Behavior Research Methods
2010, 42 (1), 42-54
doi:10.3758/BRM.42.1.42

multiTree: A computer program for the analysis of multinomial processing tree models

Morten Moshagen
University of Mannheim, Mannheim, Germany
and University of Düsseldorf, Düsseldorf, Germany

Multinomial processing tree (MPT) models are a family of stochastic models for psychology and related sciences that can be used to model observed categorical frequencies as a function of a sequence of latent states. For the analysis of such models, the present article presents a platform-independent computer program called multiTree, which simplifies the creation and the analysis of MPT models. This makes them more convenient to implement and analyze. Also, multiTree offers advanced modeling features. It provides estimates of the parameters and their variability, goodness-of-fit statistics, hypothesis testing, checks for identifiability, parametric and nonparametric bootstrapping, and power analyses. In this article, the algorithms underlying multiTree are given, and a user guide is provided. The multiTree program can be downloaded from http://psycho3.uni-mannheim.de/multitree.

Multinomial processing tree (MPT) models are a family of substantively motivated stochastic models for the analysis of categorical data. The main idea of this modeling approach is to explain behavioral data by a function of a sequence of latent states that can be interpreted as psychological processes. The statistical properties of MPT models are well understood (Hu & Batchelder, 1994; Riefer & Batchelder, 1988), and there is a substantial body of research using these models (for reviews, see Batchelder & Riefer, 1999; Erdfelder et al., 2009).

To illustrate MPT models and the computer program presented in this article, a model of source monitoring (Batchelder & Riefer, 1990) is used as a running example. The source monitoring paradigm usually inducts two stages. In the learning phase, memory items are presented in one of two (or more) formats—for instance, nouns printed in either red (Source A) or blue (Source B). In the test phase, a recognition memory test is given, where participants have to identify the words from the learning phase and new words as being learned from Source A, Source B, or as new items. Thus, for each item one of three discrete outcomes (A, B, new) is observed. Completing this task involves several processes—for example, item recognition, source discrimination, and various response biases. Figure 1 shows how such a source monitoring experiment with two sources may be represented as a processing tree structure. The model assumes that participants' responses to Source A, Source B, and new items are a function of the alleged underlying processes. Item detection of Source A and B items is represented by the parameters D_1 and D_2, respectively. Depending on whether they correctly recognize an item as being old, participants may be able to correctly identify the source of that item. The parameters d_1 and d_2 are defined as probabilities of correctly discriminating the source of detected Source A and Source B items, respectively. Finally, if unable to retrieve sufficient information to correctly identify an item, participants may engage in guessing processes, which are represented by the parameters a (guessing that a detected but nondiscriminated item belongs to Source A), b (bias for responding "old" to a nondetected item), and g (guessing that a nondetected item belongs to Source A).

Several computer programs have been developed in the past decade that can be used to estimate MPT models—namely, AppleTree (Rothkegel, 1999), GPT (Hu & Phillips, 1999) and HMMTree (Stahl & Klauer, 2007). Moreover, the parameters of MPT models can be estimated using the SOLVER library from Microsoft Excel with macros provided by Dodson, Prinzmetal, and Shimamura (1998). A comparison of features across these programs is shown in Table 1. Three main motivations led to the development of a new program called multiTree. First, with the advent of the Mac operating system OS X Leopard, AppleTree is no longer supported, due to the lack of the classic environment. Since there are no programs available for Unix systems, there is a need for a platform-independent software for the analysis of MPT models. Second, except for GPT, none of the existing programs offers an easy-to-use option to conduct a priori and post hoc power analyses, which is of vital importance for the rational design of experiments. Finally, most existing programs lack advanced modeling features such as bootstrap procedures and identifiability checks.

M. Moshagen, moshagen@uni-mannheim.de

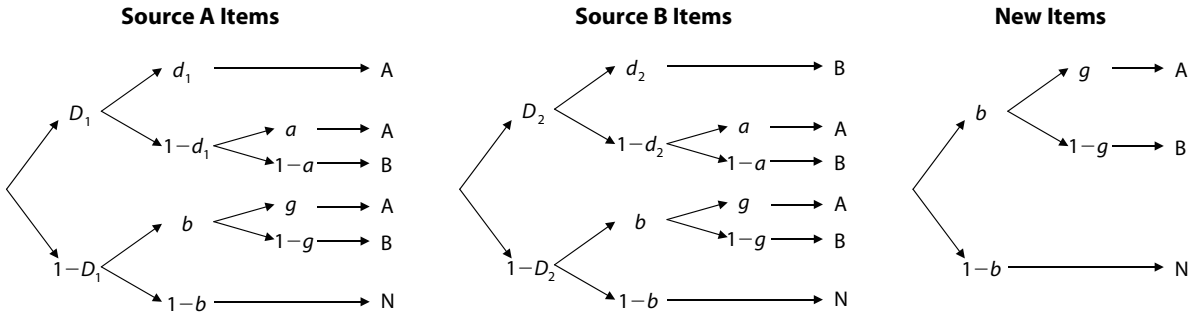

Figure 1. A multinomial processing tree model of source monitoring.

MULTINOMIAL PROCESSING TREE MODELS

MPT models attempt to estimate latent parameters from observed category frequency counts. An MPT model comprises a set of branches that lead to observable response categories. Each branch consists of a series of conditional link probabilities of one stage to another. The probability of a branch i leading to category j equals the product of the corresponding link probabilities,

$$p_{ij}(\Theta) = c_{ij} \prod_{s=1}^{S} \theta_s^{a_{ijs}} \left(1 - \theta_s\right)^{b_{ijs}}, \qquad (1)$$

where $\Theta = (\theta_1, \ldots, \theta_s)$ is a vector of length s of the parameters representing the conditional link probabilities, a_{ijs} and b_{ijs} are count variables that represent the frequency of a parameter θ_s or its complement $1 - \theta_s$ in a branch, and c_{ij} is a nonnegative real representing the product of constants on the links. A particular response category may be reached by more than one branch. Thus, the probability of observing a response in the jth category equals the sum of the branch probabilities leading to that category,

$$p_j(\Theta) = \sum_{i=1}^{I_j} p_{ij}(\Theta), \qquad (2)$$

where I_j is the number of branches leading to category j and

$$\sum_{j=1}^{J} p_j(\Theta) = 1.$$

To illustrate the definition of MPT models, consider the tree for Source A items in the model of source monitoring shown in Figure 1. The Source A tree consists of six branches leading to three response categories ("Source A," "Source B," "Old") and comprises five independent parameters (D_1, d_1, a, b, g). If a participant correctly recognizes a Source A item as being old (D_1), he or she may either correctly identify (d_1) or fail to correctly identify ($1 - d_1$) the source of that item. After failure to identify the source of a detected item, guessing the source can lead either to a correct (a) or to an incorrect ($1 - a$) response. In contrast, if a participant fails to detect a Source A item ($1 - D_1$), he or she guesses this item as being old (b) or new ($1 - b$). If the item is guessed as old following nondetection of that item, the participant engages in another

Table 1
Comparison of Different Programs for MPT Modeling

	AppleTree	GPT	HMMTree	Microsoft Excel†	multiTree
Operating system	MacOS††	Windows	Windows	MacOS, Windows	Linux, MacOS, Windows†††
Parameter estimation	+	+	+	+	+
Hypothesis testing	+	+	+	+	+
Information criteria			AIC, BIC		AIC, BIC, ΔAIC, ΔBIC, *w*AIC, *w*BIC
Variability of parameter estimates	*SE*, CI	*SE*	CI		*SE*, CI
Fisher information	+	+	+		+
Jacobian					+
Power analysis	+††††	+			+
Bootstrapped CI		Parametric		Nonparametric	Parametric and nonparametric
Bootstrapped goodness of fit					+
Latent class models			+		
Analysis of continuous data		+			
Simulation option		+			+
Simulated identifiability					+
Nonbinary trees		+		+	
Automatic reparameterizations					+
Multiple data sets	+	+			+
Graphical model builder	+	+			+

