Package 'TreeBUGS'

May 22, 2023

Version 1.5.0

Date 2023-05-21

Title Hierarchical Multinomial Processing Tree Modeling

Maintainer Daniel W. Heck <daniel.heck@uni-marburg.de>

Depends R (>= 4.0.0)

Imports Rcpp (>= 0.12.6), runjags, stats, graphics, utils, grDevices, coda, parallel, rjags, MASS, hypergeo, logspline

Suggests knitr, rmarkdown, testthat, R.rsp

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr, R.rsp

NeedsCompilation yes

SystemRequirements JAGS (https://mcmc-jags.sourceforge.io/)

Description User-friendly analysis of hierarchical multinomial processing tree (MPT) models that are often used in cognitive psychology. Implements the latent-trait MPT approach (Klauer, 2010) <DOI:10.1007/s11336-009-9141-0> and the beta-MPT approach (Smith & Batchelder, 2010) <DOI:10.1016/j.jmp.2009.06.007> to model heterogeneity of participants. MPT models are conveniently specified by an .eqn-file as used by other MPT software and data are provided by a .csv-file or directly in R. Models are either fitted by calling JAGS or by an MPT-tailored Gibbs sampler in C++ (only for nonhierarchical and beta MPT models). Provides tests of heterogeneity and MPT-tailored summaries and plotting functions. A detailed documentation is available in Heck, Arnold, & Arnold (2018) <DOI:10.3758/s13428-017-0869-7> and a tutorial on MPT modeling can be found in Schmidt, Erdfelder, & Heck (2022) <DOI:10.31234/osf.io/gh8md>.

License GPL-3

Encoding UTF-8

URL https://github.com/danheck/TreeBUGS

RoxygenNote 7.2.3

LazyData TRUE

Author Daniel W. Heck [aut, cre] (<https://orcid.org/0000-0002-6302-9252>),
Nina R. Arnold [aut, dtc],
Denis Arnold [aut],
Alexander Ly [ctb],
Marius Barth [ctb] (<https://orcid.org/0000-0002-3421-6665>)

Repository CRAN

Date/Publication 2023-05-21 22:40:02 UTC

R topics documented:

`TreeBUGS-package` *TreeBUGS: Hierarchical Multinomial Processing Tree Modeling*

Description

Uses standard MPT files in the .eqn-format (Moshagen, 2010) to fit hierarchical Bayesian MPT models. Note that the software JAGS is required (`https://mcmc-jags.sourceforge.io/`).

The core functions either fit a Beta-MPT model (`betaMPT`; Smith & Batchelder, 2010) or a latent-trait MPT model (`traitMPT`; Klauer, 2010). A fitted model can be inspected using convenient summary and plot functions tailored to hierarchical MPT models.

Detailed explanations and examples can be found in the package vignette, accessible via `vignette("TreeBUGS")`

Citation

If you use TreeBUGS, please cite the software as follows:

Heck, D. W., Arnold, N. R., & Arnold, D. (2018). TreeBUGS: An R package for hierarchical multinomial-processing-tree modeling. *Behavior Research Methods, 50*, 264–284. doi:10.3758/s1342801708697

Tutorial

For a tutorial on MPT modeling (including hierarchical modeling in TreeBUGS), see:

Schmidt, O., Erdfelder, E., & Heck, D. W. (2022). Tutorial on multinomial processing tree modeling: How to develop, test, and extend MPT models. `https://psyarxiv.com/gh8md/`

Author(s)

Daniel W. Heck, Denis Arnold, & Nina Arnold

References

Klauer, K. C. (2010). Hierarchical multinomial processing tree models: A latent-trait approach. *Psychometrika, 75*, 70-98. doi:10.1007/s1133600991410

Matzke, D., Dolan, C. V., Batchelder, W. H., & Wagenmakers, E.-J. (2015). Bayesian estimation of multinomial processing tree models with heterogeneity in participants and items. *Psychometrika, 80*, 205-235. doi:10.1007/s1133601393749

Moshagen, M. (2010). multiTree: A computer program for the analysis of multinomial processing tree models. *Behavior Research Methods, 42*, 42-54. doi:10.3758/BRM.42.1.42

Smith, J. B., & Batchelder, W. H. (2008). Assessing individual differences in categorical data. *Psychonomic Bulletin & Review, 15*, 713-731. doi:10.3758/PBR.15.4.713

Smith, J. B., & Batchelder, W. H. (2010). Beta-MPT: Multinomial processing tree models for addressing individual differences. *Journal of Mathematical Psychology, 54*, 167-183. doi:10.1016/j.jmp.2009.06.007

See Also

Useful links:

- https://github.com/danheck/TreeBUGS

`arnold2013` *Data of a Source-Monitoring Experiment*

Description

Dataset of a source-monitoring experiment by Arnold, Bayen, Kuhlmann, and Vaterrodt (2013) using a 2 (Source; within) x 3 (Expectancy; within) x 2 (Time of Schema Activation; between) mixed factorial design.

Usage

```
arnold2013
```

Format

A data frame 13 variables:

`subject` Participant code

`age` Age in years

`group` Between-subject factor "Time of Schema Activation": Retrieval vs. encoding condition

`pc` perceived contingency

`EE` Frequency of "Source E" responses to items from source "E"

`EU` Frequency of "Source U" responses to items from source "E"

`EN` Frequency of "New" responses to items from source "E"

`UE` Frequency of "Source E" responses to items from source "E"

`UU` Frequency of "Source U" responses to items from source "E"

`UN` Frequency of "New" responses to items from source "E"

`NE` Frequency of "Source E" responses to new items

`NU` Frequency of "Source U" responses to new items

`NN` Frequency of "New" responses to new items

Details

Eighty-four participants had to learn statements that were either presented by a doctor or a lawyer (Source) and were either typical for doctors, typical for lawyers, or neutral (Expectancy). These two types of statements were completely crossed in a balanced way, resulting in a true contingency of zero between Source and Expectancy. Whereas the profession schemata were activated at the time of encoding for half of the participants (encoding condition), the other half were told about the profession of the sources just before the test (retrieval condition). After the test, participants were asked to judge the contingency between item type and source (perceived contingency pc).

References

Arnold, N. R., Bayen, U. J., Kuhlmann, B. G., & Vaterrodt, B. (2013). Hierarchical modeling of contingency-based source monitoring: A test of the probability-matching account. Psychonomic Bulletin & Review, 20, 326-333.

Examples

```
head(arnold2013)

## Not run:
# fit hierarchical MPT model for encoding condition:
EQNfile <- system.file("MPTmodels/2htsm.eqn", package = "TreeBUGS")
d.encoding <- subset(arnold2013, group == "encoding", select = -(1:4))
fit <- betaMPTcpp(EQNfile, d.encoding,
  n.thin = 5,
  restrictions = list("D1=D2=D3", "d1=d2", "a=g")
)
# convergence
plot(fit, parameter = "mean", type = "default")
summary(fit)

## End(Not run)
```

`BayesFactorMPT` *Bayes Factors for Simple (Nonhierarchical) MPT Models*

Description

Computes Bayes factors for simple (fixed-effects, nonhierarchical) MPT models with beta distributions as priors on the parameters.

Usage

```
BayesFactorMPT(
  models,
  dataset = 1,
  resample,
  batches = 5,
```

```
  scale = 1,
  store = FALSE,
  cores = 1
)
```

Arguments

models	list of models fitted with simpleMPT, e.g., list(mod1, mod2)
dataset	for which data set should Bayes factors be computed?
resample	how many of the posterior samples of the MPT parameters should be resampled per model
batches	number of batches. Used to compute a standard error of the estimate.
scale	how much should posterior-beta approximations be downscaled to get fatter importance-sampling density
store	whether to save parameter samples
cores	number of CPUs used

Details

Currently, this is only implemented for a single data set!

Uses a Rao-Blackwellized version of the product-space method (Carlin & Chib, 1995) as proposed by Barker and Link (2013). First, posterior distributions of the MPT parameters are approximated by independent beta distributions. Second, for one a selected model, parameters are sampled from these proposal distributions. Third, the conditional probabilities to switch to a different model are computed and stored. Finally, the eigenvector with eigenvalue one of the matrix of switching probabilities provides an estimate of the posterior model probabilities.

References

Barker, R. J., & Link, W. A. (2013). Bayesian multimodel inference by RJMCMC: A Gibbs sampling approach. The American Statistician, 67(3), 150-156.

Carlin, B. P., & Chib, S. (1995). Bayesian model choice via Markov chain Monte Carlo methods. Journal of the Royal Statistical Society. Series B (Methodological), 57(3), 473-484.

See Also

marginalMPT

BayesFactorSlope *Bayes Factor for Slope Parameters in Latent-Trait MPT*

Description

Uses the Savage-Dickey method to compute the Bayes factor that the slope parameter of a continuous covariate in traitMPT is zero vs. positive/negative/unequal to zero.

Usage

```
BayesFactorSlope(
  fittedModel,
  parameter,
  direction = "!=",
  approx = "normal",
  plot = TRUE,
  ...
)
```

Arguments

fittedModel	a fitted latent-trait model fitted with traitMPT with predictor variables that have been defined via predStructure.
parameter	name of the slope parameter (e.g., "slope_d_covariate").
direction	alternative hypothesis: whether slope is smaller or larger than zero ("<" or ">") or unequal to zero ("!=").
approx	how to approximate the posterior density of the slope parameter at zero: approx="normal" uses a normal approximation to all samples and approx="logspline" uses a nonparametric density estimate of the package logspline. Usually, both methods provide similar results.
plot	if TRUE, the prior and posterior densities and the ratio at slope=0 are plotted.
...	further arguments passed to logspline, which is used to approximate the density of the posterior distribution.

Details

The Bayes factor is computed with the Savage-Dickey method, which is defined as the ratio of the density of the posterior and the density of the prior evaluated at slope=0 (Heck, 2019). Note that this method cannot be used with default JZS priors (IVprec="dgamma(.5,.5)") if more than one predictor is added for an MPT parameter. As a remedy, a g-prior (normal distribution) can be used on the slopes by setting the hyperprior parameter g to a fixed constant when fitting the model: traitMPT(...,IVprec = 1) (see Heck, 2019).

References

Heck, D. W. (2019). A caveat on the Savage-Dickey density ratio: The case of computing Bayes factors for regression parameters. *British Journal of Mathematical and Statistical Psychology, 72*, 316–333. doi:10.1111/bmsp.12150

Examples

```
## Not run:
# latent-trait MPT model for the encoding condition (see ?arnold2013):
EQNfile <- system.file("MPTmodels/2htsm.eqn", package = "TreeBUGS")
d.enc <- subset(arnold2013, group == "encoding")

fit <- traitMPT(EQNfile,
  data = d.enc[, -(1:4)], n.thin = 5,
  restrictions = list("D1=D2=D3", "d1=d2", "a=g"),
  covData = d.enc[, c("age", "pc")],
  predStructure = list("D1 ; age")
)
plot(fit, parameter = "slope", type = "default")
summary(fit)

BayesFactorSlope(fit, "slope_D1_age", direction = "<")

## End(Not run)
```

`betaMPT` *Fit a Hierarchical Beta-MPT Model*

Description

Fits a Beta-MPT model (Smith & Batchelder, 2010) based on a standard MPT model file (.eqn) and individual data table (.csv).

