Package 'hBayesDM'

February 12, 2019

Title Hierarchical Bayesian Modeling of Decision-Making Tasks

Version 0.7.2

Date 2019-02-11

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://rpubs.com/CCSL/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' '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' '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'

Repository CRAN

Date/Publication 2019-02-12 10:13:18 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

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` *2-Armed Bandit Task (Erev et al., 2010; Hertwig et al., 2004)*

Description

Hierarchical Bayesian Modeling of the 2-Armed Bandit Task with the following parameters: "A" (learning rate), "tau" (inverse temperature).

MODEL: Rescorla-Wagner (Delta) Model

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- bandit2arm_delta("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`

4-Armed Bandit Task (modified)

Description

Hierarchical Bayesian Modeling of the 4-Armed Bandit Task (modified) with 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).

Contributor: Yoonseo Zoh, Lei Zhang

MODEL: Kalman Filter (Daw et al., 2006, Nature)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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.

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 and store results in "output"
output <- bandit4arm2_kalman_filter("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` *4-Armed Bandit Task*

Description

Hierarchical Bayesian Modeling of the 4-Armed Bandit Task with the following parameters: "Arew" (reward learning rate), "Apun" (punishment learning rate), "xi" (noise).

MODEL: 3 Parameter Model, without C (choice perseveration), R (reward sensitivity), and P (punishment sensitivity). But with xi (noise) (Aylward et al., 2018, PsyArXiv)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- bandit4arm_2par_lapse("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-Armed Bandit Task*

Description

Hierarchical Bayesian Modeling of the 4-Armed Bandit Task with the following parameters: "Arew" (reward learning rate), "Apun" (punishment learning rate), "R" (reward sensitivity), "P" (punishment sensitivity).

MODEL: 4 Parameter Model, without C (choice perseveration) (Seymour et al., 2012, J Neuro)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.

`...` Not used for this model.

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 ("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.

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 and store results in "output"
output <- bandit4arm_4par("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` *4-Armed Bandit Task*

Description

Hierarchical Bayesian Modeling of the 4-Armed Bandit Task with the following parameters: "Arew" (reward learning rate), "Apun" (punishment learning rate), "R" (reward sensitivity), "P" (punishment sensitivity), "xi" (noise).

MODEL: 5 Parameter Model, without C (choice perseveration) but with xi (noise) (Seymour et al., 2012, J Neuro)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- bandit4arm_lapse("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` *4-Armed Bandit Task*

Description

Hierarchical Bayesian Modeling of the 4-Armed Bandit Task with the following parameters: "Arew" (reward learning rate), "Apun" (punishment learning rate), "R" (reward sensitivity), "P" (punishment sensitivity), "xi" (noise), "d" (decay rate).

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, PsyArXiv)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- bandit4arm_lapse_decay("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-Armed Bandit Task*

Description

Hierarchical Bayesian Modeling of the 4-Armed Bandit Task with the following parameters: "A" (learning rate), "R" (reward sensitivity), "P" (punishment sensitivity), "xi" (noise).

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

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- bandit4arm_singleA_lapse("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` *Balloon Analogue Risk Task (Ravenzwaaij et al., 2011)*

Description

Hierarchical Bayesian Modeling of the Balloon Analogue Risk Task with the following parameters: "phi" (prior belief of balloon not bursting), "eta" (updating rate), "gam" (risk-taking parameter), "tau" (inverse temperature).

Contributor: Harhim Park, Jaeyeong Yang, Ayoung Lee, Jeongbin Oh, Jiyoon Lee, Junha Jang

MODEL: Re-parameterized version (by Harhim Park & Jaeyeong Yang) of BART Model (Ravenzwaaij et al., 2011) with 4 parameters

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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.

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 and store results in "output"
output <- bart_par4("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` *Choice Reaction Time Task*

Description

Hierarchical Bayesian Modeling of the Choice Reaction Time Task with the following parameters: "alpha" (boundary separation), "beta" (bias), "delta" (drift rate), "tau" (non-decision time).

