Package 'hBayesDM'

January 3, 2018

Title Hierarchical Bayesian Modeling of Decision-Making Tasks

Version 0.5.0

Date 2017-12-25

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.

Depends R (>= 3.3.0), Rcpp (>= 0.12.14), methods

Imports rstan (>= 2.17.0), rstantools (>= 1.4.0), loo (>= 1.1.0), grid, parallel, ggplot2

URL http://rpubs.com/CCSL/hBayesDM

License GPL-3

LazyData true

Author Woo-Young Ahn [aut, cre], Nate Haines [aut], Lei Zhang [aut]

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-01-03 08:05:04 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

Delay Discounting Constant Sensitivity — dd_cs
Constant Sensitivity for single subject — dd_cs_single
Exponential — dd_exp
Hyperbolic — dd_hyperbolic
Hyperbolic for 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 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 Fictitious Update — prl_fictitious
Fictitious Update and multiple blocks per subject — prl_fictitious_multipleB
Experience-Weighted Attraction — prl_ewa
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

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

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

Choice/Reaction time Drift Diffusion Model — choiceRT_ddm
Drift Diffusion Model for single subject — choiceRT_ddm_single
Linear Ballistic Accumulator — choiceRT_lba
Linear Ballistic Accumulator for single subject — choiceRT_lba_single

Author(s)

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

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*. 1:1. https://doi.org/10.1101/064287

See Also

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

`bandit2arm_delta` *Two-Arm Bandit Task*

Description

Hierarchical Bayesian Modeling of the Two-Arm Bandit Task (e.g., Erev et al., 2010; Hertwig et al., 2004) using the following parameters: "A" (learning rate), "tau" (inverse temperature).

MODEL: Rescorla-Wagner (delta) model

Usage

```
bandit2arm_delta(data = "choose", niter = 3000, nwarmup = 1000,
  nchain = 4, ncore = 1, 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", and "outcome". 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 the Two-Arm Bandit Task, there should be three columns of data with the labels "subjID", "choice", and "outcome". 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"` Should contain a unique identifier for each subject within data-set to be analyzed.

`"choice"` Should contain a integer value representing the chosen choice option within the given trial (e.g., 1 or 2 in 2-arm bandit task).

`"outcome"` Should contain outcomes within each given trial (e.g., 1 = reward, -1 = loss).

*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 (`"bandit2arm_delta"`).

`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

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

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 <- bandit2arm_delta(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)
```

`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), and "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 = 2000,
  nchain = 4, ncore = 1, 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", "gain", and "loss". 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 the 4-armed bandit task, there should be four columns of data with the labels "subjID", "choice", "gain", and "loss". 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"` A nominal integer representing which choice was chosen within the given trial (e.g. 1, 2, 3, or 4).

`"gain"` A floating number representing the amount of currency won on the given trial (e.g. 50, 50, 100).

`"loss"` A floating number representing the amount of currency lost on the given trial (e.g. 0, -50).

*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 (`"bandit4arm_4par"`).

`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

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", 3000, 1000, 4, 4)  # 4 chains, 4 cores (parallel processing)

# 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)
```

`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), and "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 = 2000,
  nchain = 4, ncore = 1, 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", "gain", and "loss". 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 the 4-armed bandit task, there should be four columns of data with the labels "subjID", "choice", "gain", and "loss". 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"` A nominal integer representing which choice was chosen within the given trial (e.g. 1, 2, 3, or 4).

`"gain"` A floating number representing the amount of currency won on the given trial (e.g. 50, 50, 100).

`"loss"` A floating number representing the amount of currency lost on the given trial (e.g. 0, -50).

*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 (`"bandit4arm_lapse"`).

`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

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", 3000, 1000, 4, 4)  # 4 chains, 4 cores (parallel processing)

# 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_ddm *Choice Reaction Time task, drift diffusion modeling*

Description

Hierarchical Bayesian Modeling of choice/reaction time data with the following parameters: "alpha" (boundary separation), "beta" (bias), "delta" (drift rate), "tau" (non-decision time). The code is based on codes/comments by Guido Biele, Joseph Burling, Andrew Ellis, and potentially others @ Stan mailing

MODEL: Ratcliff drift diffusion model - multiple subjects. 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.

Usage

```
choiceRT_ddm(data = "choose", niter = 3000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "fixed", indPars = "mean",
  saveDir = NULL, RTbound = 0.1, 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", and "RT". 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.