Note—AIC, Akaike information criterion; BIC, Bayesian information criterion; ΔAIC, ΔBIC, AIC and BIC in relation to a saturated H_0 model; *w*AIC, *w*BIC, weighted AIC and BIC; *SE*, standard error; CI, confidence interval. †Using the SOLVER library (not available in Excel 2008 for Mac), with macros provided by Dodson et al. (1998). ††Requires the classic environment (discontinued with Mac OS X Leopard). †††Requires the Java runtime environment. ††††Power analysis is only available for the effect size *w* (Cohen, 1988).

guessing process judging the item as belonging to either Source A (g) or Source B ($1 - g$). From this model, it is easy to calculate the implied branch probabilities. The probabilities of the three branches leading to a "Source A" response are

$$p_{1,A} = D_1 d_1$$
$$p_{2,A} = D_1(1 - d_1)a$$
$$p_{3,A} = (1 - D_1)bg.$$

The probability of observing a particular response is simply the sum of the probabilities of corresponding branches. For example, the probability of observing a "Source A" response is

$$p_A = \sum_{i=1}^{3} p_{i,A} = D_1 d_1 + D_1(1 - d_1)a + (1 - D_1)bg.$$

The model of source monitoring is a so-called *joint MPT model*, because it is based on a joint multinomial distribution of several independent item classes (Source A, Source B, and new items), consisting of different trees for each item class (Riefer & Batchelder, 1988). A special case of joint MPT models occurs in a between-subjects experiment, where it is usually assumed that the same model applies to each experimental condition (with possible restrictions on the parameters across conditions). For example, in an experiment by Saegert, Hamayan, and Ahmar (1975), participants memorized a mixed list of words in different languages (sources) presented either alone (word condition) or embedded in sentences (sentence condition). The model of source monitoring applied to these data thus consists of the same tree structure (as shown in Figure 1) for each condition (Batchelder & Riefer, 1990). For the sake of clarity and simplicity, it is assumed in the remainder of this article that simple MPT models are analyzed; however, the statistical methods presented next also apply to joint multinomial models (with minor changes in notation; see Hu & Batchelder, 1994, for details).

STATISTICAL METHODS AND NUMERICAL ALGORITHMS

A brief presentation of the algorithms underlying multiTree follows. This presentation relies heavily on the more thorough treatments by Efron and Tibshirani (1997), Hu and Batchelder (1994), and Read and Cressie (1988), and interested readers are referred to these authors.

Parameter Estimation

Parameter estimation proceeds by employing the expectation maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977) tailored for binary tree models (Hu & Batchelder, 1994). The algorithm attempts to obtain a set of parameters $\Theta = (\theta_1, \ldots, \theta_S)$ that minimize the distance between the observed frequencies $R = (n_1, \ldots, n_J)$ and the expected frequencies $P(\Theta) = [Np_1(\Theta), \ldots, Np_J(\Theta)]$ (where $N = \Sigma_{j=1}^{J} n_j$), as measured by a member of the power divergence family (PD^λ; see below). The EM algorithm is an iteration of trials consisting of an expectation (E) and a maximization (M) step. In the E step of the EM algorithm, expected frequencies m_{ij} for each branch are computed given the parameter vector Θ from the previous trial:

$$m_{ij}(\Theta) = \frac{n_j\, p_{ij}(\Theta)}{p_j(\Theta)}. \tag{3}$$

The M step calculates revised parameter estimates $\Phi^\lambda = (\phi_1^\lambda, \ldots, \phi_S^\lambda) = M_\lambda(\Theta)$ given the expected frequencies of the E step:

$$\phi_s^\lambda(\Theta) = \frac{\sum_{j=1}^{J}\left[\left(\frac{n_j}{Np_j(\Theta)}\right)^\lambda \sum_{i=1}^{I_j}\left(\frac{n_j\, p_{ij}(\Theta)}{p_j(\Theta)} a_{ijs}\right)\right]}{\sum_{j=1}^{J}\left[\left(\frac{n_j}{Np_j(\Theta)}\right)^\lambda \sum_{i=1}^{I_j}\left(\frac{n_j\, p_{ij}(\Theta)}{p_j(\Theta)}\left(a_{ijs} + b_{ijs}\right)\right)\right]}. \tag{4}$$

In order to ensure that each iteration reduces the distance between observed and expected frequencies when $\lambda \neq 0$, the parameter estimates after each iteration t are corrected by the step-width parameter ε,

$$\Theta^{(t)} = \Theta^{(t-1)} - \varepsilon\left[\Theta^{(t-1)} - M_\lambda\left(\Theta^{(t-1)}\right)\right], \tag{5}$$

where the condition $\varepsilon(1 + \lambda) > 0$ must be satisfied (Hu & Batchelder, 1994). Regardless of the value of λ, ε may also be used to speed up the iteration process, since the algorithm may converge faster for higher values of ε.

Parameter estimation begins with initializing the parameter vector $\Theta = (\theta_1, \ldots, \theta_S)$ with a set of start values. Equations 3, 4, and 5 are then applied repeatedly and the change in the distance between observed and expected frequencies is computed after each iteration: $\mathrm{PD}^\lambda[R, P(\Theta)]^{(t-1)} - \mathrm{PD}^\lambda[R, P(\Theta)]^{(t)}$. The algorithm stops if either the criterion of convergence (e.g., 1.0E−8 = 10^{-8}) is reached or the maximum number of iterations (e.g., 1,000) is exceeded.

Assessment of Model Fit

multiTree offers two ways to assess the fit of a model to the data. The generic goodness-of-fit statistic for MPT models is given by the asymptotically χ^2 distributed power divergence family PD^λ, which is used both in parameter estimation and model evaluation. The fit of a model can also be evaluated by means of information criteria. The latter approach is particularly attractive when comparing models not hierarchically nested.

Power divergence statistic. The goodness of fit of a model is determined by measuring the distance between the model-implied frequencies and the observed category frequencies. MPT models utilize distance measures that can be characterized as a power divergence family (Read & Cressie, 1988). PD^λ defines an asymptotically χ^2 distributed family of distance measures:

$$\mathrm{PD}^\lambda[R, P(\Theta)] = \frac{2}{\lambda(\lambda + 1)} \sum_{j=1}^{J} n_j \left[\left(\frac{n_j}{Np_j(\Theta)}\right)^\lambda - 1\right], \tag{6}$$

where λ is the family parameter. The term *power divergence* describes the fact that the divergence of observed from expected frequencies is measured through a weighted sum of λ-powers. Several well-known statistics can be derived from this general formula by choosing appropriate values for λ. These may include $\lambda = 0$ and $\lambda = -1$ by taking the limit

$$\lim_{\lambda \to (0,-1)} \left[\mathrm{PD}^{\lambda} \right].$$

For example, the log-likelihood ratio statistic G^2 is a special case with $\lambda = 0$:

$$\mathrm{PD}^{\lambda=0}[R, P(\Theta)] = 2 \sum_{j=1}^{J} n_j \log \left(\frac{n_j}{N p_j(\Theta)} \right). \tag{7}$$

Similarly, $\mathrm{PD}^{\lambda=1}$ reduces to the Pearson χ^2 distance measure:

$$\mathrm{PD}^{\lambda=1}[R, P(\Theta)] = 2 \sum_{j=1}^{J} \left(\frac{\left[n_j - N p_j(\Theta) \right]^2}{N p_j(\Theta)} \right). \tag{8}$$

Other well-known statistics are the Cressie–Read statistic with $\lambda = 0.66$, the Freeman–Tukey statistic with $\lambda = -0.5$, and the Neyman-modified χ^2 with $\lambda = -2$. The properties of the different members of the power-divergence family have been extensively studied (García-Pérez, 1994; Read & Cressie, 1988; Riefer & Batchelder, 1991). Generally speaking, λ should be chosen to lie in the interval $-2 < \lambda < 2$ to achieve a good approximation of the χ^2 distribution and to minimize the sensitivity to outliers (i.e., local misfit).

