STATISTICS

GPT.EXE: A powerful tool for the visualization and analysis of general processing tree models

XIANGEN HU and GLENN A. PHILLIPS
University of Memphis, Memphis, Tennessee

This paper introduces GPT.EXE, a computer program for designing and implementing general processing tree (GPT) models. First, designing and building GPT models using this program is discussed. The second major emphasis is a description of various statistical procedures that can be carried out with GPT.EXE. There is also a brief section on the on-line documentation of this program. Throughout the text, pictures of windows from the program are displayed to help explain the procedures being described by the text.

Multinomial processing tree (MPT) models as introduced by Riefer and Batchelder (1988) are a family of substantive models for cognitive psychology. This family of models has a basic assumption that certain cognitive processes are serial in nature (i.e., memory storage taking place prior to retrieval; Riefer & Batchelder, 1988). These processes are modeled in terms of branching trees in which parameters are the conditional probabilities that link one stage to another (conditional link probabilities). MPT models emphasize the branching tree structure of mental processes (to see what is meant by *branching tree structure* look ahead to Figure 4) and have been used to analyze data from various cognitive experiments (Batchelder & Riefer, 1997). Hu and Batchelder (1994) formalized MPT models mathematically as a subfamily of multinomial models with a specific probability of observing a given category (categorical probability; note that the mathematical aspects of this formula will not be pursued here). This subfamily is the *general processing tree* (GPT) models, which emphasize the mathematical and statistical aspects of multinomial models. The purpose of this paper is to introduce the GPT.EXE computer program and the GPT resource center on the World-Wide Web, which was developed to aid in the design and implementation of GPT models.

GPT.EXE was developed as a user-friendly way of analyzing and visually representing GPT models. This program has two predecessors that are still being used to design and test various MPTs. The first of these programs is SOURCE.EXE (Hu, 1990), designed to analyze multinomial models for source monitoring (Batchelder & Riefer, 1990). This program limits the size of the model to be tested. Although it was user-friendly at the time it

was designed, it is not by today's standards. This program is also limited to conducting parameter estimations, obtaining confidence intervals, and hypothesis testing; its ability to carry out these functions is restricted to models specifically for source monitoring. SOURCE.EXE has grown outdated as new algorithms have begun to be used with MPT models. The second of these previous programs is MBT.EXE, designed to work with multinomial binary tree (MBT) models. This program again limits the size of the model that can be used, and it has a level of user friendliness similar to SOURCE.EXE. MBT.EXE restricts the models it can process to those with only two branches per parameter, limiting possible applications of the program to binary tree models. Others have attempted to design programs to implement MPT models using various approaches.

Three types of computer programs have been designed to implement MPT models. One of these approaches is to write a program for a specific model (SOURCE.EXE above falls into this category). Another example would be a program for the storage-retrieval model (Batchelder & Riefer, 1980) written by Erdfelder (1992). These programs are inherently restricted in that they can only be used with a single model. Another programming method used to implement MPT models is to write a program for a specific model with a specific purpose. An example of this is a program used by Riefer and Batchelder (1991b) for the simulation study of the storage-retrieval model. This approach to developing a program for MPT is inherently restricted not only to a model but to a specific application of that model. The third option is to develop a way to handle the tree models in general, not limiting it to any specific model or purpose. An example of this has been implemented by Dodson (Dodson, Prinzmetal, & Shimamura, 1998), who set up Microsoft Excel solver to estimate parameters for any MPT model. The limit of Dodson's method is that it can only be used to estimate parameters. MBT.EXE (described above) is another ex-

The authors thank Paul Butler for his useful suggestions and GPT.EXE users who have helped in development with suggestions. Correspondence should be addressed to X. Hu, Department of Psychology, University of Memphis, Memphis, TN 38152 (e-mail: xhu@latte.memphis.edu).

ample of the third type of program, as is GPT.EXE introduced here—both of which have fewer restrictions than other programs for dealing with these models.

GPT.EXE is a Windows-based application specifically designed to address many of the previously mentioned problems. This program is much less restricted than its predecessors, the primary limitation being the capacity of the computer on which it is running. Additionally, because it is Windows based, this program is more user-friendly than others previously mentioned (e.g., SOURCE.EXE and MBT.EXE are DOS based). GPT.EXE is designed for use with various tree models for various purposes; it is a flexible program allowing the modeler to design a processing tree model for whatever purpose is needed. GPT.EXE is a powerful tool for the analysis and visual representation of GPT models. Additionally, this program incorporates the most recent research on GPT models, allowing its users to take full advantage of modeling in studying mental processes. An example of recent research that has been incorporated into GPT.EXE is the ability to analyze response time data, which can aid in comparing models in order to determine the one that best fits observed data. This program provides a format for conducting various modeling tasks, including designing, building, and editing GPT models, conducting parameter estimations, hypotheses testing, simulations, and power analyses. GPT.EXE also provides extensive output options that can be used to view all aspects of the various analyses and create an archive of GPT models and the relevant data.

