Appendix A: Additional Details about the PSP Algorithm

The PSP algorithm can be summarized in the following three steps:

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

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

Step 2. Set i = 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, $\theta_m = \theta_y$, and 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 above, $q_i(\cdot|\theta_i)$ denotes the jumping distribution of the region corresponding to pattern i, centered at θ_i. The subscript $i\ (1 \le i \le m)$ indexes the region from which we are currently sampling, and m represents the number of region (or number of data patterns) found so far. The algorithm terminates when a preset number of search trials (or size of an MCMC sample) is obtained for *each* of the discovered regions. To compensate for the fact that regions discovered early in the search process are likely to be sampled more than regions discovered later, the algorithm concentrates more on the newly discovered regions in such a way that the total number of trials in the search history will eventually be the same for all regions. An implication of this is that the algorithm aggressively searchers for new regions in the vicinity of a newly discovered region, which makes good sense.

Details of MCMC and its Implementation in PSP

Sampling-based search using MCMC is a key component of the PSP algorithm. One of the simplest forms of Markov chains, as used in the current application, makes it possible to sample from a uniform distribution defined on a region of non-regular, arbitrary shape of high dimensions. As is standard practice in MCMC, the chain is formed with the use of a (jumping) distribution over the parameter space, which moves around (i.e., jumps) in the region by the rule described above. A hyper-sphere was employed as the jumping distribution for PSP. The reason for using a hyper-sphere 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 hyper-spherical shape of a jumping distribution is not likely to favor or disfavor a particular shape of the sampling domain a priori.

The size of the jumping distribution (*i.e.*, the radius of the hyper-sphere) 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 region. Also, rejected points 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.

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 use an adaptive jumping distribution, which on average accepts 20% to 25 % of the sample points as a result of size adaptation. This range was identified heuristically by testing the algorithm on real and toy models many times using jumping distribution that varied widely in size. This was accomplished by using the standard errors of volume estimates as a criterion of Markov chain efficiency.

The initial parameter settings (i.e, point in parameter space) that are necessary to start the algorithm can be provided in one of two ways: 1) by an initial preliminary run using SMC (*i.e.*, search by random walk) or by using values supplied by the user (e.g. those that yield the pattern generated by participants). Besides supplying the initial point, these two methods can, at the same time, help the algorithm to overcome the potential problem of discontinuity in the partition (*i.e.*, disconnected groups of regions), as will be discussed later on.

One of the reasons why PSP is superior to a random search process like SMC is because the PSP search process is "well-mixed." That is, the algorithm works by obtaining a stratified sample across the entire search. Its success is achieved by focusing on each region in equal proportion, irrespective of its size. An equal number of search trials is performed in the neighborhood of each region. As a consequence, closer attention is paid to the small regions. This means that the resulting sampling distribution over the whole parameter space is essentially a mixture distribution that gives higher density to points known to lie near many regions.

A Test of the Accuracy of Volume Estimation

The ability to analyze the volumes of regions is an attractive feature of PSP. Volume estimation, however, is nontrivial. To achieve a high degree of accuracy, a sophisticated numerical--either sampling- or nonsampling-based--method would have to be developed, which would increase computational time substantially. In this initial version of PSP, we decided to postpone such work and opted for a method that was reasonably accurate yet highly efficient. A region’s volume is estimated using an ellipsoid defined by the eigenvectors and eigenvalues of the estimated covariance matrix. We suspected that this ellipsoidal approximation would be biased toward overestimation, because an ellipsoid is an "evenly packed" object in every possible direction. Because no method exists for adjusting the bias toward the volume of an arbitrary multidimensional object with the same variance-covariance measures, it was unclear how severe the bias would be and how it might restrict analyses using this measure.

We designed a very stringent simulation test to address this question and 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 using the equation:

$$V_R = V_d\,(d+2)^{d/2}\,\left|\mathbf{S}\right|^{1/2}$$

where V_d is the volume of a *d*-dimensional sphere with radius 1, and **S** is the covariance matrix estimated from the MCMC output.

The simulation was carried out in a two-factorial design. The number of dimensions of the 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 the randomly-shaped region, which also had three levels (2, 4, and 8). Because the region-generation method, described above, was highly unconstrained (requiring only continuity between cells), the 8-cell conditions had the potential to generate extremely bizarre shapes, especially in seven and ten dimensions (e.g., multi-pronged objects in various directions and dimensions). To a lesser extent, the same could have happened in the 4-cell condition. Volume estimation was pushed to the limits in this test. The data should be interpreted with this in mind. We believe the continuity and smoothness of the functions of most cognitive models will rarely generate such patterns.