MODEL: Drift Diffusion Model (Ratcliff, 1978, Psychological Review)
*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

Parameters of the DDM (parameter names in Ratcliff), from https://github.com/gbiele/stan_wiener_test/blob/master/stan_wiener_test.R
- alpha (a): Boundary separation or Speed-accuracy trade-off (high alpha means high accuracy). 0 < alpha
- beta (b): Initial bias, for either response (beta > 0.5 means bias towards "upper" response 'A'). 0 < beta < 1
- delta (v): Drift rate; Quality of the stimulus (delta close to 0 means ambiguous stimulus or weak ability). 0 < delta
- tau (ter): Non-decision time + Motor response time + encoding time (high means slow encoding, execution). 0 < tau (in seconds)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
`inc_postpred`	**(Currently not available.)** Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE.
`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 the following **model-specific argument** to a value that you may prefer. `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 ("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.

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 and store results in "output"
output <- choiceRT_ddm("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` *Choice Reaction Time Task*

Description

Individual Bayesian Modeling of the Choice Reaction Time Task with the following parameters: "alpha" (boundary separation), "beta" (bias), "delta" (drift rate), "tau" (non-decision time).

MODEL: Drift Diffusion Model (Ratcliff, 1978, Psychological Review)
*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

Parameters of the DDM (parameter names in Ratcliff), from https://github.com/gbiele/stan_wiener_test/blob/master/stan_wiener_test.R
- alpha (a): Boundary separation or Speed-accuracy trade-off (high alpha means high accuracy). 0 < alpha
- beta (b): Initial bias, for either response (beta > 0.5 means bias towards "upper" response 'A'). 0 < beta < 1
- delta (v): Drift rate; Quality of the stimulus (delta close to 0 means ambiguous stimulus or weak ability). 0 < delta
- tau (ter): Non-decision time + Motor response time + encoding time (high means slow encoding, execution). 0 < tau (in seconds)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently not available for this model.
`vb`	Use variational inference to approximately draw from a posterior distribution. Defaults to FALSE.
`inc_postpred`	**(Currently not available.)** Include trial-level posterior predictive simulations in model output (may greatly increase file size). Defaults to FALSE.
`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 the following **model-specific argument** to a value that you may prefer. `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 ("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.

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 and store results in "output"
output <- choiceRT_ddm_single("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_lba *Choice Reaction Time task, linear ballistic accumulator modeling*

Description

Hierarchical Bayesian Modeling of choice/reaction time data with the following parameters: "d" (boundary), "A" (upper boundary of starting point), "v" (drift rate), "tau" (non-decision time). The model published in Annis, J., Miller, B. J., & Palmeri, T. J. (2016). Bayesian inference with Stan: A tutorial on adding custom distributions. Behavior research methods, 1-24.

MODEL: Brown and Heathcote LBA model - multiple subjects. Note that this implementation estimates a different drift rate for each condition-choice pair. For example, if the task involves deciding between two stimuli on each trial, and there are two different conditions throughout the task (e.g. speed versus accuracy), a total of 4 (2 stimuli by 2 conditions) drift rates will be estimated. For details on implementation, see Annis et al. (2016).

Usage

```
choiceRT_lba(data = "choose", niter = 3000, nwarmup = 1000,
  nchain = 2, ncore = 2, nthin = 1, inits = "random",
  indPars = "mean", saveDir = NULL, modelRegressor = FALSE,
  vb = FALSE, inc_postpred = FALSE, adapt_delta = 0.95,
  stepsize = 1, max_treedepth = 10)
```

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labelled as follows: "subjID", "choice", "RT", and "condition". See **Details** below for more information.
`niter`	Number of iterations, including warm-up.
`nwarmup`	Number of iterations used for warm-up only.
`nchain`	Number of chains to be run.
`ncore`	Integer value specifying how many CPUs to run the MCMC sampling on. 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. Options are "fixed" or "random" or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`saveDir`	Path to directory where .RData file of model output (`modelData`) can be saved. Leave blank if not interested.
`modelRegressor`	Exporting model-based regressors? TRUE or FALSE. Currently 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.
`adapt_delta`	Floating point number 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 that the MCMC sampler can take on each new iteration. See **Details** below.