To better describe the terminology and procedures in this paper, we use, as a running example, the model for source monitoring (Hu & Batchelder, 1994; Johnson, Hashtroudi, & Lindsay, 1993). Source monitoring is an extended recognition paradigm with experiments that typically involve two phases. The first of these is the study phase, during which participants are learning material in two different formats (Source A and Source B). The second is the testing phase, in which participants are presented learned items and some new items. The participants must then identify the items as being learned from Source A or Source B or as new items. The source monitoring paradigm has been used extensively in human memory studies. To briefly describe the task processes involved, consider that there are three possible behavioral outcomes for each item presented in the testing phase: Source A, Source B, or new. There are several processes involved in completing this task: for example, memory theoretical processes such as item recognition (similar to a typical yes/no recognition task) and source discrimination (which contributes to the source judgment accuracy). Some processes that are not necessarily memory theoretical in nature may also be involved: for example, when participants fail to retrieve information about the items presented in the testing phase, certain response strategies might be employed (e.g., guessing); this type of process would be a decision process. For the purpose of this paper, it is only important to know the basics of

source monitoring, since we will use it for a running illustration throughout the paper.

BUILDING MODELS

As we have pointed out, one limitation of previous computer programs was and is their lack of "user-friendliness." It is important then that a new program provide a user interface that allows these models to be built intuitively. This is important because it allows the programs' users to concentrate on the meaning of the model structures and their interpretations rather than technical aspects of a computer program. Additionally, increasing the ease of model building will encourage users to try new models with different underlying psychological theories. There are several steps to building a complete tree model for a given set of psychological assumptions. First, one must determine the basic structure of the tree model. The basic tree structure describes the processes involved in the task. Second, several identifiable submodels are obtained by parameter restrictions; each of the submodels corresponds to a distinct psychological hypothesis involved in the task. The third step is inputting experimental data for the purpose of analyzing this data. The fourth and final step is the documentation of the tree model, including the submodels, data from the experiments, and a description of the assumed states in the tree model.

Building the Basic Tree Structure

GPT.EXE allows the user to quickly and easily build new tree models. These models may be built as graphic models or tree view structures (as displayed in the familiar Winfile for Windows 3.11); both are easy to understand and use. The first step in building a tree model is to determine the basic branching tree structure that represents the processes involved in a task. In order to build such a tree structure, the user must first specify:

- *A set of discrete latent states and their relationships:* These states are the assumed psychological states involved in the task (e.g., the item recognition, source discrimination, and response biases in the source monitoring model).
- *A set of observable outcomes:* These are the observed categorical behaviors in the experiment. For example, the source monitoring task has three possible outcomes for each item presented in the testing phase: Source A, Source B, and new.
- *The structure of the latent states and their relationships to the observed outcomes:* This is the key to building the tree structure. In this stage, a functional relationship between the probabilities of the outcomes and the probabilities of the latent states is proposed. For example, a participant's response of Source A to a Source A item is a function of the item detection, source discrimination, and response biases. Specifically, this outcome (response Source A to Source A items) can be observed as results of (1) correct item recognition and correct source discrimination, (2) correct item recognition and unsuccessful source discrimination, but a fortunate source guess, (3) un-

successful item recognition, but fortunate guessing at both the item recognition and source discrimination states.

GPT.EXE provides two ways to create models once the above specifications have been determined: the graphic method and the tree view structure. To design a model in the graphic mode, users can simply add new states in a window where states and outcomes can be added and arranged. There is no difference between latent states or outcomes at this stage. After the specified states and outcomes have been added, modelers then link the latent states in the desired fashion. At this point, users can assign names and parameters to the states. Names can be any short string of letters, whereas parameters are

letter strings that represent the conditional probabilities of the specific states given the previous state(s) in the hierarchy of the tree. For the states that are outcomes, their names are logically the observed behavior category names (see Figure 1).

To design a model in the tree view structure, modelers can simply follow the assumed processes in the task. For example, to create a tree for Source A items first name the root of the tree [Source A]. Because there are two possibilities when an item from Source A is presented—namely, successful recognition or unsuccessful recognition—two children of [Source A] may be [Detected] and [Not Detected]. If an item from [Source A] is detected, then there are two possibilities: [Discriminated] or [Not

Figure 1. The parameters can be seen in the boxes for each node, and the name is in the pop-up box as demonstrated by the "discriminated" box.

Figure 2. It can be seen that “Source A” is at the far left, and the parameters step in to the right as in an outline. Any states immediately one step in from a parameter that also stem from that parameter are called *children* of the parameter.

Discriminated], so two more children are added after [Detected], and so on, until all of the required tree structure elements are created (see Figure 2).

