Package ‘rtmpt’

October 14, 2022

Version 1.0-0

Title Fitting RT-MPT Models

Author Raphael Hartmann [aut, cre],
Karl C. Klauer [cph, aut, ctb, ths],
Henrik Singmann [ctb],
Jean Marie Linhart [ctb],
Frederick Novomestky [ctb]

Maintainer Raphael Hartmann `<raphael.hartmann@protonmail.com>`

Imports coda, data.table, loo, methods, Ryacas, stats, stringr,
truncnorm, utils

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

SystemRequirements GSL (>=2.3)

Description Fit response-time extended multinomial processing tree (RT-MPT) models by Klauer and Kellen (2018) <doi:10.1016/j.jmp.2017.12.003>. The RT-MPT class not only incorporate frequencies like traditional multinomial processing tree (MPT) models, but also latencies. This enables it
to estimate process completion times and encoding plus motor execution times next to the process probabilities
of traditional MPTs. 'rtmpt' is a Bayesian framework and posterior samples are sampled using a Metropolis-Gibbs
sampler like the one described in the Klauer and Kellen (2018), but with some modifications. Other than in
the original C++ program we use the free and open source GNU Scientific Library (GSL). There is also the
possibility to suppress single process completion times.

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Repository CRAN

Date/Publication 2022-04-10 09:40:02 UTC

R topics documented:

`delta2delta` *Set mapping between response categories and encoding plus motor execution times*

Description

Mapping response categories with encoding and motor execution times (deltas). Unlike the processes there are no names for the different deltas and therefore a mapping from response categories to different deltas must be specified.

Usage

```
delta2delta(model, trees, categories, mappings = 0)

set_deltas_equal(model, trees, categories, mappings = 0)
```

Arguments

`model`	A list of the class `rtmpt_model`.
`trees`	Character or numerical vector giving the trees
`categories`	Character or numerical vector identifying category/ies within the specified `trees` for which the deltas should be changed.
`mappings`	Numerical vector of length `length(categories)` providing the mappings. Default is 0.

Value

A list of the class `rtmpt_model`.

Author(s)

Raphael Hartmann

See Also

`theta2const`, `tau2zero`, `theta2theta`, and `tau2tau`,

Examples

```
#####################################################################################
# Detect-Guess variant of the Two-High Threshold model.
# The encoding and motor execution times will be set to different responses
#####################################################################################

mdl_2HTM <- "
# targets
do+(1-do)*g
(1-do)*(1-g)

# lures
(1-dn)*g
dn+(1-dn)*(1-g)

# do: detect old; dn: detect new; g: guess
"

model <- to_rtmpt_model(mdl_file = mdl_2HTM)

## changing the model to have two different encoding and motor execution
## times for "old" and "new" responses.
new_model <- delta2delta(model = model, trees = c(0, 1),
                         categories = c(1,3), mappings = c(1,1))
new_model


## changing the model to have two different encoding and response execution
## times for "old" and "new" responses.
new_model <- set_deltas_equal(model = model, trees = c(0, 1),
                              categories = c(1,3), mappings = c(1,1))
new_model
```

`fit_rtmpt` *Posterior sample, diagnostics and some optional stuff for RT-MPT models*

Description

Given model and data, this function calls an altered version of the C++ program by Klauer and Kellen (2018) to sample from the posterior distribution via a Metropolis-Gibbs sampler and storing it in an mcmc.list called `samples`. Posterior predictive checks developed by Klauer (2010), deviance information criterion (DIC; Spiegelhalter et al., 2002), 99% and 95% highest density intervals (HDI) together with the median will be provided for the main parameters in a list called `diags`. Optionally, the `indices` widely applicable information criterion (WAIC; Watanabe, 2010; Vehtari et al., 2017) and leave-one-out cross-validation (LOO; Vehtari et al., 2017) can be saved. Additionally the log-likelihood (`LogLik`) can also be stored. Some specifications of the function call are also saved in `specs`.

Usage

```
fit_rtmpt(
  model,
  data,
  n.chains = 4,
  n.iter = 5000,
  n.burnin = 200,
  n.thin = 1,
  Rhat_max = 1.05,
  Irep = 1000,
  prior_params = NULL,
  indices = FALSE,
  save_log_lik = FALSE,
  old_label = FALSE
)
```

Arguments

`model`	A list of the class `rtmpt_model`.
`data`	Optimally, a list of class `rtmpt_data`. Also possible is a `data.frame` or a path to the text file. Both, `data.frame` and the text file must contain the column names "subj", "group", "tree", "cat", and "rt" preferably but not necessarily in this order. The values of the latter must be in milliseconds. It is always advised to use `to_rtmpt_data` first, which gives back an `rtmpt_data` list with informations about the changes in the data, that were needed.
`n.chains`	Number of chains to use. Default is 4. Must be larger than 1 and smaller or equal to 16.
`n.iter`	Number of samples per chain. Default is 5000.
`n.burnin`	Number of warm-up samples. Default is 200.
`n.thin`	Thinning factor. Default is 1.
`Rhat_max`	Maximal Potential scale reduction factor: A lower threshold that needs to be reached before the actual sampling starts. Default is 1.05
`Irep`	Every `Irep` samples an interim state with the current maximal potential scale reduction factor is shown. Default is 1000. The following statements must hold true for `Irep`: • `n.burnin` is smaller than or equal to `Irep`, • `Irep` is a multiple of `n.thin` and

- `n.iter` is a multiple of `Irep / n.thin`.