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 of the file, including the file extension (e.g. ".txt"), that contains the behavioral data for the subject of interest for the current analysis. The file should be a **tab-delimited** text (.txt) file whose rows represent trial-by-trial observations and columns represent variables. For choice/reaction time tasks, there should be four columns of data with the labels "subjID", "choice", "RT", and "condition". It is not necessary for the columns to be in this particular order, however it is necessary that they be labelled correctly and contain the information below:

`"subjID"` A unique identifier for each subject within data-set to be analyzed.

`"choice"` An integer representing the choice made on the current trial. (e.g., 1 1 3 2 1 2).

`"RT"` A floating number the choice reaction time in **seconds**. (e.g., 0.435 0.383 0.314 0.309, etc.).

`"condition"` An integer representing the condition of the current trail (e.g., 1 2 3 4).

*Note: The data.txt file may contain other columns of data (e.g. "Reaction_Time", "trial_number", etc.), but only the data with the column names listed above will be used for analysis/modeling. As long as the columns above are present and labelled 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 value is equivalent to a burn-in sample. Due to the nature of MCMC sampling, initial values (where the sampling chain begins) 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 representative posterior is attained. When sampling is completed, the multiple chains may be checked for convergence with the `plot(myModel, type = "trace")` command. The chains should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC samples being chosen to generate the posterior distributions. By default, **nthin** is equal to 1, hence every sample is used to generate the posterior.

Contol Parameters: adapt_delta, stepsize, and max_treedepth are advanced options that give the user more control over Stan's MCMC sampler. The Stan creators recommend that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to Hoffman & Gelman (2014, Journal of Machine Learning Research) for more information on the functioning of the sampler control parameters. One can also refer to section 58.2 of the Stan User's Manual for a less technical description of these arguments.

Value

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

`model` Character string with the name of the model (`"choiceRT_lba"`).

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

`parVals` A `'list'` where each element contains posterior samples over different model parameters.

`fit` A class `'stanfit'` object containing the fitted model.

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

References

Brown, S. D., & Heathcote, A. (2008). The simplest complete model of choice response time: Linear ballistic accumulation. Cognitive Psychology, 57(3), 153-178. http://doi.org/10.1016/j.cogpsych.2007.12.002

Annis, J., Miller, B. J., & Palmeri, T. J. (2016). Bayesian inference with Stan: A tutorial on adding custom distributions. Behavior research methods, 1-24.

Hoffman, M. D., & Gelman, A. (2014). The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. The Journal of Machine Learning Research, 15(1), 1593-1623.

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 and store results in "output"
output <- choiceRT_lba(data = "example", niter = 2000, nwarmup = 1000, nchain = 3, ncore = 3)

# Visually check convergence of the sampling chains (should like 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_lba_single` *Choice Reaction Time task, linear ballistic accumulator modeling*

Description

Individual Bayesian Modeling of choice/reaction time data with the following parameters: "d" (boundary), "A" (upper boundary of starting point), "v" (drift rate), "tau" (non-decision time). The model published in Annis, J., Miller, B. J., & Palmeri, T. J. (2016). Bayesian inference with Stan: A tutorial on adding custom distributions. Behavior research methods, 1-24.

MODEL: Brown and Heathcote LBA model - single subject. Note that this implementation estimates a different drift rate for each condition-choice pair. For example, if the task involves deciding between two stimuli on each trial, and there are two different conditions throughout the task (e.g. speed versus accuracy), a total of 4 (2 stimuli by 2 conditions) drift rates will be estimated. For details on implementation, see Annis et al. (2016).

Usage

```
choiceRT_lba_single(data = "choose", niter = 3000, nwarmup = 1000,
  nchain = 2, ncore = 2, nthin = 1, inits = "random",
  indPars = "mean", saveDir = NULL, modelRegressor = FALSE,
  vb = FALSE, inc_postpred = FALSE, adapt_delta = 0.95,
  stepsize = 1, max_treedepth = 10)
```

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labelled as follows: "subjID", "choice", "RT", and "condition". See **Details** below for more information.
`niter`	Number of iterations, including warm-up.
`nwarmup`	Number of iterations used for warm-up only.
`nchain`	Number of chains to be run.
`ncore`	Integer value specifying how many CPUs to run the MCMC sampling on. 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. Options are "fixed" or "random" or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`saveDir`	Path to directory where .RData file of model output (`modelData`) can be saved. Leave blank if not interested.
`modelRegressor`	Exporting model-based regressors? TRUE or FALSE. Currently 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.
`adapt_delta`	Floating point number 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 that the MCMC sampler can take on each new iteration. See **Details** below.