Submodels for Various Hypotheses

A basic tree structure provides an overall structural description of the processes involved in a task. Generally, the basic tree structure contains all possible parameters in a model and the relationships of these parameters. For the source monitoring example, there would be 12 parameters for the basic tree structure: 5 parameters for Source A items, 5 parameters for Source B items, and 2 parameters for new items (see Figure 3).

However, Batchelder and Riefer (1990) only use 7 parameters to start the model, because it is reasonable to assume $b_1 = b_2 = b_3$ (as b), $a_1 = a_2$ (as a), and $g_1 = g_2 = g_3$ (as g), which reduces the basic structure of the model, so it contains only $D1$, $D2$, $d1$, $d2$, a , b , and g . In most cases, the number of parameters in the basic tree structure is larger than the degrees of freedom in the data,

so the basic tree structure may not be used to directly analyze the data. In order to use tree models to analyze data and test hypotheses, one has to either increase the degrees of freedom in the data or reduce the number of independent parameters in the model, thereby producing extra degrees of freedom for model testing. Increasing degrees of freedom in the data can be achieved by adding experimental conditions¹ or reducing independent parameters in a model by applying functional constraints to the parameters. Models that result from applying parameter constraints to the basic tree structure are called submodels of the original model. For example, the 7-parameter model (Model 7) of Batchelder and Riefer (1990) is in fact a submodel of the 12-parameter model, and all other submodels (e.g., 6a, 6b, 6c, 5a, 5b, 5c, and 4, in Batchelder & Riefer, 1990) are submodels of Model 7.

In general, any tree model that is the result of parameter restrictions on the basic tree model is a submodel of the basic tree model. However, different functional constraints applied to parameters produce different effects in

Figure 3. Tree view structure of the source monitoring model.

the resulting models. As is evident in the source monitoring example, the resulting submodels are often structurally identical to the original basic tree structure. However, there are cases in which parameter constraints create submodels that are mathematically but not structurally equivalent to the basic tree model. For example, applying $a = (g)^2$ to Model 7 will produce a submodel that is mathematically equivalent but has a different tree structure (see Figure 4).

It is worth noting that, after parameter constraints have been applied to the basic tree model, the resulting submodel is generally still a GPT model or equivalent to a GPT model (Hu, 1998). It is possible that two models that are structurally unique can be viewed as submodels of a more general basic tree structure (see Figure 4). In some cases, links with probabilities of 0 or 1 can be added into the tree structure to provide a unified tree structure for all the submodels. For example, consider Figure 4 in which it was already pointed out that Model 2 is a submodel of Model 1 mathematically but not structurally. By considering a more general structure, such as Model 3, both Model 1 and Model 2 are seen as submodels of the new model. From this it is evident that, in general, any

submodel will keep the same structure as the basic tree structure.

Setting up submodels is a straightforward process. One can simply duplicate the file of a basic tree structure and change its parameters. In GPT.EXE, it is possible to have several submodels in a single file, allowing any of the submodels to be selected as needed. To illustrate this, we briefly describe the procedure used to set up Model 5c from Model 7 in the source monitoring example.

1. Save the 7-parameter model as Model 7 by using the "Save Model Version" button in the upper left corner of the window containing the basic tree structure.
2. Create submodel Model 5c by either Method 1 or Method 2 below.

Method 1:

1. Replace all d1 and d2 by d, and g by a, which produces a model containing only 5 parameters.
2. Save the current model as Model 5c by clicking the "Save Model Version" button again.

Method 2:

1. Use the pop-up menu in the main window and select "Model/Hypothesis Test."
2. The setup of the hypothesis test allows user to equate parameters.
3. After setting the parameters equal, click "Done," and a submodel will be created with the name "temp."
4. Save "temp" to Model 5c by clicking the "Save Model Version" button.

After all the needed submodels have been built, a hierarchical structure of these submodels is created that reflects the relationships among the models. This can be viewed by right clicking on the combo box displaying the submodels and selecting "View Model Overview" (see Figure 5).

Input Experimental Data

After the basic tree structure and submodels are created, data from experiments can be entered into the data grid. The grid is arranged so that the names of the observed categories are listed in the first column. Each column of the grid then contains a set of frequencies from an experiment. For example, the 9 cells of the 3×3 data table from source monitoring experiments are arranged in each column. The name of each data set is at the top of each column.

Documentation

There are several sources of important information involved in a tree model, especially when several submodels are created from a basic tree structure. It is always useful to document what has been done to the model and what hypotheses correspond to each submodel. There are several states (or nodes) in a basic tree structure, and

Figure 4. Applying $a = (g)^2$ to Model 1 produces Model 2; although Model 2 is a submodel of Model 1, it has a different structure. However, Model 1 and Model 2 are structure-preserving submodels of Model 3, because, on the basis of Model 3, Model 1 can be obtained by applying $g_3 = g$, $g_1 = a$, and $g_2 = 1$; Model 2 can be obtained by applying $g_1 = g_2 = g_3 = g$.

each represents a meaningful process involved in a task. Additionally, there may be several data sets collected from various experiments that can be analyzed using the new tree model and its submodels. Thus, it is useful to keep track of information about the model and any pertinent notes when creating the model. GPT.EXE provides a simple way for users to document important considerations and comments regarding each basic tree structure, submodel, data set, and processing state in the tree structure.