Information criteria. In addition to a goodness-of-fit statistic of the power divergence family, model fit can be evaluated by the Akaike information criterion (AIC; Akaike, 1974) and the Bayesian information criterion (BIC; Schwartz, 1978):

$$\mathrm{AIC} = 2S - 2\ln(L) \tag{9}$$

and

$$\mathrm{BIC} = 2S\ln(N) - 2\ln(L), \tag{10}$$

where S is the number of parameters and L is the maximized value of the likelihood function. These information criteria include a penalty term for the number of parameters in a model and, therefore, favor simpler over more complex models. Since these criteria are rather difficult to interpret in isolation, multiTree also provides AIC and BIC values in relation to a saturated null hypothesis model (Read & Cressie, 1988):

$$\Delta\mathrm{AIC} = \mathrm{PD}^{\lambda} - 2df \tag{11}$$

and

$$\Delta\mathrm{BIC} = \mathrm{PD}^{\lambda} - \ln(N)df. \tag{12}$$

Consequently, evaluating models with the generalized ΔAIC or the ΔBIC is equivalent to comparing the value of the goodness-of-fit statistic PD^{λ} against a linear transformation of its expectation. For ΔAIC and ΔBIC it is sufficient to check the sign: Positive values indicate to reject a model, whereas negative values indicate that the model is tenable.

Furthermore, it is useful to consider AIC and BIC weights (Wagenmakers & Farrell, 2004) to facilitate the interpretation of the difference between information criteria when comparing competing models. If only two models are compared, *w*AIC and *w*BIC are given by

$$w\mathrm{AIC} = \frac{\exp\left[-0.5\left(\mathrm{AIC}_p - \mathrm{AIC}_q \right) \right]}{1 + \exp\left[-0.5\left(\mathrm{AIC}_p - \mathrm{AIC}_q \right) \right]} \tag{13}$$

and

$$w\mathrm{BIC} = \frac{\exp\left[-0.5\left(\mathrm{BIC}_p - \mathrm{BIC}_q \right) \right]}{1 + \exp\left[-0.5\left(\mathrm{BIC}_p - \mathrm{BIC}_q \right) \right]} \tag{14}$$

and can be interpreted in terms of normalized weight ratios (evidence ratios). These evidence ratios represent the probability that model p is the better model, in terms of expected information loss, than model q.

Parameter Variability

The multiTree program offers several ways to obtain information about the variability of the parameter estimates. The asymptotic variance–covariance matrix of the parameters (and hence standard errors) can be computed by inverting the observed Fisher information matrix. In some cases, however, the observed Fisher information matrix may not provide a good approximation to the true variance–covariance matrix. Accordingly, it is sometimes desirable to draw inferences regarding the variability of parameter estimates in other ways, such as by using a bootstrap procedure.

Asymptotic variances. By default, multiTree computes the inverse of the observed Fisher information matrix as an estimate of the variance–covariance matrix of the parameters (Erdfelder, 2000; Hu & Batchelder, 1994). The observed Fisher information matrix $I(\Theta)_{s \times r}$ can be computed as shown in Equations 15–19.

To obtain standard errors of the parameters, $I(\Theta)_{s \times r}$ is inverted and the root is taken from the diagonal terms. Since the maximum likelihood estimates of the parameters can be shown to be asymptotically normally distributed (Hu, 1999), confidence intervals can be computed according to $\theta_s \pm z_{\alpha/2} SE_{\theta_s}$, where $z_{\alpha/2}$ is the tail of the standard normal distribution corresponding to the desired α level.

Bootstrapping. Although obtaining standard errors by the Fisher information matrix may, in general, be considered the method of choice, this approach may fail in certain situations. Since the estimate of the Fisher information matrix is based on observed frequencies, the estimates are heavily influenced by the sample size. If the sample size is too small, the estimate may be a rather poor approximation to the true variance–covariance matrix (Hu, 1999). Moreover, situations may arise in which the observed information matrix contains offending estimates (i.e., negative variance estimates), or may be singular and incapable of being inverted. However, even

$$I(\Theta)_{s\times r} = -\frac{\partial^2 \ln L\left(\Theta;\, <n_j>\right)}{\partial\theta_s\,\partial\theta_r} = -\begin{cases} \dfrac{\partial\alpha_s(\Theta)}{\partial\theta_r}\dfrac{}{\theta_s} - \dfrac{\partial\beta_s(\Theta)}{\partial\theta_r}\dfrac{}{1-\theta_s}, \text{ if } r \neq s \\ \dfrac{\partial\alpha_s(\Theta)}{\partial\theta_r}\dfrac{}{\theta_s} - \dfrac{\partial\beta_s(\Theta)}{\partial\theta_r}\dfrac{}{1-\theta_s} - \left[\dfrac{\alpha_s(\Theta)}{\theta_s^2} + \dfrac{\beta_s(\Theta)}{\left(1-\theta_s\right)^2}\right], \text{ if } r = s \end{cases} \tag{15}$$

where

$$\alpha_s(\Theta) = \sum_{j=1}^{J} n_j \sum_{i=1}^{I_j}\left[a_{ijs}\frac{p_{ij}(\Theta)}{p_j(\Theta)}\right], \tag{16}$$

$$\beta_s(\Theta) = \sum_{j=1}^{J} n_j \sum_{i=1}^{I_j}\left[b_{ijs}\frac{p_{ij}(\Theta)}{p_j(\Theta)}\right], \tag{17}$$

$$\frac{\partial\alpha_s(\Theta)}{\partial\theta_r} = -\sum_{j=1}^{J} n_j \left[\sum_{i=1}^{I_j} a_{ijs}\frac{p_{ij}(\Theta)\sum_{k=1}^{I_j}\left[\dfrac{a_{kjr}}{\theta_r} - \dfrac{b_{kjr}}{\left(1-\theta_r\right)}\right]p_{kj}(\Theta)}{p_j(\Theta)^2} - \sum_{i=1}^{I_j} a_{ijs}\frac{\left[\dfrac{a_{ijr}}{\theta_r} - \dfrac{b_{ijr}}{\left(1-\theta_r\right)}\right]p_{ij}(\Theta)}{p_j(\Theta)}\right], \tag{18}$$

and

$$\frac{\partial\beta_s(\Theta)}{\partial\theta_r} = -\sum_{j=1}^{J} n_j \left[\sum_{i=1}^{I_j} b_{ijs}\frac{p_{ij}(\Theta)\sum_{k=1}^{I_j}\left[\dfrac{a_{kjr}}{\theta_r} - \dfrac{b_{kjr}}{\left(1-\theta_r\right)}\right]p_{kj}(\Theta)}{p_j(\Theta)^2} - \sum_{i=1}^{I_j} b_{ijs}\frac{\left[\dfrac{a_{ijr}}{\theta_r} - \dfrac{b_{ijr}}{\left(1-\theta_r\right)}\right]p_{ij}(\Theta)}{p_j(\Theta)}\right]. \tag{19}$$

if the sample size is large and the model is identified, confidence intervals based on the information matrix may exceed the admissible region of 0–1, for example, when there are parameter estimates near the boundaries of 0 or 1. In such situations, it is desirable to rely on other procedures to obtain information about the variability of the parameter estimates. Fortunately, multiTree provides the option to perform nonparametric and parametric bootstrapping analyses.

