PSP's generality is even evident in the algorithm's implementation: A model is a module in the algorithm, making it possible to insert many types of models.

The three example applications demonstrate only some of what can be learned with the method. In the case of ALCOVE, PSP was used to perform a basic task in model analysis, that is, evaluating the model's soundness in capturing an empirical result. Not only did the analysis inform us about whether the model can mimic human performance (its local behavior), but, much more importantly, by studying the partitioned parameter space, we learned how representative this behavior is of the model and how many other data patterns the model can produce. Some of these were plausible alternative response patterns. Others occupied such a small region in the parameter space that they should be considered spurious and uncharacteristic of the model.

In the second and third applications, PSP was used to study the behavioral consequences of slight design differences between two localist connectionist models. To do this, we compared data patterns and their corresponding volumes in parameter space across models in two experimental settings that were intended to bring out design differences. Many more similarities than differences were found in classification behavior. RT analyses of the region occupied by the empirical data led to the same conclusion. Qualitatively, TRACE and Merge were indistinguishable. Only in the volume analyses did differences emerge. These took the form of biases in emphasizing some behaviors over others.

Although one might not be surprised by the models' similarities, the PSP analysis provides the evidence to solidify such a claim. Furthermore, the analyses suggest how difficult it could be to find a situation in which they make different qualitative predictions that could be tested in an experiment. That said, PSP is just the tool to assist in such an endeavor, by enabling the researcher to essentially pretest an experiment to learn whether it can discriminate between models. Once an experiment is designed, the models can be run through the PSP algorithm to determine which data patterns are shared and which are unique. To the extent that participants are likely to produce those in the latter category, the experiment has the potential to discriminate between the models.

Of course, the meaningfulness of any such analysis depends on how a data pattern is defined. Conclusions may be specific to a definition, which is why it is recommended to use more than one. Because ordinal predictions dominate in most experimental work, the task of defining a data pattern can be relatively straightforward. This is another reason why PSP is widely applicable. It does not depend on quantitative fits.

PSP can also be of considerable use in model development by making the modeler aware of the broader consequences of the choices made in model design. For example, there may be four key experimental results that any model of memory must capture to be taken seriously. By running the algorithm in each experimental setting, a comprehensive analysis of the model's design can be undertaken. Desirable and undesirable behaviors can be readily identified. The model can then be refined on the basis of this knowledge and the entire process repeated. From successive iterations, one can develop a clear understanding of the model's inner workings (e.g., how parameters contribute to model behavior, whether there are redundancies across parameters such as interdependency, the appropriateness of their ranges, and even optimal parameter values).

Figure 15 provides a glimpse of what can be learned from such detailed analyses. Shown are distributions of parameter values for three of TRACE's parameters that were obtained within the empirical region of the parameter space in the indirect inhibition experiment (no BUP). Each distribution contains 10,000 values and gives a sense of how the parameter contributes in capturing the experimental result. For the phoneme excitation parameter, the distribution of possible values is fairly broad, although there is a clear optimum. The phoneme-to-word excitation parameter is much more tightly constrained, so much so that one might wonder whether its range should be cut in half, from 0 to 0.5. The distribution for the phoneme-to-phoneme inhibition parameter is essentially flat, suggesting a high degree of context sensitivity, which goes along with Norris et al.'s (2000) observations about the stability of indirect inhibition. Analyses such as these expose the

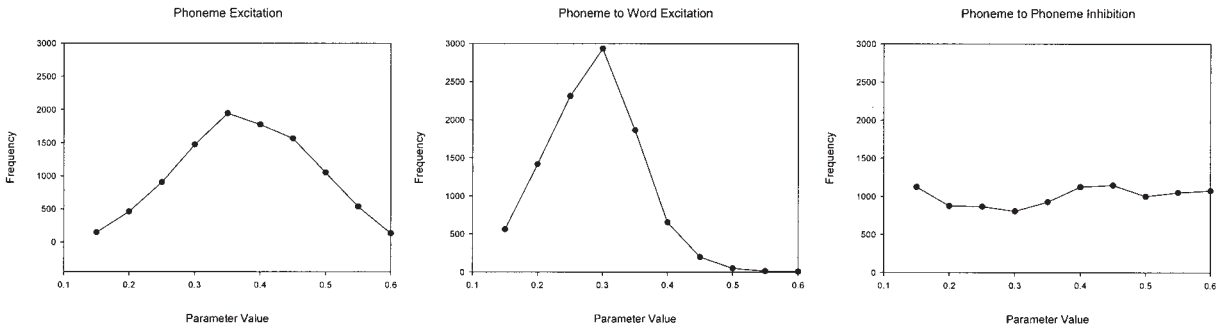


Figure 15. Distributions of values of three TRACE parameters from 10,000 samples of the region in parameter space corresponding to the empirical data pattern (without the priority rule). Parameter names are listed above each graph.

modeler to tendencies and behaviors that can be difficult to anticipate but crucial to know.