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 of the file, including the file extension (e.g. ".txt"), that contains the behavioral data of all subjects of interest for the current analysis. The file should be a **tab-delimited** text (.txt) file whose rows represent trial-by-trial observations and columns represent variables. For choice/reaction time tasks, there should be four columns of data with the labels "choice", "RT", and "condition". It is not necessary for the columns to be in this particular order, however it is necessary that they be labelled correctly and contain the information below:

`"subjID"` A unique identifier for each subject within data-set to be analyzed.

`"choice"` An integer representing the choice made on the current trial. (e.g., 1 1 3 2 1 2).

`"RT"` A floating number the choice reaction time in **seconds**. (e.g., 0.435 0.383 0.314 0.309, etc.).

`"condition"` An integer representing the condition of the current trail (e.g., 1 2 3 4).

*Note: The data.txt file may contain other columns of data (e.g. "Reaction_Time", "trial_number", etc.), but only the data with the column names listed above will be used for analysis/modeling. As long as the columns above are present and labelled 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 value is equivalent to a burn-in sample. Due to the nature of MCMC sampling, initial values (where the sampling chain begins) 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 representative posterior is attained. When sampling is completed, the multiple chains may be checked for convergence with the `plot(myModel, type = "trace")` command. The chains should resemble a "furry caterpillar".

nthin is a numerical value that specifies the "skipping" behavior of the MCMC samples being chosen to generate the posterior distributions. By default, **nthin** is equal to 1, hence every sample is used to generate the posterior.

Contol Parameters: adapt_delta, stepsize, and max_treedepth are advanced options that give the user more control over Stan's MCMC sampler. The Stan creators recommend that only advanced users change the default values, as alterations can profoundly change the sampler's behavior. Refer to Hoffman & Gelman (2014, Journal of Machine Learning Research) for more information on the functioning of the sampler control parameters. One can also refer to section 58.2 of the Stan User's Manual for a less technical description of these arguments.

Value

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

`model` Character string with the name of the model (`"choiceRT_lba_single"`).

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

`parVals` A `'list'` where each element contains posterior samples over different model parameters.

`fit` A class `'stanfit'` object containing the fitted model.

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

References

Brown, S. D., & Heathcote, A. (2008). The simplest complete model of choice response time: Linear ballistic accumulation. Cognitive Psychology, 57(3), 153-178. http://doi.org/10.1016/j.cogpsych.2007.12.002

Annis, J., Miller, B. J., & Palmeri, T. J. (2016). Bayesian inference with Stan: A tutorial on adding custom distributions. Behavior research methods, 1-24.

Hoffman, M. D., & Gelman, A. (2014). The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. The Journal of Machine Learning Research, 15(1), 1593-1623.

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 and store results in "output"
output <- choiceRT_lba_single(data = "example", niter = 2000, nwarmup = 1000, nchain = 3, ncore = 3)

# Visually check convergence of the sampling chains (should like 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 *Choice Under Risk and Ambiguity Task*

Description

Hierarchical Bayesian Modeling of the Choice Under Risk and Ambiguity Task with the following parameters: "alpha" (risk attitude), "beta" (ambiguity attitude), "gamma" (inverse temperature).

Contributor: Jaeyeong Yang

MODEL: Exponential Subjective Value Model (Hsu et al., 2005, Science)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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 and store results in "output"
output <- cra_exp("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` *Choice Under Risk and Ambiguity Task*

Description

Hierarchical Bayesian Modeling of the Choice Under Risk and Ambiguity Task with the following parameters: "alpha" (risk attitude), "beta" (ambiguity attitude), "gamma" (inverse temperature).

Contributor: Jaeyeong Yang

MODEL: Linear Subjective Value Model (Levy et al., 2010, J Neurophysiol)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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 and store results in "output"
output <- cra_linear("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 *Description Based Decison Making Task*

Description

Hierarchical Bayesian Modeling of the Description Based Decison Making Task with the following parameters: "tau" (probability weight function), "rho" (subject utility function), "lambda" (loss aversion parameter), "beta" (inverse softmax temperature).