The bootstrap is based on the idea that an empirical probability distribution $\hat{F}$ of a random sample drawn from a probability distribution F is a good estimate for aspects of the "true" distribution F (Efron & Tibshirani, 1997). In the bootstrap, random frequencies are repeatedly generated or sampled, and the model is fitted to these bootstrap samples. When this process is repeated a number of times (e.g., 100), the resulting distribution of the estimates for each parameter can be used to draw inferences regarding some aspects of its "true" distribution, such as the variability of a parameter. The standard deviation in the estimates of a parameter is the bootstrap estimate of its standard error, which may then be used to construct bootstrapped confidence intervals in the usual way. Another approach in obtaining confidence intervals via the bootstrap is to use the $\alpha/2$th and $(1 - \alpha/2)$th percentile of the bootstrap distribution as estimates of the desired α confidence interval. Note that the latter approach usually requires more than 100 bootstrap samples to obtain reliable estimates of the confidence intervals (Efron & Tibshirani, 1997).

The nonparametric and parametric bootstraps differ with respect to the resampling mechanism. In the nonparametric bootstrap, random data sets are repeatedly drawn with replacement from given observed frequencies. Since the nonparametric bootstrap does not assume that a model fits the data, it can primarily be used to draw inferences with respect to the variability of parameter estimates. In the parametric bootstrap, however, data sets are generated as implied by a particular MPT model; that is, it assumes that the model under consideration is valid. Therefore, the parametric bootstrap additionally allows for evaluating the exact distribution of the PD^{λ} statistic, for example, if the assumption of an asymptotic χ^2 distribution under the null hypothesis may be violated due to parameters at the boundaries of the parameter space (cf. Klauer & Oberauer, 1995).

The implementation of the parametric bootstrap in multiTree is similar to the one in GPT (Hu & Phillips, 1999). Given an MPT model, the desired number of observations, and means and standard deviations of the parameters, a random parameter vector is obtained from a multivariate beta distribution (see below). Then, beginning

at the root node, a random number drawn from a uniform distribution is compared with the link probabilities at that node to decide which link to follow. When reaching the end of a branch, the frequency count of the corresponding category is incremented and the procedure starts again at the root node until the frequency table is completed. Finally, the model parameters are estimated given the generated frequencies.

Moreover, in a manner similar to GPT, multiTree extends the parametric bootstrap to a more general simulation option by constructing a beta distribution for each parameter with desired mean and standard deviation. The beta distribution with parameters α and β (with $\alpha, \beta > 0$) is given by

$$f(x; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha) + \Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \qquad (20)$$

where Γ is the gamma function and $0 \leq x \leq 1$. Given the mean and standard deviation, multiTree tries to obtain α and β on the basis of

$$M = \frac{\alpha}{(\alpha + \beta)} \qquad (21)$$

and

$$VAR = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}. \qquad (22)$$

From Equations 21 and 22 it is evident that it is not possible to construct a beta distribution for arbitrary means and standard deviations, because α and β must be greater than zero. If the standard deviations of the parameters differ, this analysis can be used to evaluate the robustness of a model against violations of the identical distribution (iid) assumption (cf. Hu & Phillips, 1999). For example, MPT models assume that the parameters are identically distributed for all items across all participants. However, interindividual differences and item differences may result in a violation of the iid conditions. In a robustness study, the performance of an MTP model can be evaluated under various degrees of violations of the identical distribution assumption by setting different probability distributions for each parameter.

Identifiability

A model is globally identified if there is a one-to-one relationship between an arbitrary location in the entire parameter space and certain category probabilities. Identifiability in the neighborhood of the true parameter values (local identifiability) is required for the existence of unique parameter estimates. In contrast, if a model is not at least locally identified, multiple parameter sets may relate equally well to some category probabilities. In such a scenario, applying the EM algorithm repeatedly to identical category frequencies will not yield the same parameter estimates.

Whereas it is often difficult to demonstrate analytically that a particular model is identified, there are simpler ways to show that a model is not locally identified: (1) Count free parameters and independent categories; (2) determine the rank of the Jacobian matrix (Bamber & van Santen, 1985, 2000); and (3) perform an identifiability simulation study (Rouder & Batchelder, 1998).

Generally, a model cannot be identifiable if there are more free parameters S than independent categories J—that is, if $S > \Sigma_{k=1}^{K}(J_k - 1)$, where K is the number of trees. However, $S \leq \Sigma_{k=1}^{K}(J_k - 1)$ is a necessary but not sufficient condition for identifiability. Therefore, it is often useful to compute the rank of the Jacobian matrix. The Jacobian is the matrix of the first partial derivatives of the model equations with respect to the parameters evaluated at a particular location in the parameter space $\Theta_o = (\theta_1^o, \ldots, \theta_S^o)$ and a vector of category probabilities $f(\Theta_o) = [p_1(\Theta_o), \ldots, p_J(\Theta_o)]$. The Jacobian can be computed by Equation 23 below, where a_{ijs}, b_{ijs}, c_{ij}, and I_j are defined in Equations 1 and 2.

If the maximum rank of the Jacobian is smaller than the number of free parameters, the model is neither locally nor, by implication, globally identified (Bamber & van Santen, 1985, 2000). Conversely, evidence for local identifiability can be obtained by comparing the rank of the Jacobian at various points in the parameter space with the number of free parameters.[1] It is important to note, however, that a full rank, even in the entire parameter space, does not imply global identification of the model (P. L. Smith, 1998). Consequently, if the rank of the Jacobian is equal to or greater than the number of free parameters, one should use a procedure called *simulated identifiability* (Rouder & Batchelder, 1998). In such an analysis, model-implied category probabilities are generated from a random parameter location within the admissible parameter space. These expected category probabilities are then used to reestimate the parameters on the basis of random starting values. This procedure is repeated several times (e.g., 1,000). If the model is globally identified, the estimated parameters should match the values of the parameters initially used to generate the expected category probabilities (up to a tolerance accounting for the fact that the convergence criterion must be greater than zero). Although

$$J\left(\Theta_o, f\right)_{j\times s} = \begin{cases} \sum_{i=1}^{I_j} c_{ij}\left(a_{ijs}\prod_{l=1}^{S}\left[\left(1-\theta_l^o\right)^{b_{ijl}}\theta_l^{o\,a_{ijl}}\right] + b_{ijs}\prod_{l=1}^{S}\left[\left(1-\theta_l^o\right)^{b_{ijl}}\theta_l^{o\,a_{ijl}}\right]\right) & \text{if } l \neq s \\ \sum_{i=1}^{I_j} c_{ij}\left(a_{ijs}\prod_{l=1}^{S}\left[\left(1-\theta_l^o\right)^{b_{ijl}}\theta_l^{o\,a_{ijl}-1}\right] + b_{ijs}\prod_{l=1}^{S}\left[\left(1-\theta_l^o\right)^{b_{ijl}-1}\theta_l^{o\,a_{ijl}}\right]\right) & \text{if } l = s \end{cases} \qquad (23)$$

this procedure cannot be considered as an equivalent to a mathematical proof of identifiability, it nevertheless makes identifiability of a model plausible.

When an analytically identified MPT model is fitted to observed data, it may turn out that the model is empirically underidentified for particular data sets. Empirical identification can be technically defined as obtaining a nonsingular information matrix at the estimated parameter vector (Kenny, 1979). A measure of how close a matrix is to singularity is the condition number, defined as the square root of the ratio of the largest to the smallest eigenvalue. The condition number is infinite for a singular matrix, whereas a very large condition number indicates that the model is weakly identified. In such a scenario, the observed Fisher information matrix is still invertible, but the estimates of the variances will be large (and sometimes negative).

PROGRAM HANDLING

Using multiTree for parameter estimation of a model involves 4 steps: (1) Define a model; (2) select a data set; (3) review the model specification; and (4) press the "Analyze" button, the green arrow in the toolbar, or Alt + R to run the analysis and view the results.

Defining a Model

Two ways are provided by multiTree to define a model. Equations may be directly entered in the text field in the model tab. Alternatively, multiTree offers a graphical input method (the model builder). It is also possible to import existing equation files from other programs.