When documenting a submodel, descriptions of the relationship of the submodel to the basic tree structure will make it easier to understand the hypothesis for the submodel. Other important information may be a simple description of the submodel. For example, important information about Model 7 could simply be a description of the model, such as the number of parameters and definitions of those parameters (see Figure 6).

To document the submodel and each data set, the modeler has to be in the data input window and then clicks

“Show Model Memo.” Descriptions about each data set can be anything related to the experiments that produced the data. For instance, one could simply write the time and day the data were collected or an entire method section from a manuscript could be included here (see Figure 7).

There are a variety of options for describing states in a model, and different options can be used depending on how the information is being used. For example, when a state is added, several things can be specified: name of the state (observed category name if it is the end of a branch), parameter of the state (the conditional probability of that state given the prior state), and a description of the state (see Figure 8).

GPT.EXE affords the modeler the ability to include detailed documentation about the modeling processes. This feature is a significant improvement over previous modeling programs that were far more limited in their ability to document models built with them. Additionally, the documenting features of GPT.EXE allow research-

Figure 5. The combo box used to access “View Model Overview” so that the hierarchy of the submodels may be viewed.

ers to archive their models and data analyses so other researchers can understand what considerations were given when the data were analyzed.

STATISTICAL ANALYSIS

After building a model and entering data from experiments, the modeler will want to conduct statistical analyses using the model. GPT.EXE includes several built-in statistical procedures that allow GPT models to be tested—namely, parameter estimation, hypothesis testing, simulations, and power analysis. These statistical analyses are based on the mathematical relationships drawn from the tree structure of the model.

Parameter Estimation

Usually, the parameters in GPT model will not be linearly related to the observed category probabilities. For example, certain observed categories are the result of combinations of branches in the tree model. In this case, the expected category probabilities will generally be the sum of the products of the parameters, which are no longer linear or log-linear; this feature of GPT models permits the straightforward application of the EM algorithm (Dempster, Laird, & Rubin, 1977; adapted to multinomial modeling by Hu & Batchelder, 1994). The EM algorithm of parameter estimation is an iterative process. The basic idea of this process is that an expected (E) branch frequency for a given parameter is estimated

Figure 6. The window used for documentation of submodels.

Figure 7. The window used for documentation of data sets.

from the previous step; this expected branch frequency is then used to maximize (M) the likelihood estimate, which is in turn used to start the next step. GPT.EXE starts the iteration at a randomly selected initial value; however, users can specify an initial value for any parameter they so choose. Parameters in GPT models are conditional probabilities, and parameter estimates are necessarily elements of the (0,1) interval. GPT.EXE provides parameter estimates, and it also provides estimated standard deviations (SDs) and a goodness-of-fit index (i.e., the G^2 statistic). Furthermore, GPT.EXE computes the degrees of freedom of the model by considering the number of trees and the number of free parameters in the model. Certain features of GPT.EXE make parameter estimation very user-friendly. For example, users can estimate parameters for several submodels and several data sets in a "batch mode." Modelers can use the batch mode to estimate parameters for all the data with all the submodels in only one run. Final results can be saved in Rich Text Format (RTF) so they can be edited in Microsoft Word or other Windows-based word processors. GPT.EXE also provides the option of saving descriptions of the submodels and data sets in the output with the final results. This is very helpful because the data sets and submodels will frequently create over 100 possible combinations. When working with so much information, well-documented output is vital at the time of interpretation.

Hypothesis Testing

GPT models are a structurally and mathematically well-defined family of parametric multinomial models. This family of models is closed under certain parameter restrictions (Hu & Batchelder, 1994). This aspect of GPT models provides a mathematical foundation for hypothesis testing. As we have previously mentioned, there are several issues involved in creating submodels for vari-

ous hypotheses. However, once submodels are created, hypothesis testing is rather straightforward. Starting with the assumption that all submodels have the same structural representation as the original model, it follows that the hypotheses for a given basic tree structure consists of a hierarchical structure. This means that all the hypotheses are nested. For these nested hypotheses, we use a log-likelihood ratio framework to construct the tests. Basically, this involves parameter estimations for each of the submodels, which produce goodness-of-fit statistics for each of the submodels. The modeler can then compare the goodness-of-fit statistics for the submodels to determine which model best fits the data. For example, Model 6c and Model 5c are both submodels of Model 7. In Model 6c (a saturated model with 6 parameters and data with 6 degrees of freedom), our null hypothesis (H_0): $d_1 = d_2$ corresponds to Model 5c. In this situation, testing H_0 requires only two parameter estimations, one for Model 6c and one for Model 5c. The difference between the goodness-of-fit indices of these two models for a given data set is used as the chi-square value to determine the significance for the test. Because the two models differ only by one parameter constraint (namely, $d_1 = d_2$), the degree of freedom for the likelihood ratio is 1.

