

Running head: HMMTree

HMMTree: A computer program for latent-class  
hierarchical multinomial processing tree models

Christoph Stahl and Karl Christoph Klauer

Albert-Ludwigs-Universität Freiburg

Address for correspondence:

Christoph Stahl

Institut für Psychologie, Universität Freiburg, D-79085 Freiburg, Germany

E-mail: [stahl@psychologie.uni-freiburg.de](mailto:stahl@psychologie.uni-freiburg.de)

Tel.: +49 761 203 2418

Fax: +49 761 203 2417

### Abstract

Latent-class hierarchical multinomial models are an important extension to the widely-used family of multinomial processing tree models: they allow for testing of the parameter homogeneity assumption and provide a framework for modeling parameter heterogeneity. In this article, the computer program HMMTree is introduced that implements latent-class hierarchical multinomial processing tree models. HMMTree computes parameter estimates, confidence intervals, and goodness-of-fit statistics for these models, as well as the Fisher information, expected category means and variances, and posterior probabilities for class membership. A brief guide to using the program is provided.

**Keywords:** computer program, multinomial processing tree models, latent-class hierarchical multinomial models, parameter heterogeneity

(102 words)

HMMTree: A computer program for latent-class hierarchical multinomial  
processing tree models

Multinomial processing tree (MPT) models are models for categorical data that have been successfully applied to a wide range of paradigms within cognitive and social psychology (Batchelder & Riefer, 1999; Erdfelder, 2000; Riefer & Batchelder, 1988). A multinomial model incorporates assumptions about the psychological processes underlying the data obtained in that paradigm. The estimates of the model parameters can usually be interpreted as probabilities of these processes. For example, consider a standard recognition task, where participants are presented with a list of items – half of which were presented before – and are to decide whether an item is old or new. In that situation, an “old” decision to an old item can be based either on memory for the item (parameter  $D$ ) or on guessing “old” (parameter  $G$ ) when no memory is present ( $1 - D$ ), whereas a “new” decision to an old item would imply no memory for the item ( $1 - D$ ) and guessing “new” ( $1 - G$ ). For sake of simplicity, it is assumed in this example that decisions to new items are based only on guessing (but see Bayen, Murnane & Erdfelder, 1996). These assumptions are easily cast in the form of a multinomial model, the so-called one-high threshold model of recognition. The observed categorical events (“old” and “new” responses to old and new items) are connected to the parameters representing the psychological processes of guessing and memory by the following equations:  $P(\text{“old”} \mid \text{old}) = D + (1 - D) * G$ ;  $P(\text{“new”} \mid \text{old}) = (1 - D) * (1 - G)$ ;  $P(\text{“old”} \mid \text{new}) = G$ ;  $P(\text{“new”} \mid \text{new}) = 1 - G$ . Given a

validated model and an acceptable fit to the data, the parameter values for the memory and guessing parameters  $D$  and  $G$  can be used as separate measures for the contribution of these processes to the observed behavior in the recognition task.

MPT models have been applied to paradigms in many research areas in psychology, including memory (e.g., Bayen et al., 1996; Brainerd, Stein & Reyna, 1998; Schweickert, 1993), perception (e.g., Ashby, Prinzmetal, Ivry & Maddox, 1996; Batchelder & Crowther, 1997), reasoning (e.g., Evans, 1977; Klauer, Musch & Naumer, 2000), social cognition (e.g., Klauer & Wegener, 1998), and psychological assessment (e.g., Batchelder, 1998). Typically, MPT models are applied to category frequencies summed across participants. In these cases, equal parameter values across participants have to be assumed. This assumption of parameter homogeneity usually remains untested although it is likely to be often violated in models of cognitive tasks such as reasoning or memory tasks, in which there is a substantial amount of variation between participants.