Entering equations. Equations defining a model can be entered directly in the text field in the equations tab. The syntax of the model files closely follows the format also used by AppleTree, GPT, and HMMTree (which is usually given a filename suffix .eqn), with the exception that it is neither required nor allowed to place the total number of branches ahead of the equations. When exporting an equation file, multiTree automatically adds this line; similarly, multiTree ignores this line when importing an equation file.

Beginning with the first line, the equation defining a branch of a tree is given. Each line consists of three parts, separated by one or more spaces or tabs: the label of the tree, the label of the category, and the actual equation. No spaces are allowed within any of these parts. The model equations for the Source A tree of the source monitoring model shown in Figure 1 could look as follows:

SourceA	AA	$D_1 * d_1$
SourceA	AA	$D_1 * (1 - d_1) * a$
SourceA	AB	$D_1 * (1 - d_1) * (1 - a)$
SourceA	AA	$(1 - D_1) * b * g$
SourceA	AB	$(1 - D_1) * b * (1 - g)$
SourceA	AN	$(1 - D_1) * (1 - b)$,

where the first line defines a branch of the "SourceA" tree leading to a response category labeled "AA." When each equation defining a model has been entered, it is necessary to notify multiTree about the new model: When a file is saved by the choice of "Save multiTree File" from the file menu, multiTree performs several checks to make sure that the equations describe an MPT model as defined in Equation 1. If multiTree accepts the model equations, it will read the parameter labels from the equations and update the parameter tab accordingly.

Using the model builder. In addition to entering model equations directly, multiTree offers a more intui-

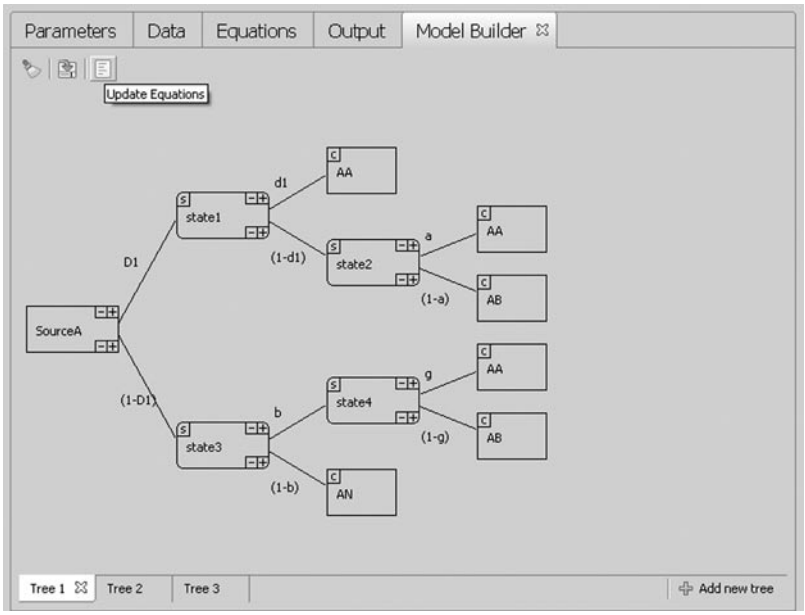

Figure 2. The graphical input facility for defining MPT models.

tive way of defining a model via a graphical input method. The model builder can be assessed by selecting "Open Model Builder" from the "Model" item in the menu bar. The model builder window comprises one or more tabs each representing a single tree (Figure 2). If a model contains multiple trees, new trees can be added by clicking on the "Add new tree" button at the lower right of the model builder window. The model builder can also be used to create a graphical tree representation from existing equations.

When building a tree, the first step is to define a label for the tree (e.g., "SourceA") by double-clicking on the default label of the root node. Branches can then be added by clicking on one of the "+" buttons on the right side of a node. Upon clicking on a "+" button, a new node appears, along with a link connecting the current with the new node. An arbitrary label can be attached to a node in the same way as is done with the root node; however, labels for nodes representing latent states are not required. The name of the parameter connecting two nodes needs to be defined on the corresponding link. The same parameter label must be used for links departing from the same node and the probability of the lower link [e.g., "$(1 - D_1)$"] needs to be the complement of the upper link (e.g., "D_1"). With the repetition of this process, new nodes and links can be easily added to a tree. By default, a node always represents a latent state. In order to change a node representing a latent state to a node representing a response category, the button labeled "s" in the upper left corner of a node has to be clicked, and the label of the response category can be entered the same way as in the case of latent states. Nodes representing latent states have a rectangular shape with rounded corners, whereas nodes representing response categories are displayed as rectangles.

When the model has been defined, equations can be generated by clicking on the "Update equations" button at the top of the model builder window; multiTree first checks whether the model is properly defined. If the model is accepted, the model equations are shown in the equations tab. In order to prompt multiTree to update the parameters given the newly created equations, it is necessary to save the current file.

Entering Data

The format of the data sets conforms to the format also used by Hu (1999), Rothkegel (1999), and Stahl and Klauer (2007), which is usually given a filename suffix .mdt. The first line must contain a title describing the data set. Beginning with the second line, a response category and a frequency count are given (separated by one or more blanks). It is important that the labels of the response categories in a data set match the category labels in the model equations. For example, a data set for the source monitoring model could look as follows (only the Source A tree is shown):

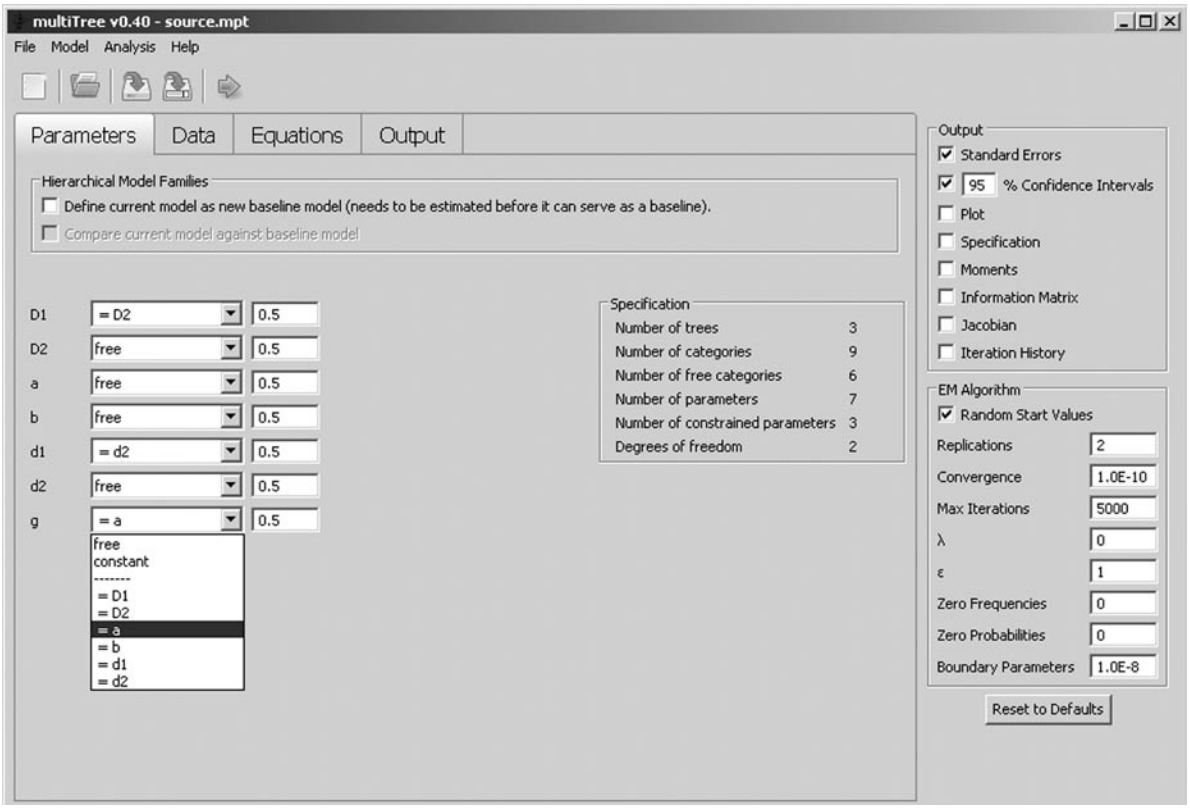

Figure 3. Setting constraints on parameters.