Not only can the modeler use this information to enhance the model itself, but it can also be incorporated into the PSP analyses by, for example, setting prior distributions over parameter ranges to emphasize reasonable behaviors and discount unreasonable ones. By default, a PSP analysis assumes the density of points within a region is uniform (i.e., no prior knowledge), but if modelers have reason to emphasize certain parameter ranges over others, then this information can be incorporated into the analysis. This was done in the ALCOVE example, in which parameter values were restricted to permit only monotonic learning curves because humans do not produce functions of other shapes.

Throughout the article, the global perspective on model behavior that PSP provides has been examined from the standpoint of the model to determine its flexibility or scope (Cutting, 2000). It can be equally productive to view the situation from the vantage point of the experiment, by asking how much the experiment is likely to challenge the model. To the extent that it can, the experiment has the potential to be a good, meaningful test of the model. In this regard, one lesson that comes out of our applications of PSP is that the fewer the number of conditions and the fewer the possible relationships between them, the less informative the test may be.

The complementary nature of global and local model analysis runs through the preceding discussion. Each informs the other and together they improve inference by providing a richer understanding of a model's relationship to data. Qualitative and quantitative methods are similarly complementary, particularly when examining local model behavior (see Figure 1). Clear model predictions cannot only be tested in an experiment but can further be evaluated by assessing the model's fit to data. The use of both can make a compelling case in favor of a model. By going a step further and performing a global analysis with PSP, additional light can be shed on the quality of the demonstration.

Although fairly general purpose, the current version of PSP is not applicable to nonstationary models, that is, those whose performance can change or adapt as a result of feedback or learning (e.g., some distributed connectionist and algorithmic models). Of the large set of models to which PSP is applicable, challenges in implementation exist on a few fronts. One is in defining a data pattern. In some situations there may be no obvious "natural" definition of a data pattern. A case in point is the power model of retention (forgetting). The response probability y, representing proportion recall, is a continuous function of retention interval t in the form of $y = at^{-b}$ where a and b ($a, b > 0$) are two parameters. Application of PSP requires one to decide whether the retention curve associated with parameter values $(a, b) = (1.0, 0.5)$ should be considered indistinguishable, in some defined sense, from the curve associated with parameter values $(a, b) = (1.1, 0.4)$. In the absence of a justifiable definition of a data pattern in a situation like this, it would be preferable to use MDL or Bayesian methods when possible. Alternatively, one could devise a "generic," suitably interpretable definition of a data pattern.

Another point to keep in mind when applying PSP is the generality of the two types of analyses, counting and volume. A counting analysis is completely generalizable in the sense that the results are independent of how the model is parameterized—a property known as *reparameterization invariance* in statistics. What this means is that the data-fitting abilities of the model will remain unchanged despite different formulations of the model. To illustrate, consider the functions $y = \exp[-(a_1x_1 + a_2x_2 + a_3x_3)]$ and $y = b_1^{x_1}b_2^{x_2}b_3^{x_3}$. These functions are equivalent because the two are related through the reparameterization $b_i = \exp(-a_i)$, i = 1,2,3. Therefore both functions will fit any give data set identically, and consequently, the counting analysis will yield the same number of qualitatively different patterns with either parameterization. On the other hand, a volume analysis is not reparameterization invariant but rather is specific to a particular parameterization. If each parameter were transformed in an arbitrary way (e.g., $\theta_i' = \theta_i + \theta_i^2$), one would obtain a different volume analysis unless the transformation were positively scaled (i.e., $\theta_i' = \alpha\theta_i$, $\alpha > 0$), in which case the results would remain the same.

Depending on the models and testing situation, this lack of invariance may be of little practical importance, especially when a specific parameterization of a model is justified. This is arguably a necessary condition to be considered a model of cognition (Estes, 2002). When psychological constructs are associated with particular parameters, constraints can be imposed that prevent arbitrary transformations. For instance, in the above example, if we assume that the parameters (a_1, a_2, a_3) represent attention weights, then, by definition, they must satisfy the constraints and therefore do not permit any reparameterization because such a transformation would alter the interpretations associated with these parameters.

Some models in cognitive science (e.g., FLMP, Oden & Massaro, 1978) are *nonidentifiable*, that is, two or more parameters are perfectly correlated. What this means is that more than one set of parameters can yield the same fit to a data set. Parameter redundancy like this is not a problem for PSP, per se, which will successfully map out the region corresponding to each data pattern. However, as with all models that are nonidentifiable, interpretation of model performance measures, volume estimates in this case, becomes tricky. Each point in a region could correspond to more than one set of parameters. When it does, which interpretation is correct? Identifying these cases could be challenging as well. Because of these difficulties, before applying PSP, the model should be constrained to make it identifiable (e.g., by fixing the value of one [or more] parameters or equating parameters), a typical practice in statistics.