Usage

```
betaMPT(
  eqnfile,
  data,
  restrictions,
  covData,
  transformedParameters,
  corProbit = FALSE,
  alpha = "dgamma(1, 0.1)T(1,)",
  beta = "dgamma(1, 0.1)T(1,)",
  n.iter = 20000,
  n.adapt = 2000,
  n.burnin = 2000,
  n.thin = 5,
```

```
  n.chains = 3,
  dic = FALSE,
  ppp = 0,
  monitorIndividual = TRUE,
  modelfilename,
  parEstFile,
  posteriorFile,
  autojags = NULL,
  ...
)
```

Arguments

`eqnfile`	The (relative or full) path to the file that specifies the MPT model (standard .eqn syntax). Note that category labels must start with a letter (different to multiTree) and match the column names of `data`. Alternatively, the EQN-equations can be provided within R as a character value (cf. `readEQN`). Note that the first line of an .eqn-file is reserved for comments and always ignored.
`data`	The (relative or full) path to the .csv file with the data (comma separated; category labels in first row). Alternatively: a data frame or matrix (rows=individuals, columns = individual category frequencies, category labels as column names)
`restrictions`	Specifies which parameters should be (a) constant (e.g., `"a=b=.5"`) or (b) constrained to be identical (e.g., `"Do=Dn"`) or (c) treated as fixed effects (i.e., identical for all participants; `"a=b=FE"`). Either given as the path to a text file with restrictions per row or as a list of restrictions, e.g., `list("D1=D2","g=0.5")`. Note that numbers in .eqn-equations (e.g., `d*(1-g)*.50`) are directly interpreted as equality constraints.
`covData`	Data that contains covariates, for which correlations with individual MPT parameters will be sampled. Either the path to a .csv file (comma-separated: rows=individuals in the same order as `data`; first row must contain covariate labels). Alternatively: a data frame or matrix (rows=individuals, columns = variables; covariate labels as column names). Note that in `betaMPT`, correlations are computed for discrete variables that are coded numerically (in `traitMPT`, this can be suppressed by using `predType="f"`)
`transformedParameters`	list with parameter transformations that should be computed based on the posterior samples of the group-level means (e.g., for testing parameter differences: `list("diffD=Do-Dn")`), or path to a text file containing one transformation per line. Transformations of individual-level parameters can also be performed after fitting a model using `transformedParameters`.
`corProbit`	whether to use probit-transformed MPT parameters to compute correlations (probit-values of `+Inf` are truncated to `max(5,max(probit))`; similarly for `-Inf`). Default for beta-MPT: MPT parameters are used on the probability scale [0,1].
`alpha`	Hyperprior for the shape parameters α of the group-level beta distributions (in JAGS syntax). Default: Truncated gamma distributions for α and β with shape 1 and rate 0.1 and truncated to be larger than 1 (see plotPrior). A named vector

can be used to specify separate hyperpriors for each MPT parameter (if unnamed, the order of parameters is determined by the default order as shown by readEQN with paramOrder = TRUE). Originally, Smith and Batchelder (2008) used the "WinBUGS-zeros-trick" (available in TreeBUGS if alpha="zero" or beta="zero"), which approximates uniform priors on the group-level mean and SD (but often results convergence issues).

beta — Hyperprior for β of group-level distributions, see alpha.

n.iter — Number of iterations per chain (including burnin samples). See run.jags for details.

n.adapt — number of adaption samples to adjust MCMC sampler in JAGS. The sampler will be more efficient if it is tuned well. However, MCMC sampling will still give correct results even if the warning appears: "Adaptation incomplete." (this just means that sampling efficiency could be better).

n.burnin — Number of samples for burnin (samples will not be stored and removed from n.iter)

n.thin — Thinning rate.

n.chains — number of MCMC chains (sampled in parallel).

dic — whether to compute DIC using extract.runjags, which requires additional sampling. Can also be computed and added after fitting the model by fittedModel$summary$dic <- runjags::extract(fittedModel$runjags,"dic"). As an alternative information criterion, WAIC can be computed for fitted models.

ppp — number of samples to compute posterior predictive p-value (see posteriorPredictive)

monitorIndividual — whether to store MCMC samples of the MPT parameters theta at the individual level (i.e., the random effects). If FALSE, it is not possible to perform posterior-predictive checks.

modelfilename — name of the generated JAGS model file. Default is to write this information to the tempdir as required by CRAN standards.

parEstFile — Name of the file to with the estimates should be stored (e.g., "parEstFile.txt")

posteriorFile — path to RData-file where to save the model including MCMC posterior samples (an object named fittedModel; e.g., posteriorFile="mcmc.RData")

autojags — JAGS first fits the MPT model as usual and then draws MCMC samples repeatedly until convergence. For this, the function autoextend.jags is used with the arguments provided in autojags (this can be an empty list, in which case the defaults are used). Possible arguments for autoextend.jags are: list(startburnin = 1000, startsample = 5000, adapt = 2000,max.time="30m") (the last of these arguments restricts sampling to 30 minutes, see autoextend.jags).

... — further arguments to be passed to the JAGS sampling function (i.e., to run.jags. Note that reproducible results are obtained by setting a random seed before fitting a model (i.e., set.seed(12345)).

Details

Note that, in the Beta-MPT model, correlations of individual MPT parameters with covariates are sampled. Hence, the covariates do not affect the estimation of the actual Beta-MPT parameters.

Therefore, the correlation of covariates with the individual MPT parameters can equivalently be performed after fitting the model using the sampled posterior parameter values stored in `betaMPT$mcmc`

Value

a list of the class `betaMPT` with the objects:

- `summary`: MPT tailored summary. Use `summary(fittedModel)`
- `mptInfo`: info about MPT model (eqn and data file etc.)
- `runjags`: the object returned from the MCMC sampler. Note that the object `fittedModel$runjags` is an runjags object, whereas `fittedModel$runjags$mcmc` is a `mcmc.list` as used by the coda package (mcmc)

Author(s)

Daniel W. Heck, Nina R. Arnold, Denis Arnold

References

Heck, D. W., Arnold, N. R., & Arnold, D. (2018). TreeBUGS: An R package for hierarchical multinomial-processing-tree modeling. *Behavior Research Methods, 50*, 264–284. doi:10.3758/s1342801708697

Smith, J. B., & Batchelder, W. H. (2010). Beta-MPT: Multinomial processing tree models for addressing individual differences. *Journal of Mathematical Psychology, 54*, 167-183. doi:10.1016/j.jmp.2009.06.007

Examples

```
## Not run:
# fit beta-MPT model for encoding condition (see ?arnold2013):
EQNfile <- system.file("MPTmodels/2htsm.eqn", package = "TreeBUGS")
d.encoding <- subset(arnold2013, group == "encoding", select = -(1:4))
fit <- betaMPT(EQNfile, d.encoding,
  n.thin = 5,
  restrictions = list("D1=D2=D3", "d1=d2", "a=g")
)
# convergence
plot(fit, parameter = "mean", type = "default")
summary(fit)

## End(Not run)
```

betaMPTcpp *C++ Sampler for Hierarchical Beta-MPT Model*

Description

Fast Gibbs sampler in C++ that is tailored to the beta-MPT model.

Usage

```
betaMPTcpp(
  eqnfile,
  data,
  restrictions,
  covData,
  corProbit = FALSE,
  n.iter = 20000,
  n.burnin = 2000,
  n.thin = 5,
  n.chains = 3,
  ppp = 0,
  shape = 1,
  rate = 0.1,
  parEstFile,
  posteriorFile,
  cores = 1
)
```

Arguments

eqnfile	The (relative or full) path to the file that specifies the MPT model (standard .eqn syntax). Note that category labels must start with a letter (different to multiTree) and match the column names of data. Alternatively, the EQN-equations can be provided within R as a character value (cf. readEQN). Note that the first line of an .eqn-file is reserved for comments and always ignored.
data	The (relative or full) path to the .csv file with the data (comma separated; category labels in first row). Alternatively: a data frame or matrix (rows=individuals, columns = individual category frequencies, category labels as column names)
restrictions	Specifies which parameters should be (a) constant (e.g., "a=b=.5") or (b) constrained to be identical (e.g., "Do=Dn") or (c) treated as fixed effects (i.e., identical for all participants; "a=b=FE"). Either given as the path to a text file with restrictions per row or as a list of restrictions, e.g., list("D1=D2","g=0.5"). Note that numbers in .eqn-equations (e.g., d*(1-g)*.50) are directly interpreted as equality constraints.
covData	Data that contains covariates, for which correlations with individual MPT parameters will be sampled. Either the path to a .csv file (comma-separated: rows=individuals in the same order as data; first row must contain covariate

	labels). Alternatively: a data frame or matrix (rows=individuals, columns = variables; covariate labels as column names). Note that in `betaMPT`, correlations are computed for discrete variables that are coded numerically (in `traitMPT`, this can be suppressed by using `predType="f"`)
`corProbit`	whether to use probit-transformed MPT parameters to compute correlations (probit-values of `+Inf` are truncated to `max(5,max(probit))`; similarly for `-Inf`). Default for beta-MPT: MPT parameters are used on the probability scale [0,1].
`n.iter`	Number of iterations per chain (including burnin samples). See `run.jags` for details.
`n.burnin`	Number of samples for burnin (samples will not be stored and removed from n.iter)
`n.thin`	Thinning rate.
`n.chains`	number of MCMC chains (sampled in parallel).
`ppp`	number of samples to compute posterior predictive p-value (see `posteriorPredictive`)
`shape`	shape parameter(s) of Gamma-hyperdistribution for the hierarchical beta-parameters α_s and β_s (can be a named vector to provide different hyperpriors for each parameter)
`rate`	rate parameter(s) of Gamma-hyperdistribution
`parEstFile`	Name of the file to with the estimates should be stored (e.g., "parEstFile.txt")
`posteriorFile`	path to RData-file where to save the model including MCMC posterior samples (an object named `fittedModel`; e.g., `posteriorFile="mcmc.RData"`)
`cores`	number of CPUs to be used

Author(s)

Daniel Heck

Examples

```
## Not run:
# fit beta-MPT model for encoding condition (see ?arnold2013):
EQNfile <- system.file("MPTmodels/2htsm.eqn", package = "TreeBUGS")
d.encoding <- subset(arnold2013, group == "encoding", select = -(1:4))
fit <- betaMPTcpp(EQNfile, d.encoding,
  n.thin = 5,
  restrictions = list("D1=D2=D3", "d1=d2", "a=g")
)
# convergence
plot(fit, parameter = "mean", type = "default")
summary(fit)

## End(Not run)
```

betweenSubjectMPT *Between-Subject Comparison of Parameters*

Description

Computes differencesor other statistics of MPT parameters for two hierarchical MPT models fitted separately to between-subjects data

Usage

```
betweenSubjectMPT(
  model1,
  model2,
  par1,
  par2 = par1,
  stat = c("x-y", "x<y"),
  plot = FALSE
)
```

Arguments

model1	fitted hierarchical MPT model for first between-subjects condition
model2	fitted hierarchical MPT model for second between-subjects condition
par1	label of parameter from first model for which statistic should be computed
par2	label of parameter from second model. Default: The same parameter as in the first model
stat	one or more functions of the parameters using "x" and "y" as placeholders for the parameters from the first and second model, respectively. Default: Compute (A) the difference between parameters and (B) a Bayesian p-value (by counting how often x<y).
plot	whether to plot the convergence of the difference in parameters

Value

a list of the class betweenMPT with the values:

- summary: Summary for parameter difference
- mptInfo1, mptInfo2: info about MPT models (eqn and data file etc.)
- mcmc: the MCMC samples of the differences in parameters

Author(s)

Daniel Heck

`correlationPosterior` *Posterior Distribution for Correlations*

Description

Adjusts the posterior distribution of correlations for the sampling error of a population correlation according to the sample size (i.e., the number of participants; Ly, Marsman, & Wagenmakers, 2018).