Rose, Rose, King, and Perez (1975)
AA 181
AB 39
AN 20

Multiple data sets referring to the same model may be stored by multiTree. There is a separate tab for each data set in the data window. New data sets can be added by clicking on the "Add new data set" button to the lower right. To discard a particular data set, it is sufficient to close the associated tab. Data sets can also be imported from an existing data file.

The drop-down menu at the top of the data window is used to specify the data set that should be used in an analysis. The drop-down menu also contains an entry labeled "Batch Analysis." If a batch analysis is requested, multiTree analyzes each of the currently available data sets and gives a summary output.

Placing Constraints on the Parameters

Constraints can be placed on a parameter in the window showing the parameters of the model along with a drop-down menu and a text field (Figure 3). There are two basic parameter constraints. A parameter may be fixed to a constant, or a parameter may be constrained to be equal to another parameter. A slightly more complicated situation arises if one wishes to apply constraints of the form $\theta_1 \leq \theta_2$. Parametric order constraints cannot be added directly to a model, but it is possible to reparameterize a given model such that order constraints on the parameters are reflected (Knapp & Batchelder, 2004).

Assign a constant to a parameter. To fix a parameter to a constant, it is necessary to select "constant" from the drop-down menu next to the parameter. The value of the parameter is entered in the corresponding text field. For example, to set the b parameter of the source monitoring model to zero, we select "constant" in the drop-down menu next to b and enter "0.0" in the associated text field. Note that constant values at the boundaries of the parameter space (0.0 and 1.0) cause a division by zero when computing the observed Fisher information matrix (Equation 15). Therefore, multiTree adjusts parameters at boundaries by a constant as defined in the preferences related to the EM algorithm in the "Boundary Parameters" text field. By default, boundary parameters are adjusted by $\pm$1.0E$-$8; that is, $\theta = 0.0$ is replaced by $\theta = 0.0 + 10^{-8}$ and $\theta = 1.0$ is replaced by $\theta = 1.0 - 10^{-8}$.

Equality restrictions. To restrict a parameter to be equal to another parameter, one selects the label of the target parameter from the drop-down menu next to the source parameter. For example, to impose the equality restriction $D_1 = D_2$, one selects "$= D_2$" from the drop-down menu next to D_1 (or vice versa). Consequently, one of the equality-constrained parameters is always freely estimated or fixed to a constant. To avoid circularity errors, a parameter cannot be set to be equal to a parameter itself constrained to be equal to another parameter. For example, to apply the restriction $D_1 = D_2 = d_1$, it is necessary to select "$= d_1$" from the drop-down menus of both D_1 and D_2, where d_1 is freely estimated or assigned a constant value.

Add order restrictions on parameters. Occasionally, one may want to model change in certain parameters. For example, in a learning experiment with R trials, θ_1 may represent a cognitive process or skill likely to improve with repeating trials. The natural modeling approach is to place an order restriction on this parameter of the form $\theta_{1,1} \leq \theta_{1,r} \leq \ldots \leq \theta_{1,R}$. In order to add such constraints to an MPT model as defined in Equation 1, it is necessary to apply some simple reparameterizations. For example, to add the constraint $\theta_{1,1} \leq \theta_{1,2}$, it is sufficient to replace every occurrence of $\theta_{1,1}$ with $\theta_{1,1} = \alpha_{1,1}\theta_{1,2}$ (Knapp & Batchelder, 2004), yielding a statistically equivalent model to the original model with parameters $\theta_{1,1}$ and $\theta_{1,2}$ subjected to the order constraint $\theta_{1,1} \leq \theta_{1,2}$. The newly introduced parameters $\alpha_{1,r}$ can then be interpreted as rates. It should be noted that these reparameterizations do not reduce the dimensionality of the model and, therefore, do not allow for using standard procedures of nested model testing.

Although in principle it is easily possible to reparameterize a model to reflect certain order constraints, the number of branches of a given model increases substantially. For example, Knapp and Batchelder (2004) reported an application of a pair clustering model, where adding order constraints on three parameters over six learning trials increased the number of branches from 36 to 256. It is clear that manually reparameterizing a model of this size is tedious and error-prone. Therefore, multiTree offers automatic reparameterizations of a model reflecting linear order constraints on the parameters. To request such a reparameterization, select "Add parametric order constraints" from the "Model" item in the menu bar. Order constraints can then be placed on the parameters in the same way as for ordinary equality constraints. When all desired constraints are set, multiTree generates the equations reflecting the specified order constraints on the parameters.

Changing Preferences Related to the Estimation Process

Several preferences related to the estimation process can be changed in the window to the right of the main window (Figure 3), such as the λ value of the power divergence statistic, the maximum number of iterations, and the criterion of convergence.

In some cases, the EM algorithm may get stuck in a local minimum. Thus, repeated estimations of the parameters may be requested in the text field labeled "Number of replications." Obviously, repeating the estimation process using identical start values will lead to the same parameter estimates, so random start values must be used for each run of the EM algorithm. If random start values are disabled, multiTree uses the values entered in the parameter tab next to the parameter labels as start values.

Depending on the value of λ, the power divergence statistic may be undefined when there are empty cells in the frequency tables (observed category frequencies of zero).[2]

Therefore, multiTree offers to replace zero frequencies by a constant value as defined in the text field labeled "Zero frequencies." Similarly, expected frequencies of zero may occur in some rare cases. Since both the power divergence statistic and the Fisher information matrix are undefined in this case, expected probabilities of zero may be replaced by a constant defined in the text field labeled "Zero probabilities."

Nested Model Testing

Since multinomial modeling often involves testing hierarchically nested model families, multiTree provides built-in support for computing the difference of the goodness of fit between two models. To define a baseline model, the box labeled "Define current model as new baseline model" in the parameter tab (Figure 3) has to be checked and the baseline model estimated. Subsequently defined models can then be compared against this baseline model when the box labeled "Compare current model against baseline model" is checked and the analysis is run. The output shows the ΔPD^{λ} statistic, the difference in the information criteria, and the normalized weight ratios (evidence ratios) of the AIC and the BIC. The evidence ratios represent the probability that the current model is the better model (in terms of the AIC and BIC) than the baseline model.

For example, suppose that the model of source memory was estimated with the equality restrictions $a = g$ and $D_1 = D_2$ and yielded $PD^{\lambda=0}$ ($df = 2$) = 0.82. We are now interested in determining whether the additional restriction $d_1 = d_2$ results in a significant loss in model fit. First, we have to define the baseline model by checking the box labeled "New baseline model" and running the analysis with $a = g$ and $D_1 = D_2$. Next, we have to define the more restrictive model. Thus, we additionally restrict $d_1 = d_2$, mark the box labeled "Compare against baseline," and click "analyze" to view the results. The output shows that the change in model fit is far from being significant with $\Delta PD^{\lambda=0}$ ($\Delta df = 1$) = 0.16.

Performing Power Analyses

The statistical power of a test is defined as the complement of the β-error probability of falsely retaining an incorrect null hypothesis (Cohen, 1988; Faul, Erdfelder, Lang, & Buchner, 2007). Generally, the power of a test is a function of the α-error probability, the sample size, and the degree of deviation between null (H_0) and alternative hypothesis (H_1). The β-error probability can be computed by evaluating the noncentral χ^2 distribution at a given α with the difference of the PD^{λ} fit statistics of a more restrictive H_0 model and a less restrictive H_1 model as an estimate of the noncentrality parameter (Erdfelder, Faul, & Buchner, 2005).