Contributor: Yoonseo Zoh

MODEL: Probability Weight Function (Erev et al., 2010; Hertwig et al., 2004; Jessup et al., 2008)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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.

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 and store results in "output"
output <- dbdm_prob_weight("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` *Delay Discounting Task*

Description

Hierarchical Bayesian Modeling of the Delay Discounting Task with the following parameters: "r" (exponential discounting rate), "s" (impatience), "beta" (inverse temperature).

MODEL: Constant-Sensitivity (CS) Model (Ebert & Prelec, 2007, Management Science)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- dd_cs("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 *Delay Discounting Task*

Description

Individual Bayesian Modeling of the Delay Discounting Task with the following parameters: "r" (exponential discounting rate), "s" (impatience), "beta" (inverse temperature).

MODEL: Constant-Sensitivity (CS) Model (Ebert & Prelec, 2007, Management Science)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- dd_cs_single("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 *Delay Discounting Task*

Description

Hierarchical Bayesian Modeling of the Delay Discounting Task with the following parameters: "r" (exponential discounting rate), "beta" (inverse temperature).

MODEL: Exponential Model (Samuelson, 1937, The Review of Economic Studies)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- dd_exp("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` *Delay Discounting Task*

Description

Hierarchical Bayesian Modeling of the Delay Discounting Task with the following parameters: "k" (discounting rate), "beta" (inverse temperature).

MODEL: Hyperbolic Model (Mazur, 1987)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- dd_hyperbolic("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 *Delay Discounting Task*

Description

Individual Bayesian Modeling of the Delay Discounting Task with the following parameters: "k" (discounting rate), "beta" (inverse temperature).