Usage

```
correlationPosterior(
  fittedModel,
  r,
  N,
  kappa = 1,
  ci = 0.95,
  M = 1000,
  precision = 0.005,
  maxiter = 10000,
  plot = TRUE,
  nCPU = 4
)
```

Arguments

`fittedModel`	a fitted betaMPT or traitMPT model with covariates (added during fitting by the argument `covData`)
`r`	optional: a vector of posterior correlations (instead of `fittedModel`)
`N`	only if `r` is used: the number of participants the correlation is based on
`kappa`	parameter for the prior of the correlation, that is, a scaled beta distribution: Beta(1/kappa, 1/kappa). The default `kappa=1` defines a uniform distribution on [-1,1], whereas `kappa<1` defines a unimodal prior centered around zero.
`ci`	credibility interval
`M`	number of subsamples from the fitted model
`precision`	precision on the interval [-1,1] to approximate the posterior density
`maxiter`	maximum number of iterations in `genhypergeo`. Higher values might be necessary to increase numerical stability for large correlations (r>.95).
`plot`	whether to plot (a) the unadjusted posterior correlations (gray histogram) and (b) the corrected posterior (black line with red credibility intervals)
`nCPU`	number of CPUs used for parallel computation of posterior distribution

Details

This function (1) uses all posterior samples of a correlation to (2) derive the posterior of the correlation corrected for sampling error and (3) averages these densities across the posterior samples. Thereby, the method accounts for estimation uncertainty of the MPT model (due to the use of the posterior samples) and also for sampling error of the population correlation due to sample size (cf. Ly, Boehm, Heathcote, Turner, Forstmann, Marsman, & Matzke, 2016).

Author(s)

Daniel W. Heck, Alexander Ly

References

Ly, A., Marsman, M., & Wagenmakers, E.-J. (2018). Analytic posteriors for Pearson's correlation coefficient. *Statistica Neerlandica, 72*, 4–13. doi:10.1111/stan.12111

Ly, A., Boehm, U., Heathcote, A., Turner, B. M. , Forstmann, B., Marsman, M., and Matzke, D. (2017). A flexible and efficient hierarchical Bayesian approach to the exploration of individual differences in cognitive-model-based neuroscience. https://osf.io/evsyv/. doi:10.1002/9781119159193

Examples

```
# test effect of number of participants:
set.seed(123)
cors <- rbeta(50, 100, 70)
correlationPosterior(r = cors, N = 10, nCPU = 1)
correlationPosterior(r = cors, N = 100, nCPU = 1)
```

`extendMPT` *Extend MCMC Sampling for MPT Model*

Description

Adds more MCMC samples to the fitted MPT model.

Usage

```
extendMPT(fittedModel, n.iter = 10000, n.adapt = 1000, n.burnin = 0, ...)
```

Arguments

`fittedModel`	a fitted `traitMPT` or `betaMPT`
`n.iter`	Number of iterations per chain (including burnin samples). See `run.jags` for details.

n.adapt	number of adaption samples to adjust MCMC sampler in JAGS. The sampler will be more efficient if it is tuned well. However, MCMC sampling will still give correct results even if the warning appears: "Adaptation incomplete." (this just means that sampling efficiency could be better).
n.burnin	Number of samples for burnin (samples will not be stored and removed from n.iter)
...	further arguments passed to `extend.jags` (see arguments listed in: run.jags). When drawing more samples, JAGS requires an additional adaptation phase, in which the MCMC sampling procedure is adjusted. Note that the MCMC sampling will still give correct results even if the warning appears: "Adaptation incomplete." (this just means that sampling efficiency is not optimal).

`genBetaMPT` *Generate Data for Beta MPT Models*

Description

Generating a data file with known parameter structure using the Beta-MPT. Useful for simulations and robustness checks.

Usage

```
genBetaMPT(
  N,
  numItems,
  eqnfile,
  restrictions,
  mean = NULL,
  sd = NULL,
  alpha = NULL,
  beta = NULL,
  warning = TRUE
)
```

Arguments

N	number of participants
numItems	number of responses per tree (a named vector with tree labels)
eqnfile	The (relative or full) path to the file that specifies the MPT model (standard .eqn syntax). Note that category labels must start with a letter (different to multiTree) and match the column names of `data`. Alternatively, the EQN-equations can be provided within R as a character value (cf. `readEQN`). Note that the first line of an .eqn-file is reserved for comments and always ignored.

restrictions	Specifies which parameters should be (a) constant (e.g., "a=b=.5") or (b) constrained to be identical (e.g., "Do=Dn") or (c) treated as fixed effects (i.e., identical for all participants; "a=b=FE"). Either given as the path to a text file with restrictions per row or as a list of restrictions, e.g., list("D1=D2","g=0.5"). Note that numbers in .eqn-equations (e.g., d*(1-g)*.50) are directly interpreted as equality constraints.
mean	Named vector of true group means of individual MPT parameters. If the vector is not named, the internal order of parameters is used (can be obtained using readEQN).
sd	named vector of group standard deviations of individual MPT parameters.
alpha	Alternative specification of the group-level distribution using the shape parameters of the beta distribution (see dbeta).
beta	see alpha
warning	whether to show warning in case the naming of data-generating parameters are unnamed or do not match

Details

Data are generated in a two-step procedure. First, person parameters are sampled from the specified beta distributions for each paramter (either based on mean/sd or based on alpha/beta). In a second step, response frequencies are sampled for each person using genMPT.

Value

a list including the generated frequencies (data) and the true, underlying parameters (parameters) on the group and individual level.

References

Smith, J. B., & Batchelder, W. H. (2010). Beta-MPT: Multinomial processing tree models for addressing individual differences. Journal of Mathematical Psychology, 54, 167-183.

See Also

genMPT

Examples

```
# Example: Standard Two-High-Threshold Model (2HTM)
EQNfile <- system.file("MPTmodels/2htm.eqn", package = "TreeBUGS")
genDat <- genBetaMPT(
  N = 100,
  numItems = c(Target = 250, Lure = 250),
  eqnfile = EQNfile,
  mean = c(Do = .7, Dn = .5, g = .5),
  sd = c(Do = .1, Dn = .1, g = .05)
)
head(genDat$data, 3)
plotFreq(genDat$data, eqn = EQNfile)
```

genMPT *Generate MPT Frequencies*

Description

Uses a matrix of individual MPT parameters to generate MPT frequencies.

Usage

```
genMPT(theta, numItems, eqnfile, restrictions, warning = TRUE)
```

Arguments

theta	matrix of MPT parameters (rows: individuals; columns: parameters). Parameters are assigned by column names of the matrix. all of the parameters in the model file need to be included.
numItems	number of responses per tree (a named vector with tree labels)
eqnfile	The (relative or full) path to the file that specifies the MPT model (standard .eqn syntax). Note that category labels must start with a letter (different to multiTree) and match the column names of data. Alternatively, the EQN-equations can be provided within R as a character value (cf. readEQN). Note that the first line of an .eqn-file is reserved for comments and always ignored.
restrictions	Specifies which parameters should be (a) constant (e.g., "a=b=.5") or (b) constrained to be identical (e.g., "Do=Dn") or (c) treated as fixed effects (i.e., identical for all participants; "a=b=FE"). Either given as the path to a text file with restrictions per row or as a list of restrictions, e.g., list("D1=D2","g=0.5"). Note that numbers in .eqn-equations (e.g., d*(1-g)*.50) are directly interpreted as equality constraints.
warning	whether to show warning in case the naming of data-generating parameters are unnamed or do not match

See Also

genTraitMPT and genBetaMPT to generate data for latent normal/beta hierarchical distributions.

Examples

```
# Example: Standard Two-High-Threshold Model (2HTM)
EQNfile <- system.file("MPTmodels/2htm.eqn", package = "TreeBUGS")
theta <- matrix(
  c(
    .8, .4, .5,
    .6, .3, .4
  ),
  nrow = 2, byrow = TRUE,
  dimnames = list(NULL, c("Do", "Dn", "g"))
)
```

```
genDat <- genMPT(
  theta, c(Target = 250, Lure = 250),
  EQNfile
)
genDat
```

---

genTraitMPT *Generate Data for Latent-Trait MPT Models*

Description

Generating a data set with known parameter structure using the Trait-MPT. Useful for simulations and robustness checks.