Finally, because PSP is a search problem, it is subject to many of the same difficulties found in nonlinear function optimization and integration. Because there are no proven methods for avoiding them, users must find work-arounds to minimize their impact, such as repeating the simulation with different starting values. PSP is no different in this regard, and this is why we performed multiple runs in the analyses. Multiple runs provide a measure of the stability, and hence accuracy, of the results. To use PSP productively, the user must be aware of the difficulties faced by search methods and have a good understanding of the model being analyzed. Appendix A elaborates on these issues.[4]

In conclusion, analysis of the global behavior of a model can provide a wealth of information that is useful for understanding a model's performance in a particular testing situation. PSP is a flexible and powerful method of global model analysis, as the examples presented here attest, suggesting it has considerable potential to assist researchers in their quest to model human behavior.

[4] Further details on parameter space partitioning are available at http://quantrm2.psy.ohio-state.edu/injae/psp.html.

References

Box, G. E. P. (1976). Science and statistics. *Journal of the American Statistical Association, 71,* 791–799.

Browne, M. W. (2000). Cross-validation methods. *Journal of Mathematical Psychology, 44,* 108–132.

Chib, S., & Greenberg, E. (1995). Understanding the Metropolis–Hastings algorithm. *The American Statistician, 49,* 327–335.

Cowles, M. K., & Carlin, B. P. (1996). Markov chain Monte Carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association, 91,* 883–904.

Cutting, J. (2000). Accuracy, scope, and flexibility of models. *Journal of Mathematical Psychology, 44,* 3–19.

Dawson, R. W., & Shamanski, K. S. (1994). Connectionism, confusion, and cognitive science. *The Journal of Intelligent Systems, 4,* 215–262.

Dunn, J. C., & James, R. N. (2003). Signed difference analysis: Theory and application. *Journal of Mathematical Psychology, 47,* 389–416.

Estes, W. K. (2002). Traps in the route to models of memory and decision. *Psychonomic Bulletin and Review, 9,* 3–25.

Feldman, J. (2000). Minimization of Boolean complexity in human concept learning. *Nature, 407,* 630–633.

Frauenfelder, U. H., Segui, J., & Dijkstra, T. (1990). Lexical effects in phonemic processing: Facilitatory or inhibitory? *Journal of Experimental Psychology: Human Perception and Performance, 16,* 77–91.

Gelman, A. (1996). Inference and monitoring convergence. In W. R. Gilks, S. Richardson, & D. J. Spiegelhalter (Eds.), *Markov chain Monte Carlo in practice* (pp. 131–140). Boca Raton, FL: Chapman & Hall/CRC.

Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (1996). *Markov chain Monte Carlo in practice.* Boca Raton, FL: Chapman & Hall.

Grünwald. P. (1998). *The minimum description length principle and reasoning under uncertainty* (ILLG Dissertation Series D5 1998–03). Doctoral dissertation, Centrum voor Wiskunde en Informatica, Amsterdam, the Netherlands.

Grünwald, P., Myung, I. J., & Pitt, M. A. (2005). *Advances in minimum description length: Theory and application.* Cambridge, MA: MIT Press.

Johansen, M. A., & Palmeri, T. J. (2002). Are there representational shifts during category learning? *Cognitive Psychology, 45,* 482–553.

Kass, R. E., & Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association, 90,* 773–795.

Kim, W., Navarro, D. J., Pitt, M. A., & Myung, I. J. (2004). An MCMC-based method of comparing connectionist models in cognitive science. *Advances in Neural Information Processing Systems, 16,* 937–944.

Kontkanen, P., Myllymäki, P., Buntine, W., Rissanen, J., & Tirri, H. (2005). An MDL framework for data clustering. In P. Grünwald, I. J. Myung, & M. A. Pitt (Eds.), *Advances in minimum description length: Theory and applications.* Cambridge, MA: MIT Press.

Kruschke, J. K. (1992). ALCOVE: An exemplar-based connectionist model of category learning. *Psychological Review, 99,* 22–44.

Kruschke, J. K. (1993). Three principles for models of category learning. *The Psychology of Learning and Motivation, 29,* 57–90.

Kruschke, J. K., & Erikson, M. A. (1995, November). *Six principles for models of category learning.* Paper presented at the 36th annual meeting of the Psychonomic Society, Los Angeles.

Lee, M. D., & Navarro, D. J. (2002). Extending the ALCOVE model of category learning to featural stimulus domains. *Psychonomic Bulletin and Review, 9,* 43–58.

Luce, R. D. (1963). Detection and recognition. In R. D. Luce, R. R. Bush, & E. Galanter (Eds.), *Handbook of mathematical psychology* (Vol. 1, pp. 103–190). New York: Wiley.

Marslen-Wilson, W., & Warren, P. (1994). Levels of perceptual representation and process in lexical access: Words, phonemes, and features. *Psychological Review, 101,* 653–675.

McClelland, J. L., & Elman, J. L. (1986). The TRACE model of speech perception. *Cognitive Psychology, 18,* 1–86.

McCloskey, M. (1991). Networks and theories: The place of connectionism in cognitive science. *Psychological Science, 2,* 387–395.

McQueen, J., Norris, D., & Cutler, A. (1999). Lexical influence in phonetic decision making: Evidence from subcategorical mismatches. *Journal of Experimental Psychology: Human Perception and Performance, 25,* 1363–1389.