Two different types of power analysis can be computed by multiTree: In a priori power analyses, the required number of observations to reject an H_0 if it is false is

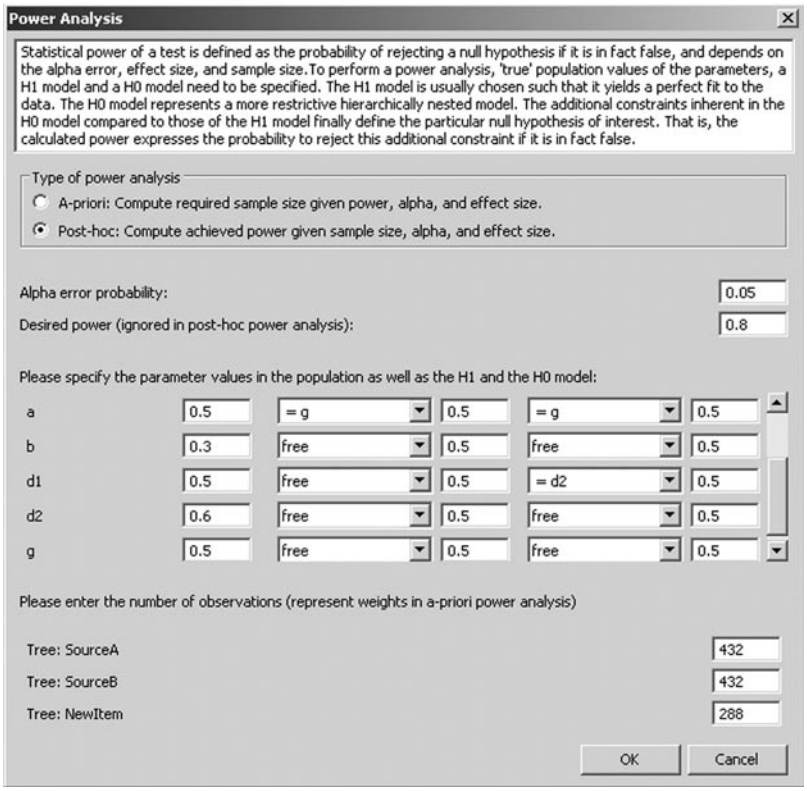

Figure 4. Performing power analyses.

computed and given a significance level α and the desired power. In post hoc power analyses, the achieved power to reject an H_0 if it is false is computed and given α and a prespecified number of observations. It should be noted that restrictions on parameters must lie in the admissible interval between 0 and 1 to satisfy the regularity condition of the likelihood ratio test (Birch, 1964).

Suppose we are interested in the power to detect a difference between the d_1 and d_2 parameters of $|d_1 - d_2| = .1$ in the model of source monitoring with $a = g$ and $D_1 = D_2$, given $\alpha = .05$ and a certain number of observations per tree (say, 432 for Source A items, 432 for Source B items, and 288 for new items). When "power analysis" is selected from the analysis menu, the power-analysis window appears (Figure 4). Since we are interested in the achieved power given a particular sample size, we request a post hoc power analysis by marking the checkbox labeled "post hoc." Next, the "true" parameter values in the population, the H_1 and the H_0 model, need to be specified. Suppose that previous research suggests population parameters of $a = g = .5$, $D_1 = D_2 = .8$, and $b = .3$. The hypothesis of interest is reflected by specifying a certain population value for d_1 (say, $d_1 = .5$) and d_2 (say, $d_2 = .6$), such that the difference between d_1 and d_2 equals $|d_1 - d_2| = .1$. After the population values have been set, the H_1 model needs to be specified. The H_1 model is usually chosen in such a way that it yields a perfect fit to the data. Consequently, we restrict $a = g$ and $D_1 = D_2$, whereas the remaining parameters may be freely estimated. Therefore, the parameters of the H_1 model will be estimated as defined in the population—that is, $a = g = .5$, $D_1 = D_2 = .8$, $b = .3$, $d_1 = .5$, and $d_2 = .6$. Finally, the H_0 model needs to be specified. The H_0 model contains the same restrictions as the H_1 model ($a = g$ and $D_1 = D_2$) and at least one additional constraint reflecting the hypothesis of interest. Since we are interested in the power to detect a difference between d_1 and d_2, the H_0 model posits that d_1 and d_2 are actually equal ($d_1 = d_2$). The power analysis may now be initiated by pressing the OK button. The results show that the power to detect the specified difference between the d_1 and d_2 parameters is $1 - \beta = .62$.

Suppose we also want to know the number of observations required to detect a difference between d_1 and d_2 of $|d_1 - d_2| = .1$ with a power of $1 - \beta = .8$. Performing an a priori power analysis is very similar to performing a post hoc power analysis, with three exceptions. Of course, the type of power analysis needs to be changed to "a priori" and the desired power has to be set, in the present example to $1 - \beta = .8$. Furthermore, the trees can be weighted when distributing the total number of observations across trees. For example, if we wanted to assign two times as many observations to the Source A and Source B trees as to the tree for new items, we could represent this restriction by assigning the Source A and Source B trees a weight of 2 and the new item tree a weight of 1. Performing this a priori power analysis shows that we would need about 1,650 observations in total (660 observations each for Source A and Source B; 330 observations for new items) to detect a difference between d_1 and d_2 of $|d_1 - d_2| = .1$ with a power of $1 - \beta = .8$.

ACCURACY OF THE ALGORITHMS UNDERLYING MULTITREE

Several tests were performed to validate the statistical algorithms implemented in multiTree. The accuracy of parameter estimation was established by analyzing simulated data sets generated from models with known parameter values. Model-implied category probabilities were created from a model of source monitoring involving 5 independent parameters (Batchelder & Riefer, 1990, model 5c) and from a more complex model of the hindsight bias involving 13 independent parameters (Erdfelder & Buchner, 1998). For each of these two models, 200 simulated data sets with 1,000 observations per tree were generated on the basis of random values of the parameters. Given that the generated category frequencies were not rounded to the nearest integer, analyzing the data sets with multiTree (with random starting values) should yield parameter estimates closely matching the true parameters. The results show that less than 1% of the $200 \times 5 + 200 \times 13 = 3{,}200$ unique parameter estimates differed in the fifth decimal place and less than 10% differed in the sixth decimal place from the true values. The maximum deviation of an estimate from the true parameter value was 4.2×10^{-5} and the mean deviation was 9.1×10^{-8}.

Moreover, the estimates of the PD^{λ} statistic and of the variances of the parameters were validated by analyzing five different data sets (Erdfelder & Buchner, 1998, Experiments 1–3; Johnson, Foley, & Leach, 1988; Rose et al., 1975, Experiment 1) for each of the two models with multiTree and GPT (Hu & Phillips, 1999). The two programs yielded variance estimates that agreed to at least five decimal places. The estimates of the PD^{λ} statistic agreed to the seventh decimal place. Finally, the implementation of the noncentral χ^2 distribution for power analyses was compared with G*Power 3 (Faul et al., 2007). G*Power and multiTree produced identical results.

LIMITATIONS

Several limitations apply to the current version of multiTree. First, it can handle only binary tree models of the form defined in Equation 1. There are, of course, models that do not comply with this general form and require more than two links at a given stage. Although such models can easily be reparameterized to fit the definition of binary MPT models, problems may arise in interpreting the parameters and confidence intervals of the reparameterized model (Batchelder & Riefer, 1999). Second, it would be useful to link multiTree to more general statistical programming environments such as the R project. Finally, multiTree offers no means to diagnose or handle heterogeneity across items and/or participants (Klauer, 2006; J. B. Smith & Batchelder, 2008). This is considered a major limitation and will be addressed in future versions.

SYSTEM REQUIREMENTS AND PROGRAM AVAILABILITY

The multiTree program is Java-based and runs under Linux, MacOS, and Windows operation systems, provided that at least version 6.0 of the Java runtime environment is installed on the target machine. The latest Java runtime environment may be freely downloaded from www.java.com. Depending on the platform, multiTree comes as archive file (Linux), disk image (Mac), or installer (Windows). In addition to the program itself, every distribution contains sample data (among others, the model of source memory used as a running example in the present article) and a license file.

