Package 'hBayesDM'

November 13, 2019

Title Hierarchical Bayesian Modeling of Decision-Making Tasks

Version 1.0.2

Date 2019-11-13

Author Woo-Young Ahn [aut, cre],
Nate Haines [aut],
Lei Zhang [aut],
Harhim Park [ctb],
Jaeyeong Yang [ctb],
Jethro Lee [ctb]

Maintainer Woo-Young Ahn <wooyoung.ahn@gmail.com>

Description Fit an array of decision-making tasks with computational models in a hierarchical Bayesian framework. Can perform hierarchical Bayesian analysis of various computational models with a single line of coding (Ahn et al., 2017) <doi:10.1162/CPSY_a_00002>.

Depends R (>= 3.4.0), Rcpp (>= 0.12.0), methods

Imports rstan (>= 2.18.1), loo (>= 2.0), grid, parallel, ggplot2, data.table

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

URL https://github.com/CCS-Lab/hBayesDM

BugReports https://github.com/CCS-Lab/hBayesDM/issues

License GPL-3

LazyData true

NeedsCompilation yes

Encoding UTF-8

RoxygenNote 6.1.1

SystemRequirements GNU make

Collate 'HDIofMCMC.R' 'preprocess_funcs.R' 'stanmodels.R' 'settings.R' 'hBayesDM_model.R' 'bandit2arm_delta.R' 'bandit4arm2_kalman_filter.R' 'bandit4arm_2par_lapse.R'

'bandit4arm_4par.R' 'bandit4arm_lapse.R'
'bandit4arm_lapse_decay.R' 'bandit4arm_singleA_lapse.R'
'bart_par4.R' 'cgt_cm.R' 'choiceRT_ddm.R'
'choiceRT_ddm_single.R' 'choiceRT_lba.R'
'choiceRT_lba_single.R' 'cra_exp.R' 'cra_linear.R'
'dbdm_prob_weight.R' 'dd_cs.R' 'dd_cs_single.R' 'dd_exp.R'
'dd_hyperbolic.R' 'dd_hyperbolic_single.R' 'estimate_mode.R'
'extract_ic.R' 'gng_m1.R' 'gng_m2.R' 'gng_m3.R' 'gng_m4.R'
'hBayesDM.R' 'igt_orl.R' 'igt_pvl_decay.R' 'igt_pvl_delta.R'
'igt_vpp.R' 'multiplot.R' 'peer_ocu.R' 'plot.hBayesDM.R'
'plotDist.R' 'plotHDI.R' 'plotInd.R' 'printFit.R' 'prl_ewa.R'
'prl_fictitious.R' 'prl_fictitious_multipleB.R'
'prl_fictitious_rp.R' 'prl_fictitious_rp_woa.R'
'prl_fictitious_woa.R' 'prl_rp.R' 'prl_rp_multipleB.R'
'pst_gainloss_Q.R' 'ra_noLA.R' 'ra_noRA.R' 'ra_prospect.R'
'rdt_happiness.R' 'rhat.R' 'ts_par4.R' 'ts_par6.R' 'ts_par7.R'
'ug_bayes.R' 'ug_delta.R' 'wcs_sql.R' 'zzz.R'

Suggests testthat

Repository CRAN

Date/Publication 2019-11-13 10:40:02 UTC

R topics documented:

`hBayesDM-package` *Hierarchical Bayesian Modeling of Decision-Making Tasks*

Description

Fit an array of decision-making tasks with computational models in a hierarchical Bayesian framework. Can perform hierarchical Bayesian analysis of various computational models with a single line of coding. Bolded tasks, followed by their respective models, are itemized below.

Bandit 2-Armed Bandit (Rescorla-Wagner (delta)) — bandit2arm_delta
4-Armed Bandit with fictive updating + reward/punishment sensitvity (Rescorla-Wagner (delta)) — bandit4arm_4par
4-Armed Bandit with fictive updating + reward/punishment sensitvity + lapse (Rescorla-Wagner (delta)) — bandit4arm_lapse

Bandit2 Kalman filter — bandit4arm2_kalman_filter

Cambridge Gambling Task Cumulative Model — cgt_cm

Choice RT Drift Diffusion Model — choiceRT_ddm
Drift Diffusion Model for a single subject — choiceRT_ddm_single
Linear Ballistic Accumulator (LBA) model — choiceRT_lba
Linear Ballistic Accumulator (LBA) model for a single subject — choiceRT_lba_single

Choice under Risk and Ambiguity Exponential model — cra_exp
Linear model — cra_linear

Description-Based Decision Making probability weight function — dbdm_prob_weight

Delay Discounting Constant Sensitivity — dd_cs
Constant Sensitivity for a single subject — dd_cs_single
Exponential — dd_exp
Hyperbolic — dd_hyperbolic
Hyperbolic for a single subject — dd_hyperbolic_single

Orthogonalized Go/Nogo RW + Noise — gng_m1
RW + Noise + Bias — gng_m2
RW + Noise + Bias + Pavlovian Bias — gng_m3
RW(modified) + Noise + Bias + Pavlovian Bias — gng_m4

Iowa Gambling Outcome-Representation Learning — igt_orl
Prospect Valence Learning-DecayRI — igt_pvl_decay
Prospect Valence Learning-Delta — igt_pvl_delta
Value-Plus_Perseverance — igt_vpp

Peer influence task OCU model — peer_ocu

Probabilistic Reversal Learning Experience-Weighted Attraction — prl_ewa
Fictitious Update — prl_fictitious
Fictitious Update w/o alpha (indecision point) — prl_fictitious_woa
Fictitious Update and multiple blocks per subject — prl_fictitious_multipleB
Reward-Punishment — prl_rp
Reward-Punishment and multiple blocks per subject — prl_rp_multipleB
Fictitious Update with separate learning for Reward-Punishment — prl_fictitious_rp
Fictitious Update with separate learning for Reward-Punishment w/o alpha (indecision point) — prl_fictitious_rp_woa

Probabilistic Selection Task Q-learning with two learning rates — pst_gainloss_Q

Risk Aversion Prospect Theory (PT) — ra_prospect
PT without a loss aversion parameter — ra_noLA
PT without a risk aversion parameter — ra_noRA

Risky Decision Task Happiness model — rdt_happiness

Two-Step task Full model (7 parameters) — ts_par7
6 parameter model (without eligibility trace, lambda) — ts_par6
4 parameter model — ts_par4

Ultimatum Game Ideal Bayesian Observer — ug_bayes
Rescorla-Wagner (delta) — ug_delta

Author(s)

Woo-Young Ahn <wahn55@snu.ac.kr>

Nathaniel Haines <haines.175@osu.edu>

Lei Zhang <bnuzhanglei2008@gmail.com>

References

Please cite as: Ahn, W.-Y., Haines, N., & Zhang, L. (2017). Revealing neuro-computational mechanisms of reinforcement learning and decision-making with the hBayesDM package. *Computational Psychiatry*. 1, 24-57. https://doi.org/10.1162/CPSY_a_00002

See Also

For tutorials and further readings, visit : http://rpubs.com/CCSL/hBayesDM.

`bandit2arm_delta` *Rescorla-Wagner (Delta) Model*

Description

Hierarchical Bayesian Modeling of the 2-Armed Bandit Task using Rescorla-Wagner (Delta) Model. It has the following parameters: `A` (learning rate), `tau` (inverse temperature).

- **Task**: 2-Armed Bandit Task (Erev et al., 2010; Hertwig et al., 2004)
- **Model**: Rescorla-Wagner (Delta) Model

Usage

```
bandit2arm_delta(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "outcome". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.

nchain	Number of Markov chains to run. Defaults to 4.
ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). Not available for this model.
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the 2-Armed Bandit Task, there should be 3 columns of data with the labels "subjID", "choice", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer value representing the option chosen on the given trial: 1 or 2.

outcome Integer value representing the outcome of the given trial (where reward == 1, and loss == -1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in

samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output, type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"bandit2arm_delta").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Erev, I., Ert, E., Roth, A. E., Haruvy, E., Herzog, S. M., Hau, R., et al. (2010). A choice prediction competition: Choices from experience and from description. Journal of Behavioral Decision Making, 23(1), 15-47. http://doi.org/10.1002/bdm.683

Hertwig, R., Barron, G., Weber, E. U., & Erev, I. (2004). Decisions From Experience and the Effect of Rare Events in Risky Choice. Psychological Science, 15(8), 534-539. http://doi.org/10.1111/j.0956-7976.2004.00715.x

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- bandit2arm_delta(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- bandit2arm_delta(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`bandit4arm2_kalman_filter`
Kalman Filter

Description

Hierarchical Bayesian Modeling of the 4-Armed Bandit Task (modified) using Kalman Filter. It has the following parameters: `lambda` (decay factor), `theta` (decay center), `beta` (inverse softmax temperature), `mu0` (anticipated initial mean of all 4 options), `sigma0` (anticipated initial sd (uncertainty factor) of all 4 options), `sigmaD` (sd of diffusion noise).

- **Task**: 4-Armed Bandit Task (modified)
- **Model**: Kalman Filter (Daw et al., 2006)

Usage

```
bandit4arm2_kalman_filter(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "outcome". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred"
`adapt_delta`	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
`stepsize`	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
`max_treedepth`	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
`...`	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the 4-Armed Bandit Task (modified), there should be 3 columns of data with the labels "subjID", "choice", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer value representing the option chosen on the given trial: 1, 2, 3, or 4.

outcome Integer value representing the outcome of the given trial (where reward == 1, and loss == -1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output, type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Yoonseo Zoh <<zohyos7@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"bandit4arm2_kalman_filter").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Daw, N. D., O'Doherty, J. P., Dayan, P., Seymour, B., & Dolan, R. J. (2006). Cortical substrates for exploratory decisions in humans. Nature, 441(7095), 876-879.

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- bandit4arm2_kalman_filter(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- bandit4arm2_kalman_filter(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`bandit4arm_2par_lapse` *3 Parameter Model, without C (choice perseveration), R (reward sensitivity), and P (punishment sensitivity). But with xi (noise)*

Description

Hierarchical Bayesian Modeling of the 4-Armed Bandit Task using 3 Parameter Model, without C (choice perseveration), R (reward sensitivity), and P (punishment sensitivity). But with xi (noise). It has the following parameters: `Arew` (reward learning rate), `Apun` (punishment learning rate), `xi` (noise).

- **Task**: 4-Armed Bandit Task
- **Model**: 3 Parameter Model, without C (choice perseveration), R (reward sensitivity), and P (punishment sensitivity). But with xi (noise) (Aylward et al., 2018)

Usage

```
bandit4arm_2par_lapse(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "gain", "loss". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred"
`adapt_delta`	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
`stepsize`	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
`max_treedepth`	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
`...`	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial

observations and columns represent variables.
For the 4-Armed Bandit Task, there should be 4 columns of data with the labels "subjID", "choice", "gain", "loss". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer value representing the option chosen on the given trial: 1, 2, 3, or 4.

gain Floating point value representing the amount of currency won on the given trial (e.g. 50, 100).

loss Floating point value representing the amount of currency lost on the given trial (e.g. 0, -50).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"bandit4arm_2par_lapse").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Aylward, Valton, Ahn, Bond, Dayan, Roiser, & Robinson (2018) Altered decision-making under uncertainty in unmedicated mood and anxiety disorders. PsyArxiv. 10.31234/osf.io/k5b8m

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- bandit4arm_2par_lapse(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- bandit4arm_2par_lapse(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

bandit4arm_4par *4 Parameter Model, without C (choice perseveration)*

Description

Hierarchical Bayesian Modeling of the 4-Armed Bandit Task using 4 Parameter Model, without C (choice perseveration). It has the following parameters: `Arew` (reward learning rate), `Apun` (punishment learning rate), `R` (reward sensitivity), `P` (punishment sensitivity).

- **Task**: 4-Armed Bandit Task
- **Model**: 4 Parameter Model, without C (choice perseveration) (Seymour et al., 2012)

Usage

```
bandit4arm_4par(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

data	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, "example" to use example data, or "choose" to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "gain", "loss". See **Details** below for more information.
niter	Number of iterations, including warm-up. Defaults to 4000.
nwarmup	Number of iterations used for warm-up only. Defaults to 1000.
nchain	Number of Markov chains to run. Defaults to 4.
ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). Not available for this model.
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial

observations and columns represent variables.
For the 4-Armed Bandit Task, there should be 4 columns of data with the labels "subjID", "choice", "gain", "loss". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer value representing the option chosen on the given trial: 1, 2, 3, or 4.

gain Floating point value representing the amount of currency won on the given trial (e.g. 50, 100).

loss Floating point value representing the amount of currency lost on the given trial (e.g. 0, -50).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"bandit4arm_4par").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Seymour, Daw, Roiser, Dayan, & Dolan (2012). Serotonin Selectively Modulates Reward Value in Human Decision-Making. J Neuro, 32(17), 5833-5842.