Myung, I. J. (2000). The importance of complexity in model selection. *Journal of Mathematical Psychology, 44,* 190–204.

Myung, I. J., Balasubramanian, V., & Pitt, M. A. (2000). Counting probability distributions: Differential geometry and model selection. *Proceedings of the National Academy of Sciences, USA, 97,* 11170–11175.

Myung, I. J., & Pitt, M. A. (1997). Applying Occam's razor in modeling cognition: A Bayesian approach. *Psychonomic Bulletin and Review, 4,* 79–95.

Navarro, D. J., & Lee, M. D. (2005, September). *An application of minimum description length clustering to partitioning learning curves.* Paper presented at the 2005 IEEE International Symposium on Information Theory, Adelaide, Australia.

Navarro, D. J., Pitt, M. A., & Myung, I. J. (2004). Assessing the distinguishability of models and the informativeness of data. *Cognitive Psychology, 49,* 47–84.

Norris, D., McQueen, J. M., & Cutler, A. (2000). Merging information in speech recognition: Feedback is never necessary. *Behavioral and Brain Sciences, 23,* 299–370.

Nosofsky, R. M. (1986). Attention, similarity and the identification–categorization relationship. *Journal of Experimental Psychology: General, 115,* 39–57.

Nosofsky, R. M., Gluck, M. A., Palmeri, T. J., McKinley, S. C., & Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepard, Hovland, and Jenkins. *Memory and Cognition, 22,* 352–369.

Oden, G. C., & Massaro, D. W. (1978). Integration of featural information in speech perception. *Psychological Review, 85,* 172–191.

Pitt, M. A., Myung, I. J., & Zhang, S. (2002). Toward a method of selecting among computational models of cognition. *Psychological Review, 109,* 472–491.

Platt, J. R. (1964). Strong Inference, *Science, 146,* 347–353.

Rissanen, J. (1996). Fisher information and stochastic complexity. *IEEE Transactions on Information Theory, 42,* 40–47.

Rissanen, J. (2001). Strong optimality of the normalized ML models as universal codes and information in data. *IEEE Transactions on Information Theory, 47,* 1712–1717.

Robert, C. P., & Casella, G. (1999). *Monte Carlo statistical methods.* New York: Springer.

Roberts, S., & Pashler, H. (2000). How persuasive is a good fit? A comment on theory testing. *Psychological Review, 107,* 358–367.

Shepard, R. N. (1987). Toward a universal law of generalization for psychological science. *Science, 237,* 1317–1323.

Shepard, R. N., Hovland, C. L., & Jenkins, H. M. (1961). Learning and memorization of classification. *Psychological Monographs 75*(13, Whole No. 517).

Shiffrin, R., & Nobel, P. (1998). The art of model development and testing. *Behavior Research Methods, Instruments and Computers, 29,* 6–14.

Shiffrin, R., & Steyvers, M. (1997). A model for recognition memory: REM—Retrieving effectively from memory. *Psychonomic Bulletin and Review, 4,* 145–166.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions [with discussion]. *Journal of Royal Statistical Society, Series B, 36,* 111–147.

Wagenmakers, E.-J., Ratcliff, R., Gomez, P., & Iverson, G. J. (2004). Assessing model mimicry using the parametric bootstrap. *Journal of Mathematical Psychology, 48,* 28–50.

(*Appendixes follow*)

Appendix A

The PSP Algorithm in Greater Detail

PSP is a search problem that attempts to answer the question, How does one determine the range of data patterns a model can generate? The problem is addressed by first adopting a particular definition of a data pattern. In doing so, one induces an unknown but finite partition on the parameter space of a model, with each region corresponding to a data pattern. The goal is to design a search procedure that visits each of these unknown regions at least once. Model analyses follow from what is learned from the search.

Difficulties Inherent in a Sampling-Based Approach

An intuitive approach for finding regions is to sample the parameter space enough times to ensure that each region is sampled at least once. In this approach, one is effectively randomly sampling parameter values from some distribution and then checking to see whether these produce a new region. In a naive, simple Monte Carlo (SMC) approach, this would be a uniform distribution over the whole parameter space. As the number of samples drawn increases, this procedure will uncover all regions of nonzero volume with probability 1. However, as the dimensionality of the model increases, SMC becomes inefficient quite rapidly. An unreasonably large number of samples is required for accurate estimation. This problem is commonly referred to as the *curse of dimensionality*, and it forces us to turn to more efficient search methods. In essence, this amounts to finding a better probability distribution from which to sample.

One of the major inefficiencies in SMC search is that it takes no account of the fact that some regions are much larger than others. With the goal of PSP being to visit each search region at least once, it is inefficient to spend most of one's time in a few big regions, which is what would happen with a uniform distribution. Thus, an ideal sampling distribution would be one that assigns equal probability to every region in the partition. In short, a natural way to improve on SMC would be to rescale the parameter space so that all regions are treated as being the same size. Our PSP algorithm uses a Markov chain Monte Carlo (MCMC) procedure to perform this kind of rescaling.