MODEL: Hyperbolic Model (Mazur, 1987)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- dd_hyperbolic_single("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(modelData = NULL, ic = "looic", ncore = 2)
```

Arguments

`modelData`	Object returned by `'hBayesDM'` model function
`ic`	Information Criterion. 'looic', 'waic', or 'both'
`ncore`	Number of corse 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` *Orthogonalized Go/Nogo Task*

Description

Hierarchical Bayesian Modeling of the Orthogonalized Go/Nogo Task with the following parameters: "xi" (noise), "ep" (learning rate), "rho" (effective size).

MODEL: RW + noise (Guitart-Masip et al., 2012, Neuroimage)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".

`modelRegressor`	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.
`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.
`...`	Not used for this model.

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 ("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 and store results in "output"
output <- gng_m1("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 *Orthogonalized Go/Nogo Task*

Description

Hierarchical Bayesian Modeling of the Orthogonalized Go/Nogo Task with the following parameters: "xi" (noise), "ep" (learning rate), "b" (action bias), "rho" (effective size).

MODEL: RW + noise + bias (Guitart-Masip et al., 2012, Neuroimage)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	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.
`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.
`...`	Not used for this model.

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 ("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 and store results in "output"
output <- gng_m2("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 *Orthogonalized Go/Nogo Task*

Description

Hierarchical Bayesian Modeling of the Orthogonalized Go/Nogo Task with the following parameters: "xi" (noise), "ep" (learning rate), "b" (action bias), "pi" (Pavlovian bias), "rho" (effective size).

MODEL: RW + noise + bias + pi (Guitart-Masip et al., 2012, Neuroimage)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	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.
`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.
`...`	Not used for this model.

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 ("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 and store results in "output"
output <- gng_m3("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 *Orthogonalized Go/Nogo Task*

Description

Hierarchical Bayesian Modeling of the Orthogonalized Go/Nogo Task with the following parameters: "xi" (noise), "ep" (learning rate), "b" (action bias), "pi" (Pavlovian bias), "rhoRew" (reward sensitivity), "rhoPun" (punishment sensitivity).

MODEL: RW (rew/pun) + noise + bias + pi (Cavanagh et al., 2013, J Neuro)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	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.
`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.
`...`	Not used for this model.

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 ("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 and store results in "output"
output <- gng_m4("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. Downloaded from John Kruschke's website http://www.indiana.edu/~kruschke/DoingBayesianDataAnalysis/

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 *Iowa Gambling Task*

Description

Hierarchical Bayesian Modeling of the Iowa Gambling Task with the following parameters: "Arew" (reward learning rate), "Apun" (punishment learning rate), "K" (perseverance decay), "betaF" (outcome frequency weight), "betaP" (perseverance weight).

Contributor: Nate Haines

MODEL: Outcome-Representation Learning Model (Haines et al., 2018, Cognitive Science)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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 the following **model-specific argument** to a value that you may prefer. `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 ("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.

References

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 and store results in "output"
output <- igt_orl("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` *Iowa Gambling Task*

Description

Hierarchical Bayesian Modeling of the Iowa Gambling Task with the following parameters: "A" (decay rate), "alpha" (outcome sensitivity), "cons" (response consistency), "lambda" (loss aversion).

MODEL: Prospect Valence Learning (PVL) Decay-RI (Ahn et al., 2014, Frontiers in Psychology)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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 the following **model-specific argument** to a value that you may prefer. `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 ("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.

References

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 and store results in "output"
output <- igt_pvl_decay("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 *Iowa Gambling Task (Ahn et al., 2008)*

Description

Hierarchical Bayesian Modeling of the Iowa Gambling Task with the following parameters: "A" (learning rate), "alpha" (outcome sensitivity), "cons" (response consistency), "lambda" (loss aversion).

MODEL: Prospect Valence Learning (PVL) Delta (Ahn et al., 2008, Cognitive Science)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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 the following **model-specific argument** to a value that you may prefer. `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 ("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.

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/10.1080/036402108023529

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 and store results in "output"
output <- igt_pvl_delta("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 *Iowa Gambling Task*

Description

Hierarchical Bayesian Modeling of the Iowa Gambling Task with 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).

MODEL: Value-Plus-Perseverance (Worthy et al., 2013, Frontiers in Psychology)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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 the following **model-specific argument** to a value that you may prefer. `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 ("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.

References

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 and store results in "output"
output <- igt_vpp("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` *Peer Influence Task (Chung et al., 2015)*

Description

Hierarchical Bayesian Modeling of the Peer Influence Task with the following parameters: "rho" (risk preference), "tau" (inverse temperature), "ocu" (other-conferred utility).

Contributor: Harhim Park

MODEL: Other-Conferred Utility (OCU) Model

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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.

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 and store results in "output"
output <- peer_ocu("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.*

Description

Based on John Kruschke's codes http://www.indiana.edu/~kruschke/DoingBayesianDataAnalysis/

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` *Probabilistic Reversal Learning Task*

Description

Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task with the following parameters: "phi" (1 - learning rate), "rho" (experience decay factor), "beta" (inverse temperature).

Contributor: (for model-based regressors) Jaeyeong Yang and Harhim Park

MODEL: Experience-Weighted Attraction Model (Ouden et al., 2013, Neuron)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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 and store results in "output"
output <- prl_ewa("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` *Probabilistic Reversal Learning Task*

Description

Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task with the following parameters: "eta" (learning rate), "alpha" (indecision point), "beta" (inverse temperature).

Contributor: (for model-based regressors) Jaeyeong Yang and Harhim Park

MODEL: Fictitious Update Model (Glascher et al., 2009, Cerebral Cortex)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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 and store results in "output"
output <- prl_fictitious("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` *Probabilistic Reversal Learning Task*

Description

Multiple-Block Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task with the following parameters: "eta" (learning rate), "alpha" (indecision point), "beta" (inverse temperature).

Contributor: (for model-based regressors) Jaeyeong Yang and Harhim Park

MODEL: Fictitious Update Model (Glascher et al., 2009, Cerebral Cortex)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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 and store results in "output"
output <- prl_fictitious_multipleB("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 *Probabilistic Reversal Learning Task*

Description

Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task with the following parameters: "eta_pos" (learning rate, +PE), "eta_neg" (learning rate, -PE), "alpha" (indecision point), "beta" (inverse temperature).

Contributor: (for model-based regressors) Jaeyeong Yang and Harhim Park

MODEL: Fictitious Update Model (Glascher et al., 2009, Cerebral Cortex), with separate learning rates for positive and negative prediction error (PE)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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 and store results in "output"
output <- prl_fictitious_rp("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` *Probabilistic Reversal Learning Task*

Description

Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task with the following parameters: "eta_pos" (learning rate, +PE), "eta_neg" (learning rate, -PE), "beta" (inverse temperature).

Contributor: (for model-based regressors) Jaeyeong Yang and Harhim Park

MODEL: Fictitious Update Model (Glascher et al., 2009, Cerebral Cortex), with separate learning rates for positive and negative prediction error (PE), without alpha (indecision point)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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 and store results in "output"
output <- prl_fictitious_rp_woa("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 *Probabilistic Reversal Learning Task*

Description

Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task with the following parameters: "eta" (learning rate), "beta" (inverse temperature).

Contributor: (for model-based regressors) Jaeyeong Yang and Harhim Park

MODEL: Fictitious Update Model (Glascher et al., 2009, Cerebral Cortex), without alpha (indecision point)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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 and store results in "output"
output <- prl_fictitious_woa("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` *Probabilistic Reversal Learning Task*

Description

Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task with the following parameters: "Apun" (punishment learning rate), "Arew" (reward learning rate), "beta" (inverse temperature).

Contributor: (for model-based regressors) Jaeyeong Yang and Harhim Park

MODEL: Reward-Punishment Model (Ouden et al., 2013, Neuron)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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 and store results in "output"
output <- prl_rp("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` *Probabilistic Reversal Learning Task*

Description

Multiple-Block Hierarchical Bayesian Modeling of the Probabilistic Reversal Learning Task with the following parameters: "Apun" (punishment learning rate), "Arew" (reward learning rate), "beta" (inverse temperature).

Contributor: (for model-based regressors) Jaeyeong Yang and Harhim Park

MODEL: Reward-Punishment Model (Ouden et al., 2013, Neuron)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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 and store results in "output"
output <- prl_rp_multipleB("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 *Probabilistic Selection Task*

Description

Hierarchical Bayesian Modeling of the Probabilistic Selection Task with the following parameters: "alpha_pos" (learning rate for positive feedbacks), "alpha_neg" (learning rate for negative feedbacks), "beta" (inverse temperature).

Contributor: Jaeyeong Yang

MODEL: Gain-Loss Q Learning Model (Frank et al., 2007, PNAS)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 below:

Code	Stimulus	Probability to win
1	A	80%
2	B	20%
3	C	70%
4	D	30%
5	E	60%
6	F	40%

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.

Value

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

`model` Character value that is the name of the model ("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.

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 and store results in "output"
output <- pst_gainloss_Q("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` *Risk Aversion Task*

Description

Hierarchical Bayesian Modeling of the Risk Aversion Task with the following parameters: "rho" (risk aversion), "tau" (inverse temperature).

MODEL: Prospect Theory (Sokol-Hessner et al., 2009, PNAS), without loss aversion (LA) parameter

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- ra_noLA("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)

## Not run:
# Paths to data published in Sokol-Hessner et al. (2009)
path_to_attend_data <- system.file("extdata", "ra_data_attend.txt", package = "hBayesDM")
path_to_regulate_data <- system.file("extdata", "ra_data_reappraisal.txt", package = "hBayesDM")

## End(Not run)
```