When parameter heterogeneity is present, this could lead to an erroneous rejection of valid MPT models (Riefer & Batchelder, 1991). It could also result in underestimated standard errors and confidence intervals of the model parameters; as a consequence, tests of significant differences between parameters could exhibit  $\alpha$ -errors inflated beyond the nominal level of significance. Finally, parameter estimates can be biased away from the underlying values. These problems are addressed by the recently introduced

latent-class hierarchical approach to MPT modeling (Klauer, in press). This extension of MPT models allows one to test the homogeneity assumption, and, in case of violation, parameter heterogeneity can be modeled by a set of latent classes with different parameter values. In this article, the computer program HMMTree is introduced that computes parameter estimates, confidence intervals, and goodness-of-fit statistics for hierarchical multinomial processing tree models. It can also be used to analyze data in the traditional way under the assumption of parameter homogeneity. HMMTree is available for download from <http://www.psychologie.uni-freiburg.de/Members/stahl/HMMTree>.

First, a short introduction is given to the new class of latent-class hierarchical MPT models and the computation of the model parameters and test statistics (notation follows Klauer, in press, see that paper also for mathematical details). It is followed by a description of the HMMTree program, its user interface, and the equation and data file formats used. Third, a step-by-step guide to computing latent-class hierarchical multinomial models with HMMTree is offered, illustrating the computation of parameter estimates and confidence intervals, goodness-of-fit statistics, hypotheses tests regarding parameter values, means as well as variances and covariances of expected and observed category frequencies, and the posterior probability of class membership for a given participant.

#### Latent-class hierarchical MPT models

Hierarchical multinomial models (HMM models; Klauer, in press) extend the class of MPT models originally formulated by Riefer and

Batchelder (1988, see also Hu & Batchelder, 1994) by a latent-class approach to capture parameter heterogeneity. MPT models refer to category probabilities  $p_{kj}$  from  $K$  different groups with  $J_k$  categories in group  $k$ ,  $k = 1, \dots, K, j = 1, \dots, J_k$ . The corresponding category frequencies,  $n_{kj}$ , are usually obtained by aggregating the observed frequencies of several participants. The model equations of the MPT model,  $p_{kj}(\boldsymbol{\theta})$ , express these category probabilities in terms of parameters  $\boldsymbol{\theta}$ , thereby formulating assumptions about how these probabilities are generated by the processes postulated by the MPT model. Equation 1 gives the probability  $P$  of a vector of category frequencies,  $\mathbf{n}$ , given the vector of model parameters  $\boldsymbol{\theta}$ .

$$(1) \quad P(\mathbf{N} = \mathbf{n} \mid \boldsymbol{\theta}) = \prod_{k=1}^K \left[ \binom{N_k}{n_{k1} \dots n_{kJ_k}} \prod_{j=1}^{J_k} [p_{kj}(\boldsymbol{\theta})]^{n_{jk}} \right]$$

The model parameters  $\boldsymbol{\theta}$  are estimated using the Maximum-Likelihood method as implemented in the EM-algorithm. The asymptotically  $\chi^2$ -distributed log-likelihood statistic  $G^2$  is used to assess goodness of fit of MPT models (see Hu & Batchelder, 1994). A good fit (indicated by a  $G^2$  below the critical value) is expected when the model can accommodate the aggregated category frequencies and the true parameter values are homogeneous across participants. A bad fit can result if the model structure does not approximate the structure of participants' response process well. However, a bad fit can also result if the model accurately describes the structure of a person's response process, but parameters differ across persons (Klauer, in press). The traditional MPT approach does not distinguish between these situations. When parameter

heterogeneity is present, simple MPT models are often rejected even if they accurately describe the structure of each individual's response process, and more complex models with more parameters are then often fitted to the data. As a consequence, when parameter heterogeneity is present, substantive conclusions drawn from MPT models can be misleading.

HMM models, in contrast, are computed on the basis of the participants' individual category frequencies. Equation 2 describes the *core* model of an HMM model by applying Equation 1 to the vector  $\mathbf{n}_t$  of category frequencies of a single participant  $t$ ,  $t = 1, \dots, T$  (with  $\boldsymbol{\theta}_t$  representing the vector of parameter values for that participant).

$$(2) \quad P(\mathbf{N} = \mathbf{n}_t \mid \boldsymbol{\theta}_t) = \prod_{k=1}^K \left[ \binom{N_k}{n_{k1t} \dots n_{kJ_k t}} \prod_{j=1}^{J_k} [p_{kj}(\boldsymbol{\theta}_t)]^{n_{jkt}} \right]$$