PSP Sampling by MCMC: The Basic Idea

MCMC (Gilks et al, 1996) is a powerful method for sampling a complicated target distribution, denoted $f(\theta)$. The assumption is that we know how to calculate the value of $f(\theta)$, but have no simple method of sampling from it. The Metropolis–Hastings algorithm (see Chib & Greenberg, 1995) provides such a method. One begins by specifying a *jumping distribution* (described more fully below), denoted $q(\theta_k|\theta_{k-1})$, which is used to sample candidate points by "jumping" around in the target distribution. With probability $a = \min\{[f(\theta_k)q(\theta_{k-1}|\theta_k)] / [f(\theta_{k-1})q(\theta_k|\theta_{k-1})], 1\}$, the next sampled point is θ_k. However, with probability $1 - a$, we stay in the same spot (i.e., the next point is θ_{k-1}). The ergodicity of the algorithm guarantees that the frequency distribution of the sampled points converges to the target distribution.

Recall that our goal was to specify a target distribution that is uniform over the *regions* in parameter space. Obviously it is impossible to specify this distribution directly, given that the partition itself is unknown. However, we can specify a series of approximations to this ideal distribution that will eventually converge to it, under certain assumptions. At any given point in the search, the PSP algorithm will have discovered some number of regions. To ensure that each of the discovered regions is treated equally, we run a separate Markov chain, using the Metropolis–Hastings algorithm for each of these regions, each with a different target distribution that is uniform on that region (i.e., $f(\theta) = c_0$ for some positive constant c_0 if θ belongs to the region, and $f(\theta) = 0$ otherwise). A uniform target distribution is used because we want to generate a series of accepted points that are evenly sampled over the region. On any given sampling trial, we randomly pick one of these chains from which to draw the next sample (or systematically sweep through all of them in order to be more efficient), thereby ensuring that samples are drawn from each of the discovered regions equally often irrespective of size. After enough trials, this mixture of samples will approximate the target distribution over the regions.

"Growing" the Partition: How PSP Finds Regions

The procedure outlined above has the rather awkward property of converging to the ideal target distribution (uniform on each region) so long as the algorithm has already discovered all the regions, which rather defeats the point of the search. In order to address this problem, we need to ensure that the algorithm looks outside of the discovered regions often enough to ensure that it finds new regions. Our approach to discovering new regions is based on the assumptions that new regions will be found along the edges of old regions ("contiguity"; see below). It also helps if adjacent regions are correlated in size. A natural way to find regions that satisfy this assumption is to adapt the jumping distribution to ensure that the Markov chains jump outside of these regions at an ideal rate. To understand why this approach succeeds, recall that the target distribution for a given MCMC routine is uniform on a particular region so that points that lie outside the region will always be rejected. If the parameter space meets the aforementioned conditions, all regions will be found.

Another aspect of our approach is that whenever a new region is encountered, the PSP algorithm concentrates solely on that region until it has "caught up" to the pre-existing regions in terms of the number of samples drawn from it. (If multiple new regions are found, they are queued until they all catch up.) The reason for this is that the discovery of a new region is essentially a source of information about a potentially informative area of the parameter space that has not yet been fully explored. Accordingly, as we wish to treat all regions equally, the algorithm concentrates on a new region until it is on an equal footing with all the regions that were previously discovered. By deliberately ensuring that the various chains frequently jump outside their borders, the PSP algorithm is constantly "looking" for new regions. In sum, the algorithm aggressively searches for new regions in the vicinity of a newly discovered region at a level of granularity that is scaled to the size of the current region.

The Jumping Distribution

As the previous discussion should make clear, the choice of a jumping distribution is critical to the success of the PSP algorithm, even though it is chosen independently of the target distribution and serves only as the source from which to generate candidate points. We have used a uniform distribution over a hypersphere centered on the current location in the parameter space. The reason for using a hypersphere is that it is widely used in cases where the sampling region is constrained and no prior information is available about the shape of the region; the hyperspherical shape of a jumping distribution is least likely to favor a particular shape of the sampling domain a priori.

In PSP, the size of the jumping distribution (i.e., the radius of the hypersphere) must adapt to each region. If it is too small, almost all candidate points will be accepted, but every jump will be so small that it will take too many jumps for an exhaustive search of a new region. Also, rejected points, by which new regions outside the current region are being discovered, will rarely be generated. In contrast, if the size of the jumping distribution is too large, candidate points will be rejected too often, and the granularity of the jumps will not be small enough to define the edges of a

region, which requires a properly sized jumping distribution to succeed. Unfortunately, unless one is dealing with a normal distribution, no theory exists that defines the optimal jumping distribution.

With the PSP algorithm, we have found it best to adapt the size of the jumping distribution so that the Markov chain on average accepts 20% to 25% of the candidate points. In other words, each chain spends 20–25% of its time moving around inside the region and the rest of its time searching for new regions just outside the borders. This range was identified heuristically by testing the algorithm on real and toy models many times with jumping distributions that varied widely in size. Two performance measures were used in these tests: search speed and the variance of the region's volume estimates. The goal was to find an acceptance rate for the jumping distribution that maximized search speed and minimized the variance of volume estimates.