Usage

```
genTraitMPT(
  N,
  numItems,
  eqnfile,
  restrictions,
  mean,
  mu,
  sigma,
  rho,
  warning = TRUE
)
```

Arguments

N	number of participants
numItems	number of responses per tree (a named vector with tree labels)
eqnfile	The (relative or full) path to the file that specifies the MPT model (standard .eqn syntax). Note that category labels must start with a letter (different to multiTree) and match the column names of data. Alternatively, the EQN-equations can be provided within R as a character value (cf. readEQN). Note that the first line of an .eqn-file is reserved for comments and always ignored.
restrictions	Specifies which parameters should be (a) constant (e.g., "a=b=.5") or (b) constrained to be identical (e.g., "Do=Dn") or (c) treated as fixed effects (i.e., identical for all participants; "a=b=FE"). Either given as the path to a text file with restrictions per row or as a list of restrictions, e.g., list("D1=D2","g=0.5"). Note that numbers in .eqn-equations (e.g., d*(1-g)*.50) are directly interpreted as equality constraints.
mean	named vector of data-generating group means of the individual MPT parameters on the probability scale. If the vector is not named, the internal order of parameters is used (can be obtained using readEQN).

mu	an alternative way to define the group-level means on the latent-probit scale (i.e., mu = qnorm(mean) or equivalently, mean = pnorm(mu)).
sigma	(named) vector of group standard deviations of individual MPT parameters on the latent probit scale. Default is zero (no person heterogeneity).
rho	(named) correlation matrix for individual MPT parameters on the latent probit scale. Must be symmetric and positive definite (e.g., no correlations of 1 or -1 allowed). Default: a diagonal matrix (i.e., zero correlations).
warning	whether to show warning in case the naming of data-generating parameters are unnamed or do not match

Details

This functions implements a two-step sampling procedure. First, the person parameters on the latent probit-scale are sampled from the multivariate normal distribution (based on the mean mu = qnorm(mean), the standard deviations sigma, and the correlation matrix rho). These person parameters are then transformed to the probability scale using the probit-link. In a last step, observed frequencies are sampled for each person using the MPT equations.

Note that the user can generate more complex structures for the latent person parameters, and then supply these person parameters to the function genMPT.

Value

a list including the generated frequencies per person (data) and the sampled individual parameters (parameters) on the probit and probability scale (thetaLatent and theta, respectively).

References

Klauer, K. C. (2010). Hierarchical multinomial processing tree models: A latent-trait approach. Psychometrika, 75, 70-98.

See Also

genMPT

Examples

```
# Example: Standard Two-High-Threshold Model (2HTM)
EQNfile <- system.file("MPTmodels/2htm.eqn", package = "TreeBUGS")
rho <- matrix(c(
  1, .8, .2,
  .8, 1, .1,
  .2, .1, 1
), nrow = 3)
colnames(rho) <- rownames(rho) <- c("Do", "Dn", "g")
genDat <- genTraitMPT(
  N = 100,
  numItems = c(Target = 250, Lure = 250),
  eqnfile = EQNfile,
  mean = c(Do = .7, Dn = .7, g = .5),
  sigma = c(Do = .3, Dn = .3, g = .15),
```

```
  rho = rho
)
head(genDat$data, 3)
plotFreq(genDat$data, eqn = EQNfile)
```

`getGroupMeans` *Get Mean Parameters per Group*

Description

For hierarchical latent-trait MPT models with discrete predictor variables as fitted with `traitMPT(..., predStructure = list("f"))`.

Usage

```
getGroupMeans(
  traitMPT,
  factor = "all",
  probit = FALSE,
  file = NULL,
  mcmc = FALSE
)
```

Arguments

`traitMPT`	a fitted `traitMPT` model
`factor`	whether to get group estimates for all combinations of factor levels (default) or only for specific factors (requires the names of the covariates in covData)
`probit`	whether to use probit scale or probability scale
`file`	filename to export results in .csv format (e.g., `file="fit_group.csv"`)
`mcmc`	if `TRUE`, the raw MCMC samples for the group means are returned as an `mcmc.list` object. This allows pairwise tests of group means (see `transformedParameters`).

Author(s)

Daniel Heck

See Also

`getParam` for parameter estimates

Examples

```
## Not run:
# save group means (probability scale):
getGroupMeans(traitMPT, file = "groups.csv")

## End(Not run)
```

getParam *Get Parameter Posterior Statistics*

Description

Returns posterior statistics (e.g., mean, median) for the parameters of a hierarchical MPT model.

Usage

```
getParam(fittedModel, parameter = "mean", stat = "mean", file = NULL)
```

Arguments

fittedModel	a fitted latent-trait MPT model (see traitMPT) or beta MPT model (see betaMPT)
parameter	which parameter(s) of the (hierarchical) MPT model should be returned? (see details in getParam).
stat	whether to get the posterior "mean", "median", "sd", or "summary" (includes mean, SD, and 95% credibility interval)
file	filename to export results in .csv format (e.g., file="est_param.csv")

Details

This function is a convenient way to get the information stored in fittedModel$mcmc.summ.

The latent-trait MPT includes the following parameters:

- "mean" (group means on probability scale)
- "mu" (group means on probit scale)
- "sigma" (SD on probit scale)
- "rho" (correlations on probit scale)
- "theta" (individual MPT parameters)

The beta MPT includes the following parameters:

- "mean" (group means on probability scale)
- "sd" (SD on probability scale)
- "alph","bet" (group parameters of beta distribution)
- "theta" (individual MPT parameters)

Author(s)

Daniel Heck

See Also

getGroupMeans mean group estimates

Examples

```
## Not run:
# mean estimates per person:
getParam(fittedModel, parameter = "theta")

# save summary of individual estimates:
getParam(fittedModel,
  parameter = "theta",
  stat = "summary", file = "ind_summ.csv"
)

## End(Not run)
```

`getSamples` *Get Posterior Samples from Fitted MPT Model*

Description

Extracts MCMC posterior samples as an `coda::mcmc.list` and relabels the MCMC variables.

Usage

```
getSamples(
  fittedModel,
  parameter = "mean",
  select = "all",
  names = "par_label"
)
```

Arguments

`fittedModel`	a fitted latent-trait MPT model (see `traitMPT`) or beta MPT model (see `betaMPT`)
`parameter`	which parameter(s) of the (hierarchical) MPT model should be returned? (see details in `getParam`).
`select`	character vector of parameters to be plotted (e.g., `select = c("d", "g")`. Can be used to plot subsets of parameters and change the order of parameters.
`names`	whether and how to rename the variables in the MCMC output: `par` (internal parameter labels such as `mu[1]`), `label` (MPT label from EQN file such as `"d"`), or `par_name` (concatenation of both such as `"mu[1]_d"`).

Examples

```
## Not run:
getSamples(fittedModel, "mu", select = c("d", "g"))

## End(Not run)
```

`marginalMPT` *Marginal Likelihood for Simple MPT*

Description

Computes the marginal likelihood for simple (fixed-effects, nonhierarchical) MPT models.

Usage

```
marginalMPT(
  eqnfile,
  data,
  restrictions,
  alpha = 1,
  beta = 1,
  dataset = 1,
  method = "importance",
  posterior = 500,
  mix = 0.05,
  scale = 0.9,
  samples = 10000,
  batches = 10,
  show = TRUE,
  cores = 1
)
```

Arguments

`eqnfile`	The (relative or full) path to the file that specifies the MPT model (standard .eqn syntax). Note that category labels must start with a letter (different to multiTree) and match the column names of `data`. Alternatively, the EQN-equations can be provided within R as a character value (cf. `readEQN`). Note that the first line of an .eqn-file is reserved for comments and always ignored.
`data`	The (relative or full) path to the .csv file with the data (comma separated; category labels in first row). Alternatively: a data frame or matrix (rows=individuals, columns = individual category frequencies, category labels as column names)
`restrictions`	Specifies which parameters should be (a) constant (e.g., `"a=b=.5"`) or (b) constrained to be identical (e.g., `"Do=Dn"`) or (c) treated as fixed effects (i.e., identical for all participants; `"a=b=FE"`). Either given as the path to a text file with restrictions per row or as a list of restrictions, e.g., `list("D1=D2","g=0.5")`. Note that numbers in .eqn-equations (e.g., `d*(1-g)*.50`) are directly interpreted as equality constraints.
`alpha`	first shape parameter(s) for the beta prior-distribution of the MPT parameters θ_s (can be a named vector to use a different prior for each MPT parameter)
`beta`	second shape parameter(s)

dataset	for which data set should Bayes factors be computed?
method	either "importance" (importance sampling using a mixture of uniform and beta-aproximation of the posterior) or "prior" (brute force Monte Carlo sampling from prior)
posterior	number of posterior samples used to approximate importance-sampling densities (i.e., beta distributions)
mix	mixture proportion of the uniform distribution for the importance-sampling density
scale	how much should posterior-beta approximations be downscaled to get fatter importance-sampling density
samples	total number of samples from parameter space
batches	number of batches. Used to compute a standard error of the estimate.
show	whether to show progress
cores	number of CPUs used

Details

Currently, this is only implemented for a single data set!

If method = "prior", a brute-force Monte Carlo method is used and parameters are directly sampled from the prior.Then, the likelihood is evaluated for these samples and averaged (fast, but inefficient).

Alternatively, an importance sampler is used if method = "importance", and the posterior distributions of the MPT parameters are approximated by independent beta distributions. Then each parameter s is sampled from the importance density:

$mix * U(0,1) + (1 - mix) * Beta(scale * a_s, scale * b_s)$

References

Vandekerckhove, J. S., Matzke, D., & Wagenmakers, E. (2015). Model comparison and the principle of parsimony. In Oxford Handbook of Computational and Mathematical Psychology (pp. 300-319). New York, NY: Oxford University Press.

See Also

BayesFactorMPT

Examples

```
# 2-High-Threshold Model
eqn <- "## 2HTM ##
   Target  Hit  d
   Target  Hit  (1-d)*g
   Target  Miss (1-d)*(1-g)
   Lure    FA   (1-d)*g
   Lure    CR   (1-d)*(1-g)
   Lure    CR   d"
data <- c(
```

```
  Hit = 46, Miss = 14,
  FA = 14, CR = 46
)

# weakly informative prior for guessing
aa <- c(d = 1, g = 2)
bb <- c(d = 1, g = 2)
curve(dbeta(x, aa["g"], bb["g"]))

# compute marginal likelihood
htm <- marginalMPT(eqn, data,
  alpha = aa, beta = bb,
  posterior = 200, samples = 1000
)
# second model: g=.50
htm.g50 <- marginalMPT(eqn, data, list("g=.5"),
  alpha = aa, beta = bb,
  posterior = 200, samples = 1000
)

# Bayes factor
# (per batch to get estimation error)
bf <- htm.g50$p.per.batch / htm$p.per.batch
mean(bf) # BF
sd(bf) / sqrt(length(bf)) # standard error of BF estimate
```

`plot.betaMPT` *Plot Convergence for Hierarchical MPT Models*

Description

Plot Convergence for Hierarchical MPT Models

Usage

```
## S3 method for class 'betaMPT'
plot(x, parameter = "mean", type = "default", ...)

## S3 method for class 'simpleMPT'
plot(x, type = "default", ...)

## S3 method for class 'traitMPT'
plot(x, parameter = "mean", type = "default", ...)
```

Arguments

x	fitted hierarchical MPT model (traitMPT, betaMPT)

parameter	which parameter to plot (e.g., "theta", "mean", "rho", "slope"). Parameters are matched partially, in order to plot all entries of vector valued parameters (see getParam to get a list of parameters). Moreover, parameter labels can be used, e.g., "theta[D]" or "rho[D,g]"
type	type of convergence plot. Can be one of "default" (trace+density), "acf" (auto-correlation function), "trace", "autocorr", "crosscorr","density", "gelman". See plotting functions in the coda package (plot.mcmc.list, acfplot, traceplot, autocorr.plot, crosscorr.plot, densplot, gelman.plot).
...	further arguments passed to the plotting functions in coda

Methods (by class)

- plot(betaMPT): Plot convergence for beta MPT
- plot(simpleMPT): Plot convergence for nonhierarchical MPT model
- plot(traitMPT): Plot convergence for latent-trait MPT

---

plotDistribution *Plot Distribution of Individual Estimates*

Description

Plots histograms of the posterior-means of individual MPT parameters against the group-level distribution given by the posterior-mean of the hierarchical parameters (e.g., the beta distribution in case of the beta-MPT)

Usage

```
plotDistribution(fittedModel, scale = "probability", ...)
```

Arguments

fittedModel	fitted latent-trait or beta MPT model (traitMPT, betaMPT)
scale	only for latent-trait MPT: should estimates be plotted on the "latent" or the "probability" scale (i.e., as MPT parameters). Can be abbreviated by "l" and "p".
...	further arguments passed to hist (e.g., breaks=50 to get a more fine-grained histogram)

Details

For the latent-trait MPT, differences due to continuous predictors or discrete factors are currently not considered in the group-level predictions (red density). Under such a model, individual estimates are not predicted to be normally distributed on the latent scale as shown in the plot.