`prior_params` Named list with prior parameters. All parameters have default values, that lead to uninformative priors. Vectors are not allowed. Allowed parameters are:

- `mean_of_exp_mu_beta`: This is the a priori expected exponential rate (`E(exp(beta)) = E(lambda)`) and `1/mean_of_exp_mu_beta` is the a priori expected process time (`1/E(exp(beta)) = E(tau)`). The default mean is set to `10`, such that the expected a priori process time is `0.1` seconds.
- `var_of_exp_mu_beta`: The a priori group-specific variance of the exponential rates. Since `exp(mu_beta)` is Gamma distributed, the rate of the distribution is just mean divided by variance and the shape is the mean times the rate. The default is set to `100`.
- `mean_of_mu_gamma`: This is the a priori expected *mean parameter* of the encoding and response execution times, which follow a normal distribution truncated from below at zero, so `E(mu_gamma) < E(gamma)`. The default is `0`.
- `var_of_mu_gamma`: The a priori group-specific variance of the *mean parameter*. Its default is `10`.
- `mean_of_omega_sqr`: This is the a priori expected residual variance (`E(omega^2)`). Its distribution differs from the one used in the paper. Here it is a Gamma distribution instead of an improper one. The default is `0.005`.
- `var_of_omega_sqr`: The a priori variance of the residual variance (`Var(omega^2)`). The default is `0.01`. The default of the mean and variance is equivalent to a shape and rate of `0.0025` and `0.5`, respectivly.
- `df_of_sigma_sqr`: A priori degrees of freedom for the individual variance of the response executions. The individual variance has a scaled inverse chi-squared prior with `df_of_sigma_sqr` degrees of freedom and `omega^2` as scale. `2` is the default and it should be an integer.
- `sf_of_scale_matrix_SIGMA`: The original scaling matrix (S) of the (scaled) inverse Wishart distribution for the process related parameters is an identity matrix `S=I`. `sf_of_scale_matrix_SIGMA` is a scaling factor, that scales this matrix (`S=sf_of_scale_matrix_SIGMA*I`). Its default is `1`.
- `sf_of_scale_matrix_GAMMA`: The original scaling matrix (S) of the (scaled) inverse Wishart distribution for the encoding and motor execution parameters is an identity matrix `S=I`. `sf_of_scale_matrix_GAMMA` is a scaling factor, that scales this matrix (`S=sf_of_scale_matrix_GAMMA*I`). Its default is `1`.
- `prec_epsilon`: This is epsilon in the paper. It is the precision of mu_alpha and all xi (scaling parameter in the scaled inverse Wishart distribution). Its default is also `1`.
- `add_df_to_invWish`: If `P` is the number of parameters or rather the size of the scale matrix used in the (scaled) inverse Wishart distribution then `add_df_to_invWish` is the number of degrees of freedom that can be added to it. So `DF = P + add_df_to_invWish`. The default for `add_df_to_invWish` is `1`, such that the correlations are uniformly distributed within `[-1, 1]`.

`indices` Model selection indices. If set to `TRUE` the log-likelihood for each iteration and trial will be stored temporarily and with that the WAIC and LOO will be calculated via the `loo` package. If you want to have this log-likelihood matrix stored

	in the output of this function, you can set `save_log_lik` to `TRUE`. The default for `indices` is `FALSE`.
`save_log_lik`	If set to `TRUE` and `indices = TRUE` the log-likelihood matrix for each iteration and trial will be saved in the output as a matrix. Its default is `FALSE`.
`old_label`	If set to `TRUE` the old labels of "subj" and "group" of the data will be used in the elements of the output list. Default is `FALSE`.

Value

A list of the class `rtmpt_fit` containing

- `samples`: the posterior samples as an `mcmc.list` object,
- `diags`: some diagnostics like deviance information criterion, posterior predictive checks for the frequencies and latencies, potential scale reduction factors, and also the 99% and 95% HDIs and medians for the group-level parameters,
- `specs`: some model specifications like the model, arguments of the model call, and information about the data transformation,
- `indices` (optional): if enabled, WAIC and LOO,
- `LogLik` (optional): if enabled, the log-likelihood matrix used for WAIC and LOO.
- `summary` includes posterior mean and median of the main parameters.

Author(s)

Raphael Hartmann

References

Hartmann, R., Johannsen, L., & Klauer, K. C. (2020). rtmpt: An R package for fitting response-time extended multinomial processing tree models. *Behavior Research Methods, 52*(3), 1313–1338.

Hartmann, R., & Klauer, K. C. (2020). Extending RT-MPTs to enable equal process times. *Journal of Mathematical Psychology, 96*, 102340.

Klauer, K. C. (2010). Hierarchical multinomial processing tree models: A latent-trait approach. *Psychometrika, 75(1)*, 70-98.

Klauer, K. C., & Kellen, D. (2018). RT-MPTs: Process models for response-time distributions based on multinomial processing trees with applications to recognition memory. *Journal of Mathematical Psychology, 82*, 111-130.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & Van Der Linde, A. (2002). Bayesian measures of model complexity and fit. *Journal of the royal statistical society: Series b (statistical methodology), 64(4)*, 583-639.

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.

Watanabe, S. (2010). Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory. *Journal of Machine Learning Research, 11(Dec)*, 3571-3594.

Examples

```
########################################################################################
# Detect-Guess variant of the Two-High Threshold model.
# The encoding and motor execution times are assumed to be equal for each response.
########################################################################################

mdl_2HTM <- "
# targets
do+(1-do)*g
(1-do)*(1-g)

# lures
(1-dn)*g
dn+(1-dn)*(1-g)

# do: detect old; dn: detect new; g: guess
"

model <- to_rtmpt_model(mdl_file = mdl_2HTM)

data_file <- system.file("extdata/data.txt", package="rtmpt")
data <- read.table(file = data_file, header = TRUE)
data_list <- to_rtmpt_data(raw_data = data, model = model)

# This might take some time
rtmpt_out <- fit_rtmpt(model = model, data = data_list, Rhat_max = 1.1)
rtmpt_out

# Type ?SimData for another working example.
```

`fit_rtmpt_SBC` *Simulation-based calibration for RT-MPT models*

Description

Simulate data from RT-MPT models using `rtmpt_model` objects. The difference to `sim_rtmpt_data` is that here only scalars are allowed. This makes it usable for simulation-based calibration (SBC; Talts et al., 2018). You can specify the random seed, number of subjects, number of trials, and some parameters (same as `prior_params` from `fit_rtmpt`).