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- bandit4arm_4par(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- bandit4arm_4par(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

bandit4arm_lapse *5 Parameter Model, without C (choice perseveration) but with xi (noise)*

Description

Hierarchical Bayesian Modeling of the 4-Armed Bandit Task using 5 Parameter Model, without C (choice perseveration) but with xi (noise). It has the following parameters: `Arew` (reward learning rate), `Apun` (punishment learning rate), `R` (reward sensitivity), `P` (punishment sensitivity), `xi` (noise).

- **Task**: 4-Armed Bandit Task
- **Model**: 5 Parameter Model, without C (choice perseveration) but with xi (noise) (Seymour et al., 2012)

Usage

```
bandit4arm_lapse(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

data	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, "example" to use example data, or "choose" to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "gain", "loss". See **Details** below for more information.
niter	Number of iterations, including warm-up. Defaults to 4000.
nwarmup	Number of iterations used for warm-up only. Defaults to 1000.
nchain	Number of Markov chains to run. Defaults to 4.
ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). Not available for this model.
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial

observations and columns represent variables.
For the 4-Armed Bandit Task, there should be 4 columns of data with the labels "subjID", "choice", "gain", "loss". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer value representing the option chosen on the given trial: 1, 2, 3, or 4.

gain Floating point value representing the amount of currency won on the given trial (e.g. 50, 100).

loss Floating point value representing the amount of currency lost on the given trial (e.g. 0, -50).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"bandit4arm_lapse").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Seymour, Daw, Roiser, Dayan, & Dolan (2012). Serotonin Selectively Modulates Reward Value in Human Decision-Making. J Neuro, 32(17), 5833-5842.

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- bandit4arm_lapse(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- bandit4arm_lapse(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

bandit4arm_lapse_decay

5 Parameter Model, without C (choice perseveration) but with xi (noise). Added decay rate (Niv et al., 2015, J. Neuro).

Description

Hierarchical Bayesian Modeling of the 4-Armed Bandit Task using 5 Parameter Model, without C (choice perseveration) but with xi (noise). Added decay rate (Niv et al., 2015, J. Neuro).. It has the following parameters: `Arew` (reward learning rate), `Apun` (punishment learning rate), `R` (reward sensitivity), `P` (punishment sensitivity), `xi` (noise), `d` (decay rate).

- **Task**: 4-Armed Bandit Task
- **Model**: 5 Parameter Model, without C (choice perseveration) but with xi (noise). Added decay rate (Niv et al., 2015, J. Neuro). (Aylward et al., 2018)

Usage

```
bandit4arm_lapse_decay(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "gain", "loss". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred"
`adapt_delta`	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
`stepsize`	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
`max_treedepth`	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
`...`	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the 4-Armed Bandit Task, there should be 4 columns of data with the labels "subjID", "choice", "gain", "loss". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer value representing the option chosen on the given trial: 1, 2, 3, or 4.

gain Floating point value representing the amount of currency won on the given trial (e.g. 50, 100).

loss Floating point value representing the amount of currency lost on the given trial (e.g. 0, -50).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"bandit4arm_lapse_decay").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Aylward, Valton, Ahn, Bond, Dayan, Roiser, & Robinson (2018) Altered decision-making under uncertainty in unmedicated mood and anxiety disorders. PsyArxiv. 10.31234/osf.io/k5b8m

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- bandit4arm_lapse_decay(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- bandit4arm_lapse_decay(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`bandit4arm_singleA_lapse`

4 Parameter Model, without C (choice perseveration) but with xi (noise). Single learning rate both for R and P.

Description

Hierarchical Bayesian Modeling of the 4-Armed Bandit Task using 4 Parameter Model, without C (choice perseveration) but with xi (noise). Single learning rate both for R and P.. It has the following parameters: `A` (learning rate), `R` (reward sensitivity), `P` (punishment sensitivity), `xi` (noise).

- **Task**: 4-Armed Bandit Task
- **Model**: 4 Parameter Model, without C (choice perseveration) but with xi (noise). Single learning rate both for R and P. (Aylward et al., 2018)

Usage

```
bandit4arm_singleA_lapse(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "gain", "loss". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.

`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred"
`adapt_delta`	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
`stepsize`	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
`max_treedepth`	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
`...`	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the 4-Armed Bandit Task, there should be 4 columns of data with the labels "subjID", "choice", "gain", "loss". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer value representing the option chosen on the given trial: 1, 2, 3, or 4.

gain Floating point value representing the amount of currency won on the given trial (e.g. 50, 100).

loss Floating point value representing the amount of currency lost on the given trial (e.g. 0, -50).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"bandit4arm_singleA_lapse").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Aylward, Valton, Ahn, Bond, Dayan, Roiser, & Robinson (2018) Altered decision-making under uncertainty in unmedicated mood and anxiety disorders. PsyArxiv. 10.31234/osf.io/k5b8m

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- bandit4arm_singleA_lapse(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- bandit4arm_singleA_lapse(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)
```

```
# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

bart_par4 *Re-parameterized version of BART model with 4 parameters*

Description

Hierarchical Bayesian Modeling of the Balloon Analogue Risk Task using Re-parameterized version of BART model with 4 parameters. It has the following parameters: `phi` (prior belief of balloon not bursting), `eta` (updating rate), `gam` (risk-taking parameter), `tau` (inverse temperature).

- **Task**: Balloon Analogue Risk Task
- **Model**: Re-parameterized version of BART model with 4 parameters (van Ravenzwaaij et al., 2011)

Usage

```
bart_par4(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "pumps", "explosion". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.

vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Balloon Analogue Risk Task, there should be 3 columns of data with the labels "subjID", "pumps", "explosion". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

pumps The number of pumps.

explosion 0: intact, 1: burst

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: adapt_delta, stepsize, and max_treedepth are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for stan for a less technical description of these arguments.

Contributors: Harhim Park <<hrpark12@gmail.com>>, Jaeyeong Yang <<jaeyeong.yang1125@gmail.com>>, Ayoung Lee <<aylee2008@naver.com>>, Jeongbin Oh <<ows0104@gmail.com>>, Jiyoon Lee <<nicole.lee2001@gmail.com>>, Junha Jang <<andy627robo@naver.com>>

Value

A class "hBayesDM" object modelData with the following components:

model Character value that is the name of the model (\code"bart_par4").

allIndPars Data.frame containing the summarized parameter values (as specified by indPars) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class stanfit object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

van Ravenzwaaij, D., Dutilh, G., & Wagenmakers, E. J. (2011). Cognitive model decomposition of the BART: Assessment and application. Journal of Mathematical Psychology, 55(1), 94-105.

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- bart_par4(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- bart_par4(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")
```

```
# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

cgt_cm *Cumulative Model*

Description

Hierarchical Bayesian Modeling of the Cambridge Gambling Task using Cumulative Model. It has the following parameters: alpha (probability distortion), c (color bias), rho (relative loss sensitivity), beta (discounting rate), gamma (choice sensitivity).

- **Task**: Cambridge Gambling Task (Rogers et al., 1999)
- **Model**: Cumulative Model

Usage

```
cgt_cm(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

data	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, "example" to use example data, or "choose" to choose data with an interactive window. Columns in the dataset must include: "subjID", "gamble_type", "percentage_staked", "trial_initial_points", "assessment_stage", "red_chosen", "n_red_boxes". See **Details** below for more information.
niter	Number of iterations, including warm-up. Defaults to 4000.
nwarmup	Number of iterations used for warm-up only. Defaults to 1000.
nchain	Number of Markov chains to run. Defaults to 4.
ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.

indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). For this model they are: "y_hat_col", "y_hat_bet", "bet_utils".
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. Not available for this model.
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Cambridge Gambling Task, there should be 7 columns of data with the labels "subjID", "gamble_type", "percentage_staked", "trial_initial_points", "assessment_stage", "red_chosen", "n_red_boxes". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

gamble_type Integer value representng whether the bets on the current trial were presented in descending (0) or ascending (1) order.

percentage_staked Integer value representing the bet percentage (not proportion) selected on the current trial: 5, 25, 50, 75, or 95.

trial_initial_points Floating point value representing the number of points that the subject has at the start of the current trial (e.g., 100, 150, etc.).

assessment_stage Integer value representing whether the current trial is a practice trial (0) or a test trial (1). Only test trials are used for model fitting.

red_chosen Integer value representing whether the red color was chosen (1) versus the blue color (0).

n_red_boxes Integer value representing the number of red boxes shown on the current trial: 1, 2, 3,..., or 9.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the

necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Nathaniel Haines <<haines.175@osu.edu>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"cgt_cm").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Rogers, R. D., Everitt, B. J., Baldacchino, A., Blackshaw, A. J., Swainson, R., Wynne, K., Baker, N. B., Hunter, J., Carthy, T., London, M., Deakin, J. F. W., Sahakian, B. J., Robbins, T. W. (1999). Dissociable deficits in the decision-making cognition of chronic amphetamine abusers, opiate abusers, patients with focal damage to prefrontal cortex, and tryptophan-depleted normal volunteers: evidence for monoaminergic mechanisms. Neuropsychopharmacology, 20, 322–339.

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- cgt_cm(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- cgt_cm(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

choiceRT_ddm *Drift Diffusion Model*

Description

Hierarchical Bayesian Modeling of the Choice Reaction Time Task using Drift Diffusion Model. It has the following parameters: `alpha` (boundary separation), `beta` (bias), `delta` (drift rate), `tau` (non-decision time).

- **Task**: Choice Reaction Time Task
- **Model**: Drift Diffusion Model (Ratcliff, 1978)

Usage

```
choiceRT_ddm(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "RT". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. Not available for this model.
`adapt_delta`	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
`stepsize`	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
`max_treedepth`	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
`...`	For this model, it's possible to set **model-specific argument(s)** as follows:
	RTbound Floating point value representing the lower bound (i.e., minimum allowed) reaction time. Defaults to 0.1 (100 milliseconds).

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Choice Reaction Time Task, there should be 3 columns of data with the labels "subjID", "choice", "RT". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Choice made for the current trial, coded as 1/2 to indicate lower/upper boundary or left/right choices (e.g., 1 1 1 2 1 2).

RT Choice reaction time for the current trial, in **seconds** (e.g., 0.435 0.383 0.314 0.309, etc.).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"choiceRT_ddm").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

Note

Notes: Note that this implementation is NOT the full Drift Diffusion Model as described in Ratcliff (1978). This implementation estimates the drift rate, boundary separation, starting point, and non-decision time; but not the between- and within-trial variances in these parameters. Code for this model is based on codes/comments by Guido Biele, Joseph Burling, Andrew Ellis, and potential others @ Stan mailing.

References

Ratcliff, R. (1978). A theory of memory retrieval. Psychological Review, 85(2), 59-108. http://doi.org/10.1037/0033-295X.85.2.59

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- choiceRT_ddm(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- choiceRT_ddm(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`choiceRT_ddm_single` *Drift Diffusion Model*

Description

Individual Bayesian Modeling of the Choice Reaction Time Task using Drift Diffusion Model. It has the following parameters: `alpha` (boundary separation), `beta` (bias), `delta` (drift rate), `tau` (non-decision time).