In principle, one could compute a different set of parameter estimates for each participant; in practice, the number of available observations is often insufficiently small. In the latent-class approach, different parameter vectors are computed for a fixed number  $C$  of latent classes. The data obtained from  $T$  participants can thus be described by Equation 3,

$$(3) \quad P((\mathbf{N}_1, \dots, \mathbf{N}_T) = (\mathbf{n}_1, \dots, \mathbf{n}_T)) = \prod_{t=1}^T \left[ \sum_{c=1}^C \lambda_c P(\mathbf{N} = \mathbf{n}_t \mid \boldsymbol{\theta}_c) \right]$$

where  $\boldsymbol{\theta}_c$  is the vector of parameter values for class  $c$ , and the sizes of the  $C$  latent classes,  $\lambda_1, \dots, \lambda_c$ , sum up to 1.

For HMM models, estimates for the  $C$  class size parameters  $\lambda_1, \dots, \lambda_c$  have to be computed, as well as different estimates for the  $S$  core model

parameters for each of the  $C$  classes,  $\theta_{11}, \dots, \theta_{S1}, \dots, \theta_{SC}$ . HMMTree computes parameter estimates using a combination of an EM algorithm (Klauer, in press) and the conjugate-gradient method. After a few initial iterations with the EM algorithm, the more efficient conjugate-gradient method is used to maximize the likelihood. A number of final EM algorithm iterations serve to increase the precision of the results.

In the case of a single class ( $C = 1$ ), the HMM model is equivalent to an MPT model. Thus, HMMTree can also be used to compute traditional MPT models. In the single-class case, the  $G^2$  statistic is computed. If  $C > 1$ , HMMTree provides goodness-of-fit statistics that assess the fit of the means structure (statistics  $M_1$  and  $M_2$ ) as well as the fit of the variance-covariance structure of the individuals' category frequencies (statistics  $S_1$  and  $S_2$ ; Klauer, in press). The  $M_1$  and  $M_2$  statistics assess the mean structure, whereas the  $S_1$  and  $S_2$  statistics assess the variance-covariance structure; all four are asymptotically  $\chi^2$ -distributed (with degrees of freedom as specified in Klauer, in press, and computed by HMMTree). Thus, if  $M_1$  and  $M_2$  are below the critical  $\chi^2$  value, the means of the observed and expected category frequencies closely correspond. If  $M_1$  and/or  $M_2$  exceed the respective critical value, the model does not describe the mean structure of the data well. If  $S_1$  and  $S_2$  are below the respective critical  $\chi^2$  values, the model describes the variance-covariance structure of the data well. If  $S_1$  and/or  $S_2$  exceed their respective critical values, the model does not describe the variance-covariance structure of

the data well, indicating that the model does not describe the extant heterogeneity adequately.

In addition to the mentioned goodness-of-fit statistics, the Akaike Information Criterion (AIC; Akaike, 1974) and the Bayes Information Criterion (BIC; Schwarz, 1978) are computed (the reported BIC is based on the number of observations; in case of multiple classes, a second BIC is reported that is based on the number of persons).

To obtain confidence intervals for the parameters and to compute  $M_2$  and  $S_2$ , HMMTree computes the inverse of the expected Fisher information. The expected Fisher information can be easily obtained for MPT models in most cases. Its computation is also possible for HMM models; however, computation becomes very resource-intensive as the number of categories and the number of observations per person increases, and is thus not always feasible. To accommodate for this, the *observed* Fisher information matrix can be used alternatively; it is computed more easily. A Monte Carlo estimate of the observed Fisher information can also be computed. The downside of using the observed Fisher information or the Monte Carlo Fisher information is that the goodness-of-fit statistic  $M_2$  cannot be computed correctly; one has to resort to using  $M_1$  in those cases (see Klauer, in press, for a mathematical discussion of the problem).

HMMTree also computes the observed and expected category frequencies and their variances and covariances. A comparison of observed and expected means as well as variances and covariances is helpful in determining

the categories that contribute to a bad fit. Finally, HMMTree computes each participant's posterior probabilities for membership in each class; these values can serve as a basis of mapping participants to classes.