Usage

```
fit_rtmpt_SBC(
  model,
  seed,
  n.eff_samples = 99,
  n.chains = 4,
  n.iter = 5000,
  n.burnin = 200,
```

```
  n.thin = 1,
  Rhat_max = 1.05,
  Irep = 1000,
  n.subj = 40,
  n.trials = 30,
  prior_params = NULL,
  sim_list = NULL
)
```

Arguments

`model`	A list of the class `rtmpt_model`.
`seed`	Random seed number.
`n.eff_samples`	Number of effective samples. Default is 99, leading to 100 possible ranks (from 0 to 99).
`n.chains`	Number of chains to use. Default is 4. Must be larger than 1 and smaller or equal to 16.
`n.iter`	Number of samples per chain. Default is 5000. Must be larger or equal to `n.eff_samples`.
`n.burnin`	Number of warm-up samples. Default is 200.
`n.thin`	Thinning factor. Default is 1.
`Rhat_max`	Maximal Potential scale reduction factor: A lower threshold that needs to be reached before the actual sampling starts. Default is 1.05
`Irep`	Every `Irep` samples an interim state with the current maximal potential scale reduction factor is shown. Default is 1000. The following statements must hold true for `Irep`: • `n.burnin` is smaller than or equal to `Irep`, • `Irep` is a multiple of `n.thin` and • `n.iter` is a multiple of `Irep / n.thin`.
`n.subj`	Number of subjects. Default is 40.
`n.trials`	Number of trials per tree. Default is 30.
`prior_params`	Named list of parameters from which the data will be generated. This must be the same named list as `prior_params` from `fit_rtmpt` and has the same defaults. It is not recommended to use the defaults since they lead to many probabilities close or equal to `0` and/or `1` and to RTs close or equal to `0`. Allowed parameters are: • `mean_of_exp_mu_beta`: This is the expected exponential rate (`E(exp(beta)) = E(lambda)`) and `1/mean_of_exp_mu_beta` is the expected process time (`1/E(exp(beta)) = E(tau)`). The default mean is set to `10`, such that the expected process time is `0.1` seconds. • `var_of_exp_mu_beta`: The group-specific variance of the exponential rates. Since `exp(mu_beta)` is Gamma distributed, the rate of the distribution is just mean divided by variance and the shape is the mean times the rate. The default is set to `100`.

- `mean_of_mu_gamma`: This is the expected *mean parameter* of the encoding and response execution times, which follow a normal distribution truncated from below at zero, so `E(mu_gamma)` < `E(gamma)`. The default is `0`.
- `var_of_mu_gamma`: The group-specific variance of the *mean parameter*. Its default is `10`.
- `mean_of_omega_sqr`: This is the expected residual variance (`E(omega^2)`). The default is `0.005`.
- `var_of_omega_sqr`: The variance of the residual variance (`Var(omega^2)`). The default is `0.01`. The default of the mean and variance is equivalent to a shape and rate of `0.0025` and `0.5`, respectivly.
- `df_of_sigma_sqr`: degrees of freedom for the individual variance of the response executions. The individual variance follows a scaled inverse chi-squared distribution with `df_of_sigma_sqr` degrees of freedom and `omega^2` as scale. `2` is the default and it should be an integer.
- `sf_of_scale_matrix_SIGMA`: The original scaling matrix (S) of the (scaled) inverse Wishart distribution for the process related parameters is an identity matrix `S=I`. `sf_of_scale_matrix_SIGMA` is a scaling factor, that scales this matrix (`S=sf_of_scale_matrix_SIGMA*I`). Its default is `1`.
- `sf_of_scale_matrix_GAMMA`: The original scaling matrix (S) of the (scaled) inverse Wishart distribution for the encoding and motor execution parameters is an identity matrix `S=I`. `sf_of_scale_matrix_GAMMA` is a scaling factor that scales this matrix (`S=sf_of_scale_matrix_GAMMA*I`). Its default is `1`.
- `prec_epsilon`: This is epsilon in the paper. It is the precision of mu_alpha and all xi (scaling parameter in the scaled inverse Wishart distribution). Its default is also `1`.
- `add_df_to_invWish`: If `P` is the number of parameters or rather the size of the scale matrix used in the (scaled) inverse Wishart distribution then `add_df_to_invWish` is the number of degrees of freedom that can be added to it. So `DF` = `P` + `add_df_to_invWish`. The default for `add_df_to_invWish` is `1`, such that the correlations are uniformly distributed within `[-1, 1]`.

`sim_list`	Object of class `rtmpt_sim`. This is also an output object. Can be used to re-fit the model if `n.eff_samples` was not achieved in a previous fitting attempt. It will then use the data stored in this object. Default is NULL and this object will be created anew.

Value

A list of the class `rtmpt_sbc` containing

- `ranks`: the rank statistic for all parameters,
- `sim_list`: an object of the class `rtmpt_sim`,
- `fit_list`: an object of the class `rtmpt_fit`,
- `specs`: some specifications like the model, seed number, etc.,

Author(s)

Raphael Hartmann

References

Talts, S., Betancourt, M., Simpson, D., Vehtari, A., & Gelman, A. (2018). Validating Bayesian inference algorithms with simulation-based calibration. *arXiv preprint arXiv:1804.06788*.

Examples

```
########################################################################################
# Detect-Guess variant of the Two-High Threshold model.
# The encoding and motor execution times are assumed to be different for each response.
########################################################################################

mdl_2HTM <- "
# targets
d+(1-d)*g     ; 0
(1-d)*(1-g)   ; 1

# lures
(1-d)*g       ; 0
d+(1-d)*(1-g) ; 1

# d: detect; g: guess
"

model <- to_rtmpt_model(mdl_file = mdl_2HTM)

params <- list(mean_of_exp_mu_beta = 10,
               var_of_exp_mu_beta = 10,
               mean_of_mu_gamma = 0.5,
               var_of_mu_gamma = 0.0025,
               mean_of_omega_sqr = 0.005,
               var_of_omega_sqr = 0.000025,
               df_of_sigma_sqr = 10,
               sf_of_scale_matrix_SIGMA = 0.1,
               sf_of_scale_matrix_GAMMA = 0.01,
               prec_epsilon = 10,
               add_df_to_invWish = 5)

R = 2 # typically 2000 with n.eff_samples = 99, but this will run many days
rank_mat <- matrix(NA, ncol = 393, nrow = 2)
for (r in 1:R) {
  SBC_out <- fit_rtmpt_SBC(model, seed = r*123, prior_params = params,
                           n.eff_samples = 99, n.thin = 5,
                           n.iter = 5000, n.burnin = 2000, Irep = 5000)
  rank_mat[r, ] <- SBC_out$ranks
}
```

`set_params` *Set constants for probability parameters and suppress process times in a `rtmpt_model` list*

Description

By using `parameter = "probs"` you can specify which of the probability parameters should be set to a constant by using values between zero and one. If you use `NA` the probability will be estimated. By using `parameter = "tau_minus"` or `parameter = "tau_plus"` you can suppress process times/rates. Here `0` will suppress the named process and `NA` allows the process time/rate to be estimated.