- **Task**: Choice Reaction Time Task
- **Model**: Drift Diffusion Model (Ratcliff, 1978)

Usage

```
choiceRT_ddm_single(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "RT". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. Not available for this model.
`adapt_delta`	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
`stepsize`	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
`max_treedepth`	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.

... For this model, it's possible to set **model-specific argument(s)** as follows:

RTbound Floating point value representing the lower bound (i.e., minimum allowed) reaction time. Defaults to 0.1 (100 milliseconds).

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Choice Reaction Time Task, there should be 3 columns of data with the labels "subjID", "choice", "RT". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Choice made for the current trial, coded as 1/2 to indicate lower/upper boundary or left/right choices (e.g., 1 1 1 2 1 2).

RT Choice reaction time for the current trial, in **seconds** (e.g., 0.435 0.383 0.314 0.309, etc.).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"choiceRT_ddm_single").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

Note

Notes: Note that this implementation is NOT the full Drift Diffusion Model as described in Ratcliff (1978). This implementation estimates the drift rate, boundary separation, starting point, and non-decision time; but not the between- and within-trial variances in these parameters. Code for this model is based on codes/comments by Guido Biele, Joseph Burling, Andrew Ellis, and potential others @ Stan mailing.

References

Ratcliff, R. (1978). A theory of memory retrieval. Psychological Review, 85(2), 59-108. http://doi.org/10.1037/0033-295X.85.2.59

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- choiceRT_ddm_single(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- choiceRT_ddm_single(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)
```

```
# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`cra_exp` *Exponential Subjective Value Model*

Description

Hierarchical Bayesian Modeling of the Choice Under Risk and Ambiguity Task using Exponential Subjective Value Model. It has the following parameters: `alpha` (risk attitude), `beta` (ambiguity attitude), `gamma` (inverse temperature).

- **Task**: Choice Under Risk and Ambiguity Task
- **Model**: Exponential Subjective Value Model (Hsu et al., 2005)

Usage

```
cra_exp(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "prob", "ambig", "reward_var", "reward_fix", "choice". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). For this model they are: "sv", "sv_fix", "sv_var", "p_var".

vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Choice Under Risk and Ambiguity Task, there should be 6 columns of data with the labels "subjID", "prob", "ambig", "reward_var", "reward_fix", "choice". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

prob Objective probability of the variable lottery.

ambig Ambiguity level of the variable lottery (0 for risky lottery; greater than 0 for ambiguous lottery).

reward_var Amount of reward in variable lottery. Assumed to be greater than zero.

reward_fix Amount of reward in fixed lottery. Assumed to be greater than zero.

choice If the variable lottery was selected, choice == 1; otherwise choice == 0.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The nwarmup argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple

chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Jaeyeong Yang `<<jaeyeong.yang1125@gmail.com>>`

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"cra_exp").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Hsu, M., Bhatt, M., Adolphs, R., Tranel, D., & Camerer, C. F. (2005). Neural systems responding to degrees of uncertainty in human decision-making. Science, 310(5754), 1680-1683. https://doi.org/10.1126/science.1115327

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: `https://rpubs.com/CCSL/hBayesDM`

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- cra_exp(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- cra_exp(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)
```

```
# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`cra_linear` *Linear Subjective Value Model*

Description

Hierarchical Bayesian Modeling of the Choice Under Risk and Ambiguity Task using Linear Subjective Value Model. It has the following parameters: `alpha` (risk attitude), `beta` (ambiguity attitude), `gamma` (inverse temperature).

- **Task**: Choice Under Risk and Ambiguity Task
- **Model**: Linear Subjective Value Model (Levy et al., 2010)

Usage

```
cra_linear(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "prob", "ambig", "reward_var", "reward_fix", "choice". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.

nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). For this model they are: "sv", "sv_fix", "sv_var", "p_var".
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Choice Under Risk and Ambiguity Task, there should be 6 columns of data with the labels "subjID", "prob", "ambig", "reward_var", "reward_fix", "choice". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

prob Objective probability of the variable lottery.

ambig Ambiguity level of the variable lottery (0 for risky lottery; greater than 0 for ambiguous lottery).

reward_var Amount of reward in variable lottery. Assumed to be greater than zero.

reward_fix Amount of reward in fixed lottery. Assumed to be greater than zero.

choice If the variable lottery was selected, choice == 1; otherwise choice == 0.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Jaeyeong Yang `<<jaeyeong.yang1125@gmail.com>>`

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"cra_linear").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Levy, I., Snell, J., Nelson, A. J., Rustichini, A., & Glimcher, P. W. (2010). Neural representation of subjective value under risk and ambiguity. Journal of Neurophysiology, 103(2), 1036-1047.

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- cra_linear(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- cra_linear(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

dbdm_prob_weight *Probability Weight Function*

Description

Hierarchical Bayesian Modeling of the Description Based Decison Making Task using Probability Weight Function. It has the following parameters: `tau` (probability weight function), `rho` (subject utility function), `lambda` (loss aversion parameter), `beta` (inverse softmax temperature).

- **Task**: Description Based Decison Making Task
- **Model**: Probability Weight Function (Erev et al., 2010; Hertwig et al., 2004; Jessup et al., 2008)

Usage

```
dbdm_prob_weight(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

data	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, "example" to use example data, or "choose" to choose data with an interactive window. Columns in the dataset must include: "subjID", "opt1hprob", "opt2hprob", "opt1hval", "opt1lval", "opt2hval", "opt2lval", "choice". See **Details** below for more information.
niter	Number of iterations, including warm-up. Defaults to 4000.
nwarmup	Number of iterations used for warm-up only. Defaults to 1000.
nchain	Number of Markov chains to run. Defaults to 4.
ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). Not available for this model.
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Description Based Decison Making Task, there should be 8 columns of data with the labels "subjID", "opt1hprob", "opt2hprob", "opt1hval", "opt1lval", "opt2hval", "opt2lval", "choice". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

opt1hprob Possiblity of getting higher value of outcome(opt1hval) when choosing option 1.

opt2hprob Possiblity of getting higher value of outcome(opt2hval) when choosing option 2.

opt1hval Possible (with opt1hprob probability) outcome of option 1.

opt1lval Possible (with (1 - opt1hprob) probability) outcome of option 1.

opt2hval Possible (with opt2hprob probability) outcome of option 2.

opt2lval Possible (with (1 - opt2hprob) probability) outcome of option 2.

choice If option 1 was selected, choice == 1; else if option 2 was selected, choice == 2.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Yoonseo Zoh <<zohyos7@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"dbdm_prob_weight").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class stanfit object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Erev, I., Ert, E., Roth, A. E., Haruvy, E., Herzog, S. M., Hau, R., ... & Lebiere, C. (2010). A choice prediction competition: Choices from experience and from description. Journal of Behavioral Decision Making, 23(1), 15-47.

Hertwig, R., Barron, G., Weber, E. U., & Erev, I. (2004). Decisions from experience and the effect of rare events in risky choice. Psychological science, 15(8), 534-539.

Jessup, R. K., Bishara, A. J., & Busemeyer, J. R. (2008). Feedback produces divergence from prospect theory in descriptive choice. Psychological Science, 19(10), 1015-1022.

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- dbdm_prob_weight(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- dbdm_prob_weight(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

dd_cs *Constant-Sensitivity (CS) Model*

Description

Hierarchical Bayesian Modeling of the Delay Discounting Task using Constant-Sensitivity (CS) Model. It has the following parameters: `r` (exponential discounting rate), `s` (impatience), `beta` (inverse temperature).

- **Task**: Delay Discounting Task
- **Model**: Constant-Sensitivity (CS) Model (Ebert et al., 2007)

Usage

```
dd_cs(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", "choice". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred"

adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Delay Discounting Task, there should be 6 columns of data with the labels "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", "choice". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

delay_later An integer representing the delayed days for the later option (e.g. 1, 6, 28).

amount_later A floating point number representing the amount for the later option (e.g. 10.5, 13.4, 30.9).

delay_sooner An integer representing the delayed days for the sooner option (e.g. 0).

amount_sooner A floating point number representing the amount for the sooner option (e.g. 10).

choice If amount_later was selected, choice == 1; else if amount_sooner was selected, choice == 0.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"dd_cs").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Ebert, J. E. J., & Prelec, D. (2007). The Fragility of Time: Time-Insensitivity and Valuation of the Near and Far Future. Management Science. http://doi.org/10.1287/mnsc.1060.0671

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- dd_cs(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- dd_cs(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
```

```
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`dd_cs_single` *Constant-Sensitivity (CS) Model*

Description

Individual Bayesian Modeling of the Delay Discounting Task using Constant-Sensitivity (CS) Model. It has the following parameters: `r` (exponential discounting rate), `s` (impatience), `beta` (inverse temperature).

- **Task**: Delay Discounting Task
- **Model**: Constant-Sensitivity (CS) Model (Ebert et al., 2007)

Usage

```
dd_cs_single(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", "choice". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.

indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). Not available for this model.
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Delay Discounting Task, there should be 6 columns of data with the labels "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", "choice". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

delay_later An integer representing the delayed days for the later option (e.g. 1, 6, 28).

amount_later A floating point number representing the amount for the later option (e.g. 10.5, 13.4, 30.9).

delay_sooner An integer representing the delayed days for the sooner option (e.g. 0).

amount_sooner A floating point number representing the amount for the sooner option (e.g. 10).

choice If amount_later was selected, choice == 1; else if amount_sooner was selected, choice == 0.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The nwarmup argument

can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"dd_cs_single").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Ebert, J. E. J., & Prelec, D. (2007). The Fragility of Time: Time-Insensitivity and Valuation of the Near and Far Future. Management Science. http://doi.org/10.1287/mnsc.1060.0671

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- dd_cs_single(
```

```
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- dd_cs_single(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

dd_exp *Exponential Model*

Description

Hierarchical Bayesian Modeling of the Delay Discounting Task using Exponential Model. It has the following parameters: `r` (exponential discounting rate), `beta` (inverse temperature).

- **Task**: Delay Discounting Task
- **Model**: Exponential Model (Samuelson, 1937)

Usage

```
dd_exp(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", "choice". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.

ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). Not available for this model.
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Delay Discounting Task, there should be 6 columns of data with the labels "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", "choice". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

delay_later An integer representing the delayed days for the later option (e.g. 1, 6, 28).

amount_later A floating point number representing the amount for the later option (e.g. 10.5, 13.4, 30.9).

delay_sooner An integer representing the delayed days for the sooner option (e.g. 0).

amount_sooner A floating point number representing the amount for the sooner option (e.g. 10).

choice If amount_later was selected, choice == 1; else if amount_sooner was selected, choice == 0.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"dd_exp").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Samuelson, P. A. (1937). A Note on Measurement of Utility. The Review of Economic Studies, 4(2), 155. http://doi.org/10.2307/2967612

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- dd_exp(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- dd_exp(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

dd_hyperbolic *Hyperbolic Model*

Description

Hierarchical Bayesian Modeling of the Delay Discounting Task using Hyperbolic Model. It has the following parameters: k (discounting rate), beta (inverse temperature).

- **Task**: Delay Discounting Task
- **Model**: Hyperbolic Model (Mazur, 1987)

Usage

```
dd_hyperbolic(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

data	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, "example" to use example data, or "choose" to choose data with an interactive window. Columns in the dataset must include: "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", "choice". See **Details** below for more information.
niter	Number of iterations, including warm-up. Defaults to 4000.
nwarmup	Number of iterations used for warm-up only. Defaults to 1000.
nchain	Number of Markov chains to run. Defaults to 4.
ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). Not available for this model.
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Delay Discounting Task, there should be 6 columns of data with the labels "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", "choice". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

delay_later An integer representing the delayed days for the later option (e.g. 1, 6, 28).

amount_later A floating point number representing the amount for the later option (e.g. 10.5, 13.4, 30.9).

delay_sooner An integer representing the delayed days for the sooner option (e.g. 0).

amount_sooner A floating point number representing the amount for the sooner option (e.g. 10).

choice If amount_later was selected, choice == 1; else if amount_sooner was selected, choice == 0.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"dd_hyperbolic").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Mazur, J. E. (1987). An adjustment procedure for studying delayed reinforcement.

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- dd_hyperbolic(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- dd_hyperbolic(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

dd_hyperbolic_single *Hyperbolic Model*

Description

Individual Bayesian Modeling of the Delay Discounting Task using Hyperbolic Model. It has the following parameters: k (discounting rate), beta (inverse temperature).