### Introduction to HMMTree

The HMMTree program computes HMM models on the Microsoft Windows platform. It also provides a simple and convenient way to compute MPT models on that platform (see also Hu & Phillips, 1999). It consists of a comfortable visual user interface and a FORTRAN library that implements the computation algorithms described in Klauer (in press). In order to minimize computation times, interaction between the computation process and the visual interface was minimized. In the following, the HMMTree user interface is introduced, followed by a description of the equations and data files.

#### *User interface*

HMMTree presents four empty tabbed windows and a set of computation options (Figure 1). On the right side of the main window, the number of latent classes is specified that are to be estimated; the desired outputs are selected from the available options; and computations are executed and aborted. The *Best of N* option allows multiple sets of parameter estimates to be computed using different randomly selected starting values. Also, basic information about model and data are displayed.

(Figure 1 about here)

On the left side of the main window, the *Parameters* window is shown in front upon startup. Here, restrictions can be imposed upon the model parameters by fixing a parameter value to a constant, by setting two or more parameters equal within the core model, or by setting a core model parameter equal across classes. Parameter restrictions for different latent classes can be imposed by selecting the desired class from the drop-down box at the top of the window. For reasons we will discuss below, HMMTree allows editing of parameter estimates before computation of Fisher information, moments, goodness-of-fit statistics, and posterior probabilities.

After computation, the *Output* window is presented, displaying the results. Each output contains the names of model equation and data files and the log-likelihood value for the computed model along with the number of parameters estimated. In addition, the selected outputs are printed; by default, these include estimates of the model parameters and confidence intervals of these estimates generated from the inverse of the Fisher information matrix. Additional output options include printing the inverse of the Fisher matrix, the means as well as variances and covariances of observed and expected category frequencies, and participants' posterior probabilities of class membership.

The *File* menu contains commands to load model equation and data files; selection of correct model equation and data files can be verified by looking at the *Model* and *Data* windows. For the *Output* window, editing options are available from the *Edit* menu. In the *Settings* menu, the display font can be specified, a shortcut to removing all parameter restrictions is provided,

and a choice between 90% or 95% confidence intervals is offered. A brief user guide and information about the program can be accessed from the *Help* menu.

### *Model equation files*

All files read and written by HMMTree are of plain text format. For the core model equation files, the EQN syntax is used that is compatible with the MBT (Hu, 1999) and AppleTree (Rothkegel, 1999) programs for analyzing traditional MPT models. Figure 2 illustrates the EQN syntax: The first line of an EQN file gives the number of equations it contains. Beginning on the second line, the equations are given, with each line representing one branch of the processing tree. Each of these lines contains three values (separated by one or more spaces or tabs) – the first being the number of the tree or sub-model the equation belongs to, the second being the number of the category that this branch is associated with, and the third value being the product of parameters that are postulated to contribute to performance in that branch of the processing tree. Tree numbers must begin with 1 and increase by 1 for each tree. Similarly, category numbers must begin with 1 and increase by 1 for each category, and they must provide a unique number for each category across all trees (i.e., if category 1 occurs within tree 1, it must not occur within tree 2).

(Figure 2 about here)

### *Data files*

Data files are also of plain text format. Two different types of files can be read by HMMTree: (1) Data are read from the common MDT file format (see Rothkegel, 1999; Hu, 1999) that contains the category frequencies for each category (see Figure 3 for an example). The first line contains a description of the data set. On each subsequent line, the number of the category is given, followed by its frequency (separated by one or more spaces or tabs). A data set is terminated by a line beginning with three (or more) '=' characters (*equal signs*). MDT files can contain multiple data sets; they are treated as individual participants' data sets by HMMTree. (2) Individual participants' category frequencies can also be given in a category  $\times$  participant matrix where the first line contains the category numbers as column headers and each additional line contains a participant's category frequency vector (again separated by one or more spaces or tabs). This is referred to as the DAT file type (see Figure 4 for an example); it can be generated manually or exported from most general-purpose statistics packages.

(Figures 3 and 4 about here)

### Using HMMTree

In the first part of the following section, the basic steps to computing HMM models with HMMTree are described. These include selecting model equation and data files, specifying the desired number of latent classes,

specifying output options, and executing computation. Next, additional options are described, including selecting additional outputs, imposing parameter restrictions, and different types of Fisher information matrices. Practical limits regarding the handling of models with large numbers of observations per participant and categories are addressed.