Formal Description of the PSP Algorithm

Having presented the main concepts underlying PSP, we are now in a position to define the search algorithm formally.

Initialize. Given θ_1 and Pattern 1, set $m = i = 1$.

Step 1. Establish $q_m(\cdot|\cdot)$ by adapting the size of its hyperspherical domain. Go to Step 2.

Step 2. Set $i = \text{mod}(i, m) + 1$. Go to Step 3.

Step 3. Sample θ_y from $q_i(\cdot|\theta_i)$. If θ_y generates a new valid pattern, set $m = m + 1$, set $\theta_m = \theta_y$, record the new pattern as pattern m, and then go to Step 1. If θ_y generates pattern i, set $\theta_i = \theta_y$ and go to Step 2. Otherwise, go to Step 2.

In the algorithm, $q_i(\cdot|\theta_i)$ denotes the jumping distribution of the region corresponding to pattern i, centered at θ_i. The subscript $1 \leq i \leq m$ indexes the region from which we are currently sampling, and m represents the number of regions found thus far. The algorithm terminates when a preset number of search trials (i.e., size of an MCMC chain) is obtained for each of the discovered regions. Several analytic methods are available to help determine how many search trials must be used for the series to converge to the target distribution and to determine the number of initial draws, called "burn-ins," to discard from the analysis (Cowles & Carlin, 1996; Gelman, 1996).

Note that the initial conditions (Step 0) require an initial parameter set (i.e., point in parameter space) in order to start the algorithm. These can be provided in one of two ways. First, we could run a preliminary SMC search to find a parameter set that yields a valid pattern. Alternatively, if the modeler has good intuitions about which parameter sets generate valid data patterns, these can be used as the initial starting values.

Assumptions When Using PSP

For the PSP algorithm to succeed in finding all data patterns, some assumptions must be met. They are the following:

1. *Continuity*. A data pattern occupies a single continuous region in the parameter space, such that any two points that produce the pattern can be joined by a path that passes only through points that produce that same pattern.
2. *Contiguity*. Regions are contiguous with one another in the sense that any two regions can be joined by a path that passes only through points that produce a valid data pattern.
3. *Stationarity*. Model behavior is stationary in the sense that a given parameter set always generates a single, fixed data pattern. This means that the boundaries of the regions are fixed, not varying every time the model generates a data pattern.

Among these three conditions, the first two (continuity and contiguity) are key conditions necessary for the algorithm to perform satisfactorily. That is, the assumption that the region is continuous is implicit in Step 1 and the contiguity condition is necessary for Step 3. In the following passage, we discuss how potential violations of all three can be detected and handled if suspected.

The assumptions of continuity and contiguity are reasonable if the computational model being analyzed is composed of mathematical functions that are known to exhibit these properties (i.e., the functions are continuous and smooth). If these assumptions are suspected of being violated (e.g., there are breaks in the range of a parameter), the modeler can check the consistency of the PSP solution through multiple runs of the algorithm. Estimated center or mean points and covariance matrices of discovered regions, which are output by the algorithm, will reveal inconsistencies, such as discrepant volume estimates or the number of data patterns found. Inspection of these values across multiple PSP runs should validate any suspicions and reveal the nature of the inconsistency, if it exists.

For example, if one suspects the existence of noncontiguous regions in the parameter space, creating "islands" of data patterns, then there are at least two remedies one can use to ensure PSP finds them. One is to run the PSP algorithm supplied with different initial starting points. The modeler, who is most familiar with the behavior of the model, should be able to provide parameter values that generate typical patterns as well as less typical patterns that could come from another region in the parameter space. The more parameter values the modeler can provide, the more likely the algorithm will find the noncontiguous regions, if they are present. Another method for finding noncontiguous regions is to use SMC. That is, one would randomly sample the parameter space to find such islands. Large regions are of most importance because the data from them could influence PSP performance. Fortunately, size works to the modeler's advantage in this circumstance because a reasonably long run of SMC should be able to find at least one point in an island. Once this is done, the PSP algorithm can take over from there and find the other elements of the partition that belong to the island.

To test this idea, we ran a simulation test in which islands of data patterns were placed in various regions of the parameter space. The results showed that SMC worked quite well and found all islands in a reasonable amount of time for models of up to seven dimensions. At higher dimensions, the islands become such tiny specks in the model's parameter space that exponentially longer runs are needed to find them, which can quickly become impractical, leading to a drop in the success rate. Work is currently under way to devise a more efficient method for finding islands in higher dimensions.

These same two approaches to discovering noncontiguous regions can be used to identify discontinuous regions that correspond to the same data pattern. Multiple runs with different initial points or a preliminary SMC run can help detect such discontinuous regions, again especially if they occupy considerable individual volumes. Output analyses conducted with estimated center points, volumes, and the covariance matrix of discovered regions will reveal their existence and location through discrepancies across repeated measurements.