- **Task**: Delay Discounting Task
- **Model**: Hyperbolic Model (Mazur, 1987)

Usage

```
dd_hyperbolic_single(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

data	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, "example" to use example data, or "choose" to choose data with an interactive window. Columns in the dataset must include: "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", "choice". See **Details** below for more information.
niter	Number of iterations, including warm-up. Defaults to 4000.
nwarmup	Number of iterations used for warm-up only. Defaults to 1000.
nchain	Number of Markov chains to run. Defaults to 4.
ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). Not available for this model.
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for

the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Delay Discounting Task, there should be 6 columns of data with the labels "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", "choice". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

delay_later An integer representing the delayed days for the later option (e.g. 1, 6, 28).

amount_later A floating point number representing the amount for the later option (e.g. 10.5, 13.4, 30.9).

delay_sooner An integer representing the delayed days for the sooner option (e.g. 0).

amount_sooner A floating point number representing the amount for the sooner option (e.g. 10).

choice If amount_later was selected, choice == 1; else if amount_sooner was selected, choice == 0.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"dd_hyperbolic_single").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Mazur, J. E. (1987). An adjustment procedure for studying delayed reinforcement.

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- dd_hyperbolic_single(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- dd_hyperbolic_single(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

estimate_mode *Function to estimate mode of MCMC samples*

Description

Based on codes from 'http://stackoverflow.com/questions/2547402/is-there-a-built-in-function-for-finding-the-mode' see the comment by Rasmus Baath

Usage

```
estimate_mode(x)
```

Arguments

x	MCMC samples or some numeric or array values.

---

extract_ic *Extract Model Comparison Estimates*

Description

Extract Model Comparison Estimates

Usage

```
extract_ic(model_data = NULL, ic = "looic", ncore = 2)
```

Arguments

model_data	Object returned by 'hBayesDM' model function
ic	Information Criterion. 'looic', 'waic', or 'both'
ncore	Number of cores to use when computing LOOIC

Value

IC Leave-One-Out and/or Watanabe-Akaike information criterion estimates.

Examples

```
## Not run:
library(hBayesDM)
output = bandit2arm_delta("example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 1)
# To show the LOOIC model fit estimates (a detailed report; c)
extract_ic(output)
# To show the WAIC model fit estimates
extract_ic(output, ic = "waic")

## End(Not run)
```

gng_m1 *RW + noise*

Description

Hierarchical Bayesian Modeling of the Orthogonalized Go/Nogo Task using RW + noise. It has the following parameters: `xi` (noise), `ep` (learning rate), `rho` (effective size).

- **Task**: Orthogonalized Go/Nogo Task
- **Model**: RW + noise (Guitart-Masip et al., 2012)

Usage

```
gng_m1(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "cue", "keyPressed", "outcome". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). For this model they are: "Qgo", "Qnogo", "Wgo", "Wnogo".
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred"
`adapt_delta`	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.

stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Orthogonalized Go/Nogo Task, there should be 4 columns of data with the labels "subjID", "cue", "keyPressed", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

cue Nominal integer representing the cue shown for that trial: 1, 2, 3, or 4.

keyPressed Binary value representing the subject's response for that trial (where Press == 1; No press == 0).

outcome Ternary value representing the outcome of that trial (where Positive feedback == 1; Neutral feedback == 0; Negative feedback == -1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to

'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"gng_m1").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Guitart-Masip, M., Huys, Q. J. M., Fuentemilla, L., Dayan, P., Duzel, E., & Dolan, R. J. (2012). Go and no-go learning in reward and punishment: Interactions between affect and effect. Neuroimage, 62(1), 154-166. http://doi.org/10.1016/j.neuroimage.2012.04.024

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- gng_m1(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- gng_m1(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
```

```
printFit(output)

## End(Not run)
```

---

gng_m2 *RW + noise + bias*

Description

Hierarchical Bayesian Modeling of the Orthogonalized Go/Nogo Task using RW + noise + bias. It has the following parameters: `xi` (noise), `ep` (learning rate), `b` (action bias), `rho` (effective size).

- **Task**: Orthogonalized Go/Nogo Task
- **Model**: RW + noise + bias (Guitart-Masip et al., 2012)

Usage

```
gng_m2(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "cue", "keyPressed", "outcome". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). For this model they are: "Qgo", "Qnogo", "Wgo", "Wnogo".
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.

inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Orthogonalized Go/Nogo Task, there should be 4 columns of data with the labels "subjID", "cue", "keyPressed", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

cue Nominal integer representing the cue shown for that trial: 1, 2, 3, or 4.

keyPressed Binary value representing the subject's response for that trial (where Press == 1; No press == 0).

outcome Ternary value representing the outcome of that trial (where Positive feedback == 1; Neutral feedback == 0; Negative feedback == -1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The nwarmup argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"gng_m2").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Guitart-Masip, M., Huys, Q. J. M., Fuentemilla, L., Dayan, P., Duzel, E., & Dolan, R. J. (2012). Go and no-go learning in reward and punishment: Interactions between affect and effect. Neuroimage, 62(1), 154-166. http://doi.org/10.1016/j.neuroimage.2012.04.024

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- gng_m2(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- gng_m2(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
```

```
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

gng_m3 *RW + noise + bias + pi*

Description

Hierarchical Bayesian Modeling of the Orthogonalized Go/Nogo Task using RW + noise + bias + pi. It has the following parameters: `xi` (noise), `ep` (learning rate), `b` (action bias), `pi` (Pavlovian bias), `rho` (effective size).

- **Task**: Orthogonalized Go/Nogo Task
- **Model**: RW + noise + bias + pi (Guitart-Masip et al., 2012)

Usage

```
gng_m3(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "cue", "keyPressed", "outcome". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). For this model they are: "Qgo", "Qnogo", "Wgo", "Wnogo", "SV".

vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Orthogonalized Go/Nogo Task, there should be 4 columns of data with the labels "subjID", "cue", "keyPressed", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

cue Nominal integer representing the cue shown for that trial: 1, 2, 3, or 4.

keyPressed Binary value representing the subject's response for that trial (where Press == 1; No press == 0).

outcome Ternary value representing the outcome of that trial (where Positive feedback == 1; Neutral feedback == 0; Negative feedback == -1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The nwarmup argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"gng_m3").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Guitart-Masip, M., Huys, Q. J. M., Fuentemilla, L., Dayan, P., Duzel, E., & Dolan, R. J. (2012). Go and no-go learning in reward and punishment: Interactions between affect and effect. Neuroimage, 62(1), 154-166. http://doi.org/10.1016/j.neuroimage.2012.04.024

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- gng_m3(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- gng_m3(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")
```

```
# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

gng_m4 *RW (rew/pun) + noise + bias + pi*

Description

Hierarchical Bayesian Modeling of the Orthogonalized Go/Nogo Task using RW (rew/pun) + noise + bias + pi. It has the following parameters: `xi` (noise), `ep` (learning rate), `b` (action bias), `pi` (Pavlovian bias), `rhoRew` (reward sensitivity), `rhoPun` (punishment sensitivity).

- **Task**: Orthogonalized Go/Nogo Task
- **Model**: RW (rew/pun) + noise + bias + pi (Cavanagh et al., 2013)

Usage

```
gng_m4(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "cue", "keyPressed", "outcome". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.

indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). For this model they are: "Qgo", "Qnogo", "Wgo", "Wnogo", "SV".
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Orthogonalized Go/Nogo Task, there should be 4 columns of data with the labels "subjID", "cue", "keyPressed", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

cue Nominal integer representing the cue shown for that trial: 1, 2, 3, or 4.

keyPressed Binary value representing the subject's response for that trial (where Press == 1; No press == 0).

outcome Ternary value representing the outcome of that trial (where Positive feedback == 1; Neutral feedback == 0; Negative feedback == -1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The nwarmup argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated

from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"gng_m4").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Cavanagh, J. F., Eisenberg, I., Guitart-Masip, M., Huys, Q., & Frank, M. J. (2013). Frontal Theta Overrides Pavlovian Learning Biases. Journal of Neuroscience, 33(19), 8541-8548. http://doi.org/10.1523/JNEUROSCI.5754-12.2013

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- gng_m4(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- gng_m4(
```

```
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`HDIofMCMC` *Compute Highest-Density Interval*

Description

Computes the highest density interval from a sample of representative values, estimated as shortest credible interval. Based on John Kruschke's codes.

Usage

```
HDIofMCMC(sampleVec, credMass = 0.95)
```

Arguments

`sampleVec`	A vector of representative values from a probability distribution (e.g., MCMC samples).
`credMass`	A scalar between 0 and 1, indicating the mass within the credible interval that is to be estimated.

Value

A vector containing the limits of the HDI

igt_orl *Outcome-Representation Learning Model*

Description

Hierarchical Bayesian Modeling of the Iowa Gambling Task using Outcome-Representation Learning Model. It has the following parameters: `Arew` (reward learning rate), `Apun` (punishment learning rate), `K` (perseverance decay), `betaF` (outcome frequency weight), `betaP` (perseverance weight).

- **Task**: Iowa Gambling Task (Ahn et al., 2008)
- **Model**: Outcome-Representation Learning Model (Haines et al., 2018)

Usage

```
igt_orl(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "gain", "loss". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred"
`adapt_delta`	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.

`stepsize` Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.

`max_treedepth` Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.

`...` For this model, it's possible to set **model-specific argument(s)** as follows:

payscale Raw payoffs within data are divided by this number. Used for scaling data. Defaults to 100.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Iowa Gambling Task, there should be 4 columns of data with the labels "subjID", "choice", "gain", "loss". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer indicating which deck was chosen on that trial (where A==1, B==2, C==3, and D==4).

gain Floating point value representing the amount of currency won on that trial (e.g. 50, 100).

loss Floating point value representing the amount of currency lost on that trial (e.g. 0, -50).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users

change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Nate Haines <<haines.175@osu.edu>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"igt_orl").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Ahn, W. Y., Busemeyer, J. R., & Wagenmakers, E. J. (2008). Comparison of decision learning models using the generalization criterion method. Cognitive Science, 32(8), 1376-1402. http://doi.org/10.1080/03640210802352

Haines, N., Vassileva, J., & Ahn, W.-Y. (2018). The Outcome-Representation Learning Model: A Novel Reinforcement Learning Model of the Iowa Gambling Task. Cognitive Science. https://doi.org/10.1111/cogs.12688

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- igt_orl(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- igt_orl(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)
```

```
# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`igt_pvl_decay` *Prospect Valence Learning (PVL) Decay-RI*

Description

Hierarchical Bayesian Modeling of the Iowa Gambling Task using Prospect Valence Learning (PVL) Decay-RI. It has the following parameters: `A` (decay rate), `alpha` (outcome sensitivity), `cons` (response consistency), `lambda` (loss aversion).

- **Task**: Iowa Gambling Task (Ahn et al., 2008)
- **Model**: Prospect Valence Learning (PVL) Decay-RI (Ahn et al., 2014)

Usage

```
igt_pvl_decay(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "gain", "loss". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".

`modelRegressor` Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.

`vb` Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.

`inc_postpred` Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred"

`adapt_delta` Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.

`stepsize` Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.

`max_treedepth` Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.

`...` For this model, it's possible to set **model-specific argument(s)** as follows:

payscale Raw payoffs within data are divided by this number. Used for scaling data. Defaults to 100.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Iowa Gambling Task, there should be 4 columns of data with the labels "subjID", "choice", "gain", "loss". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer indicating which deck was chosen on that trial (where A==1, B==2, C==3, and D==4).

gain Floating point value representing the amount of currency won on that trial (e.g. 50, 100).

loss Floating point value representing the amount of currency lost on that trial (e.g. 0, -50).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated

from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"igt_pvl_decay").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Ahn, W. Y., Busemeyer, J. R., & Wagenmakers, E. J. (2008). Comparison of decision learning models using the generalization criterion method. Cognitive Science, 32(8), 1376-1402. http://doi.org/10.1080/03640210802352

Ahn, W.-Y., Vasilev, G., Lee, S.-H., Busemeyer, J. R., Kruschke, J. K., Bechara, A., & Vassileva, J. (2014). Decision-making in stimulant and opiate addicts in protracted abstinence: evidence from computational modeling with pure users. Frontiers in Psychology, 5, 1376. http://doi.org/10.3389/fpsyg.2014.00849

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- igt_pvl_decay(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)
```

```
# Run the model with example data
output <- igt_pvl_decay(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`igt_pvl_delta` *Prospect Valence Learning (PVL) Delta*

Description

Hierarchical Bayesian Modeling of the Iowa Gambling Task using Prospect Valence Learning (PVL) Delta. It has the following parameters: `A` (learning rate), `alpha` (outcome sensitivity), `cons` (response consistency), `lambda` (loss aversion).