Simulations

An advantage of using GPT models to study cognitive phenomena is the intuitive nature of the branching tree representation of processes involved in a cognitive task. Such representations help researchers interpret findings, providing insight into the underlying processes involved in these tasks. Tree representations also help researchers carry out statistical analyses, such as parameter estimation and hypothesis testing. The advantage from a computational point of view is that the tree representation provides an efficient way of conducting simulations using the

Figure 8. (A) Illustration for editing descriptions of states in tree view. (B) Illustration for editing descriptions of states in the “Property Editing Panel.” (C) Illustration for editing descriptions of states in the “Drawing Panel.”

basic tree structure and its submodels. Assume a basic tree structure and some possible submodels have been created for a task. A simulation study will require only two more steps: first, generate frequency tables by a given model and then use those frequency tables to test the original or perhaps a new model.

Following is a brief description of the stages involved in a simulation given that the basic tree structure and submodels have already been created.

1. Specify values:
 - A. Mean and *SD* of the model parameters
 - B. Number of items involved in each tree
 - C. Number of data sets to be simulated
2. Generate frequency tables: For each trial, the simulation starts at the source of a tree. At each node (including the source), the simulation process can be described in several steps:
 - A. A random vector is obtained from a prespecified distribution (either a uniform or a multivariate beta, depending on user specifications), and a multinomial distribution (binomial, if two branches) is obtained for that node.
 - B. A number is randomly drawn from a uniform distribution.
 - C. The obtained number is then compared with the link probabilities at that node to decide which link to follow. A link probability is prespecified or sampled, the random number from the previous step is compared with the link probabilities extending from the current node, and the appropriate link is followed.
 - D. If the current node is the end of a branch, the trial ends by adding 1 to the frequency of the category represented by the branch. A frequency table is obtained after several trials.
3. Estimate model parameters based on the simulated frequency table: This step is simply parameter estimation of a model given the data generated in the previous step.

These procedures can be implemented in GPT.EXE, so modelers can conduct simulation analyses with GPT models. One example of how GPT.EXE's simulation fea-

ture may be used is to obtain simulated *SDs* for parameters that could in turn be employed to examine the robustness of a model and to simulate power of statistical tests.

Simulate *SD*. When the sample size is sufficiently large, GPT.EXE provides parameter estimates and estimated *SDs* for the maximum likelihood estimate (MLE) of the parameters. GPT.EXE has also been used in situations in which the sample size was not large enough for stable estimation of the *SDs* of the MLEs. For example, it is sometimes the case that the estimated *SD* is so large that boundaries of the estimated confidence intervals for certain parameters may exceed the defined interval of the parameters (0,1). In such cases, the modeler has to conduct simulations to obtain confidence intervals instead of reporting the confidence intervals computed from the estimated covariance matrix.

The simulation procedure previously outlined also works for the simulation of *SDs*. The basic idea is to generate multiple frequency tables from given parameters with a specified sample size. For each of the generated frequency tables, GPT.EXE estimates the parameters. At the end of the simulation, there will be a vector of estimated values for each of the parameters. The *SDs* of these vectors are the estimated *SDs* for the corresponding parameters. Finally, the estimated *SDs* can be used to obtain confidence intervals for the model parameters. As an illustration, we have simulated the confidence intervals for the parameters of Model 5c, from the source monitoring model, with data from Johnson, Kounios, and Reeder (1994). First, the parameters for Model 5c were estimated; then, using the simulation option, a model was simulated with the estimated parameter values. Each table consists of 600 Type A items, 598 Type B items, and 1,198 new items, which were the same values as the observed frequencies in the original experiment. With 1,000 simulated frequency tables, we were able to obtain the *SDs* of the parameters (see Figure 9).

Conducting simulations on GPT models not only provides real *SDs* for the parameters but also provides more information about the parameters. After conducting a simulation, a plot of the obtained parameter values shows certain distributional characteristics that require more information than a mean and *SD* could provide. For example, the distributions of the simulated values for pa-

Parameters	Desired Mean	Desired SD	Distribution	Actual Mean	Actual SD	Simulated Mean	Simulated SD
a	0.2083000	0.0000000	Const	0.2083000	0.0000000	0.2124899	0.0820460
b	0.0200000	0.0000000	Const	0.0200000	0.0000000	0.0201227	0.0040739
D	0.9531500	0.0000000	Const	0.9531500	0.0000000	0.9534461	0.0061116
d1	0.8032100	0.0000000	Const	0.8032100	0.0000000	0.7996771	0.0308863
d2	0.7582400	0.0000000	Const	0.7582400	0.0000000	0.7146545	0.1570574

Figure 9. Output window for simulations of that obtain simulated *SDs*.

parameter D are different from that of d (see Figure 10). Such detailed information about MLE is useful in understanding the model.