The stationarity assumption implies that PSP is not applicable to simulation-based, probabilistic models (e.g., distributed connectionist models, algorithmic models such as REM; Shiffrin & Steyvers, 1997). For this reason, a probabilistic model can be analyzed by PSP only if its simulational component can be replaced with a closed-form probability density (or mass) function, so that data patterns are defined on the space of probability distributions. Implementation of an algorithm such as PSP requires dealing with other issues as well. Many of these are practical in nature but have the potential to influence the results. For example, if neighboring regions are similar in size, there is a good chance of finding them because the probability of sampling from a similarly-sized region is higher than that of a much smaller region. If one suspects that tiny regions are nestled between larger ones, then the best course of action is to increase the sample size so that the likelihood

(*Appendixes continue*)

of landing in these tiny regions isgreater. The use of multiple runs, as discussed above, will also help in this regard. Other pragmatic issues include ensuring that the parameter ranges are finite and defining appropriate stopping criteria.[A1] The preceding discussion is intended to alert potential users of PSP to the issues that must be considered when applying it. Our sampling-based approach to partitioning the parameter space makes it a powerful tool for model analysis, but like other numerical methods that work in high dimensions, including MCMC and nonlinear optimization, correct answers are not guaranteed and cannot be proved in general settings, except for trivial cases. This is why heuristics must be devised and used during application of the method to combat potential problems. Their purpose is to minimize anomalous results and ensure that an answer is accurate and trustworthy.

[A1] To assist the user in addressing these and related issues when using PSP, the Web site (see Footnote 4) provides the Matlab source code of the current algorithm and future updates, along with a tutorial that covers these topics in detail.

Appendix B

A Test of the Accuracy of Volume Estimation

The ability to analyze the volumes of regions is an attractive feature of PSP. Nevertheless, it must be recognized that it is nontrivial to estimate the volume of an arbitrary multidimensional object with no closed-form shape and location. Volume estimation was performed in two steps, both involving sampling-based methods. In the first, a region's border is approximated with an ellipsoid defined by the estimated mean vector and covariance matrix of the MCMC sample of the region. If we use the volume of this ellipsoid as a first approximation to the volume of the region, the result should be biased toward overestimation because an ellipsoid is an "evenly packed" object in every possible direction, and data patterns are not likely to have similarly simple, compact shapes. Because of this, the amount of bias should be related to the degree to which the region is concave (all or part of it). This bias can affect the volume analysis because the degree of concavity may differ across regions being compared.

The second step of volume estimation adjusts for this bias with the hit-or-miss method of Monte Carlo integration. First, a random sample of size n is drawn from the uniform distribution over the ellipsoid. Next, the ratio of the number of sample points within the region (i.e., the "hits") to the total number of samples, n, is used as a factor of bias correction. This value is then multiplied by the volume of the ellipsoid to obtain the final volume estimate of the region. Our expectation in using this two-step procedure was that any bias of underestimation after adjustment would be negligible compared with the overestimation before adjustment. Nonetheless, it was necessary to assess the severity of the bias in order to determine how it might limit analyses using the volume measure.

We therefore designed a very stringent simulation test to assess the accuracy of volume estimation. A region in a multidimensional space was randomly generated in the following way. On an evenly spaced grid in this space, a cell was selected as a starting point of the region. The region was then expanded (i.e., grew) into one of its orthogonally neighboring cells in a random direction, to form a two-celled region. The region was expanded again into a neighboring cell, adjacent to any part of the region, not just the new cell. This process was repeated until this randomly shaped region, R, filled a preset number or cells. Its volume, V_R, was estimated by PSP with the equation,

$$V_R = \frac{k}{n} V_d (d+2)^{d/2} |\mathbf{S}|^{1/2}$$

where V_d is the volume of a d-dimensional sphere of radius 1, $\mathbf{S}$ is the covariance matrix estimated from the MCMC output, and k is the number of sample points in region R out of all n points in the ellipsoid.

Accuracy was measured by comparing the estimated volumes of two of these arbitrarily shaped regions. The ratio of these two volumes was used as the measure of accuracy, for which the known, true ratio served as the reference point. It is important to understand that it is not appropriate to compare the estimated volume of a region with its "true" volume. The reason is that the scale of the parameter space is arbitrary, so the value of a region's volume is not meaningful by itself. The value is meaningful only in comparison with another region's volume, which is how the measure was used in our three application examples and how it was used in this test.