Usage

```
set_params(model, parameter, names, values = NA)
```

Arguments

`model`	A list of the class `rtmpt_model`.
`parameter`	Character of length one indicating the parameter to change. Allowed characters: • "probs": probability parameters • "tau_minus": rate parameters of the exponential distribution of the process times that lead to a negative outcome • "tau_plus": rate parameters of the exponential distribution of the process times that lead to a positive outcome
`names`	Character vector with process names.
`values`	Numerical vector of length `length(names)`. By using `parameter = "probs"` you have the following options • `NA`: estimate the named probability • `0 < values < 1`: set the named probability to a constant value between zero and one Example: `set_params(model = model, parameter = "probs", names = c("do", "dn", "g"), values = c(NA, NA, .5))` will set the guessing "old" (g) to the constant `0.5` in the 2HT model. By using `parameter = "tau_minus"` or `parameter = "tau_plus"` you have two options: • `NA`: estimate the process time/rate • `0`: suppress the process time/rate Example: `set_params(model = model, parameter = "tau_minus", names = c("do", "dn", "g"), values = c(NA, NA, 0))` will suppress the process-completion time for guessing "new" in the 2HT model. This of course does not make sense here, but for some models it might be useful if you assume that a time-consuming process is not associated with certain process-outcome pairs (e.g., for technical parameters not corresponding to a psychological process).

Value

A list of the class `rtmpt_model`.

Author(s)

Raphael Hartmann

See Also

`delta2delta`

Examples

```
####################################################################################
# Detect-Guess variant of the Two-High Threshold model.
# The encoding and motor execution times are assumed to be equal for each category.
# The process completion times for both failed detections will be suppressed.
####################################################################################

mdl_2HTM <- "
# targets
do+(1-do)*g
(1-do)*(1-g)

# lures
(1-dn)*g
dn+(1-dn)*(1-g)

# do: detect old; dn: detect new; g: guess
"

model <- to_rtmpt_model(mdl_file = mdl_2HTM)

## removing the process times for the failed detection ("tau_minus")
## of the detection parameters ("dn", "do")
model <- set_params(model = model, parameter = "tau_minus",
                    names = c("dn", "do"), values = c(0,0))
```

---

set_resps *Set responses in an* `rtmpt_model`

Description

Change the responses for a tree and the categories within that tree.

Usage

```
set_resps(model, tree, categories, values = 0)
```

Arguments

`model`	A list of the class `rtmpt_model`.
`tree`	Character or numerical value of the tree for which the responses should be changed.
`categories`	Character or numerical vector identifying category/ies within the specified `tree` for which the responses should be changed.
`values`	Numerical vector of length `length(categories)` providing the responses. Default is 0.

Value

A list of the class `rtmpt_model`.

Author(s)

Raphael Hartmann

See Also

`set_params`

Examples

```
##########################################################################
# Detect-Guess variant of the Two-High Threshold model.
# The encoding and motor execution times will be set to different values
#   for each response.
##########################################################################

mdl_2HTM <- "
# targets
do+(1-do)*g
(1-do)*(1-g)

# lures
(1-dn)*g
dn+(1-dn)*(1-g)

# do: detect old; dn: detect new; g: guess
"

model <- to_rtmpt_model(mdl_file = mdl_2HTM)

## changing the model to have two different encoding and response execution
## times for "old" and "new" responses.
for(i in c(0,1)) model <- set_resps(model = model, tree = i,
                                    categories = i*2+1, values = 1)
```

`SimData` *Data simulated from the restricted 2HTM*

Description

Data set generated from a restricted Two-High Threshold model.

Usage

```
SimData
```

Format

A data frame with five variables:

`subj` subjects number

`group` group label of the subjects

`tree` condition of the current trial

`cat` observed response category

`rt` observed response time in ms

Details

Fourty subjects with thirty trials per condition (Studied items, new Items) were simulated.

Examples

```
####################################################################
# Detect-Guess variant of the restricted Two-High Threshold model.
####################################################################

head(SimData)

mdl_2HTM <- "
# targets
d+(1-d)*g     ; 0
(1-d)*(1-g)   ; 1

# lures
(1-d)*g       ; 0
d+(1-d)*(1-g) ; 1

# d: detect; g: guess
"

model <- to_rtmpt_model(mdl_file = mdl_2HTM)

data <- to_rtmpt_data(raw_data = SimData, model = model)
```

```
# this might take some time to run
rtmpt_out <- fit_rtmpt(model = model, data = data)

# convergence
## traceplot and summary of the first six parameters
coda::traceplot(rtmpt_out$samples[,1:6])
summary(rtmpt_out)
```

`sim_rtmpt_data` *Simulate data from RT-MPT models*

Description

Simulate data from RT-MPT models using `rtmpt_model` objects. You can specify the random seed, number of subjects, number of trials per tree, and some parameters (mainly the same as `prior_params` from `fit_rtmpt`).

Usage

```
sim_rtmpt_data(model, seed, n.subj, n.trials, params = NULL)
```

Arguments

`model`	A list of the class `rtmpt_model`.
`seed`	Random seed number.
`n.subj`	Number of subjects.
`n.trials`	Number of trials per tree.
`params`	Named list of parameters from which the data will be generated. This must be the same named list as `prior_params` from `fit_rtmpt`, except for "mean_of_mu_alpha" and "var_of_mu_alpha", and has the same defaults. The difference to `prior_params` is, that vectors are allowed, but must match the length of the parameters in the `model`. It is not recommended to use the defaults since they lead to many probabilities close or equal to `0` and/or `1` and to RTs close or equal to `0`. Allowed parameters are: • `mean_of_mu_alpha`: Probit transformed mean probability. If you want to have a group-level mean probability of `0.6`, use `mean_of_mu_alpha = qnorm(0.6)` in the `params` list. Default is `0` or `qnorm(.5)`. • `var_of_mu_alpha`: Variance of the probit transformed group-level mean probability. If specified, mu_alpha will be sampled from N(`mean_of_mu_alpha, var_of_mu_alpha`). If not, mu_alpha = `mean_of_mu_alpha`. • `mean_of_exp_mu_beta`: This is the expected exponential rate (`E(exp(beta)) = E(lambda)`) and `1/mean_of_exp_mu_beta` is the expected process time (`1/E(exp(beta)) = E(tau)`). The default mean is set to `10`, such that the expected process time is `0.1` seconds. For a mean process time of `200` ms, wirte `mean_of_exp_mu_beta = 1000/200`.