See Also

plot.traitMPT

plotFit *Plot Posterior Predictive Mean Frequencies*

Description

Plots observed means/covariances of individual frequencies against the means/covariances sampled from the posterior distribution (posterior predictive distribution).

Usage

```
plotFit(fittedModel, M = 1000, stat = "mean", ...)
```

Arguments

fittedModel	fitted latent-trait or beta MPT model (traitMPT, betaMPT)
M	number of posterior predictive samples. As a maximum, the number of posterior samples in fittedModel is used.
stat	whether to plot mean frequencies ("mean") or covariances of individual frequencies ("cov")
...	arguments passed to boxplot

Details

If posterior predictive p-values were computed when fitting the model (e.g., by adding the argument traitMPT(...,ppp=1000)), the stored posterior samples are re-used for plotting. Note that the last category in each MPT tree is dropped, because one category per multinomial distribution is fixed.

Examples

```
## Not run:
# add posterior predictive samples to fitted model (optional step)
fittedModel$postpred$freq.pred <-
  posteriorPredictive(fittedModel, M = 1000)

# plot model fit
plotFit(fittedModel, stat = "mean")

## End(Not run)
```

plotFreq *Plot Raw Frequencies*

Description

Plot observed individual and mean frequencies.

Usage

```
plotFreq(x, freq = TRUE, select = "all", boxplot = TRUE, eqnfile, ...)
```

Arguments

x	either a fitted hierarchical MPT model (see traitMPT, betaMPT); or a matrix/data frame of response frequencies (can be provided as a path to a .csv-file with individual frequencies).
freq	whether to plot absolute frequencies or relative frequencies (which sum up to one within each tree; only if x is a hierarchical model or if eqnfile is provided)
select	a numeric vector with participant indices to select which raw frequencies to plot (default: "all")
boxplot	if FALSE, lines and points are drawn instead of boxplots
eqnfile	optional: EQN description of an MPT model, that is, either the path to an EQN file or as a character string (only used if x refers to a matrix/data frame or .csv-file)
...	further arguments passed to boxplot and plot

Examples

```
# get frequency data and EQN file
freq <- subset(arnold2013, group == "encoding", select = -(1:4))
eqn <- system.file("MPTmodels/2htsm.eqn", package = "TreeBUGS")
plotFreq(freq, eqnfile = eqn)
plotFreq(freq, freq = FALSE, eqnfile = eqn)
```

plotParam *Plot Parameter Estimates*

Description

Plot parameter estimates for hierarchical MPT models.

Usage

```
plotParam(
  x,
  includeIndividual = TRUE,
  addLines = FALSE,
  estimate = "mean",
  select = "all",
  ...
)
```

Arguments

x	a fitted Beta or latent-trait MPT model
includeIndividual	whether to plot individual estimates
addLines	whether to connect individual parameter estimates by lines
estimate	type of point estimates for group-level and individual parameters (either "mean" or "median")
select	character vector of parameters to be plotted (e.g., select = c("d", "g"). Can be used to plot subsets of parameters and change the order of parameters.
...	further arguments passed to the standard plot function

Author(s)

Daniel Heck

See Also

betaMPT, traitMPT, plotDistribution

Examples

```
## Not run:
plotParam(fit,
  addLines = TRUE,
  estimate = "median",
  select = c("d1", "d2")
)

## End(Not run)
```

plotPrior *Plot Prior Distributions*

Description

Plots prior distributions for group means, standard deviation, and correlations of MPT parameters across participants.

Usage

```
plotPrior(prior, probitInverse = "mean", M = 5000, nCPU = 3, ...)
```

Arguments

prior	a named list defining the priors. For the traitMPT, the default is `list(mu = "dnorm(0,1)", xi="dunif(0,10)", V=diag(S),df=S+1)`, where S is the number of free parameters. For the betaMPT, the default is `list(alpha ="dgamma(1,.1)", beta = "dgamma(1,.1)")`. Note that the normal distribution `"dnorm(mu,prec)"` is parameterized as in JAGS by the mean and precision (= 1/variance).
probitInverse	which latent-probit parameters (for `traitMPT` model) to transform to probability scale. Either `"none"`, `"mean"` (simple transformation $\Phi(\mu)$), or `"mean_sd"` (see `probitInverse`)
M	number of random samples to approximate priors of group-level parameters
nCPU	number of CPUs used for parallel sampling. For large models and many participants, this may require a lot of memory.
...	further arguments passed to `plot`

Details

This function samples from a set of hyperpriors (either for hierarchical traitMPT or betaMPT structure) to approximate the implied prior distributions on the parameters of interest (group-level mean, SD, and correlations of MPT parameters). Note that the normal distribution `"dnorm(mu,prec)"` is parameterized as in JAGS by the mean and precision (= 1/variance).

See Also

`priorPredictive`

Examples

```
## Not run:
# default priors for traitMPT:
plotPrior(list(
  mu = "dnorm(0, 1)",
  xi = "dunif(0, 10)",
  V = diag(2),
  df = 2 + 1
```

```
), M = 4000)

# default priors for betaMPT:
plotPrior(list(
  alpha = "dgamma(1, 0.1)",
  beta = "dgamma(1, 0.1)"
), M = 4000)

## End(Not run)
```

`plotPriorPost` *Plot Prior vs. Posterior Distribution*

Description

Allows to judge how much the data informed the parameter posterior distributions compared to the prior.

Usage

```
plotPriorPost(
  fittedModel,
  probitInverse = "mean",
  M = 2e+05,
  ci = 0.95,
  nCPU = 3,
  ...
)
```

Arguments

`fittedModel`	fitted latent-trait or beta MPT model (`traitMPT`, `betaMPT`)
`probitInverse`	which latent-probit parameters (for `traitMPT` model) to transform to probability scale. Either `"none"`, `"mean"` (simple transformation $\Phi(\mu)$), or `"mean_sd"` (see `probitInverse`)
`M`	number of random samples to approximate prior distributions
`ci`	credibility interval indicated by vertical red lines
`nCPU`	number of CPUs used for parallel sampling. For large models and many participants, this may require a lot of memory.
`...`	arguments passed to `boxplot`

Details

Prior distributions are shown as blue, dashed lines, whereas posterior distributions are shown as solid, black lines.

`posteriorPredictive` *Get Posterior Predictive Samples*

Description

Draw predicted frequencies based on posterior distribution of (a) individual estimates (default) or (b) for a new participant (if `numItems` is provided; does not consider continuous or discrete predictors in traitMPT).

Usage

```
posteriorPredictive(
  fittedModel,
  M = 100,
  numItems = NULL,
  expected = FALSE,
  nCPU = 4
)
```

Arguments

`fittedModel`	fitted latent-trait or beta MPT model (`traitMPT`, `betaMPT`)
`M`	number of posterior predictive samples. As a maximum, the number of posterior samples in `fittedModel` is used.
`numItems`	optional: a vector with the number of items per MPT tree to sample predicted data for a new participant (first, a participant vector θ is sampled from the hierarchical posterior; second, frequencies are generated).
`expected`	if `TRUE`, the expected frequencies per person are returned (without additional sampling from a multinomial distribution)
`nCPU`	number of CPUs used for parallel sampling. For large models and many participants, this requires considerable computer-memory resources (as a remedy, use `nCPU=1`).

Value

by default, a list of `M` posterior-predictive samples (i.e., matrices) with individual frequencies (rows=participants, columns=MPT categories). For `M=1`, a single matrix is returned. If `numItems` is provided, a matrix with samples for a new participant is returned (rows=samples)

Examples

```
## Not run:
# add posterior predictive samples to fitted model
#     (facilitates plotting using ?plotFit)
fittedModel$postpred$freq.pred <-
  posteriorPredictive(fittedModel, M = 1000)

## End(Not run)
```

PPP *Compute Posterior Predictive P-Values*

Description

Computes posterior predictive p-values to test model fit.

Usage

```
PPP(fittedModel, M = 1000, nCPU = 4, T2 = TRUE, type = "X2")
```

Arguments

fittedModel	fitted latent-trait or beta MPT model (traitMPT, betaMPT)
M	number of posterior predictive samples. As a maximum, the number of posterior samples in fittedModel is used.
nCPU	number of CPUs used for parallel sampling. For large models and many participants, this requires considerable computer-memory resources (as a remedy, use nCPU=1).
T2	whether to compute T2 statistic to check coveriance structure (can take a lot of time). If some participants do not have responses for some trees, (co)variances are computed by pairwise deletion of the corresponding persons.
type	whether the T1 statistic of expected means is computed using Person's "X2" or the likelihood-ratio statistic "G2"

Author(s)

Daniel Heck

References

Klauer, K. C. (2010). Hierarchical multinomial processing tree models: A latent-trait approach. Psychometrika, 75, 70-98.

priorPredictive *Prior Predictive Samples*

Description

Samples full data sets (i.e., individual response frequencies) or group-level MPT parameters based on prior distribution for group-level parameters.