In each of the nine conditions, the region-generation and volume-estimation process was repeated 1,000 times. The results are shown in Figure 1A. Each panel shows the distribution of volume estimates for a given dimensionality, d, across the different numbers of cells, n. The true volume of an n-cell region is indicated by a dotted vertical line. Volume estimates are plotted on a log scale given the wide variation in dimensionality.

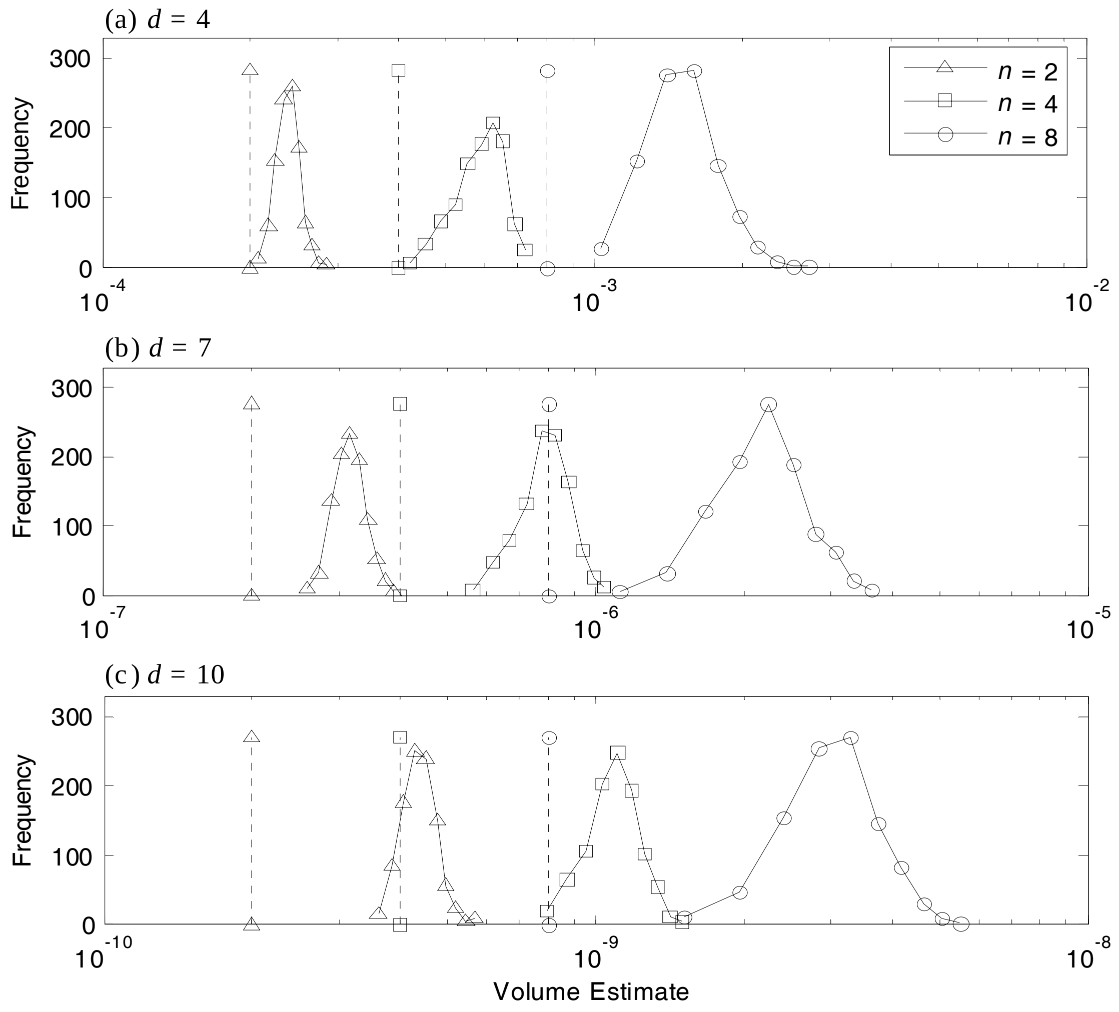

Figure 1A. Note: *d* is the number of dimensions of the space and *n* is the number of cells used to form a randomly shaped region. Volume estimate is on a log scale.

As expected, the ellipsoid approximation overestimates region volume. Inspection of the shapes and locations of the distributions relative to the true volume shows that bias in overestimation increased as dimensionality increased. The distributions are shifted further to the right across panels a-c. Within each graph, it is easy to see that the variability of the estimate increased as *n* increased. This latter result is to be expected given that the variety of possible shapes expanded as *n* increased. Despite the bias of the volume estimation method, what is most remarkable about these results is that volume estimation was precise enough under this stringent test to recover the rank orders of the volumes of the three regions. Their distributions are sufficiently narrow to be easily distinguished from one another. This held true even when d=10. The results of this simulation test demonstrate that the volume estimation method is sufficiently accurate to use rank-ordered estimates in analyzing the volumes of parameter regions.