- `var_of_exp_mu_beta`: The group-specific variance of the exponential rates. Since `exp(mu_beta)` is Gamma distributed, the rate of the distribution is just mean divided by variance and the shape is the mean times the rate. If specified, exp(mu_beta) is sampled from Gammashape `= mean_of_exp_mu_beta^2/var_of_exp_m` `rate = mean_of_exp_mu_beta/var_of_exp_mu_beta`). If not, mu_alpha `= mean_of_exp_mu_beta`.
- `mean_of_mu_gamma`: This is the expected *mean parameter* of the encoding and response execution times, which follow a normal distribution truncated from below at zero, so `E(mu_gamma) < E(gamma)`. The default is `0`. For a mean motor time of `550` ms write `mean_of_mu_gamma = 550/1000`.
- `var_of_mu_gamma`: The group-specific variance of the *mean parameter*. If specified, mu_gamma is sampled from N(`mean_of_mu_gamma, var_of_mu_gamma`). If not, mu_gamma `= mean_of_mu_gamma`.
- `mean_of_omega_sqr`: This is the expected residual variance (`E(omega^2)`). The default is `0.005`.
- `var_of_omega_sqr`: The variance of the residual variance (`Var(omega^2)`). If specified, omega_sqr is sampled from GAMMA(`shape = mean_of_omega_sqr^2/var_of_omega_` `rate = mean_of_omega_sqr/var_of_omega_sqr`). If not, omega_sqr `= mean_of_omega_sqr`. `0.01`. The default of the mean and variance is equivalent to a shape and rate of `0.0025` and `0.5`, respectivly.
- `df_of_sigma_sqr`: Degrees of freedom for the individual variance of the response executions. The individual variance follows a scaled inverse chi-squared distribution with `df_of_sigma_sqr` degrees of freedom and `omega^2` as scale. `2` is the default and it should be an integer.
- `sf_of_scale_matrix_SIGMA`: The original scaling matrix (S) of the (scaled) inverse Wishart distribution for the process related parameters is an identity matrix `S=I`. `sf_of_scale_matrix_SIGMA` is a scaling factor, that scales this matrix (`S=sf_of_scale_matrix_SIGMA*I`). Its default is `1`.
- `sf_of_scale_matrix_GAMMA`: The original scaling matrix (S) of the (scaled) inverse Wishart distribution for the encoding and motor execution parameters is an identity matrix `S=I`. `sf_of_scale_matrix_GAMMA` is a scaling factor that scales this matrix (`S=sf_of_scale_matrix_GAMMA*I`). Its default is `1`.
- `prec_epsilon`: This is epsilon in the paper. It is the precision of xi (scaling parameter in the scaled inverse Wishart distribution). Its default is also `1`.
- `add_df_to_invWish`: If `P` is the number of parameters or rather the size of the scale matrix used in the (scaled) inverse Wishart distribution then `add_df_to_invWish` is the number of degrees of freedom that can be added to it. So `DF = P + add_df_to_invWish`. The default for `add_df_to_invWish` is `1`, such that the correlations are uniformly distributed within `[-1, 1]`.
- `SIGMA`: Variance-covariance matrix of the process-related parameters. It must match the number of process-related parameters to be estimated. If scalars or vectors are given, they will be transformed into diagonal matrices using `diag(SIGMA)`. If not specified it will be randomly generated using `diag(xi)%*%rinvwishart(nu, S)%*%diag(xi)`, where nu is the number of process-related group-level parameters to be estimated plus `add_df_to_invWish`, S is the identity matrix multiplied by `sf_of_scale_matrix_SIGMA`, and xi

(randomly generated from N(`1, 1/prec_epsilon`)) are the scaling factors for the scaled inverse wishart distribution. If `SIGMA` is used, `sf_of_scale_matrix_SIGMA` and `add_df_to_invWish` will be ignored for the process-related parameters.

- `GAMMA`: Variance-covariance matrix of the motor time parameters. It must match the number of motor time parameters to be estimated. If scalars or vectors are given, they will be transformed into diagonal matrices using `diag(SIGMA)`. If not specified it will be randomly generated using `diag(xi)%*%rinvwishart(nu, S)%*%diag(xi)`, where nu is the number of motor time group-level parameters to be estimated plus `add_df_to_invWish`, S is the identity matrix multiplied by `sf_of_scale_matrix_GAMMA`, and xi (randomly generated from N(`1, 1/prec_epsilon`)) are the scaling factors for the scaled inverse wishart distribution. If `GAMMA` is used, `sf_of_scale_matrix_GAMMA` and `add_df_to_invWish` will be ignored for the motor time parameters.

Value

A list of the class `rtmpt_sim` containing

- `data`: the data.frame with the simulated data,
- `gen_list`: a list containing lists of the group-level and subject-specific parameters for the process-related parameters and the motor-related parameters, and the trial-specific probabilities, process-times, and motor-times,
- `specs`: some specifications like the model, seed number, etc.,

Author(s)

Raphael Hartmann

Examples

```
########################################################################################
# Detect-Guess variant of the Two-High Threshold model.
# The encoding and motor execution times are assumed to be different for each response.
########################################################################################

mdl_2HTM <- "
# targets
do+(1-do)*g     ; 0
(1-do)*(1-g)    ; 1

# lures
(1-dn)*g        ; 0
dn+(1-dn)*(1-g) ; 1

# do: detect old; dn: detect new; g: guess
"

model <- to_rtmpt_model(mdl_file = mdl_2HTM)
```

```
# random group-level parameters
params <- list(mean_of_mu_alpha = 0,
               #var_of_mu_alpha = 1
               mean_of_exp_mu_beta = 10,
               var_of_exp_mu_beta = 10,
               mean_of_mu_gamma = 0.5,
               var_of_mu_gamma = 0.0025,
               mean_of_omega_sqr = 0.005,
               var_of_omega_sqr = 0.000025,
               df_of_sigma_sqr = 10,
               sf_of_scale_matrix_SIGMA = 0.1,
               sf_of_scale_matrix_GAMMA = 0.01,
               prec_epsilon = 10,
               add_df_to_invWish = 5)

sim_dat <- sim_rtmpt_data(model, seed = 123, n.subj = 40, n.trials = 30, params = params)

# fixed group-level parameters
params <- list(mean_of_mu_alpha = 0,
               mean_of_exp_mu_beta = 10,
               mean_of_mu_gamma = 0.5,
               mean_of_omega_sqr = 0.005,
               df_of_sigma_sqr = 10,
               sf_of_scale_matrix_SIGMA = 0.1,
               sf_of_scale_matrix_GAMMA = 0.01,
               prec_epsilon = 10,
               add_df_to_invWish = 5,
               SIGMA = diag(9),   # independent process-related params
               GAMMA = diag(2))   # independent motor time params

sim_dat <- sim_rtmpt_data(model, seed = 123, n.subj = 40, n.trials = 30, params = params)
```

`sim_rtmpt_data_SBC` *Simulate data from an RT-MPT model*

Description

Simulate data from RT-MPT models using `rtmpt_model` objects. The difference to `sim_rtmpt_data` is that here only scalars are allowed. This makes it usable for simulation-based calibration (SBC; Talts et al., 2018). You can specify the random seed, number of subjects, number of trials, and some parameters (same as `prior_params` from `fit_rtmpt`).