- **Task**: Iowa Gambling Task (Ahn et al., 2008)
- **Model**: Prospect Valence Learning (PVL) Delta (Ahn et al., 2008)

Usage

```
igt_pvl_delta(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "gain", "loss". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.

ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). Not available for this model.
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, it's possible to set **model-specific argument(s)** as follows:
	payscale Raw payoffs within data are divided by this number. Used for scaling data. Defaults to 100.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Iowa Gambling Task, there should be 4 columns of data with the labels "subjID", "choice", "gain", "loss". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer indicating which deck was chosen on that trial (where A==1, B==2, C==3, and D==4).

gain Floating point value representing the amount of currency won on that trial (e.g. 50, 100).

loss Floating point value representing the amount of currency lost on that trial (e.g. 0, -50).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output, type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"igt_pvl_delta").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Ahn, W. Y., Busemeyer, J. R., & Wagenmakers, E. J. (2008). Comparison of decision learning models using the generalization criterion method. Cognitive Science, 32(8), 1376-1402. http://doi.org/10.1080/03640210802352

Ahn, W. Y., Busemeyer, J. R., & Wagenmakers, E. J. (2008). Comparison of decision learning models using the generalization criterion method. Cognitive Science, 32(8), 1376-1402. http://doi.org/10.1080/03640210802352

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- igt_pvl_delta(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- igt_pvl_delta(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

igt_vpp *Value-Plus-Perseverance*

Description

Hierarchical Bayesian Modeling of the Iowa Gambling Task using Value-Plus-Perseverance. It has the following parameters: `A` (learning rate), `alpha` (outcome sensitivity), `cons` (response consistency), `lambda` (loss aversion), `epP` (gain impact), `epN` (loss impact), `K` (decay rate), `w` (RL weight).

- **Task**: Iowa Gambling Task (Ahn et al., 2008)
- **Model**: Value-Plus-Perseverance (Worthy et al., 2013)

Usage

```
igt_vpp(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "gain", "loss". See **Details** below for more information.

niter	Number of iterations, including warm-up. Defaults to 4000.
nwarmup	Number of iterations used for warm-up only. Defaults to 1000.
nchain	Number of Markov chains to run. Defaults to 4.
ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). Not available for this model.
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, it's possible to set **model-specific argument(s)** as follows:
	payscale Raw payoffs within data are divided by this number. Used for scaling data. Defaults to 100.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Iowa Gambling Task, there should be 4 columns of data with the labels "subjID", "choice", "gain", "loss". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer indicating which deck was chosen on that trial (where A==1, B==2, C==3, and D==4).

gain Floating point value representing the amount of currency won on that trial (e.g. 50, 100).

loss Floating point value representing the amount of currency lost on that trial (e.g. 0, -50).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"igt_vpp").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Ahn, W. Y., Busemeyer, J. R., & Wagenmakers, E. J. (2008). Comparison of decision learning models using the generalization criterion method. Cognitive Science, 32(8), 1376-1402. http://doi.org/10.1080/03640210802352

Worthy, D. A., & Todd Maddox, W. (2013). A comparison model of reinforcement-learning and win-stay-lose-shift decision-making processes: A tribute to W.K. Estes. Journal of Mathematical Psychology, 59, 41-49. http://doi.org/10.1016/j.jmp.2013.10.001

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- igt_vpp(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- igt_vpp(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`multiplot` *Function to plot multiple figures*

Description

Plots multiple figures Based on codes from 'http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/'

Usage

```
multiplot(..., plots = NULL, cols = NULL)
```

Arguments

`...`	Plot objects
`plots`	List containing plot objects
`cols`	Number of columns within the multi-figure plot

peer_ocu *Other-Conferred Utility (OCU) Model*

Description

Hierarchical Bayesian Modeling of the Peer Influence Task using Other-Conferred Utility (OCU) Model. It has the following parameters: `rho` (risk preference), `tau` (inverse temperature), `ocu` (other-conferred utility).

- **Task**: Peer Influence Task (Chung et al., 2015)
- **Model**: Other-Conferred Utility (OCU) Model

Usage

```
peer_ocu(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "condition", "p_gamble", "safe_Hpayoff", "safe_Lpayoff", "risky_Hpayoff", "risky_Lpayoff", "choice". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred"

adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Peer Influence Task, there should be 8 columns of data with the labels "subjID", "condition", "p_gamble", "safe_Hpayoff", "safe_Lpayoff", "risky_Hpayoff", "risky_Lpayoff", "choice". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

condition 0: solo, 1: info (safe/safe), 2: info (mix), 3: info (risky/risky).

p_gamble Probability of receiving a high payoff (same for both options).

safe_Hpayoff High payoff of the safe option.

safe_Lpayoff Low payoff of the safe option.

risky_Hpayoff High payoff of the risky option.

risky_Lpayoff Low payoff of the risky option.

choice Which option was chosen? 0: safe, 1: risky.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Harhim Park <<hrpark12@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"peer_ocu").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Chung, D., Christopoulos, G. I., King-Casas, B., Ball, S. B., & Chiu, P. H. (2015). Social signals of safety and risk confer utility and have asymmetric effects on observers' choices. Nature Neuroscience, 18(6), 912-916.

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- peer_ocu(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- peer_ocu(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
```

```
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`plot.hBayesDM` *General Purpose Plotting for hBayesDM. This function plots hyper parameters.*

Description

General Purpose Plotting for hBayesDM. This function plots hyper parameters.

Usage

```
## S3 method for class 'hBayesDM'
plot(x = NULL, type = "dist", ncols = NULL,
  fontSize = NULL, binSize = NULL, ...)
```

Arguments

`x`	Model output of class hBayesDM
`type`	Character value that specifies the plot type. Options are: "dist", "trace", or "simple". Defaults to "dist".
`ncols`	Integer value specifying how many plots there should be per row. Defaults to the number of parameters.
`fontSize`	Integer value specifying the size of the font used for plotting. Defaults to 10.
`binSize`	Integer value specifying how wide the bars on the histogram should be. Defaults to 30.
`...`	Additional arguments to be passed on

plotDist *Plots the histogram of MCMC samples.*

Description

Plots the histogram of MCMC samples.

Usage

```
plotDist(sample = NULL, Title = NULL, xLab = "Value",
  yLab = "Density", xLim = NULL, fontSize = NULL, binSize = NULL,
  ...)
```

Arguments

sample	MCMC samples
Title	Character value containing the main title for the plot
xLab	Character value containing the x label
yLab	Character value containing the y label
xLim	Vector containing the lower and upper x-bounds of the plot
fontSize	Size of the font to use for plotting. Defaults to 10
binSize	Size of the bins for creating the histogram. Defaults to 30
...	Arguments that can be additionally supplied to geom_histogram

Value

h1 Plot object

plotHDI *Plots highest density interval (HDI) from (MCMC) samples and prints HDI in the R console. HDI is indicated by a red line. Based on John Kruschke's codes.*

Description

Plots highest density interval (HDI) from (MCMC) samples and prints HDI in the R console. HDI is indicated by a red line. Based on John Kruschke's codes.

Usage

```
plotHDI(sample = NULL, credMass = 0.95, Title = NULL,
  xLab = "Value", yLab = "Density", fontSize = NULL, binSize = 30,
  ...)
```

Arguments

sample	MCMC samples
credMass	A scalar between 0 and 1, indicating the mass within the credible interval that is to be estimated.
Title	Character value containing the main title for the plot
xLab	Character value containing the x label
yLab	Character value containing the y label
fontSize	Integer value specifying the font size to be used for the plot labels
binSize	Integer value specifyin ghow wide the bars on the histogram should be. Defaults to 30.
...	Arguments that can be additionally supplied to geom_histogram

Value

A vector containing the limits of the HDI

---

plotInd — *Plots individual posterior distributions, using the stan_plot function of the rstan package*

Description

Plots individual posterior distributions, using the stan_plot function of the rstan package

Usage

```
plotInd(obj = NULL, pars, show_density = T, ...)
```

Arguments

obj	An output of the hBayesDM. Its class should be 'hBayesDM'.
pars	(from stan_plot's help file) Character vector of parameter names. If unspecified, show all user-defined parameters or the first 10 (if there are more than 10)
show_density	T(rue) or F(alse). Show the density (T) or not (F)?
...	(from stan_plot's help file) Optional additional named arguments passed to stan_plot, which will be passed to geoms. See stan_plot's help file.

Examples

```
## Not run:
# Run a model
output <- dd_hyperbolic("example", 2000, 1000, 3, 3)

# Plot the hyper parameters ('k' and 'beta')
plot(output)

# Plot individual 'k' (discounting rate) parameters
plotInd(output, "k")

# Plot individual 'beta' (inverse temperature) parameters
plotInd(output, "beta")

# Plot individual 'beta' parameters but don't show density
plotInd(output, "beta", show_density = F)

## End(Not run)
```

`printFit` *Print model-fits (mean LOOIC or WAIC values in addition to Akaike weights) of hBayesDM Models*

Description

Print model-fits (mean LOOIC or WAIC values in addition to Akaike weights) of hBayesDM Models

Usage

```
printFit(..., ic = "looic", ncore = 2, roundTo = 3)
```

Arguments

`...`	Model objects output by hBayesDM functions (e.g. output1, output2, etc.)
`ic`	Which model comparison information criterion to use? 'looic', 'waic', or 'both
`ncore`	Number of corse to use when computing LOOIC
`roundTo`	Number of digits to the right of the decimal point in the output

Value

modelTable A table with relevant model comparison data. LOOIC and WAIC weights are computed as Akaike weights.

Examples

```
## Not run:
# Run two models and store results in "output1" and "output2"
output1 <- dd_hyperbolic("example", 2000, 1000, 3, 3)

output2 <- dd_exp("example", 2000, 1000, 3, 3)

# Show the LOOIC model fit estimates
printFit(output1, output2)

# To show the WAIC model fit estimates
printFit(output1, output2, ic = "waic")

# To show both LOOIC and WAIC
printFit(output1, output2, ic = "both")

## End(Not run)
```

---

prl_ewa *Experience-Weighted Attraction Model*

Description

Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task using Experience-Weighted Attraction Model. It has the following parameters: `phi` (1 - learning rate), `rho` (experience decay factor), `beta` (inverse temperature).

- **Task**: Probabilistic Reversal Learning Task
- **Model**: Experience-Weighted Attraction Model (Ouden et al., 2013)

Usage

```
prl_ewa(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "outcome". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.

nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). For this model they are: "ev_c", "ev_nc", "ew_c", "ew_nc".
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Probabilistic Reversal Learning Task, there should be 3 columns of data with the labels "subjID", "choice", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer value representing the option chosen on that trial: 1 or 2.

outcome Integer value representing the outcome of that trial (where reward == 1, and loss == -1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The nwarmup argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Jaeyeong Yang (for model-based regressors) <<jaeyeong.yang1125@gmail.com>>, Harhim Park (for model-based regressors) <<hrpark12@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"prl_ewa").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Ouden, den, H. E. M., Daw, N. D., Fernandez, G., Elshout, J. A., Rijpkema, M., Hoogman, M., et al. (2013). Dissociable Effects of Dopamine and Serotonin on Reversal Learning. Neuron, 80(4), 1090-1100. http://doi.org/10.1016/j.neuron.2013.08.030

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: `https://rpubs.com/CCSL/hBayesDM`

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- prl_ewa(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- prl_ewa(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

prl_fictitious *Fictitious Update Model*

Description

Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task using Fictitious Update Model. It has the following parameters: `eta` (learning rate), `alpha` (indecision point), `beta` (inverse temperature).