Depending on the platform, hard-disk space requirements vary between 6 and 8 MB, with multiTree itself requiring about 40 MB RAM. Processing speed depends on many factors, including the size of the model, the number of required iterations, and the number of times the parameter estimation is replicated. As a rule of thumb, using a 2-GHz Intel Core 2 processor, multiTree needs about 0.04 msec for each iteration and parameter in the model. For example, parameter estimation lasts for about 1 sec for a model with 25 parameters that requires 1,000 iterations to converge. However, when performing a bootstrap analysis with 1,000 simulations on a model comprising 80 parameters that needs on average 5,000 iterations to converge and requires 5 replications to assure convergence to a global minimum, the analysis will last for about 22 h.

Linux, MacOS, and Windows versions of multiTree can be downloaded from http://psycho3.uni-mannheim.de/multitree free of charge for academic and personal use. Users who wish to distribute multiTree in another way need to ask the author for permission. The charge for commercial applications is €400. Although considerable effort has been put into program development and evaluation, there is no warranty whatsoever.

AUTHOR NOTE

The author thanks Edgar Erdfelder and Jennifer Nicolai for their helpful comments on drafts of the manuscript. Furthermore, the author is grateful to all testers of previous versions of multiTree for their valuable comments, particularly Axel Buchner, Edgar Erdfelder, Benjamin Hilbig, Albert-Georg Lang, Jochen Musch, and Michael Wolf. Correspondence concerning this article should be addressed to M. Moshagen, University of Mannheim, Lehrstuhl Psychologie III, Schloss EO 254, 68131 Mannheim, Germany (e-mail: moshagen@uni-mannheim.de).

REFERENCES

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, **19**, 716-723.

Bamber, D., & van Santen, J. P. H. (1985). How many parameters can a model have and still be testable? *Journal of Mathematical Psychology*, **29**, 443-473.

Bamber, D., & van Santen, J. P. H. (2000). How to assess a model's testability and identifiability. *Journal of Mathematical Psychology*, **44**, 20-40.

Batchelder, W. H., & Riefer, D. M. (1990). Multinomial processing models of source monitoring. *Psychological Review*, **97**, 548-564.

Batchelder, W. H., & Riefer, D. M. (1999). Theoretical and empirical review of multinomial process tree modeling. *Psychonomic Bulletin & Review*, **6**, 57-86.

Birch, M. W. (1964). A new proof of the Pearson–Fisher theorem. *Annals of Mathematical Statistics*, **35**, 817-824.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences*. Hillsdale, NJ: Erlbaum.

Dempster, A. P., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society 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.

Efron, B., & Tibshirani, R. J. (1997). *An introduction to the bootstrap*. New York: Chapman & Hall.

Erdfelder, E. (2000). *Multinomiale Modele in der kognitiven Psychologie* [Multinomial models in cognitive psychology]. Unpublished habilitation thesis, Friedrich-Wilhelms University, Bonn.

Erdfelder, E., Auer, T.-S., Hilbig, B. E., Aßfalg, A., Moshagen, M., & Nadarevic, L. (2009). Multinomial processing tree models: A review of the literature. *Zeitschrift für Psychologie/Journal of Psychology*, **217**, 108-124.

Erdfelder, E., & Buchner, A. (1998). Decomposing the hindsight bias: A processing tree model for separating recollection and reconstruction biases in hindsight. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **24**, 387-414.

Erdfelder, E., Faul, F., & Buchner, A. (2005). Power analysis for categorical methods. In B. Everitt & D. Howell (Eds.), *Encyclopedia of statistics in behavioral science* (pp. 1565-1570). Chichester, U.K.: Wiley.

Faul, F., Erdfelder, E., Lang, A., & Buchner, A. (2007). G*Power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences. *Behavior Research Methods*, **39**, 175-191.

García-Pérez, M. A. (1994). Parameter estimation and goodness-of-fit testing in multinomial models. *British Journal of Mathematical & Statistical Psychology*, **47**, 247-282.

Hu, X. (1999). Multinomial processing tree models: An implementation. *Behavior Research Methods, Instruments, & Computers*, **31**, 689-695.

Hu, X., & Batchelder, W. H. (1994). The statistical analysis of general processing tree models with the EM algorithm. *Psychometrika*, **59**, 21-47.

Hu, X., & Phillips, G. A. (1999). GPT.EXE: A powerful tool for the visualization and analysis of general processing tree models. *Behavior Research Methods, Instruments, & Computers*, **31**, 220-234.

Johnson, M. K., Foley, M. A., & Leach, K. (1988). The consequences for memory of imagining in another person's voice. *Memory & Cognition*, **16**, 337-342.

Kenny, D. A. (1979). *Correlation and causality*. New York: Wiley.

Klauer, K. C. (2006). Hierarchical multinomial processing tree models: A latent-class approach. *Psychometrika*, **71**, 7-31.

Klauer, K. C., & Oberauer, K. (1995). Testing the mental model theory of propositional reasoning. *Quarterly Journal of Experimental Psychology*, **48A**, 671-687.

Knapp, B. R., & Batchelder, W. H. (2004). Representing parametric order constraints in multi-trial applications of multinomial processing tree models. *Journal of Mathematical Psychology*, **48**, 215-229.

Read, T. R. C., & Cressie, N. A. C. (1988). *Goodness-of-fit statistics for discrete multivariate data*. New York: Springer.

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. (1991). Statistical inference for multinomial processing tree models. In J.-P. Doignon & J.-C. Falmagne (Eds.), *Mathematical psychology: Current developments* (pp. 313-335). Berlin: Springer.

Rose, R. G., Rose, P. R., King, N., & Perez, A. (1975). Bilingual memory for related and unrelated sentences. *Journal of Experimental Psychology: Human Learning & Memory*, **1**, 599-606.

Rothkegel, R. (1999). AppleTree: A multinomial processing tree mod-

eling program for Macintosh computers. *Behavior Research Methods, Instruments, & Computers*, **31**, 696-700.

ROUDER, J. N., & BATCHELDER, W. H. (1998). Multinomial models for measuring storage and retrieval processes in paired associate learning. In C. E. Dowling, F. S. Roberts, & P. Theuns (Eds.), *Recent progress in mathematical psychology* (pp. 195-225). Mahwah, NJ: Erlbaum.

SAEGERT, J., HAMAYAN, E., & AHMAR, H. (1975). Memory for language of input in polyglots. *Journal of Experimental Psychology: Human Learning & Memory*, **1**, 607-613.

SCHWARTZ, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, **6**, 461-464.

SMITH, J. B., & BATCHELDER, W. H. (2008). Assessing individual differences in categorical data. *Psychonomic Bulletin & Review*, **15**, 713-731.

SMITH, P. L. (1998). Attention and luminance detection: A quantitative analysis. *Journal of Experimental Psychology: Human Perception & Performance*, **24**, 105-133.

STAHL, C., & KLAUER, K. C. (2007). HMMTree: A computer program for latent-class hierarchical multinomial processing tree models. *Behavior Research Methods*, **39**, 267-273.

WAGENMAKERS, E.-J., & FARRELL, S. (2004). AIC model selection using Akaike weights. *Psychonomic Bulletin & Review*, **11**, 192-196.

NOTES

1. MPT models have the property that the Jacobian attains its maximum rank almost everywhere in the parameter space (Bamber & van Santen, 1985). Consequently, to determine the maximum rank it suffices to obtain the rank at a random location in the parameter space.

2. In the case of $PD^{\lambda=0}$, multiTree adds zero for empty response categories and thereby avoids the undefined logarithm (see Equation 7).

(Manuscript received June 29, 2009;
revision accepted for publication August 28, 2009.)