Usage

```
priorPredictive(
  prior,
  eqnfile,
  restrictions,
  numItems,
  level = "data",
  N = 1,
  M = 100,
  nCPU = 4
)
```

Arguments

prior	a named list defining the priors. For the traitMPT, the default is `list(mu = "dnorm(0,1)", xi="dunif(0,10)", V=diag(S),df=S+1)`, where S is the number of free parameters. For the betaMPT, the default is `list(alpha ="dgamma(1,.1)", beta = "dgamma(1,.1)")`. Note that the normal distribution `"dnorm(mu,prec)"` is parameterized as in JAGS by the mean and precision (= 1/variance).
eqnfile	The (relative or full) path to the file that specifies the MPT model (standard .eqn syntax). Note that category labels must start with a letter (different to multiTree) and match the column names of `data`. Alternatively, the EQN-equations can be provided within R as a character value (cf. `readEQN`). Note that the first line of an .eqn-file is reserved for comments and always ignored.
restrictions	Specifies which parameters should be (a) constant (e.g., `"a=b=.5"`) or (b) constrained to be identical (e.g., `"Do=Dn"`) or (c) treated as fixed effects (i.e., identical for all participants; `"a=b=FE"`). Either given as the path to a text file with restrictions per row or as a list of restrictions, e.g., `list("D1=D2","g=0.5")`. Note that numbers in .eqn-equations (e.g., `d*(1-g)*.50`) are directly interpreted as equality constraints.
numItems	vector with the number of items per MPT tree (either named or assigned to alphabetically ordered tree labels)
level	either `"data"` (returns individual frequencies) or `"parameter"` (returns group-level MPT parameters; `M` and `numItems` are ignored)
N	number of participants per replication
M	number of prior predictive samples (i.e., data sets with `N` participants).
nCPU	number of CPUs used for parallel sampling. For large models and many participants, this may require a lot of memory.

Value

a list of `M` matrices with individual frequencies (rows=participants, columns=MPT categories). A single matrix is returned if `M=1` or `level="parameter"`.

Examples

```
eqnfile <- system.file("MPTmodels/2htm.eqn",
  package = "TreeBUGS"
)
### beta-MPT:
prior <- list(
  alpha = "dgamma(1,.1)",
  beta = "dgamma(1,.1)"
)

### prior-predictive frequencies:
priorPredictive(prior, eqnfile,
  restrictions = list("g=.5", "Do=Dn"),
  numItems = c(50, 50), N = 10, M = 1, nCPU = 1
)

### prior samples of group-level parameters:
priorPredictive(prior, eqnfile,
  level = "parameter",
  restrictions = list("g=.5", "Do=Dn"),
  M = 5, nCPU = 1
)

### latent-trait MPT
priorPredictive(
  prior = list(
    mu = "dnorm(0,1)", xi = "dunif(0,10)",
    df = 3, V = diag(2)
  ),
  eqnfile, restrictions = list("g=.5"),
  numItems = c(50, 50), N = 10, M = 1, nCPU = 1
)
```

---

probitInverse *Probit-Inverse of Group-Level Normal Distribution*

Description

Transform latent group-level normal distribution (latent-trait MPT) into mean and SD on probability scale.

Usage

```
probitInverse(mu, sigma, fittedModel = NULL)
```

Arguments

mu	latent-probit mean of normal distribution

sigma	latent-probit SD of normal distribution
fittedModel	optional: fitted traitMPT model. If provided, the bivariate inverse-probit transform is applied to all MCMC samples (and mu and sigma are ignored).

Value

implied mean and SD on probability scale

Examples

```
####### compare bivariate vs. univariate transformation
probitInverse(mu = 0.8, sigma = c(0.25, 0.5, 0.75, 1))
pnorm(0.8)

# full distribution
prob <- pnorm(rnorm(10000, mean = 0.8, sd = 0.7))
hist(prob, 80, col = "gray", xlim = 0:1)

## Not run:
# transformation for fitted model
mean_sd <- probitInverse(fittedModel = fit)
summarizeMCMC(mean_sd)

## End(Not run)
```

---

readEQN *Read multiTree files*

Description

Function to import MPT models from standard .eqn model files as used, for instance, by multiTree (Moshagen, 2010).

Usage

```
readEQN(file, restrictions = NULL, paramOrder = FALSE, parse = FALSE)
```

Arguments

file	The (full path to the) file that specifies the MPT model (standard .eqn syntax). Note that category labels must start with a letter (different to multiTree) and match the column names of data. Alternatively, the EQN-equations can be provided within R as a character value (see examples). Note that the first line of an .eqn-file is reserved for comments and always ignored.
restrictions	Specifies which parameters should be (a) constant (e.g., "a=b=.5") or (b) constrained to be identical (e.g., "Do=Dn") or (c) treated as fixed effects (i.e., identical for all participants; "a=b=FE"). Either given as the path to a text file with restrictions per row or as a list of restrictions, e.g., list("D1=D2","g=0.5").

Note that numbers in .eqn-equations (e.g., `d*(1-g)*.50`) are directly interpreted as equality constraints.

`paramOrder` if TRUE, the order of MPT parameters as interally used is printed.

`parse` whether to return a parsed MPT model description in terms of the matrices a and b (the powers of the θ and $(1-\theta)$, respectively, and the vector of constants c. Each branch probability is then given as $c_i \prod_s \theta^{a_{i,s}}(1-\theta)^{b_{i,s}})$

Details

The file format should adhere to the standard .eqn-syntax (note that the first line is skipped and can be used for comments). In each line, a separate branch of the MPT model is specified using the tree label, category label, and the model equations in full form (multiplication sign '*' required; not abbreviations such as 'a^2' allowed).

As an example, the standard two-high threshold model (2HTM) is defined as follows:

```
Target     Hit              Do
 Target    Hit              (1-Do)*g
 Target    Miss             (1-Do)*(1-g)
 Lure      FalseAlarm       (1-Dn)*g
 Lure      CorrectReject    (1-Dn)*(1-g)
 Lure      CorrectReject    Dn
```

Author(s)

Daniel Heck, Denis Arnold, Nina Arnold

References

Moshagen, M. (2010). multiTree: A computer program for the analysis of multinomial processing tree models. Behavior Research Methods, 42, 42-54.

Examples

```
# Example: Standard Two-High-Threshold Model (2HTM)
EQNfile <- system.file("MPTmodels/2htm.eqn",
  package = "TreeBUGS"
)
readEQN(file = EQNfile, paramOrder = TRUE)

# with equality constraint:
readEQN(
  file = EQNfile, restrictions = list("Dn = Do", "g = 0.5"),
  paramOrder = TRUE
)

# define MPT model directly within R
model <-
  "2-High Threshold Model (2HTM)
  old hit d
  old hit (1-d)*g
```

```
  old miss (1-d)*(1-g)
  new fa (1-d)*g
  new cr (1-d)*(1-g)
  new cr d"
readEQN(model, paramOrder = TRUE)
```

`simpleMPT` *C++ Sampler for Standard (Nonhierarchical) MPT Models*

Description

Fast Gibbs sampler in C++ that is tailored to the standard fixed-effects MPT model (i.e., fixed-effects, non-hierarchical MPT). Assumes independent parameters per person if a matrix of frequencies per person is supplied.

Usage

```
simpleMPT(
  eqnfile,
  data,
  restrictions,
  n.iter = 2000,
  n.burnin = 500,
  n.thin = 3,
  n.chains = 3,
  ppp = 0,
  alpha = 1,
  beta = 1,
  parEstFile,
  posteriorFile,
  cores = 1
)
```

Arguments

`eqnfile`	The (relative or full) path to the file that specifies the MPT model (standard .eqn syntax). Note that category labels must start with a letter (different to multiTree) and match the column names of `data`. Alternatively, the EQN-equations can be provided within R as a character value (cf. `readEQN`). Note that the first line of an .eqn-file is reserved for comments and always ignored.
`data`	The (relative or full) path to the .csv file with the data (comma separated; category labels in first row). Alternatively: a data frame or matrix (rows=individuals, columns = individual category frequencies, category labels as column names)
`restrictions`	Specifies which parameters should be (a) constant (e.g., `"a=b=.5"`) or (b) constrained to be identical (e.g., `"Do=Dn"`) or (c) treated as fixed effects (i.e., identical for all participants; `"a=b=FE"`). Either given as the path to a text file with restrictions per row or as a list of restrictions, e.g., `list("D1=D2","g=0.5")`.

	Note that numbers in .eqn-equations (e.g., `d*(1-g)*.50`) are directly interpreted as equality constraints.
`n.iter`	Number of iterations per chain (including burnin samples). See `run.jags` for details.
`n.burnin`	Number of samples for burnin (samples will not be stored and removed from n.iter)
`n.thin`	Thinning rate.
`n.chains`	number of MCMC chains (sampled in parallel).
`ppp`	number of samples to compute posterior predictive p-value (see `posteriorPredictive`)
`alpha`	first shape parameter(s) for the beta prior-distribution of the MPT parameters θ_s (can be a named vector to use a different prior for each MPT parameter)
`beta`	second shape parameter(s)
`parEstFile`	Name of the file to with the estimates should be stored (e.g., "parEstFile.txt")
`posteriorFile`	path to RData-file where to save the model including MCMC posterior samples (an object named `fittedModel`; e.g., `posteriorFile="mcmc.RData"`)
`cores`	number of CPUs to be used

Details

Beta distributions with fixed shape parameters α and β are used. The default $\alpha = 1$ and $\beta = 1$ assumes uniform priors for all MPT parameters.

Author(s)

Daniel Heck

Examples

```
## Not run:
# fit nonhierarchical MPT model for aggregated data (see ?arnold2013):
EQNfile <- system.file("MPTmodels/2htsm.eqn", package = "TreeBUGS")
d.encoding <- subset(arnold2013, group == "encoding", select = -(1:4))
fit <- simpleMPT(EQNfile, colSums(d.encoding),
  restrictions = list("D1=D2=D3", "d1=d2", "a=g")
)
# convergence
plot(fit)
summary(fit)

## End(Not run)
```

summarizeMCMC *MCMC Summary*

Description

TreeBUGS-specific MCMC summary for `mcmc.list`-objects.

Usage

```
summarizeMCMC(mcmc, batchSize = 50, probs = c(0.025, 0.5, 0.975))
```

Arguments

`mcmc`	a `mcmc.list` object
`batchSize`	size of batches of parameters used to reduce memory load when computing posterior summary statistics (including Rhat and effective sample size).
`probs`	quantile probabilities used to compute credibility intervals

summarizeMPT *Summarize JAGS Output for Hierarchical MPT Models*

Description

Provide clean and readable summary statistics tailored to MPT models based on the JAGS output.

Usage

```
summarizeMPT(mcmc, mptInfo, probs = c(0.025, 0.5, 0.975), summ = NULL)
```

Arguments

`mcmc`	the actual mcmc.list output of the sampler of a fitted MPT model (accesible via `fittedModel$runjags$mcmc`)
`mptInfo`	the internally stored information about the fitted MPT model (accesible via `fittedModel$mptInfo`)
`probs`	quantile probabilities used to compute credibility intervals
`summ`	optional argument for internal use

Details

The MPT-specific summary is computed directly after fitting a model. However, this function might be used manually after removing MCMC samples (e.g., extending the burnin period).