Usage

```
sim_rtmpt_data_SBC(model, seed, n.subj, n.trials, params = NULL)
```

Arguments

`model`	A list of the class `rtmpt_model`.
`seed`	Random seed number.
`n.subj`	<- Number of subjects.
`n.trials`	<- Number of trials per tree.
`params`	Named list of parameters from which the data will be generated. This must be the same named list as `prior_params` from `fit_rtmpt` and has the same defaults. It is not recommended to use the defaults since they lead to many probabilities close or equal to `0` and/or `1` and to RTs close or equal to `0`. Allowed parameters are: • `mean_of_exp_mu_beta`: This is the expected exponential rate (`E(exp(beta))` `= E(lambda)`) and `1/mean_of_exp_mu_beta` is the expected process time (`1/E(exp(beta)) = E(tau)`). The default mean is set to `10`, such that the expected process time is `0.1` seconds. • `var_of_exp_mu_beta`: The group-specific variance of the exponential rates. Since `exp(mu_beta)` is Gamma distributed, the rate of the distribution is just mean divided by variance and the shape is the mean times the rate. The default is set to `100`. • `mean_of_mu_gamma`: This is the expected *mean parameter* of the encoding and response execution times, which follow a normal distribution truncated from below at zero, so `E(mu_gamma) < E(gamma)`. The default is `0`. • `var_of_mu_gamma`: The group-specific variance of the *mean parameter*. Its default is `10`. • `mean_of_omega_sqr`: This is the expected residual variance (`E(omega^2)`). The default is `0.005`. • `var_of_omega_sqr`: The variance of the residual variance (`Var(omega^2)`). The default is `0.01`. The default of the mean and variance is equivalent to a shape and rate of `0.0025` and `0.5`, respectivly. • `df_of_sigma_sqr`: degrees of freedom for the individual variance of the response executions. The individual variance follows a scaled inverse chi-squared distribution with `df_of_sigma_sqr` degrees of freedom and `omega^2` as scale. `2` is the default and it should be an integer. • `sf_of_scale_matrix_SIGMA`: The original scaling matrix (S) of the (scaled) inverse Wishart distribution for the process related parameters is an identity matrix `S=I`. `sf_of_scale_matrix_SIGMA` is a scaling factor, that scales this matrix (`S=sf_of_scale_matrix_SIGMA*I`). Its default is `1`. • `sf_of_scale_matrix_GAMMA`: The original scaling matrix (S) of the (scaled) inverse Wishart distribution for the encoding and motor execution parameters is an identity matrix `S=I`. `sf_of_scale_matrix_GAMMA` is a scaling factor that scales this matrix (`S=sf_of_scale_matrix_GAMMA*I`). Its default is `1`. • `prec_epsilon`: This is epsilon in the paper. It is the precision of mu_alpha and all xi (scaling parameter in the scaled inverse Wishart distribution). Its default is also `1`.

- `add_df_to_invWish`: If `P` is the number of parameters or rather the size of the scale matrix used in the (scaled) inverse Wishart distribution then `add_df_to_invWish` is the number of degrees of freedom that can be added to it. So `DF = P + add_df_to_invWish`. The default for `add_df_to_invWish` is `1`, such that the correlations are uniformly distributed within `[-1, 1]`.

Value

A list of the class `rtmpt_sim` containing

- `data`: the data.frame with the simulated data,
- `gen_list`: a list containing lists of the group-level and subject-specific parameters for the process-related parameters and the motor-related parameters, and the trial-specific probabilities, process-times, and motor-times,
- `specs`: some specifications like the model, seed number, etc.,

Author(s)

Raphael Hartmann

References

Talts, S., Betancourt, M., Simpson, D., Vehtari, A., & Gelman, A. (2018). Validating Bayesian inference algorithms with simulation-based calibration. *arXiv preprint arXiv:1804.06788*.

Examples

```
#######################################################################################
# Detect-Guess variant of the Two-High Threshold model.
# The encoding and motor execution times are assumed to be different for each response.
#######################################################################################

mdl_2HTM <- "
# targets
do+(1-do)*g     ; 0
(1-do)*(1-g)    ; 1

# lures
(1-dn)*g        ; 0
dn+(1-dn)*(1-g) ; 1

# do: detect old; dn: detect new; g: guess
"

model <- to_rtmpt_model(mdl_file = mdl_2HTM)

params <- list(mean_of_exp_mu_beta = 10,
               var_of_exp_mu_beta = 10,
               mean_of_mu_gamma = 0.5,
               var_of_mu_gamma = 0.0025,
               mean_of_omega_sqr = 0.005,
```

```
                var_of_omega_sqr = 0.000025,
                df_of_sigma_sqr = 10,
                sf_of_scale_matrix_SIGMA = 0.1,
                sf_of_scale_matrix_GAMMA = 0.01,
                prec_epsilon = 10,
                add_df_to_invWish = 5)

sim_dat <- rtmpt:::sim_rtmpt_data_SBC(model, seed = 123, n.subj = 40,
                                     n.trials = 30, params = params)
```

`tau2tau` *Set process completion times equal*

Description

Setting multiple process completion times (taus) equal. This means all process times of negative outcomes will be set equal and all process times of positive outcomes will be set equal. Only two process times (one for the negative and one for the positive outcome) of the named processes will be estimated. The equality can be removed by just naming only one process name.