- **Task**: Probabilistic Reversal Learning Task
- **Model**: Fictitious Update Model (Glascher et al., 2009)

Usage

```
prl_fictitious(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "outcome". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). For this model they are: "ev_c", "ev_nc", "pe_c", "pe_nc", "dv".
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred"
`adapt_delta`	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
`stepsize`	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
`max_treedepth`	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
`...`	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Probabilistic Reversal Learning Task, there should be 3 columns of data with the labels "subjID", "choice", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer value representing the option chosen on that trial: 1 or 2.

outcome Integer value representing the outcome of that trial (where reward == 1, and loss == -1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Jaeyeong Yang (for model-based regressors) <<jaeyeong.yang1125@gmail.com>>, Harhim Park (for model-based regressors) <<hrpark12@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"prl_fictitious").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Glascher, J., Hampton, A. N., & O'Doherty, J. P. (2009). Determining a Role for Ventromedial Prefrontal Cortex in Encoding Action-Based Value Signals During Reward-Related Decision Making. Cerebral Cortex, 19(2), 483-495. http://doi.org/10.1093/cercor/bhn098

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- prl_fictitious(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- prl_fictitious(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

prl_fictitious_multipleB

Fictitious Update Model

Description

Multiple-Block Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task using Fictitious Update Model. It has the following parameters: `eta` (learning rate), `alpha` (indecision point), `beta` (inverse temperature).

- **Task**: Probabilistic Reversal Learning Task
- **Model**: Fictitious Update Model (Glascher et al., 2009)

Usage

```
prl_fictitious_multipleB(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

data	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, "example" to use example data, or "choose" to choose data with an interactive window. Columns in the dataset must include: "subjID", "block", "choice", "outcome". See **Details** below for more information.
niter	Number of iterations, including warm-up. Defaults to 4000.
nwarmup	Number of iterations used for warm-up only. Defaults to 1000.
nchain	Number of Markov chains to run. Defaults to 4.
ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). For this model they are: "ev_c", "ev_nc", "pe_c", "pe_nc", "dv".
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial

observations and columns represent variables.
For the Probabilistic Reversal Learning Task, there should be 4 columns of data with the labels "subjID", "block", "choice", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

block A unique identifier for each of the multiple blocks within each subject.

choice Integer value representing the option chosen on that trial: 1 or 2.

outcome Integer value representing the outcome of that trial (where reward == 1, and loss == -1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Jaeyeong Yang (for model-based regressors) <<jaeyeong.yang1125@gmail.com>>, Harhim Park (for model-based regressors) <<hrpark12@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"prl_fictitious_multipleB").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class stanfit object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Glascher, J., Hampton, A. N., & O'Doherty, J. P. (2009). Determining a Role for Ventromedial Prefrontal Cortex in Encoding Action-Based Value Signals During Reward-Related Decision Making. Cerebral Cortex, 19(2), 483-495. http://doi.org/10.1093/cercor/bhn098

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- prl_fictitious_multipleB(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- prl_fictitious_multipleB(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`prl_fictitious_rp` *Fictitious Update Model, with separate learning rates for positive and negative prediction error (PE)*

Description

Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task using Fictitious Update Model, with separate learning rates for positive and negative prediction error (PE). It has the following parameters: `eta_pos` (learning rate, +PE), `eta_neg` (learning rate, -PE), `alpha` (indecision point), `beta` (inverse temperature).

- **Task**: Probabilistic Reversal Learning Task
- **Model**: Fictitious Update Model, with separate learning rates for positive and negative prediction error (PE) (Glascher et al., 2009; Ouden et al., 2013)

Usage

```
prl_fictitious_rp(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "outcome". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). For this model they are: "ev_c", "ev_nc", "pe_c", "pe_nc", "dv".
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred"
`adapt_delta`	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
`stepsize`	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.

`max_treedepth` Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.

`...` For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Probabilistic Reversal Learning Task, there should be 3 columns of data with the labels "subjID", "choice", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer value representing the option chosen on that trial: 1 or 2.

outcome Integer value representing the outcome of that trial (where reward == 1, and loss == -1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Jaeyeong Yang (for model-based regressors) <<jaeyeong.yang1125@gmail.com>>, Harhim Park (for model-based regressors) <<hrpark12@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"prl_fictitious_rp").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Glascher, J., Hampton, A. N., & O'Doherty, J. P. (2009). Determining a Role for Ventromedial Prefrontal Cortex in Encoding Action-Based Value Signals During Reward-Related Decision Making. Cerebral Cortex, 19(2), 483-495. http://doi.org/10.1093/cercor/bhn098

Ouden, den, H. E. M., Daw, N. D., Fernandez, G., Elshout, J. A., Rijpkema, M., Hoogman, M., et al. (2013). Dissociable Effects of Dopamine and Serotonin on Reversal Learning. Neuron, 80(4), 1090-1100. http://doi.org/10.1016/j.neuron.2013.08.030

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- prl_fictitious_rp(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- prl_fictitious_rp(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)
```

```
# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`prl_fictitious_rp_woa` *Fictitious Update Model, with separate learning rates for positive and negative prediction error (PE), without alpha (indecision point)*

Description

Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task using Fictitious Update Model, with separate learning rates for positive and negative prediction error (PE), without alpha (indecision point). It has the following parameters: `eta_pos` (learning rate, +PE), `eta_neg` (learning rate, -PE), `beta` (inverse temperature).

- **Task**: Probabilistic Reversal Learning Task
- **Model**: Fictitious Update Model, with separate learning rates for positive and negative prediction error (PE), without alpha (indecision point) (Glascher et al., 2009; Ouden et al., 2013)

Usage

```
prl_fictitious_rp_woa(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "outcome". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".

modelRegressor	Whether to export model-based regressors (TRUE or FALSE). For this model they are: "ev_c", "ev_nc", "pe_c", "pe_nc", "dv".
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Probabilistic Reversal Learning Task, there should be 3 columns of data with the labels "subjID", "choice", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer value representing the option chosen on that trial: 1 or 2.

outcome Integer value representing the outcome of that trial (where reward == 1, and loss == -1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The nwarmup argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Jaeyeong Yang (for model-based regressors) <<jaeyeong.yang1125@gmail.com>>, Harhim Park (for model-based regressors) <<hrpark12@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"prl_fictitious_rp_woa").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Glascher, J., Hampton, A. N., & O'Doherty, J. P. (2009). Determining a Role for Ventromedial Prefrontal Cortex in Encoding Action-Based Value Signals During Reward-Related Decision Making. Cerebral Cortex, 19(2), 483-495. http://doi.org/10.1093/cercor/bhn098

Ouden, den, H. E. M., Daw, N. D., Fernandez, G., Elshout, J. A., Rijpkema, M., Hoogman, M., et al. (2013). Dissociable Effects of Dopamine and Serotonin on Reversal Learning. Neuron, 80(4), 1090-1100. http://doi.org/10.1016/j.neuron.2013.08.030

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: `https://rpubs.com/CCSL/hBayesDM`

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- prl_fictitious_rp_woa(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)
```

```
# Run the model with example data
output <- prl_fictitious_rp_woa(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`prl_fictitious_woa` *Fictitious Update Model, without alpha (indecision point)*

Description

Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task using Fictitious Update Model, without alpha (indecision point). It has the following parameters: `eta` (learning rate), `beta` (inverse temperature).

- **Task**: Probabilistic Reversal Learning Task
- **Model**: Fictitious Update Model, without alpha (indecision point) (Glascher et al., 2009)

Usage

```
prl_fictitious_woa(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "outcome". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.

ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). For this model they are: "ev_c", "ev_nc", "pe_c", "pe_nc", "dv".
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Probabilistic Reversal Learning Task, there should be 3 columns of data with the labels "subjID", "choice", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer value representing the option chosen on that trial: 1 or 2.

outcome Integer value representing the outcome of that trial (where reward == 1, and loss == -1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The nwarmup argument

can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Jaeyeong Yang (for model-based regressors) <<jaeyeong.yang1125@gmail.com>>, Harhim Park (for model-based regressors) <<hrpark12@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"prl_fictitious_woa").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Glascher, J., Hampton, A. N., & O'Doherty, J. P. (2009). Determining a Role for Ventromedial Prefrontal Cortex in Encoding Action-Based Value Signals During Reward-Related Decision Making. Cerebral Cortex, 19(2), 483-495. http://doi.org/10.1093/cercor/bhn098

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: `https://rpubs.com/CCSL/hBayesDM`

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- prl_fictitious_woa(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- prl_fictitious_woa(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

prl_rp *Reward-Punishment Model*

Description

Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task using Reward-Punishment Model. It has the following parameters: `Apun` (punishment learning rate), `Arew` (reward learning rate), `beta` (inverse temperature).

- **Task**: Probabilistic Reversal Learning Task
- **Model**: Reward-Punishment Model (Ouden et al., 2013)

Usage

```
prl_rp(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "outcome". See **Details** below for more information.

niter	Number of iterations, including warm-up. Defaults to 4000.
nwarmup	Number of iterations used for warm-up only. Defaults to 1000.
nchain	Number of Markov chains to run. Defaults to 4.
ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). For this model they are: "ev_c", "ev_nc", "pe".
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Probabilistic Reversal Learning Task, there should be 3 columns of data with the labels "subjID", "choice", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer value representing the option chosen on that trial: 1 or 2.

outcome Integer value representing the outcome of that trial (where reward == 1, and loss == -1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output, type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Jaeyeong Yang (for model-based regressors) <<jaeyeong.yang1125@gmail.com>>, Harhim Park (for model-based regressors) <<hrpark12@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"prl_rp").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Ouden, den, H. E. M., Daw, N. D., Fernandez, G., Elshout, J. A., Rijpkema, M., Hoogman, M., et al. (2013). Dissociable Effects of Dopamine and Serotonin on Reversal Learning. Neuron, 80(4), 1090-1100. http://doi.org/10.1016/j.neuron.2013.08.030

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- prl_rp(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- prl_rp(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

prl_rp_multipleB *Reward-Punishment Model*

Description

Multiple-Block Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task using Reward-Punishment Model. It has the following parameters: `Apun` (punishment learning rate), `Arew` (reward learning rate), `beta` (inverse temperature).

- **Task**: Probabilistic Reversal Learning Task
- **Model**: Reward-Punishment Model (Ouden et al., 2013)

Usage

```
prl_rp_multipleB(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

data	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, "example" to use example data, or "choose" to choose data with an interactive window. Columns in the dataset must include: "subjID", "block", "choice", "outcome". See **Details** below for more information.
niter	Number of iterations, including warm-up. Defaults to 4000.
nwarmup	Number of iterations used for warm-up only. Defaults to 1000.
nchain	Number of Markov chains to run. Defaults to 4.
ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). For this model they are: "ev_c", "ev_nc", "pe".
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Probabilistic Reversal Learning Task, there should be 4 columns of data with the labels "subjID", "block", "choice", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

block A unique identifier for each of the multiple blocks within each subject.

choice Integer value representing the option chosen on that trial: 1 or 2.

outcome Integer value representing the outcome of that trial (where reward == 1, and loss == -1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output, type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Jaeyeong Yang (for model-based regressors) <<jaeyeong.yang1125@gmail.com>>, Harhim Park (for model-based regressors) <<hrpark12@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"prl_rp_multipleB").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Ouden, den, H. E. M., Daw, N. D., Fernandez, G., Elshout, J. A., Rijpkema, M., Hoogman, M., et al. (2013). Dissociable Effects of Dopamine and Serotonin on Reversal Learning. Neuron, 80(4), 1090-1100. http://doi.org/10.1016/j.neuron.2013.08.030

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- prl_rp_multipleB(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- prl_rp_multipleB(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

pst_gainloss_Q *Gain-Loss Q Learning Model*

Description

Hierarchical Bayesian Modeling of the Probabilistic Selection Task using Gain-Loss Q Learning Model. It has the following parameters: `alpha_pos` (learning rate for positive feedbacks), `alpha_neg` (learning rate for negative feedbacks), `beta` (inverse temperature).

- **Task**: Probabilistic Selection Task
- **Model**: Gain-Loss Q Learning Model (Frank et al., 2007)

Usage

```
pst_gainloss_Q(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

data	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, "example" to use example data, or "choose" to choose data with an interactive window. Columns in the dataset must include: "subjID", "type", "choice", "reward". See **Details** below for more information.
niter	Number of iterations, including warm-up. Defaults to 4000.
nwarmup	Number of iterations used for warm-up only. Defaults to 1000.
nchain	Number of Markov chains to run. Defaults to 4.
ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (TRUE or FALSE). Not available for this model.
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial

observations and columns represent variables.
For the Probabilistic Selection Task, there should be 4 columns of data with the labels "subjID", "type", "choice", "reward". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

type Two-digit number indicating which pair of stimuli were presented for that trial, e.g. 12, 34, or 56. The digit on the left (tens-digit) indicates the presented stimulus for option1, while the digit on the right (ones-digit) indicates that for option2. Code for each stimulus type (1~6) is defined as for 80% (type 1), 20% (type 2), 70% (type 3), 30% (type 4), 60% (type 5), 40% (type 6). The modeling will still work even if different probabilities are used for the stimuli; however, the total number of stimuli should be less than or equal to 6.

choice Whether the subject chose the left option (option1) out of the given two options (i.e. if option1 was chosen, 1; if option2 was chosen, 0).

reward Amount of reward earned as a result of the trial.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Jaeyeong Yang <<jaeyeong.yang1125@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"pst_gainloss_Q").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Frank, M. J., Moustafa, A. A., Haughey, H. M., Curran, T., & Hutchison, K. E. (2007). Genetic triple dissociation reveals multiple roles for dopamine in reinforcement learning. Proceedings of the National Academy of Sciences, 104(41), 16311-16316.