Examples

```
# Remove additional burnin samples and recompute MPT summary
## Not run:
# start later or thin (see ?window)
mcmc.subsamp <- window(fittedModel$runjags$mcmc, start = 3001, thin = 2)
new.mpt.summary <- summarizeMPT(mcmc.subsamp, fittedModel$mptInfo)
new.mpt.summary

## End(Not run)
```

`testHetChi` *Chi-Square Test of Heterogeneity*

Description

Tests whether whether participants (items) are homogeneous under the assumption of item (participant) homogeneity.

Usage

```
testHetChi(freq, tree)
```

Arguments

`freq`	matrix with observed frequencies (rows: persons/items; columns: categories). Can also be the path to a .csv file with frequencies (comma-separated; first line defines category labels)
`tree`	a vector defining which columns of x belong to separate multinomial distributions (i.e., MPT trees). For instance, if `x` has five categories from two MPT trees: `tree=c(1,1,2,2,2)` or `tree=c("t1","t1","t2","t2","t2")`

Details

If an item/person has zero frequencies on all categories in an MPT tree, these zeros are neglected when computing mean frequencies per column. As an example, consider a simple recognition test with a fixed assignments of words to the learn/test list. In such an experiment, all learned words will result in hits or misses (i.e., the MPT tree of old items), whereas new words are always false alarms/correct rejections and thus belong to the MPT tree of new items (this is not necessarily the case if words are assigned randomly).

Note that the test assumes independence of observations and item homogeneity when testing participant heterogeneity. The latter assumption can be dropped when using a permutation test (`testHetPerm`).

Author(s)

Daniel W. Heck

References

Smith, J. B., & Batchelder, W. H. (2008). Assessing individual differences in categorical data. Psychonomic Bulletin & Review, 15, 713-731. doi:10.3758/PBR.15.4.713

See Also

testHetPerm, plotFreq

Examples

```
# some made up frequencies:
freq <- matrix(
  c(
    13, 16, 11, 13,
    15, 21, 18, 13,
    21, 14, 16, 17,
    19, 20, 21, 18
  ),
  ncol = 4, byrow = TRUE
)
# for a product-binomial distribution:
# (categories 1 and 2 and categories 3 and 4 are binomials)
testHetChi(freq, tree = c(1, 1, 2, 2))
# => no significant deviation from homogeneity (low power!)
```

---

testHetPerm *Permutation Test of Heterogeneity*

Description

Tests whether whether participants (items) are homogeneous without assuming item (participant) homogeneity.

Usage

```
testHetPerm(data, tree, source = "person", rep = 1000, nCPU = 4)
```

Arguments

data	matrix or data frame with three columns: person code/index, item label, response category. Can also be the path to a .csv file with frequencies (comma-separated; first line defines category labels)
tree	a list that defines which categories belong to the same multinomial distribution (i.e., the the same MPT tree). For instance: `tree = list(tree.old = c("hit","cr"), tree.new = c("fa","miss"))`. Category labels must match the values of the third column of `data`
source	whether to test for `"person"` or `"item"` homogeneity
rep	number of permutations to be sampled
nCPU	number of CPUs used for parallel Monte Carlo sampling of permutations

Details

If an item/person has zero frequencies on all categories in an MPT tree, these zeros are neglected when computing mean frequencies per column. As an example, consider a simple recognition test with a fixed assignments of words to the learn/test list. In such an experiment, all learned words will result in hits or misses (i.e., the MPT tree of old items), whereas new words are always false alarms/correct rejections and thus belong to the MPT tree of new items (this is not necessarily the case if words are assigned randomly).

Note that the test does still assume independence of observations. However, it does not require item homogeneity when testing participant heterogeneity (in contrast to the chi-square test: `testHetChi`).

Author(s)

Daniel W. Heck

References

Smith, J. B., & Batchelder, W. H. (2008). Assessing individual differences in categorical data. Psychonomic Bulletin & Review, 15, 713-731. doi:10.3758/PBR.15.4.713

See Also

`testHetChi`, `plotFreq`

Examples

```
# generate homogeneous data
# (N=15 participants, M=30 items)
data <- data.frame(
  id = rep(1:15, each = 30),
  item = rep(1:30, 15)
)
data$cat <- sample(c("h", "cr", "m", "fa"), 15 * 30,
  replace = TRUE,
  prob = c(.7, .3, .4, .6)
)
head(data)
tree <- list(
  old = c("h", "m"),
  new = c("fa", "cr")
)

# test participant homogeneity:
tmp <- testHetPerm(data, tree, rep = 200, nCPU = 1)
tmp[2:3]
```

traitMPT *Fit a Hierarchical Latent-Trait MPT Model*

Description

Fits a latent-trait MPT model (Klauer, 2010) based on a standard MPT model file (.eqn) and individual data table (.csv).

Usage

```
traitMPT(
  eqnfile,
  data,
  restrictions,
  covData,
  predStructure,
  predType,
  transformedParameters,
  corProbit = TRUE,
  mu = "dnorm(0,1)",
  xi = "dunif(0,10)",
  V,
  df,
  IVprec = "dgamma(.5,.5)",
  n.iter = 20000,
  n.adapt = 2000,
  n.burnin = 2000,
  n.thin = 5,
  n.chains = 3,
  dic = FALSE,
  ppp = 0,
  monitorIndividual = TRUE,
  modelfilename,
  parEstFile,
  posteriorFile,
  autojags = NULL,
  ...
)
```

Arguments

eqnfile	The (relative or full) path to the file that specifies the MPT model (standard .eqn syntax). Note that category labels must start with a letter (different to multiTree) and match the column names of data. Alternatively, the EQN-equations can be provided within R as a character value (cf. readEQN). Note that the first line of an .eqn-file is reserved for comments and always ignored.

`data` The (relative or full) path to the .csv file with the data (comma separated; category labels in first row). Alternatively: a data frame or matrix (rows=individuals, columns = individual category frequencies, category labels as column names)

`restrictions` Specifies which parameters should be (a) constant (e.g., `"a=b=.5"`) or (b) constrained to be identical (e.g., `"Do=Dn"`) or (c) treated as fixed effects (i.e., identical for all participants; `"a=b=FE"`). Either given as the path to a text file with restrictions per row or as a list of restrictions, e.g., `list("D1=D2","g=0.5")`. Note that numbers in .eqn-equations (e.g., `d*(1-g)*.50`) are directly interpreted as equality constraints.

`covData` Data that contains covariates, for which correlations with individual MPT parameters will be sampled. Either the path to a .csv file (comma-separated: rows=individuals in the same order as `data`; first row must contain covariate labels). Alternatively: a data frame or matrix (rows=individuals, columns = variables; covariate labels as column names). Note that in `betaMPT`, correlations are computed for discrete variables that are coded numerically (in `traitMPT`, this can be suppressed by using `predType="f"`)

`predStructure` Defines which variables in `covData` are included as predictors for which MPT parameters. Either the path to the file that specifies the assignment of MPT parameters to covariates (that is, each row assigns one or more MPT parameters to one or more covariates, separated by a semicolon, e.g., `Do g; age extraversion`). Can also be provided as a list, e.g., `list("Do Dn ; age", "g ; extraversion")`. Note that no correlations of MPT parameters and predictors are computed. However, for continuous covariates, the standardized slope parameters `slope_std_parameter_predictor` can be interpreted as a correlation if a single predictor is included for the corresponding MPT parameter (see Jobst et al., 2020).

`predType` a character vector specifying the type of continuous or discrete predictors in each column of `covData`: `"c"` = continuous covariate (which are centered to have a mean of zero); `"f"` = discrete predictor, fixed effect (default for character/factor variables); `"r"` = discrete predictor, random effect.

`transformedParameters`
list with parameter transformations that should be computed based on the posterior samples of the group-level means (e.g., for testing parameter differences: `list("diffD=Do-Dn")`), or path to a text file containing one transformation per line. Transformations of individual-level parameters can also be performed after fitting a model using `transformedParameters`.

`corProbit` whether to use probit-transformed MPT parameters to compute correlations (probit-values of `+Inf` are truncated to `max(5,max(probit))`; similarly for `-Inf`). Default for beta-MPT: MPT parameters are used on the probability scale [0,1].

`mu` hyperprior for group means of probit-transformed parameters in JAGS syntax. Default is a standard normal distribution, which implies a uniform distribution on the MPT probability parameters. A named vector can be used to specify separate hyperpriors for each MPT parameter (the order of parameters is determined by the names of the vector or by the default order as shown in `readEQN` with `paramOrder = TRUE`).

`xi` hyperprior for scaling parameters of the group-level parameter variances. Default is a uniform distribution on the interval [0,10]. Similarly as for `mu`, a vec-

tor of different priors can be used. Less informative priors can be used (e.g., `"dunif(0,100)"`) but might result in reduced stability.

`V` S x S matrix used as a hyperprior for the inverse-Wishart hyperprior parameters with as many rows and columns as there are core MPT parameters. Default is a diagonal matrix.

`df` degrees of freedom for the inverse-Wishart hyperprior for the individual parameters. Minimum is S+1, where S gives the number of core MPT parameters.

`IVprec` hyperprior on the precision parameter g (= the inverse of the variance) of the standardized slope parameters of continuous covariates. The default `IVprec=dgamma(.5,.5)` defines a mixture of g-priors (also known as JZS or Cauchy prior) with the scale parameter $s = 1$. Different scale parameters s can be set via: `IVprec=dgamma(.5,.5*s^2)`. A numeric constant `IVprec=1` implies a g-prior (a normal distribution). For ease of interpretation, TreeBUGS reports both unstandardized and standardized regression coefficients. See details below.

`n.iter` Number of iterations per chain (including burnin samples). See `run.jags` for details.

`n.adapt` number of adaption samples to adjust MCMC sampler in JAGS. The sampler will be more efficient if it is tuned well. However, MCMC sampling will still give correct results even if the warning appears: "Adaptation incomplete." (this just means that sampling efficiency could be better).

`n.burnin` Number of samples for burnin (samples will not be stored and removed from n.iter)

`n.thin` Thinning rate.

`n.chains` number of MCMC chains (sampled in parallel).

`dic` whether to compute DIC using `extract.runjags`, which requires additional sampling. Can also be computed and added after fitting the model by `fittedModel$summary$dic <- runjags::extract(fittedModel$runjags,"dic")`. As an alternative information criterion, `WAIC` can be computed for fitted models.

`ppp` number of samples to compute posterior predictive p-value (see `posteriorPredictive`)

`monitorIndividual`
whether to store MCMC samples of the MPT parameters `theta` at the individual level (i.e., the random effects). If `FALSE`, it is not possible to perform posterior-predictive checks.

`modelfilename` name of the generated JAGS model file. Default is to write this information to the tempdir as required by CRAN standards.