Usage

```
tau2tau(model, names, keep_zeros = FALSE)

set_taus_equal(model, names, keep_zeros = FALSE)
```

Arguments

`model`	A list of the class `rtmpt_model`.
`names`	Character vector giving the names of the processes for which the process completion times should be equal. If `length(names) = 1` then the corresponding process completion times (for negative and positive outcomes) will be estimates (i.e., they will be set to NA)
`keep_zeros`	Can be one of the following • logical value: `FALSE` (default) means none of the zeros for `names` in the `model` will be kept; The times of the reference process (i.e., first of `names` in alphabetical order) will be set to `NA` (i.e., will be estimated) and the others will be set to the name of the reference process (i.e., will be set to equal the reference process times). `TRUE` means the zero(s) of the reference process times (if specified) is used for the same outcome of all other processes. • numeric value: index for `names`. If 1, the zero(s) of the first process in `names` (in original order defined by the user) is used for the same outcome of all other processes in `names`. If 2, the zero(s) of the second process is used. And so on.

Value

A list of the class `rtmpt_model`.

Note

If you use `theta2theta()` and `tau2tau()` with the same process names you might just change the EQN or MDL file accordingly by using the same process name for all processes which should have equal process times and probabilities.

Author(s)

Raphael Hartmann

See Also

`delta2delta`, `theta2const`, `tau2zero` and `theta2theta`

Examples

```
#######################################################################################
# Detect-Guess variant of the Two-High Threshold model.
# The encoding and motor execution times are assumed to be equal for each category.
# The process completion times for both detection processes ("do" and "dn") will be
# set equal.
#######################################################################################

mdl_2HTM <- "
# targets
do+(1-do)*g
(1-do)*(1-g)

# lures
(1-dn)*g
dn+(1-dn)*(1-g)

# do: detect old; dn: detect new; g: guess
"

model <- to_rtmpt_model(mdl_file = mdl_2HTM)

## make do = dn
new_model <- tau2tau(model = model, names = c("do", "dn"))
new_model


## make do = dn
new_model <- set_taus_equal(model = model, names = c("do", "dn"))
new_model
```

`tau2zero` *Set process completion times to zero*

Description

Setting process completion times (taus) to zero or change it back to be estimated.

Usage

```
tau2zero(model, names, outcomes, values = 0)

set_tau_zero(model, names, outcomes, values = 0)
```

Arguments

`model`	A list of the class `rtmpt_model`.
`names`	Character vector with process names.
`outcomes`	Character vector of length `length(names)` indicating for which process outcome the process completion time should be zero or changed back to be estimated. Allowed characters are: • `"minus"`: the negative outcome of the process. • `"plus"`: the positive outcome of the process.
`values`	Numerical vector of length one or `length(names)`. You have the following options for the elements of the numeric vector: • `0`: suppress the process time/rate, i.e., set the process completion time (tau) with the specified output to zero. • `NA`: estimate the process time (tau)

Value

A list of the class `rtmpt_model`.

Author(s)

Raphael Hartmann

See Also

`delta2delta`, `theta2const`, `theta2theta` and `tau2tau`

Examples

```
########################################################################################
# Detect-Guess variant of the Two-High Threshold model.
# The encoding and motor execution times are assumed to be equal for each category.
# The process completion times for both failed detections will be suppressed.
########################################################################################

mdl_2HTM <- "
# targets
do+(1-do)*g
(1-do)*(1-g)

# lures
(1-dn)*g
dn+(1-dn)*(1-g)

# do: detect old; dn: detect new; g: guess
"

model <- to_rtmpt_model(mdl_file = mdl_2HTM)

## removing the process times (tau) for the failed (="minus") detection ("do" and "dn")
new_model <- tau2zero(model = model, names = c("dn", "do"),
                      outcomes = c("minus", "minus"), values = 0)
new_model


## removing the process times (tau) for the failed (="minus") detection ("do" and "dn")
new_model <- set_tau_zero(model = model, names = c("dn", "do"),
                          outcomes = c("minus", "minus"), values = 0)
new_model
```

`theta2const` *Set process probabilities to constants*

Description

Setting process probabilities (thetas) to constants or change it back to be estimated.

Usage

```
theta2const(model, names, constants = NA)

set_theta_const(model, names, constants = NA)
```

Arguments

`model`	An object of the class `rtmpt_model`.
`names`	Character vector with process names.

`constants`	Numerical vector of length one or `length(names)`. You have the following options for the elements of the numeric vector: • `0` < `constants` < `1`: set the named probability to a constant value between zero and one • `NA`: estimate the named probability

Value

An object of the class `rtmpt_model`.

Author(s)

Raphael Hartmann

See Also

`delta2delta`, `tau2zero`, `theta2theta` and `tau2tau`

Examples

```
#####################################################################################
# Detect-Guess variant of the Two-High Threshold model.
# The encoding and motor execution times are assumed to be equal for each category.
# The process probability for guessing (g) will be set to 0.5.
#####################################################################################

mdl_2HTM <- "
# targets
do+(1-do)*g
(1-do)*(1-g)

# lures
(1-dn)*g
dn+(1-dn)*(1-g)

# do: detect old; dn: detect new; g: guess
"

model <- to_rtmpt_model(mdl_file = mdl_2HTM)

## setting g to a constant (0.5):
new_model <- theta2const(model = model, names = c("g"), constants = c(0.5))
new_model


## setting g to a constant (0.5):
new_model <- set_theta_const(model = model, names = c("g"), constants = c(0.5))
new_model
```

`theta2theta` *Set process probabilities equal*

Description

Setting multiple process probabilities (thetas) equal. One of the process probabilities will be estimated and the other named process(es) will be set to equal the former. The equality can be removed by only using one name of a process.

Usage

```
theta2theta(model, names, keep_consts = FALSE)

set_thetas_equal(model, names, keep_consts = FALSE)
```

Arguments

`model`	A list of the class `rtmpt_model`.
`names`	Character vector giving the names of the processes for which the process probabilities should be equal. If `length(names) = 1` then the corresponding process probability will be estimates (i.e., it will be set to NA)
`keep_consts`	Can be one of the following • logical value: `FALSE` (default) means none of the constants for `names` in the `model` will be kept; The probability of the reference process (i.e., first of `names` in alphabetical order) will be set to `NA` (i.e., will be estimated) and the others will be set to the name of the reference process (i.e., will be set to equal the reference process probability). `TRUE` means the constant of the reference process probability (if specified) is used for all other processes. • numeric value: index for `names`. If 1, the constant of the first process in `names` (in original order defined by the user) is used for all other probabilities of the processes in `names`. If 2, the constant of the second process is used. And so on.