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- pst_gainloss_Q(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- pst_gainloss_Q(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`ra_noLA` *Prospect Theory, without loss aversion (LA) parameter*

Description

Hierarchical Bayesian Modeling of the Risk Aversion Task using Prospect Theory, without loss aversion (LA) parameter. It has the following parameters: `rho` (risk aversion), `tau` (inverse temperature).

- **Task**: Risk Aversion Task
- **Model**: Prospect Theory, without loss aversion (LA) parameter (Sokol-Hessner et al., 2009)

Usage

```
ra_noLA(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "gain", "loss", "cert", "gamble". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred"
`adapt_delta`	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.

`stepsize`	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
`max_treedepth`	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
`...`	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Risk Aversion Task, there should be 5 columns of data with the labels "subjID", "gain", "loss", "cert", "gamble". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

gain Possible (50%) gain outcome of a risky option (e.g. 9).

loss Possible (50%) loss outcome of a risky option (e.g. 5, or -5).

cert Guaranteed amount of a safe option. "cert" is assumed to be zero or greater than zero.

gamble If gamble was taken, gamble == 1; else gamble == 0.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman

& Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"ra_noLA").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Sokol-Hessner, P., Hsu, M., Curley, N. G., Delgado, M. R., Camerer, C. F., Phelps, E. A., & Smith, E. E. (2009). Thinking like a Trader Selectively Reduces Individuals' Loss Aversion. Proceedings of the National Academy of Sciences of the United States of America, 106(13), 5035-5040. http://www.pnas.org/content/106/13/5035

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- ra_noLA(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- ra_noLA(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
```

```
printFit(output)

## End(Not run)
```

`ra_noRA` *Prospect Theory, without risk aversion (RA) parameter*

Description

Hierarchical Bayesian Modeling of the Risk Aversion Task using Prospect Theory, without risk aversion (RA) parameter. It has the following parameters: `lambda` (loss aversion), `tau` (inverse temperature).

- **Task**: Risk Aversion Task
- **Model**: Prospect Theory, without risk aversion (RA) parameter (Sokol-Hessner et al., 2009)

Usage

```
ra_noRA(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "gain", "loss", "cert", "gamble". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.

inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Risk Aversion Task, there should be 5 columns of data with the labels "subjID", "gain", "loss", "cert", "gamble". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

gain Possible (50%) gain outcome of a risky option (e.g. 9).

loss Possible (50%) loss outcome of a risky option (e.g. 5, or -5).

cert Guaranteed amount of a safe option. "cert" is assumed to be zero or greater than zero.

gamble If gamble was taken, gamble == 1; else gamble == 0.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The nwarmup argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"ra_noRA").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Sokol-Hessner, P., Hsu, M., Curley, N. G., Delgado, M. R., Camerer, C. F., Phelps, E. A., & Smith, E. E. (2009). Thinking like a Trader Selectively Reduces Individuals' Loss Aversion. Proceedings of the National Academy of Sciences of the United States of America, 106(13), 5035-5040. http://www.pnas.org/content/106/13/5035

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- ra_noRA(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- ra_noRA(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)
```

```
# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`ra_prospect` *Prospect Theory*

Description

Hierarchical Bayesian Modeling of the Risk Aversion Task using Prospect Theory. It has the following parameters: `rho` (risk aversion), `lambda` (loss aversion), `tau` (inverse temperature).

- **Task**: Risk Aversion Task
- **Model**: Prospect Theory (Sokol-Hessner et al., 2009)

Usage

```
ra_prospect(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "gain", "loss", "cert", "gamble". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.

vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Risk Aversion Task, there should be 5 columns of data with the labels "subjID", "gain", "loss", "cert", "gamble". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

gain Possible (50%) gain outcome of a risky option (e.g. 9).

loss Possible (50%) loss outcome of a risky option (e.g. 5, or -5).

cert Guaranteed amount of a safe option. "cert" is assumed to be zero or greater than zero.

gamble If gamble was taken, gamble == 1; else gamble == 0.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The nwarmup argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"ra_prospect").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Sokol-Hessner, P., Hsu, M., Curley, N. G., Delgado, M. R., Camerer, C. F., Phelps, E. A., & Smith, E. E. (2009). Thinking like a Trader Selectively Reduces Individuals' Loss Aversion. Proceedings of the National Academy of Sciences of the United States of America, 106(13), 5035-5040. http://www.pnas.org/content/106/13/5035

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- ra_prospect(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- ra_prospect(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")
```

```
# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`rdt_happiness` *Happiness Computational Model*

Description

Hierarchical Bayesian Modeling of the Risky Decision Task using Happiness Computational Model. It has the following parameters: `w0` (baseline), `w1` (weight of certain rewards), `w2` (weight of expected values), `w3` (weight of reward prediction errors), `gam` (forgetting factor), `sig` (standard deviation of error).

- **Task**: Risky Decision Task
- **Model**: Happiness Computational Model (Rutledge et al., 2014)

Usage

```
rdt_happiness(data = NULL, niter = 4000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "vb",
  indPars = "mean", modelRegressor = FALSE, vb = FALSE,
  inc_postpred = FALSE, adapt_delta = 0.95, stepsize = 1,
  max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "gain", "loss", "cert", "type", "gamble", "outcome", "happy", "RT_happy". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.

`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred"
`adapt_delta`	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
`stepsize`	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
`max_treedepth`	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
`...`	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Risky Decision Task, there should be 9 columns of data with the labels "subjID", "gain", "loss", "cert", "type", "gamble", "outcome", "happy", "RT_happy". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

gain Possible (50%) gain outcome of a risky option (e.g. 9).

loss Possible (50%) loss outcome of a risky option (e.g. 5, or -5).

cert Guaranteed amount of a safe option.

type loss == -1, mixed == 0, gain == 1

gamble If gamble was taken, gamble == 1; else gamble == 0.

outcome Result of the trial.

happy Happiness score.

RT_happy Reaction time for answering the happiness score.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Harhim Park <<hrpark12@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"rdt_happiness").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Rutledge, R. B., Skandali, N., Dayan, P., & Dolan, R. J. (2014). A computational and neural model of momentary subjective well-being. Proceedings of the National Academy of Sciences, 111(33), 12252-12257.

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- rdt_happiness(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- rdt_happiness(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

rhat *Function for extracting Rhat values from an hBayesDM object*

Description

A convenience function for extracting Rhat values from an hBayesDM object. Can also check if all Rhat values are less than or equal to a specified value. If variational inference was used, an error message will be displayed.

Usage

```
rhat(fit = NULL, less = NULL)
```

Arguments

`fit`	Model output of class `hBayesDM`
`less`	A numeric value specifying how to check Rhat values. Defaults to FALSE.

Value

If `'less'` is specified, then `rhat(fit,less)` will return `TRUE` if all Rhat values are less than or equal to `'less'`. If any values are greater than `'less'`, `rhat(fit,less)` will return `FALSE`. If `'less'` is left unspecified (NULL), `rhat(fit)` will return a `data.frame` object containing all Rhat values.

`ts_par4` *Hybrid Model, with 4 parameters*

Description

Hierarchical Bayesian Modeling of the Two-Step Task using Hybrid Model, with 4 parameters. It has the following parameters: `a` (learning rate for both stages 1 & 2), `beta` (inverse temperature for both stages 1 & 2), `pi` (perseverance), `w` (model-based weight).

- **Task**: Two-Step Task (Daw et al., 2011)
- **Model**: Hybrid Model, with 4 parameters (Daw et al., 2011; Wunderlich et al., 2012)

Usage

```
ts_par4(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "level1_choice", "level2_choice", "reward". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred_step1", "y_pred_step2"

`adapt_delta` Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.

`stepsize` Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.

`max_treedepth` Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.

`...` For this model, it's possible to set **model-specific argument(s)** as follows:

trans_prob Common state transition probability from Stage (Level) 1 to Stage (Level) 2. Defaults to 0.7.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Two-Step Task, there should be 4 columns of data with the labels "subjID", "level1_choice", "level2_choice", "reward". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

level1_choice Choice made for Level (Stage) 1 (1: stimulus 1, 2: stimulus 2).

level2_choice Choice made for Level (Stage) 2 (1: stimulus 3, 2: stimulus 4, 3: stimulus 5, 4: stimulus 6).
Note that, in our notation, choosing stimulus 1 in Level 1 leads to stimulus 3 & 4 in Level 2 with a common (0.7 by default) transition. Similarly, choosing stimulus 2 in Level 1 leads to stimulus 5 & 6 in Level 2 with a common (0.7 by default) transition. To change this default transition probability, set the function argument 'trans_prob' to your preferred value.

reward Reward after Level 2 (0 or 1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Harhim Park <<hrpark12@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"ts_par4").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Daw, N. D., Gershman, S. J., Seymour, B., Ben Seymour, Dayan, P., & Dolan, R. J. (2011). Model-Based Influences on Humans' Choices and Striatal Prediction Errors. Neuron, 69(6), 1204-1215. http://doi.org/10.1016/j.neuron.2011.02.027

Daw, N. D., Gershman, S. J., Seymour, B., Ben Seymour, Dayan, P., & Dolan, R. J. (2011). Model-Based Influences on Humans' Choices and Striatal Prediction Errors. Neuron, 69(6), 1204-1215. http://doi.org/10.1016/j.neuron.2011.02.027

Wunderlich, K., Smittenaar, P., & Dolan, R. J. (2012). Dopamine enhances model-based over model-free choice behavior. Neuron, 75(3), 418-424.

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- ts_par4(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)
```

```
# Run the model with example data
output <- ts_par4(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

`ts_par6` *Hybrid Model, with 6 parameters*

Description

Hierarchical Bayesian Modeling of the Two-Step Task using Hybrid Model, with 6 parameters. It has the following parameters: `a1` (learning rate in stage 1), `beta1` (inverse temperature in stage 1), `a2` (learning rate in stage 2), `beta2` (inverse temperature in stage 2), `pi` (perseverance), `w` (model-based weight).