### *Basic steps*

The first step is to select a file that contains the model equations, an EQN file. It can be opened via the Load equations command from the File menu (additionally, keyboard shortcuts, described in the user guide, are available for all commands). The file is rejected if model equations do not conform to a binary processing tree structure. If accepted, equations are displayed in the Model window; now, a data file can be opened via the Load data command. Data files are accepted if they conform to the number of categories specified by the model and provide the same number of observations for each participant. Data with zero cells can be handled by HMMTree. Although a constant of one is frequently added when there are zero counts in multinomial analyses, this is not done by HMMTree and the data are used as entered into the program.

When model equations and data are accepted, the desired number of latent classes can be specified (top right corner of the program window). To compute a model with more than one class, individual participants' data are required; the single-class case can be computed with individual as well as with aggregated category frequencies.

When the number of classes has been specified, computation is initiated by pressing the *Run* button. With the default settings, parameter estimates and confidence intervals as well as the expected Fisher information are computed. The *Best of ...* option allows for the repeated execution of the parameter estimation algorithm with random start values. This provides a useful check of local identifiability: it is violated if a maximum of the likelihood function is obtained for different sets of parameter estimates. The set of parameter estimates for which the largest likelihood value is obtained then serves as a basis for the subsequent computation of the Fisher information. Because the computation of the expected Fisher information can be very time-consuming even for small class numbers, precautions were taken to avoid undesired lengthy computations. In a first step, an estimate of the time and number of operations needed for this computation is displayed in a pop-up window; it is then upon the researcher to decide whether to actually run this computation. Alternatives to computing the expected Fisher information are discussed below. Computation can be terminated by pressing the *Abort* button.

Computation results are displayed in the *Output* window (see Figure 5 for an example). Simple text editing functions, such as copy and paste, and basic undo/redo functionality is provided there. Results can be saved to a text file with the *Save* or the *Save as* commands from the *File* menu.

(Figure 5 about here)

### *Specifying parameter restrictions*

Testing hypotheses regarding model parameters is accomplished by assessing the difference in goodness-of-fit (more precisely, twice the difference between the log-likelihood values) between different sub-models. To obtain these sub-models that can be tested against each other from the model described in the equation files, HMMTree provides ways to introduce parameter restrictions in the *Parameters* window. Three types of restrictions are supported – replacing parameters with a constant, setting two parameters equal to each other, and fixing a parameter across latent classes: (1) To replace a parameter with a constant, in a first step the left of the two drop-down boxes is switched from *free* to *constant*; then, the text field containing the parameter estimate is to be changed to hold the desired constant value separately for all classes (see Figure 6); (2) to specify that a parameter A be equal to another parameter B, the name of parameter B is to be selected from the left drop-down box (see Figure 7; note that a given parameter A can be set equal to another parameter B only if no other parameter C has been set equal to parameter A. To impose an equality restriction on the three parameters at the same time, specify both parameters A and C to be equal to B); (3) to fix a parameter to be equal across all latent classes, the right drop-down box is switched from *free* to *fixed* (see Figure 8).

(Figures 6, 7, 8 about here)

These types of restrictions can be combined in the following ways: (a) a parameter can be replaced by a constant and fixed across classes at the same time (this is equivalent to replacing it by the same constant in all classes separately); (b) a parameter A can be restricted to be equal to parameter B while B is replaced by a constant (this is equivalent to replacing both parameters with the same constant in all classes); (c) a parameter A can be restricted to be equal to parameter B while B is fixed across latent classes (this results in parameter A also being fixed across latent classes).

Other types of tests of parameter restrictions might be of interest in some cases, in which restrictions are limited to a single latent class or a subset of latent classes (e.g., testing for equality of parameters A and B for each latent class separately, setting parameter A of class 1 equal to parameter B of class 2, or setting parameter A of class 1 equal to parameter A of class 2 but leaving parameter A of class 3 free). The log-likelihood ratio method cannot be used for these tests because there is no control over which latent classes the restrictions finally turn out to be applied to. However, Wald's tests can be used (see Klauer, in press).