Conditions that must be met when using PSP

All numerical methods that work in high dimensions require that regularity conditions be satisfied to perform accurately and optimally. This is also true of PSP. The following five conditions must be met: (1) regions are contiguous with one another in the sense that a path between any two regions exists within the partition; (2) the size of the regions changes smoothly in the parameter space, so small regions tend to cluster together; (3) 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; (4) the range of the parameter space to be searched is finite; (5) the data space can be discretized in such a way that the total number of data patterns to be discovered is finite to the extent that they can be found within a reasonable amount of computing time.

It is the responsibility of the modeler to ensure that these conditions are not violated or to take the appropriate action when they are. Although some may be self-evident before applying PSP (3, 4), others can be more difficult to assess (1, 2, 5). In these latter cases, in the absence of proofs to test for violations, the best way to identify and deal with them is through numerical diagnosis using multiple PSP runs. This solution is not only common when solving high-dimensional search problems, which is what PSP is doing, but has proven to be successful (e.g., solving the local minima problem in nonlinear optimization). By running the algorithm multiple times with different starting values, tolerance levels, maximum number of function evaluations, etc, one can ascertain whether a consistent solution is obtained. In the same way, the consistency of a PSP solution can be established by performing multiple runs with different starting points in the parameter space, size of the sample used for an adaptation cycle, and the maximum number of search cycles before termination. Analysis of the resulting data should provide an answer to question. Below we discuss more thoroughly how potential violations of PSP's regularity conditions can be handled.

The assumptions of continuity and smoothness of the partitions in the parameter space are reasonable to make 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 accuracy and consistency of the PSP solution through multiple runs of the algorithm. Estimated center (or mean) points and covariance matrices of discovered regions, which are outputted by the algorithm, will reveal inconsistencies, such as discrepant volume estimates or the number of data patterns found. Inspection of these values across multiple runs should validate any suspicions and reveal the nature of the inconsistency if it exists. In addition, the results of these multiple PSP runs can be combined to create a more accurate picture of model behavior.

For example, if one suspects disconnected regions exist in the model, then at least two methods can be used to ensure PSP finds and records both. One is to supply 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 (but predicted) patterns that could come from another region in the parameter space. The more sets of parameter values the modeler can provide, the more likely the algorithm will find the disconnected regions if they are present. Another way to find disconnected regions is to use SMC. That is, randomly sample the parameter space to find these “islands.” Large regions are of most importance because the data from them could influence the PSP analysis. Fortunately, size works to the modelers advantage in this circumstance because a reasonably long run of SMC should be able to find at least one point in the island, which is all that is needed to discover its existence. The PSP algorithm can take over from there and partition the region. In simulation tests in which islands of data patterns were placed in various regions of the parameter space, we have found SMC to work quite well and in a reasonable amount of time with models that have up to seven dimensions. At higher dimensions, the region becomes such a tiny speck in the model’s parameter space that exponentially longer runs are needed, which can quickly become impractical and the success rate drops.

This same approach can be used to identify noncontiguous 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 analysis using 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). 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. Study of the application of PSP to probabilistic models is currently underway. As for the requirement that the range of parameters must be finite, if some unconstrained parameters are unavoidable (i.e., plausible data patterns could still be generated from their extreme values), it is recommended to reparameterize the model utilizing log, inverse logistic, or other transformation function.

As with some of the preceding assumptions, the last one, that the data space should be discretized in a way that the total number of data patterns to discover must be finite to the extent that they can be found within a reasonable amount of computing time, has no guaranteed solution. To promote the successful application of PSP, it is wise to restrict the model's prediction space by imposing restrictions on possible data patterns when they are reasonable, as was done with ALCOVE. In most cases, such restrictive definition of a data pattern will be justifiable from experimental data or prior work with the model. By precluding nonsensical data patterns, the model analysis can be fair, realistic, and thus informative.

The preceding discussion is intended to alert the potential user to the issues that must be considered when applying PSP. It also serves as demonstration of PSP's current capabilities. The sampling-based approach to partitioning the parameter space makes it a powerful tool for model analysis. There is room for improvement, and enhancements to improve its accuracy and efficiency are being developed. Current and future updates of the algorithm, including Matlab source code and a tutorial, can be found at http://quantrm2.psy.ohio-state.edu/PSP/