Robustness of GPT models. Functionally, GPT.EXE was built for parameter estimations of GPT models. A constant concern regarding the use of any GPT model and the MLEs of the model's parameters is whether the model is robust if the data departs from the standard assumptions of multinomial distribution. To answer this question, one must consider the robustness issue as it relates to GPT models.

Theoretically, GPT models are used to analyze data from multinomial trials. Multinomial trials are series of experimental trials that satisfy the identical independent distributions (iid) conditions—specifically, trials that are independent and identically distributed with the given multinomial distribution. In practice, there are almost always violations of such statistical conditions. For example, iid assumptions assume that the model parameters remain the same for all items across all participants; however, in psychological experiments, the unavoidable individual differences and item differences produce a certain

Figure 10. Graphic depictions of the distributions for simulated D and simulated d that show the differences between these parameters. These representations are based on the total number of items in the sample and the probability associated with the parameters.

degree of violation to the iid conditions. The robustness issue of GPT models then is about the adequacy of the models when data are not observed under iid assumptions. Specifically, the robustness of a model is assessed by examining how well the model accounts for parameters from a given parameter estimation procedure.

In general, there are two ways to evaluate the robustness of a model. The first method is to determine the analytic relationships between the parameter estimates and the violation of iid assumptions. For example, Riefer and Batchelder (1991a) have studied the robustness of the storage and retrieval model (Riefer & Batchelder, 1988) by deriving an explicit relation between the MLE of the parameters and the means and *SDs* of the true parameters. The second method is to simulate data under variable degrees of violation to the iid assumptions and then examine the model to see how well it performs under each of the degrees of violation. This method is also demonstrated in Riefer and Batchelder's (1991a) simulation for the storage and retrieval model.

It is always preferable to derive an explicit relationship between the MLE and the means and *SDs* of the true parameters. However, this is generally difficult for GPT models because it is hard to obtain the explicit functional relationships. GPT.EXE tries to take the second method from above and help researchers by examining the robustness of GPT models through simulation. To illustrate this, we conducted a robustness analysis for Submodel 5c of the source monitoring model. Simulation for robustness of a GPT model is the same as the simulation for obtaining the *SDs* for the parameters. The only difference is that the assumption of the violation of the iid conditions is implemented by assuming probability distributions for each of the parameters. The independence assumption of iid is not violated, but the identical distributions assumption is. To start the simulation, one needs to specify the mean, *SD*, and distribution (uniform or multivariate beta) of the parameters. After this, specify the

number of observations for each of the tree. Finally, specify the number of data sets to be simulated and the critical values for the test (this is used to obtain the rejection rate for the model) (see Figure 11).

Detailed information about the simulation is provided in the output window in table format. For each of the data sets, frequency, generated mean, *SD*, and the corresponding power component in Dirichlet distribution (multivariate beta) are recorded (see Figure 12).

Simulate power for statistical tests. The simulation feature of GPT.EXE can also be used to simulate power for statistical tests. The idea is straightforward: Simply use data sets that are generated by the real model (H1: the statement that a specific alternative model is the real model) to obtain the goodness-of-fit for another model (H0: the statement that a specific model is assumed to be the real model). For example, consider a hypothetical situation for the source monitoring model in which the real data is generated by Model 5c with parameters $D = 0.9$, $a = 0.5$, $b = 0.5$, $d1 = 0.85$, and $d2 = 0.9$, using 30 subjects with 120 items (40 Source A, 40 Source B, and 40 new items) for each. The question is, How much power will we have if we want to reject Model 4 in which $d1$ and $d2$ are assumed to be the same? To start this simulation, specify Model 5c (H1) with the given parameters and number of items in each tree ($40 \times 30 = 1,200$ for each of the three trees). Next, specify Model 4 as the H0. The model that corresponds to H0 is used to estimate parameters for each of the data sets generated from H1. By simulating 500 data sets using Model 5c and applying Model 4 in the goodness-of-fit for each of the data sets, we find that the rejection rate (power) is 51% (for this example). Using this feature of power simulation, models can be evaluated prior to running any experiments. For example, the ability to correctly discriminate between the sources ($d1$ vs. $d2$) is heavily dependent on the correct item detection (D). If a participant's item detection (D) is only 0.8, then the power changes to 40%. Users can also ex-

Figure 11. The desired mean and desired *SD* for each parameter are specified before the simulation. Actual mean and actual *SD* are obtained by finding the best beta distributions that approximate the desired mean and *SD*. Simulated mean and simulated *SD* are simulated results. From the example in this figure, note that Model 5c is rather stable under violation of iid assumptions.