#### *Computational limits*

Neither the number of model equations, categories, subjects, nor classes is limited by design in HMMTree. However, the number of classes might be limited by practical considerations: computation times rise with the number of classes; and a larger number of classes imply a larger number of model parameters, and this increases the risk that the model cannot be identified.

Computation time for the expected Fisher information with  $C > 1$  classes depends on the number of observations per person,  $N_k$ , and on the number of categories,  $J_k$ , that determine the number  $\prod_k \binom{N_k + J_k - 1}{N_k}$  of vectors that are summed in the computation (Klauer, in press). Before the expected Fisher information is computed for a model with  $C > 1$  classes, the number of vectors is determined as an estimate for the number of operations to be computed. Whereas the number of classes does not affect the number of vectors, it increases the complexity of computing the vectors and thus affects computation time. An estimate for the required computation time is obtained by timing the computation of a fixed number of 2000 vectors and scaling the result to the total number of vectors. The obtained estimates for number of operations and required time are then displayed, and it is upon the user to decide whether to start the actual computation.

Large observed category frequencies can pose another problem for the computation of the expected Fisher information: The probability of a numerical underflow or overflow error increases with category frequencies. A warning is issued when computation is expected to encounter this problem in order to inform a user's judgment of whether it is worthwhile to run a lengthy computation.

Parameter estimates that lie on or close to the borders of the parameter space  $[0,1]$  pose a third problem for the computation of the inverse of the Fisher information matrix. If this occurs, a warning is issued, prompting the

user to replace such extreme estimates with more moderate values before an attempt to computing the Fisher information is made. Parameter estimates can be modified in the *Parameters* window.

#### *Alternatives to the expected Fisher information*

HMMTree computes the inverse of the expected Fisher information matrix as a default option because this information is needed to obtain the goodness-of-fit statistic  $M_2$ . However, as described above, one may not always want to select this option. As an alternative, the so-called observed Fisher information matrix can be computed (Klauer, in press). Its precision in estimating the expected Fisher matrix is increased by a Monte Carlo approach, where the matrix is based on a specified number (e.g., 10000) of simulated data sets randomly sampled from the observed data. This Monte Carlo variant of the observed Fisher information is available as a second alternative. Computation of both alternatives is accomplished efficiently.

#### Summary

HMM models are an important addition to the psychologist's modeling toolbox. Firstly, they allow for the testing of the parameter homogeneity assumption underlying multinomial processing tree models. Secondly, they extend the applicability of multinomial processing tree models to research on interindividual and group differences by modeling parameter heterogeneity. With HMMTree, a convenient implementation of this new class of models is now available for the Windows environment, providing the basis for their application in future research.

## References

- Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19, 716-723.
- Ashby, F. G., Prinzmetal, W., Ivry, R., & Maddox, W. T. (1996). A formal theory of feature binding in object perception. *Psychological Review*, 103, 165-192.
- Batchelder, W. H. (1998). Multinomial processing tree models and psychological assessment. *Psychological Assessment*, 10, 331-344.
- Batchelder, W. H., & Crowther, C. S. (1997). Multinomial processing tree models of factorial categorization. *Journal of Mathematical Psychology*, 41, 45-55.
- Batchelder, W. H., & Riefer, D. M. (1999). Theoretical and empirical review of multinomial process tree modeling. *Psychonomic Bulletin & Review*, 6, 57-86.
- Bayen, U. J., Murnane, K., & Erdfelder, E. (1996). Source discrimination, item detection, and multinomial models of source monitoring. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, 22, 197-215.
- Brainerd, C. J., Stein, L. M., & Reyna, V. F. (1998). On the development of conscious and unconscious memory. *Developmental Psychology*, 34, 342-357.