- **Task**: Two-Step Task (Daw et al., 2011)
- **Model**: Hybrid Model, with 6 parameters (Daw et al., 2011)

Usage

```
ts_par6(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "level1_choice", "level2_choice", "reward". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.

nchain	Number of Markov chains to run. Defaults to 4.
ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
modelRegressor	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred_step1", "y_pred_step2"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, it's possible to set **model-specific argument(s)** as follows: **trans_prob** Common state transition probability from Stage (Level) 1 to Stage (Level) 2. Defaults to 0.7.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Two-Step Task, there should be 4 columns of data with the labels "subjID", "level1_choice", "level2_choice", "reward". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

level1_choice Choice made for Level (Stage) 1 (1: stimulus 1, 2: stimulus 2).

level2_choice Choice made for Level (Stage) 2 (1: stimulus 3, 2: stimulus 4, 3: stimulus 5, 4: stimulus 6).
Note that, in our notation, choosing stimulus 1 in Level 1 leads to stimulus 3 & 4 in Level 2 with a common (0.7 by default) transition. Similarly, choosing stimulus 2 in Level 1 leads to stimulus 5 & 6 in Level 2 with a common (0.7 by default) transition. To change this default transition probability, set the function argument 'trans_prob' to your preferred value.

reward Reward after Level 2 (0 or 1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Harhim Park <<hrpark12@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"ts_par6").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Daw, N. D., Gershman, S. J., Seymour, B., Ben Seymour, Dayan, P., & Dolan, R. J. (2011). Model-Based Influences on Humans' Choices and Striatal Prediction Errors. Neuron, 69(6), 1204-1215. http://doi.org/10.1016/j.neuron.2011.02.027

Daw, N. D., Gershman, S. J., Seymour, B., Ben Seymour, Dayan, P., & Dolan, R. J. (2011). Model-Based Influences on Humans' Choices and Striatal Prediction Errors. Neuron, 69(6), 1204-1215. http://doi.org/10.1016/j.neuron.2011.02.027

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- ts_par6(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- ts_par6(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

ts_par7 *Hybrid Model, with 7 parameters (original model)*

Description

Hierarchical Bayesian Modeling of the Two-Step Task using Hybrid Model, with 7 parameters (original model). It has the following parameters: `a1` (learning rate in stage 1), `beta1` (inverse temperature in stage 1), `a2` (learning rate in stage 2), `beta2` (inverse temperature in stage 2), `pi` (perseverance), `w` (model-based weight), `lambda` (eligibility trace).

- **Task**: Two-Step Task (Daw et al., 2011)
- **Model**: Hybrid Model, with 7 parameters (original model) (Daw et al., 2011)

Usage

```
ts_par7(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "level1_choice", "level2_choice", "reward". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.
`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred_step1", "y_pred_step2"
`adapt_delta`	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
`stepsize`	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
`max_treedepth`	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
`...`	For this model, it's possible to set **model-specific argument(s)** as follows: **trans_prob** Common state transition probability from Stage (Level) 1 to Stage (Level) 2. Defaults to 0.7.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Two-Step Task, there should be 4 columns of data with the labels "subjID", "level1_choice", "level2_choice", "reward". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

level1_choice Choice made for Level (Stage) 1 (1: stimulus 1, 2: stimulus 2).

level2_choice Choice made for Level (Stage) 2 (1: stimulus 3, 2: stimulus 4, 3: stimulus 5, 4: stimulus 6).
Note that, in our notation, choosing stimulus 1 in Level 1 leads to stimulus 3 & 4 in Level 2 with a common (0.7 by default) transition. Similarly, choosing stimulus 2 in Level 1 leads to stimulus 5 & 6 in Level 2 with a common (0.7 by default) transition. To change this default transition probability, set the function argument 'trans_prob' to your preferred value.

reward Reward after Level 2 (0 or 1).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Harhim Park <<hrpark12@gmail.com>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"ts_par7").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Daw, N. D., Gershman, S. J., Seymour, B., Ben Seymour, Dayan, P., & Dolan, R. J. (2011). Model-Based Influences on Humans' Choices and Striatal Prediction Errors. Neuron, 69(6), 1204-1215. http://doi.org/10.1016/j.neuron.2011.02.027

Daw, N. D., Gershman, S. J., Seymour, B., Ben Seymour, Dayan, P., & Dolan, R. J. (2011). Model-Based Influences on Humans' Choices and Striatal Prediction Errors. Neuron, 69(6), 1204-1215. http://doi.org/10.1016/j.neuron.2011.02.027

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- ts_par7(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- ts_par7(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
```

```
printFit(output)

## End(Not run)
```

`ug_bayes` *Ideal Observer Model*

Description

Hierarchical Bayesian Modeling of the Norm-Training Ultimatum Game using Ideal Observer Model. It has the following parameters: `alpha` (envy), `beta` (guilt), `tau` (inverse temperature).

- **Task**: Norm-Training Ultimatum Game
- **Model**: Ideal Observer Model (Xiang et al., 2013)

Usage

```
ug_bayes(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "offer", "accept". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to `FALSE`.

`inc_postpred`	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to `FALSE`. If set to `TRUE`, it includes: "y_pred"
`adapt_delta`	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
`stepsize`	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
`max_treedepth`	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
`...`	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Norm-Training Ultimatum Game, there should be 3 columns of data with the labels "subjID", "offer", "accept". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

offer Floating point value representing the offer made in that trial (e.g. 4, 10, 11).

accept 1 or 0, indicating whether the offer was accepted in that trial (where accepted == 1, rejected == 0).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The `nwarmup` argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"ug_bayes").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Xiang, T., Lohrenz, T., & Montague, P. R. (2013). Computational Substrates of Norms and Their Violations during Social Exchange. Journal of Neuroscience, 33(3), 1099-1108. http://doi.org/10.1523/JNEUROSCI.1642-12.2013

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: `https://rpubs.com/CCSL/hBayesDM`

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- ug_bayes(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- ug_bayes(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
```

```
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

ug_delta *Rescorla-Wagner (Delta) Model*

Description

Hierarchical Bayesian Modeling of the Norm-Training Ultimatum Game using Rescorla-Wagner (Delta) Model. It has the following parameters: `alpha` (envy), `tau` (inverse temperature), `ep` (norm adaptation rate).

- **Task**: Norm-Training Ultimatum Game
- **Model**: Rescorla-Wagner (Delta) Model (Gu et al., 2015)

Usage

```
ug_delta(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, `"example"` to use example data, or `"choose"` to choose data with an interactive window. Columns in the dataset must include: "subjID", "offer", "accept". See **Details** below for more information.
`niter`	Number of iterations, including warm-up. Defaults to 4000.
`nwarmup`	Number of iterations used for warm-up only. Defaults to 1000.
`nchain`	Number of Markov chains to run. Defaults to 4.
`ncore`	Number of CPUs to be used for running. Defaults to 1.
`nthin`	Every `i == nthin` sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
`inits`	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Whether to export model-based regressors (`TRUE` or `FALSE`). Not available for this model.

vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Norm-Training Ultimatum Game, there should be 3 columns of data with the labels "subjID", "offer", "accept". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

offer Floating point value representing the offer made in that trial (e.g. 4, 10, 11).

accept 1 or 0, indicating whether the offer was accepted in that trial (where accepted == 1, rejected == 0).

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The nwarmup argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"ug_delta").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Gu, X., Wang, X., Hula, A., Wang, S., Xu, S., Lohrenz, T. M., et al. (2015). Necessary, Yet Dissociable Contributions of the Insular and Ventromedial Prefrontal Cortices to Norm Adaptation: Computational and Lesion Evidence in Humans. Journal of Neuroscience, 35(2), 467-473. http://doi.org/10.1523/JNEUROSCI.2906-14.2015

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- ug_delta(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- ug_delta(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)
```

```
# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

---

wcs_sql *Sequential Learning Model*

Description

Hierarchical Bayesian Modeling of the Wisconsin Card Sorting Task using Sequential Learning Model. It has the following parameters: r (reward sensitivity), p (punishment sensitivity), d (decision consistency or inverse temperature).

- **Task**: Wisconsin Card Sorting Task
- **Model**: Sequential Learning Model (Bishara et al., 2010)

Usage

```
wcs_sql(data = NULL, niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "vb", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

data	Data to be modeled. It should be given as a data.frame object, a filepath for a tab-seperated txt file, "example" to use example data, or "choose" to choose data with an interactive window. Columns in the dataset must include: "subjID", "choice", "outcome". See **Details** below for more information.
niter	Number of iterations, including warm-up. Defaults to 4000.
nwarmup	Number of iterations used for warm-up only. Defaults to 1000.
nchain	Number of Markov chains to run. Defaults to 4.
ncore	Number of CPUs to be used for running. Defaults to 1.
nthin	Every i == nthin sample will be used to generate the posterior distribution. Defaults to 1. A higher number can be used when auto-correlation within the MCMC sampling is high.
inits	Character value specifying how the initial values should be generated. Possible options are "vb" (default), "fixed", "random", or your own initial values.
indPars	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".

modelRegressor	Whether to export model-based regressors (TRUE or FALSE). Not available for this model.
vb	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
inc_postpred	Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE. If set to TRUE, it includes: "y_pred"
adapt_delta	Floating point value representing the target acceptance probability of a new sample in the MCMC chain. Must be between 0 and 1. See **Details** below.
stepsize	Integer value specifying the size of each leapfrog step that the MCMC sampler can take on each new iteration. See **Details** below.
max_treedepth	Integer value specifying how many leapfrog steps the MCMC sampler can take on each new iteration. See **Details** below.
...	For this model, there is no model-specific argument.

Details

This section describes some of the function arguments in greater detail.

data should be assigned a character value specifying the full path and name (including extension information, e.g. ".txt") of the file that contains the behavioral data-set of all subjects of interest for the current analysis. The file should be a **tab-delimited** text file, whose rows represent trial-by-trial observations and columns represent variables.
For the Wisconsin Card Sorting Task, there should be 3 columns of data with the labels "subjID", "choice", "outcome". It is not necessary for the columns to be in this particular order, however it is necessary that they be labeled correctly and contain the information below:

subjID A unique identifier for each subject in the data-set.

choice Integer value indicating which deck was chosen on that trial: 1, 2, 3, or 4.

outcome 1 or 0, indicating the outcome of that trial: correct == 1, wrong == 0.

*Note: The file may contain other columns of data (e.g. "ReactionTime", "trial_number", etc.), but only the data within the column names listed above will be used during the modeling. As long as the necessary columns mentioned above are present and labeled correctly, there is no need to remove other miscellaneous data columns.

nwarmup is a numerical value that specifies how many MCMC samples should not be stored upon the beginning of each chain. For those familiar with Bayesian methods, this is equivalent to burn-in samples. Due to the nature of the MCMC algorithm, initial values (i.e. where the sampling chains begin) can have a heavy influence on the generated posterior distributions. The nwarmup argument can be set to a high number in order to curb the effects that initial values have on the resulting posteriors.

nchain is a numerical value that specifies how many chains (i.e. independent sampling sequences) should be used to draw samples from the posterior distribution. Since the posteriors are generated from a sampling process, it is good practice to run multiple chains to ensure that a reasonably representative posterior is attained. When the sampling is complete, it is possible to check the multiple chains for convergence by running the following line of code: `plot(output,type = "trace")`. The trace-plot should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC sampler, using only every `i == nthin` samples to generate posterior distributions. By default, `nthin` is equal to 1, meaning that every sample is used to generate the posterior.

Control Parameters: `adapt_delta`, `stepsize`, and `max_treedepth` are advanced options that give the user more control over Stan's MCMC sampler. It is recommended that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to 'The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo (Hoffman & Gelman, 2014, Journal of Machine Learning Research)' for more information on the sampler control parameters. One can also refer to 'Section 34.2. HMC Algorithm Parameters' of the Stan User's Guide and Reference Manual, or to the help page for `stan` for a less technical description of these arguments.

Contributors: Dayeong Min <<mindy2801@snu.ac.kr>>

Value

A class "hBayesDM" object `modelData` with the following components:

model Character value that is the name of the model (\code"wcs_sql").

allIndPars Data.frame containing the summarized parameter values (as specified by `indPars`) for each subject.

parVals List object containing the posterior samples over different parameters.

fit A class `stanfit` object that contains the fitted Stan model.

rawdata Data.frame containing the raw data used to fit the model, as specified by the user.

modelRegressor List object containing the extracted model-based regressors.

References

Bishara, A. J., Kruschke, J. K., Stout, J. C., Bechara, A., McCabe, D. P., & Busemeyer, J. R. (2010). Sequential learning models for the Wisconsin card sort task: Assessing processes in substance dependent individuals. Journal of Mathematical Psychology, 54(1), 5-13.

See Also

We refer users to our in-depth tutorial for an example of using hBayesDM: https://rpubs.com/CCSL/hBayesDM

Examples

```
## Not run:
# Run the model with a given data.frame as df
output <- wcs_sql(
  data = df, niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Run the model with example data
output <- wcs_sql(
  data = "example", niter = 2000, nwarmup = 1000, nchain = 4, ncore = 4)

# Visually check convergence of the sampling chains (should look like 'hairy caterpillars')
```

```
plot(output, type = "trace")

# Check Rhat values (all Rhat values should be less than or equal to 1.1)
rhat(output)

# Plot the posterior distributions of the hyper-parameters (distributions should be unimodal)
plot(output)

# Show the WAIC and LOOIC model fit estimates
printFit(output)

## End(Not run)
```

Index