	D	1-D
96	(93, 100), 0.90503,(0.05416), 28.00000	(7, 100), 0.09497,(0.05416), 3.00000
97	(92, 100), 0.90669,(0.05816), 28.00000	(8, 100), 0.09331,(0.05816), 3.00000
98	(93, 100), 0.90337,(0.05447), 28.00000	(7, 100), 0.09663,(0.05447), 3.00000
99	(95, 100), 0.89559,(0.05786), 28.00000	(5, 100), 0.10441,(0.05786), 3.00000
100	(95, 100), 0.91077,(0.04597), 28.00000	(5, 100), 0.08923,(0.04597), 3.00000
Mean	90.59000	9.41000
(sd)	(2.96343)	(2.96343)

Figure 12. The columns for parameter D and (1-D) from the output table. The left column lists the data set number. The numbers in the columns for each parameter are frequency, generated mean, generated *SD*, and the power component in the beta distribution (Dirchelete distribution for nonbinary nodes). For example, the beta distribution for parameter D has power components 28 and 3 for D and (1-D). With this distribution, the simulation generated 95 counts for D and 5 for (1-D). The MLE for D based on this data set is 0.91007 with an estimated *SD* of 0.04597.

plore other aspects of the model and the experiments represented in them. For example, if data are obtained from twice as many participants, then power increases to 75%.

Power Analysis

Issues related to statistical power are very important for any hypothesis tests. Theoretically, power analysis for GPT models is not different from any other power analysis for chi-square tests, because the hypothesis tests for GPT models are log-likelihood ratio tests that are asymptotically distributed chi-squares. The key procedures for power analysis can be outlined in the following steps: (1) Obtain categorical probabilities under H1. (2) Obtain categorical probabilities under H0. (3) Obtain each outcome's effect sizes (Cohen, 1988) as a function of the categorical probabilities under H1 and H0. (4) Compute one of the following four components when the other three are given: sample size, overall effect size, alpha level, and power.

Obtaining categorical probabilities under the assumptions of H1 is straightforward because the categorical probabilities are explicit functions of the model parameters. However, to obtain categorical probabilities under H0 involves parameter estimations. For example, if Model 5c represents H1 and Model 4 represents H0, the parameters (D, a, b, and d) for Model 4 must first be obtained so that its categorical probabilities are closest to the categorical probabilities under Model 5c. For instance, Model 5c with parameter values $D = 0.9$, $a = 0.5$, $b = 0.2$, $d1 = 0.4$, and $d2 = 0.9$ corresponds to Model 4 with $D = 0.9$, $a = 0.277$, $b = 0.2$, and $d = 0.642$ (see Figure 13).

Once we have obtained categorical probabilities under H1 and H0, effect sizes for each of the trees can be obtained. With the obtained effect sizes for each of the trees, the overall effect size can be obtained. Given an alpha level and effect size, we can obtain the sample size necessary for the specific power required (see Figure 14).

The algorithm used for power analysis in GPT.EXE is the same as the algorithm for chi-square power analysis in GPower (Erdfeiler, Faul, & Buchner, 1996).

ON-LINE DOCUMENTATION

GPT.EXE was developed, in part, to provide a more user-friendly format for processing GPT models. In keeping with this goal, documentation and a users manual are provided online. The on-line manual and help can be accessed through "What's New" in the "Help" menu of GPT.EXE's main window or by typing <http://xhuoffice.psyac.memphis.edu/gpt/index.htm> into the address box of your internet browser. The on-line documentation site provides the best way to acquire GPT.EXE, by providing a downloadable installation program. Once the program is downloaded and installed, instructions are provided on the documentation site to help new users get started

Figure 13. Specify parameter values for the model of H1 and then obtain the parameter values under the H0 model.

Select models Power calculation

☐ alpha 0.05
 ☒ Sample Size 372
 Degree of Freedom 1

☐ Power 0.8
 ☐ Total effect size 0.145304

Compute Sample Size

☒ Use computed effect sizes from the model

Code for the power calculation is provided by Dr. Franz Faul and Dr. Erdfelder

Sample Size

☒ Equal sample for all trees
☐ Set sample sizes for each tree based on the effect sizes

Tree	effect size #1
SOURCEA	0.0765558
SOURCEB	0.1368867
New	0.2224750
Total Sample Size	372

Figure 14. With degrees of freedom and any three other components given, the fourth component can be obtained.

using GPT.EXE. Included in this documentation are many snapshots of windows from the program to illustrate what the written instructions describe. By referring to the snapshots, new users can be certain they are following the instructions properly. Descriptions of many uses of GPT.EXE are provided, including building and editing GPT models, parameter estimation, hypothesis testing, simulations, and output and archiving. In addition to how to use GPT.EXE, this internet site provides a great deal of information on MPT and GPT history.