- Erdfelder, E. (2000). *Multinomiale Modelle in der kognitiven Psychologie* [*Multinomial models in cognitive psychology*]. Unpublished habilitation thesis, Universität Bonn, Germany.
- Evans, J. St. B. T. (1977). Toward a statistical theory of reasoning. *Quarterly Journal of Experimental Psychology*, 29, 621-635.
- 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 engineering 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.
- Klauer, K. C. (in press). Hierarchical multinomial processing tree models: A latent-class approach. *Psychometrika*.
- Klauer, K. C., Musch, J., & Naumer, B. (2000). On belief bias in syllogistic reasoning. *Psychological Review*, 107, 852-884.
- Klauer, K. C., & Wegener, I. (1998). Unraveling social categorization in the "Who said what?" paradigm. *Journal of Personality & Social Psychology*, 75, 1155-1178.
- 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). New York: Springer.
- Rothkegel, R. (1999). AppleTree: A multinomial processing tree modeling program for Macintosh computers. *Behavior Research Methods, Instruments, & Computers*, 31, 696-700.
- Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6, 461-464.
- Schweickert, R. (1993). A multinomial processing tree model for degradation and redintegration in immediate recall. *Memory & Cognition*, 21, 168-175.

## Author Note

Christoph Stahl, Institut für Psychologie; Karl Christoph Klauer,  
Institut für Psychologie.

The authors thank Edgar Erdfelder, Christoph Kaller, Rainer Leonhart,  
Jochen Musch, Christine Sattler, and three anonymous reviewers for helpful  
comments regarding previous versions of the manuscript and the HMMTree  
software.

The research reported in this paper was supported by grant Kl 614/31-1  
from the Deutsche Forschungsgemeinschaft to the second author.

Correspondence concerning this article should be addressed to  
Christoph Stahl, Institut für Psychologie, Albert-Ludwigs-Universität Freiburg,  
D-79085 Freiburg, Germany, [stahl@psychologie.uni-freiburg.de](mailto:stahl@psychologie.uni-freiburg.de).

## Figure Captions

*Figure 1.* The HMMTree user interface.

*Figure 2.* An example EQN file.

*Figure 3.* An example MDT file.

*Figure 4.* An example DAT file.

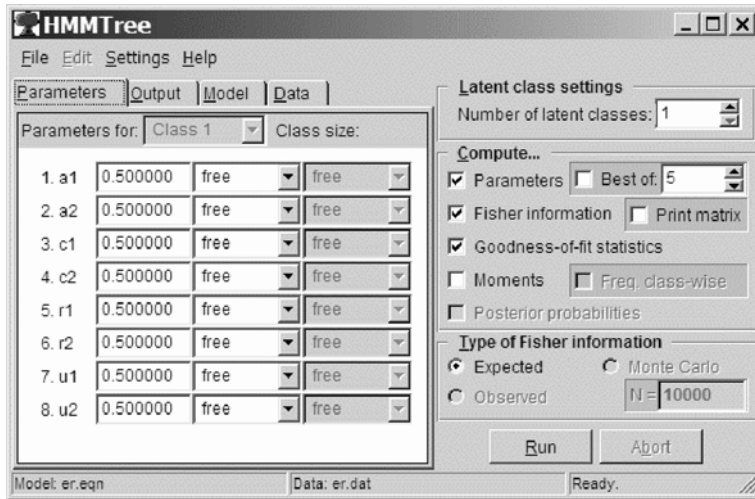
*Figure 5.* An example output with two classes. Parameter values estimated for the two classes are written on one line for each core model parameter along with confidence intervals.

*Figure 6.* Replacing a parameter with a constant.

*Figure 7.* Imposing an equality restriction.

*Figure 8.* Fixing a parameter across latent classes.

Figure 1



*Figure 2*

```
16
1 1 c1*r1
1 4 c1*(1-r1)
1 2 (1-c1)*u1*u1
1 3 (1-c1)*u1*(1-u1)
1 3 (1-c1)*(1-u1)*u1
1 4 (1-c1)*(1-u1)*(1-u1)
2 5 a1
2 6 (1-a1)
3 7 c2*r2
3 10 c2*(1-r2)
3 8 (1-c2)*u2*u2
3 9 (1-c2)*u2*(1-u2)
3 9 (1-c2)*(1-u2)*u2
3 10 (1-c2)*(1-u2)*(1-u2)
4 11 a2
4 12 (1-a2)
```

*Figure 3*

```
Participant 1
1 4
2 0
3 1
4 5
5 2
===
Participant 2
1 3
2 0
3 4
4 2
5 3
===
Participant 3
1 1
2 1
3 2
4 6
5 2
===
```