`ra_noRA` *Risk Aversion Task*

Description

Hierarchical Bayesian Modeling of the Risk Aversion Task with the following parameters: "lambda" (loss aversion), "tau" (inverse temperature).

MODEL: Prospect Theory (Sokol-Hessner et al., 2009, PNAS), without risk aversion (RA) parameter

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- ra_noRA("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)

## Not run:
# Paths to data published in Sokol-Hessner et al. (2009)
path_to_attend_data <- system.file("extdata", "ra_data_attend.txt", package = "hBayesDM")
path_to_regulate_data <- system.file("extdata", "ra_data_reappraisal.txt", package = "hBayesDM")

## End(Not run)
```

---

ra_prospect *Risk Aversion Task*

Description

Hierarchical Bayesian Modeling of the Risk Aversion Task with the following parameters: "rho" (risk aversion), "lambda" (loss aversion), "tau" (inverse temperature).

MODEL: Prospect Theory (Sokol-Hessner et al., 2009, PNAS)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- ra_prospect("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)

## Not run:
# Paths to data published in Sokol-Hessner et al. (2009)
path_to_attend_data <- system.file("extdata", "ra_data_attend.txt", package = "hBayesDM")
path_to_regulate_data <- system.file("extdata", "ra_data_reappraisal.txt", package = "hBayesDM")

## End(Not run)
```

`rdt_happiness` *Risky Decision Task*

Description

Hierarchical Bayesian Modeling of the Risky Decision Task with 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).