The documentation site includes several types of background information that new and experienced GPT modelers may find interesting and useful. One of the important components of this site is an explanation of some of the mathematics behind GPT. There is a detailed description of the mathematics of modeling and how to conceptually develop models. For those who wish to develop a better understanding of this area, there is an extensive list of studies that provide a background for MPT and GPT modeling and studies that have employed these models. This list of studies includes abstracts and complete references. In addition to the GPT model background information, there are prior programs available for download, such as SOURCE.EXE and MBT.EXE. If users are

not able to obtain the information they need or have further questions, there is a discussion group where questions can be posted, or e-mail can be sent directly to the developer of the program. The support and documentation available online effectively increase the user-friendliness of GPT.EXE, allowing anyone to use the program and develop a working understanding of GPT models.

SUMMARY

MPT models are a family of substantive models for cognitive psychology. GPT models are a subfamily of MPT models that emphasize the mathematical and statistical properties of these models. GPT.EXE is a computer program that allows the user to more easily design and analyze GPT models. This program is designed to be user-friendly and to overcome limitations of previous programs. GPT.EXE eliminates the need for writing programs for specific models and of adapting programs (intentioned for other uses) to this purpose. The program introduced here provides an easy-to-use format for designing and editing GPT models. Once the models have been created, GPT.EXE allows the modeler to analyze the tree in several ways, including parameter estimation,

hypothesis testing, simulation, and power analysis. This program also creates extensive outputs based on the various analyses; furthermore, it allows the modeler to add information to the output for the purpose of creating an archive. The on-line documentation that is provided for GPT.EXE provides step-by-step instructions for using this program. The internet site affords extensive references about MPT and GPT models that allow new users the opportunity to learn about the background of these models and some applications of these models. This site also provides an easy way to request information about GPT.EXE from the developer and from other users of the program. GPT.EXE simplifies the creation and analysis of GPT models, making them easier to implement and analyze, and the on-line documentation provides easy-to-use support for modelers using the program.

REFERENCES

- BATCHELDER, W. H., & RIEFER, D. M. (1980). Separation of storage and retrieval factors in free recall of clusterable-pairs. *Psychological Review*, *87*, 375-397.
- BATCHELDER, W. H., & RIEFER, D. M. (1990). Multinomial processing models of source monitoring. *Psychological Review*, *97*, 548-564.
- BATCHELDER, W. H., & RIEFER, D. M. (1997). *Theoretical and empirical review of multinomial processing tree models* (Tech. Rep. MPS 97-17). Irvine: University of California.
- COHEN, J. (1988). *Statistical power analysis for the behavioral sciences*. Hillsdale, NJ: Erlbaum.
- DEMPSTER, A. P., LAIRD, N. M., & RUBIN, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, *39*, 1-38.
- DODSON, C. S., PRINZMETAL, W., & SHIMAMURA, A. P. (1998). Using Excel to estimate parameters from observed data: An example from source memory data. *Behavior Research Methods, Instruments, & Computers*, *30*, 517-526.
- ERDFELDER, E. (1992). *Parameter estimation and goodness-of-fit tests for Batchelder and Riefer's storage-retrieval model: A manual for users of the SRM program* (Tech. Rep. No. 2). Universität Bonn, Psychologischen Institut.
- ERDFELDER, E., FAUL, F., & BUCHNER, A. (1996). GPOWER: A general power analysis program. *Behavior Research Methods, Instruments, & Computers*, *28*, 1-11.
- HU, X. (1990). *Source: Computer program for source monitoring experiments*. Memphis, TN: University of Memphis, Department of Psychology. (Available upon request)
- HU, X. (1998). *General processing tree (GPT) models with non-binary parameters*. Manuscript submitted for publication.
- HU, X., & BATCHELDER, W. H. (1994). The statistical analysis of general processing tree models use the EM algorithm. *Psychometrika*, *59*, 21-47.
- JOHNSON, M. K., HASHTROUDI, S., & LINDSAY, D. S. (1993). Source monitoring. *Psychological Bulletin*, *114*, 3-28.
- JOHNSON, M. K., KOUNIOS, J., & REEDER, J. A. (1994). Time-course studies of reality monitoring and recognition. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, *20*, 1409-1419.
- RIEFER, D. M., & BATCHELDER, W. H. (1988). Multinomial modeling and the measurement of cognitive processes. *Psychological Review*, *95*, 318-339.
- RIEFER, D. M., & BATCHELDER, W. H. (1991a). Age differences in storage and retrieval: A multinomial modeling analysis. *Bulletin of the Psychonomic Society*, *29*, 415-418.
- RIEFER, D. M., & BATCHELDER, W. H. (1991b). Statistical inference for multinomial processing tree models. In J. P. Doignon & J. C. Falmagne (Eds.), *Mathematical psychology: Current developments* (pp. 313-315). Berlin: Springer-Verlag.

NOTE

1. In some cases, by adding more experimental conditions, the number of degrees of freedom gained in the data exceeds the number of free parameters gained in the model.

(Manuscript received October 15, 1998;
revision accepted for publication January 27, 1999.)