*Figure 4*

1	2	3	4	5
4	0	1	5	2
3	0	4	2	3
1	1	2	6	2

*Figure 5*

```

Model: 'example.eqn'
Data: 'example.dat'
No. parameters estimated: 17
-2*log(likelihood): 3269.782989
AIC: 3303.782989
BIC: 3398.036635
BIC_2: 3340.216279
Classes: 2
Class weights [lower and upper limit of 95% CI]:
  1 0.797800 [ 0.579607 1.015992 ]
  2 0.202200 [ 0.069785 0.334616 ]
Note: Confidence intervals are based on observed Fisher information.

Parameter values [lower and upper limit of 95% CI]
  1 a1 = 0.162160 [ 0.115325 0.208995 ] 0.380702 [ 0.257192 0.504212 ]
  2 a2 = 0.285430 [ 0.228396 0.342464 ] 0.616543 [ 0.484127 0.748959 ]
  3 c1 = 0.080480 [ -0.198806 0.359766 ] 0.614728 [ 0.327694 0.901763 ]
  4 c2 = 0.445815 [ 0.271692 0.619938 ] 0.826476 [ 0.745578 0.907374 ]
  5 r1 = 1.000000 [ -2.465266 4.465266 ] 0.505053 [ 0.251648 0.758457 ]
  6 r2 = 0.444845 [ 0.263794 0.625897 ] 0.677443 [ 0.579240 0.775646 ]
  7 u1 = 0.123130 [ 0.080010 0.166249 ] 0.287173 [ 0.061778 0.512568 ]
  8 u2 = 0.280127 [ 0.186896 0.373357 ] 0.677382 [ 0.459189 0.895574 ]
Note: Confidence intervals are based on observed Fisher information.

```

Figure 6

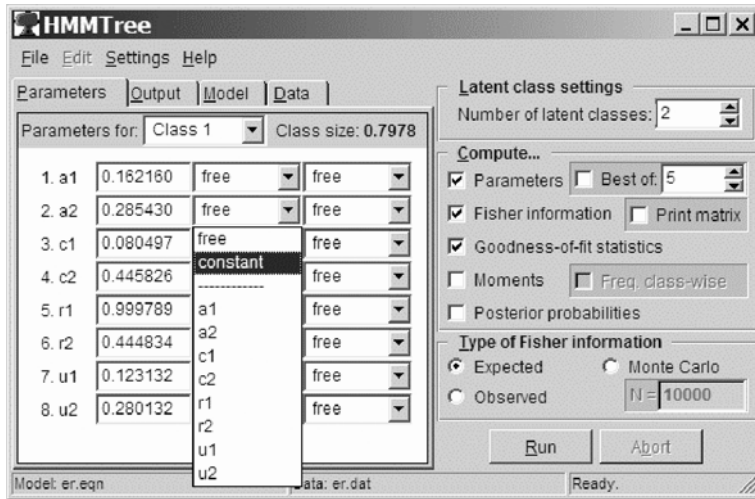


Figure 7

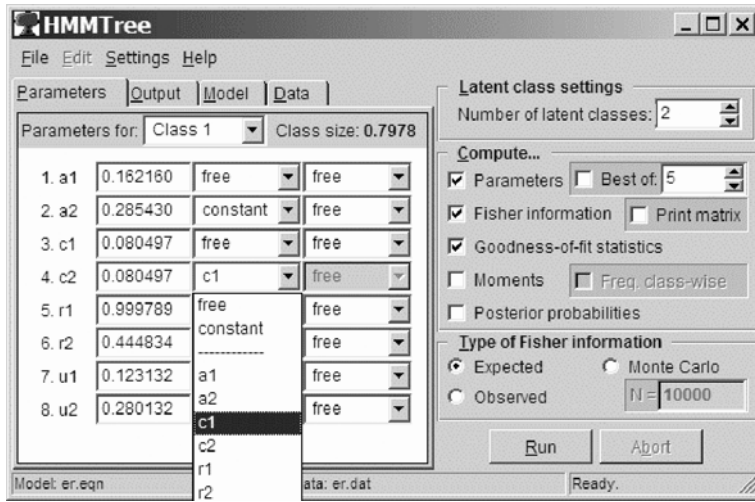


Figure 8