Contributor: Harhim Park

MODEL: Happiness Computational Model (Rutledge et al., 2014, PNAS)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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.

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 and store results in "output"
output <- rdt_happiness("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` *Two-Step Task (Daw et al., 2011)*

Description

Hierarchical Bayesian Modeling of the Two-Step Task with 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).

Contributor: Harhim Park

MODEL: Hybrid Model (Daw et al., 2011; Wunderlich et al., 2012), with 4 parameters

Usage

```
ts_par4(data = "choose", niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "random", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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 the following **model-specific argument** to a value that you may prefer. `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.

Value

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

`model` Character value that is the name of the model ("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.

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

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 and store results in "output"
output <- ts_par4("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` *Two-Step Task (Daw et al., 2011)*

Description

Hierarchical Bayesian Modeling of the Two-Step Task with 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).

Contributor: Harhim Park

MODEL: Hybrid Model (Daw et al., 2011, Neuron), with 6 parameters

Usage

```
ts_par6(data = "choose", niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "random", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".

`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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 the following **model-specific argument** to a value that you may prefer. `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.

Value

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

`model` Character value that is the name of the model ("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.

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

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 and store results in "output"
output <- ts_par6("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` *Two-Step Task (Daw et al., 2011)*

Description

Hierarchical Bayesian Modeling of the Two-Step Task with 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).

Contributor: Harhim Park

MODEL: Hybrid Model (Daw et al., 2011, Neuron), with 7 parameters (original model)

Usage

```
ts_par7(data = "choose", niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "random", indPars = "mean",
  modelRegressor = FALSE, vb = FALSE, inc_postpred = FALSE,
  adapt_delta = 0.95, stepsize = 1, max_treedepth = 10, ...)
```

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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 the following **model-specific argument** to a value that you may prefer. `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.

Value

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

`model` Character value that is the name of the model ("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.

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

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 and store results in "output"
output <- ts_par7("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 *Norm-Training Ultimatum Game*

Description

Hierarchical Bayesian Modeling of the Norm-Training Ultimatum Game with the following parameters: "alpha" (envy), "beta" (guilt), "tau" (inverse temperature).

MODEL: Ideal Observer Model (Xiang et al., 2013, J Neuro)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- ug_bayes("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` *Norm-Training Ultimatum Game*

Description

Hierarchical Bayesian Modeling of the Norm-Training Ultimatum Game with the following parameters: "alpha" (envy), "tau" (inverse temperature), "ep" (norm adaptation rate).

MODEL: Rescorla-Wagner (Delta) Model (Gu et al., 2015, J Neuro)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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 ("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.

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 and store results in "output"
output <- ug_delta("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 *Wisconsin Card Sorting Task*

Description

Hierarchical Bayesian Modeling of the Wisconsin Card Sorting Task with the following parameters: "r" (reward sensitivity), "p" (punishment sensitivity), "d" (decision consistency or inverse temperature).

Contributor: Dayeong Min

MODEL: Sequential Learning Model (Bishara et al., 2010, Journal of Mathematical Psychology)

Usage

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

Arguments

`data`	A .txt file containing the data to be modeled. Data columns should be labeled as: "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. Options are "fixed" or "random", or your own initial values.
`indPars`	Character value specifying how to summarize individual parameters. Current options are: "mean", "median", or "mode".
`modelRegressor`	Export model-based regressors? TRUE or FALSE. Currently 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.
`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.
`...`	Not used for this model.

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.

Value

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

`model` Character value that is the name of the model ("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.

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 and store results in "output"
output <- wcs_sql("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