Value

A list of the class `rtmpt_model`.

Note

If you use `theta2theta()` and `tau2tau()` with the same process names you might just change the EQN or MDL file accordingly by using the same process name for all processes which should have equal process times and probabilities.

Author(s)

Raphael Hartmann

See Also

`delta2delta`, `theta2const`, `tau2zero` and `tau2tau`

Examples

```
#######################################################################################
# Detect-Guess variant of the Two-High Threshold model.
# The encoding and motor execution times are assumed to be equal for each category.
# The process probabilities for both detection processes ("do" and "dn") will be
# set equal.
#######################################################################################

mdl_2HTM <- "
# targets
do+(1-do)*g
(1-do)*(1-g)

# lures
(1-dn)*g
dn+(1-dn)*(1-g)

# do: detect old; dn: detect new; g: guess
"

model <- to_rtmpt_model(mdl_file = mdl_2HTM)

## make do = dn
new_model <- theta2theta(model = model, names = c("do", "dn"))
new_model


## make do = dn
new_model <- set_thetas_equal(model = model, names = c("do", "dn"))
new_model
```

`to_rtmpt_data` *Transform data to be used in RT-MPT model fitting*

Description

Transform data, such that it can be used in `fit_rtmpt`. This implies changing each value/label in "subj", "group", "tree", and "cat" to numbers such that it starts from zero (e.g. data$tree = c(1,1,3,3,2,2,...) will be changed to data$tree = c(0,0,2,2,1,1,...)) and the columns will be ordered in the right way. "rt" must be provided in milliseconds. If it has decimal places it will be rounded to a whole number. `fit_rtmpt` will automatically call this function if its input is not already an `rtmpt_data` list, but it is advised to use it anyway because it provides information about the transformations of the data.

Usage

```
to_rtmpt_data(raw_data, model)
```

Arguments

`raw_data`	`data.frame` or path to data containing columns "subj", "group", "tree", "cat", and "rt". If not provided in this order it will be reordered and unused variables will be moved to the end of the new data frame.
`model`	A list of the class `rtmpt_model`.

Value

A list of the class `rtmpt_data` containing transformed data and information about the transformation that has been done.

Author(s)

Raphael Hartmann

Examples

```
####################################################################################
# Detect-Guess variant of the Two-High Threshold model.
# The encoding and motor execution times are assumed to be equal for each response.
####################################################################################

eqn_2HTM <- "
# CORE MPT EQN
# tree ; cat  ; mpt
target ; hit  ; do
target ; hit  ; (1-do)*g
target ; miss ; (1-do)*(1-g)

  lure ;  f_a ; (1-dn)*g
  lure ;  c_r ; dn
  lure ;  c_r ; (1-dn)*(1-g)
"

model <- to_rtmpt_model(eqn_file = eqn_2HTM)

data_file <- system.file("extdata/labeled_data.txt", package="rtmpt")
data <- read.table(file = data_file, header = TRUE)

data_list <- to_rtmpt_data(raw_data = data, model = model)
data_list
```

`to_rtmpt_model` *Create a model list to fit an RT-MPT*

Description

Create a model list of the class `rtmpt_model` by providing either `eqn_file` or `mdl_file`. If both are provided `mdl_file` will be used.

Usage

```
to_rtmpt_model(eqn_file = NULL, mdl_file = NULL)
```

Arguments

`eqn_file`	Character string as shown in example 2 or path to the text file that specifies the (RT-)MPT model with standard .eqn syntax (Heck et al., 2018; Hu, 1999). E.g. `studied ; hit ; (1-do)*g` for a correct guess in the detect-guess 2HT model.
`mdl_file`	Character string as shown in example 1 or path to the text file that specifies the (RT-)MPT model and gives on each line the equation of one category using `+` to separate branches and `*` to separate processes (Singmann and Kellen, 2013). E.g. `do+(1-do)*g` for the category "hit" in the detect-guess 2HT model.

Value

A list of the class `rtmpt_model`.

Note

Within a branch of a (RT-)MPT model it is not allowed to have the same process two or more times.

Author(s)

Raphael Hartmann

References

Heck, D. W., Arnold, N. R., & Arnold, D. (2018). TreeBUGS: An R package for hierarchical multinomial-processing-tree modeling. *Behavior Research Methods, 50(1)*, 264-284.

Hu, X. (1999). Multinomial processing tree models: An implementation. *Behavior Research Methods, Instruments, & Computers, 31(4)*, 689-695.

Singmann, H., & Kellen, D. (2013). MPTinR: Analysis of multinomial processing tree models in R. *Behavior Research Methods, 45(2)*, 560-575.

See Also

`delta2delta`, `theta2const`, `tau2zero`, `theta2theta`, and `tau2tau` for functions to change the model

Examples

```
########################################################################################
# Detect-Guess variant of the Two-High Threshold model
#   with constant guessing and
#   suppressed process completion times for both failed detections.
# The encoding and motor execution times are assumed to be different for each response.
########################################################################################

## 1. using the mdl syntax
mdl_2HTM <- "
# targets
do+(1-do)*g     ; 0
(1-do)*(1-g)    ; 1

# lures
(1-dn)*g        ; 0
dn+(1-dn)*(1-g) ; 1

# do: detect old; dn: detect new; g: guess

# OPTIONAL MPT CONSTRAINTS
#   set probabilities to constants:
const_prob: g=0.5

#   suppress process times:
suppress_process: dn-, do-
"

model <- to_rtmpt_model(mdl_file = mdl_2HTM)
model

## 2. using the eqn syntax
eqn_2HTM <- "
# CORE MPT EQN
# tree ; cat ; mpt
     0 ;   0 ; do
     0 ;   0 ; (1-do)*g
     0 ;   1 ; (1-do)*(1-g)

     1 ;   2 ; (1-dn)*g
     1 ;   3 ; dn
     1 ;   3 ; (1-dn)*(1-g)

# OPTIONAL MPT CONSTRAINTS
#   set probabilities to constants:
const_prob: g=0.5

#   suppress process times:
suppress_process: dn-, do-

#     tree ; cat ;  MAP
resp:    0 ;   0 ;    0
```

```
resp:    0 ;   1 ;    1
resp:    1 ;   2 ;    0
resp:    1 ;   3 ;    1
# different motor execution times for old and new responses.
"

model <- to_rtmpt_model(eqn_file = eqn_2HTM)
model
```

Index