`parEstFile` Name of the file to with the estimates should be stored (e.g., "parEstFile.txt")

`posteriorFile` path to RData-file where to save the model including MCMC posterior samples (an object named `fittedModel`; e.g., `posteriorFile="mcmc.RData"`)

`autojags` JAGS first fits the MPT model as usual and then draws MCMC samples repeatedly until convergence. For this, the function `autoextend.jags` is used with the arguments provided in `autojags` (this can be an empty list, in which case the defaults are used). Possible arguments for `autoextend.jags` are: `list(startburnin = 1000, startsample = 5000, adapt = 2000,max.time="30m")` (the last of these arguments restricts sampling to 30 minutes, see autoextend.jags).

... further arguments to be passed to the JAGS sampling function (i.e., to `run.jags`. Note that reproducible results are obtained by setting a random seed before fitting a model (i.e., `set.seed(12345)`).

Value

a list of the class `traitMPT` with the objects:

- `summary`: MPT tailored summary. Use `summary(fittedModel)`
- `mptInfo`: info about MPT model (eqn and data file etc.)
- `mcmc`: the object returned from the MCMC sampler. Note that the object `fittedModel$mcmc` is an runjags object, whereas `fittedModel$mcmc$mcmc` is an `mcmc.list` as used by the coda package (mcmc)

Regression Extensions

Continuous and discrete predictors are added on the latent-probit scale via:

$$\theta = \Phi(\mu + X\beta + \delta),$$

where X is a design matrix includes centered continuous covariates and recoded factor variables (using the orthogonal contrast coding scheme by Rouder et al., 2012). Note that both centering and recoding is done internally. TreeBUGS reports unstandardized regression coefficients β that correspond to the scale/SD of the predictor variables. Hence, slope estimates will be very small if the covariate has a large variance. TreeBUGS also reports standardized slope parameters (labeled with `std`) which are standardized both with respect to the variance of the predictor variables and the variance in the individual MPT parameters. If a single predictor variable is included, the standardized slope can be interpreted as a correlation coefficient (Jobst et al., 2020).

For continuous predictor variables, the default prior `IVprec = "dgamma(.5,.5)"` implies a Cauchy prior on the β parameters (standardized with respect to the variance of the predictor variables). This prior is similar to the Jeffreys-Zellner-Siow (JZS) prior with scale parameter $s = 1$ (for details, see: Rouder et. al, 2012; Rouder & Morey, 2012). In contrast to the JZS prior for standard linear regression by Rouder & Morey (2012), TreeBUGS implements a latent-probit regression where the prior on the coefficients β is only standardized/scaled with respect to the continuous predictor variables but not with respect to the residual variance (since this is not a parameter in probit regression). If small effects are expected, smaller scale values s can be used by changing the default to `IVprec = 'dgamma(.5, .5*s^2)'` (by plugging in a specific number for `s`). To use a standard-normal instead of a Cauchy prior distribution, use `IVprec = 'dcat(1)'`. Bayes factors for slope parameters of continuous predictors can be computed with the function BayesFactorSlope.

Uncorrelated Latent-Trait Values

The standard latent-trait MPT model assumes a multivariate normal distribution of the latent-trait values, where the covariance matrix follows a scaled-inverse Wishart distribution. As an alternative, the parameters can be assumed to be independent (this is equivalent to a diagonal covariance matrix). If the assumption of uncorrelated parameters is justified, such a simplified model has less parameters and is more parsimonious, which in turn might result in more robust estimation and more precise parameter estimates.

This alternative method can be fitted in TreeBUGS (but not all of the features of TreeBUGS might be compatible with this alternative model structure). To fit the model, the scale matrix `V` is set to `NA` (V is only relevant for the multivariate Wishart prior) and the prior on `xi` is changed: `traitMPT(..., V=NA, xi="dnorm(0,1)")`. The model assumes that the latent-trait values δ_i (=random-intercepts) are decomposed by the scaling parameter ξ and the raw deviation ϵ_i (cf. Gelman, 2006):

$$\delta_i = \xi \cdot \epsilon_i$$

$$\epsilon_i \sim Normal(0, \sigma^2)$$

$$\sigma^2 \sim Inverse - \chi^2(df)$$

Note that the default prior for ξ should be changed to `xi="dnorm(0,1)"`, which results in a half-Cauchy prior (Gelman, 2006).

Author(s)

Daniel W. Heck, Denis Arnold, Nina R. Arnold

References

Heck, D. W., Arnold, N. R., & Arnold, D. (2018). TreeBUGS: An R package for hierarchical multinomial-processing-tree modeling. *Behavior Research Methods, 50*, 264–284. doi:10.3758/s1342801708697

Gelman, A. (2006). Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper). *Bayesian Analysis, 1*, 515-534.

Jobst, L. J., Heck, D. W., & Moshagen, M. (2020). A comparison of correlation and regression approaches for multinomial processing tree models. *Journal of Mathematical Psychology, 98*, 102400. doi:10.1016/j.jmp.2020.102400

Klauer, K. C. (2010). Hierarchical multinomial processing tree models: A latent-trait approach. *Psychometrika, 75*, 70-98. doi:10.1007/s1133600991410

Matzke, D., Dolan, C. V., Batchelder, W. H., & Wagenmakers, E.-J. (2015). Bayesian estimation of multinomial processing tree models with heterogeneity in participants and items. *Psychometrika, 80*, 205-235. doi:10.1007/s1133601393749

Rouder, J. N., Morey, R. D., Speckman, P. L., & Province, J. M. (2012). Default Bayes factors for ANOVA designs. *Journal of Mathematical Psychology, 56*, 356-374. doi:10.1016/j.jmp.2012.08.001

Rouder, J. N., & Morey, R. D. (2012). Default Bayes Factors for Model Selection in Regression. *Multivariate Behavioral Research, 47*, 877-903. doi:10.1080/00273171.2012.734737

Examples

```
## Not run:
# fit beta-MPT model for encoding condition (see ?arnold2013):
EQNfile <- system.file("MPTmodels/2htsm.eqn", package = "TreeBUGS")
d.encoding <- subset(arnold2013, group == "encoding", select = -(1:4))
fit <- traitMPT(EQNfile, d.encoding,
  n.thin = 5,
  restrictions = list("D1=D2=D3", "d1=d2", "a=g")
)
```

```
# convergence
plot(fit, parameter = "mean", type = "default")
summary(fit)

## End(Not run)
```

`transformedParameters` *Get Transformed Parameters*

Description

Computes transformations of MPT parameters based on the MCMC posterior samples (e.g., differences of parameters).

Usage

```
transformedParameters(
  fittedModel,
  transformedParameters,
  level = "group",
  nCPU = 4
)
```

Arguments

`fittedModel`	either a fitted latent-trait or beta MPT model (`traitMPT`, `betaMPT`) or an `mcmc.list`.
`transformedParameters`	list with parameter transformations that should be computed based on the posterior samples (e.g., for testing parameter differences: `list("diffD=Do-Dn")`).
`level`	whether to compute transformations of `"group"` or `"individual"` estimates
`nCPU`	number of CPU cores across which the MCMC chains are distributed

Value

an mcmc.list of posterior samples for the transformed parameters

Examples

```
## Not run:
tt <- transformedParameters(fittedModel,
  list("diff = a-b", "p = a>b"),
  level = "individual"
)
summary(tt)

## End(Not run)
```

WAIC *WAIC: Widely Applicable Information Criterion*

Description

Implementation of the WAIC for model comparison.

Usage

```
WAIC(
  fittedModel,
  n.adapt = 1000,
  n.chains = 3,
  n.iter = 10000,
  n.thin = 1,
  summarize = FALSE
)

## S3 method for class 'waic'
print(x, ...)

## S3 method for class 'waic_difference'
print(x, ...)

## S3 method for class 'waic'
e1 - e2
```

Arguments

fittedModel	fitted latent-trait or beta MPT model (traitMPT, betaMPT)
n.adapt	number of adaptation samples.
n.chains	number of chains (no parallel computation).
n.iter	number of iterations after burnin.
n.thin	Thinning rate.
summarize	deprecated argument only available for backwards compatibility
x	An object of class waic or waic_difference to be printed.
...	Further arguments that may be passed to print methods.
e1, e2	Two objects of class waic to be compared.

Details

WAIC provides an approximation of predictive accuracy with respect to out-of-sample deviance. The uncertainty of the WAIC for the given number of observed nodes (i.e., number of free categories times the number of participants) is quantified by the standard error of WAIC "se_waic" (cf. Vehtari et al., 2017). In contrast, to assess whether the approximation uncertainty due to MCMC

sampling (not sample size) is sufficiently low, it is a good idea to fit each model twice and compute WAIC again to assess the stability of the WAIC values.

For more details, see Vehtari et al. (2017) and the following discussion about the JAGS implementation (which is currently an experimental feature of JAGS 4.3.0):

https://sourceforge.net/p/mcmc-jags/discussion/610036/thread/8211df61/

Value

Function `WAIC()` returns an object of class `waic`, which is basically a list containing three vectors `p_waic`, `deviance`, and `waic`, with separate values for each observed node (i.e., for all combinations of persons and free categories).

For these objects, a `print()` method exists, which also calculates the standard error of the estimate of WAIC.

For backwards compatibility, if `WAIC()` is called with `summarize = TRUE`, a vector with values `p_waic`, `deviance`, `waic`, and `se_waic` is returned.

WAIC values from two models can be compared by using the `-` operator; the result is an object of class `waic_difference`.

References

Vehtari, A., Gelman, A., & Gabry, J. (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. Statistics and Computing, 27(5), 1413–1432. doi:10.1007/s11222-016-9696-4

Examples

```
## Not run:

#### WAIC for a latent-trait MPT model:
fit <- traitMPT(...)
WAIC(fit)


#### pairwise comparison of two models:

# (1) compute WAIC per model
waic1 <- WAIC(fit1)
waic2 <- WAIC(fit2)

# (2) WAIC difference
waic1 - waic2

## End(Not run)
```

`withinSubjectEQN` *Generate EQN Files for Within-Subject Designs*

Description

Replicates an MPT model multiple times with different tree, category, and parameter labels for within-subject factorial designs.

Usage

```
withinSubjectEQN(eqnfile, labels, constant, save)
```

Arguments

`eqnfile`	The (relative or full) path to the file that specifies the MPT model (standard .eqn syntax). Note that category labels must start with a letter (different to multiTree) and match the column names of `data`. Alternatively, the EQN-equations can be provided within R as a character value (cf. `readEQN`). Note that the first line of an .eqn-file is reserved for comments and always ignored.
`labels`	a character vector defining the labels that are added to the parameters in each within-subject condition
`constant`	optional: a character vector defining which parameters are constrained to be constant across within-conditions
`save`	optional: path to an EQN output file. By default, the model is return as a string character

Examples

```
# Example: Standard Two-High-Threshold Model (2HTM)
EQNfile <- system.file("MPTmodels/2htm.eqn",
  package = "TreeBUGS"
)
withinSubjectEQN(EQNfile, c("high", "low"), constant = c("g"))
```

Index