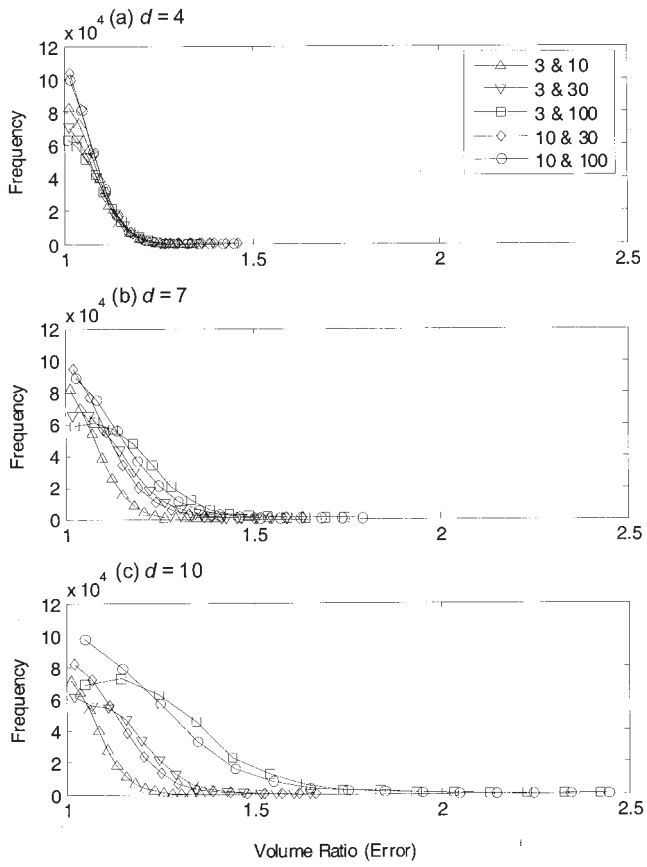


Figure B1. Volume ratio estimates for a given dimensionality, d, across the six cell–number pairs. Values greater than one indicate a bias in overestimating volume.

The simulation was carried out using a two-factorial design that yielded a variety of situations in which PSP's volume estimation method might be used in real applications. The number of dimensions of the parameter space was the first variable. There were three levels (4, 7, 10), which were chosen

to cover a range of models in use in the discipline. The second factor was the number of cells used to form each of the two randomly shaped regions (3, 10, 30, 100). All four levels were completely crossed (e.g., 3 and 10, 3 and 30, 3 and 100) to increase the diversity of possible shapes being compared, which was meant to further increase the realism and difficulty of the test. Note that because of the algorithm's scale-adapting ability, as discussed in Appendix A, it does not pose an obstacle for comparing regions whose volumes constitute a different number of cells (e.g., 3 vs. 30). The two regions were simply scaled to have the same volume so that their ratio should be one in all comparisons.

Because the region-generation method described above was highly unconstrained (requiring only continuity between cells), the 10-, 30-, and 100-cell regions had the potential to have extremely bizarre shapes (e.g., multipronged objects in various directions and dimensions). As a result, conditions 3 and 30, 3 and 100, and 10 and 100 were likely to be the most challenging pairings. The accuracy of volume estimation was pushed to the limits in these cases.

In each of the pairings across dimensionality, the region-generation and volume-estimation and -comparison process was repeated 300,000 times. Regions were sampled 60,000 times (for ellipsoid estimation) and 15,000 times (for the hit-or-miss adjustment) on each trial, which is equivalent to the five multiple runs used in the Merge and TRACE analyses. A subset of the results, which includes the most difficult cases, is shown in Figure B1. Each panel shows the distribution of volume ratio estimates for a given dimensionality d across the different cell–number pairs. The ratio of the larger estimate to the smaller was used, so values greater than one indicate the size of error in volume comparison.

Across most pairing trials, volume estimation was good, with the peaks of the distributions at or near 1.0 and the tails falling off sharply so that few values exceeded 1.5, even when $d = 10$. Errors were greatest for the pairings in which cell numbers differed most (3 and 100, 10 and 100), and this tendency was magnified with dimensionality. Nevertheless, the bias is surprisingly modest, as the 3 and 100 comparison is a full order of magnitude larger than the 3 and 10 pairing, yet the means for the two distributions differ by only 0.163 (1.234 and 1.071, respectively). When this same test was performed without the hit-or-miss volume-adjustment step, biases were much more severe, with the peaks of error distributions far beyond 1.0, especially when the number of dimensions and cells was large.

In sum, the results of this test demonstrate that although volume estimation can be biased, its robustness across a wide range of region shapes indicates that the method is reasonably accurate and trustworthy most of the time. For the models used in the present study, volume differences between regions were exponentially large. To view them all on the same graph, we had to plot them on a log scale (e.g., Figure 8). Thus, it goes without saying that the volume estimation method is more than sufficiently accurate to use rank-ordered estimates in analyzing the volumes of parameter regions, regardless of their shape. In most instances, comparison on an interval scale is appropriate.

Appendix C

TRACE and Merge Parameters

Parameter	TRACE	Merge	Range
Phoneme excitation	√	√	(0,1)
Phoneme to word excitation	√	√	(0,1)
Phoneme to word inhibition		√	(0,1)
Phoneme to target excitation		√	(0,1)
Phoneme decay	√	√	(0,1)
Word to target excitation		√	(0,1)
Word to phoneme excitation	√		(0,1)
Word to word inhibition	√	√	(0,1)
Word decay	√	√	(0,1)
Target to target inhibition		√	(0,1)
Target decay		√	(0,1)
Target momentum		√	(0,1)
Phoneme to phoneme inhibition	√		(0,1)
Cycles per input slice	√	√	Fixed

Note. The word to word inhibition parameter was used in the indirect inhibition example.

Received August 24, 2004
Revision received June 28, 2005
Accepted June 29, 2005 ■