RTbound	A floating point number representing the lower bound (i.e. minimum allowed) reaction time. Defaults to 0.1 (100 milliseconds).
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	(**Not currently available for DDM models**) 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 three columns of data with the labels "subjID", "choice", and "RT". 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. Lower/upper boundary or left/right choices should be coded as 1/2 (e.g., 1 1 1 2 1 2).

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

*Note: The data.txt file may contain other columns of data (e.g. "stimulus_name", "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_ddm"`).

`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

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

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_ddm(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_ddm_single` *Choice Reaction Time task, drift diffusion modeling*

Description

Individual Bayesian Modeling of choice/reaction time data with the following parameters: "alpha" (boundary separation), "beta" (bias), "delta" (drift rate), "tau" (non-decision time). The code is based on codes/comments by Guido Biele, Joseph Burling, Andrew Ellis, and potentially others @ Stan mailing

MODEL: Ratcliff drift diffusion model - single subject. 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.

Usage

```
choiceRT_ddm_single(data = "choose", niter = 3000, nwarmup = 1000,
  nchain = 4, ncore = 1, nthin = 1, inits = "fixed", indPars = "mean",
  saveDir = NULL, RTbound = 0.1, 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", and "RT". 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.
RTbound	A floating point number representing the lower bound (i.e. minimum allowed) reaction time. Defaults to 0.1 (100 milliseconds).
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	(**Not currently available for DDM models**) 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 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 two columns of data with the labels "subjID", "choice", and "RT". 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. Lower/upper boundary or left/right choices should be coded as 1/2 (e.g., 1 1 1 2 1 2).

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

*Note: The data.txt file may contain other columns of data (e.g. "subjID", "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_ddm_single"`).

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

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

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_ddm_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)
```

---

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)
```

---

dd_cs *Delay Discounting Task*

Description

Hierarchical Bayesian Modeling of the Delay Discounting Task using the following parameters: "r" (exponential discounting rate; impatience), "s" (time-sensitivity), "beta" (inverse temp.).

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

Usage

```
dd_cs(data = "choose", niter = 3000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "fixed", 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", "delay_later", "amount_later", "delay_sooner", "amount_sooner", and "choice". 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 the Delay Discounting Task, there should be six columns of data with the labels "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", and "choice". 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.

`"delay_later"` An integer representing the delayed days for the later option within the given trial. (e.g., 1 6 15 28 85 170).

`"amount_later"` A floating number representing the amount for the later option within the given trial. (e.g., 10.5 38.3 13.4 31.4 30.9, etc.).

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

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

`"choice"` An integer value representing the chosen option within the given trial (e.g., 0=instant amount, 1=delayed amount)

*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 (`"dd_cs"`).

`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

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

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 <- dd_cs(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)
```

---

dd_cs_single *Delay Discounting Task (Ebert & Prelec, 2007)*

Description

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

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

Usage

```
dd_cs_single(data = "choose", niter = 3000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "fixed", 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", "delay_later", "amount_later", "delay_sooner", "amount_sooner", and "choice". 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 the Delay Discounting Task, there should be six columns of data with the labels "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", and "choice". 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.

`"delay_later"` An integer representing the delayed days for the later option within the given trial. (e.g., 1 6 15 28 85 170).

`"amount_later"` A floating number representing the amount for the later option within the given trial. (e.g., 10.5 38.3 13.4 31.4 30.9, etc.).

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

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

`"choice"` An integer value representing the chosen option within the given trial (e.g., 0=instant amount, 1=delayed amount)

*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.

Value

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

`model` Character string with the name of the model ("dd_cs_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.

Examples

```
## Not run:
# Run the model and store results in "output"
```

```
output <- dd_cs_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)
```

---

dd_exp *Delay Discounting Task*

Description

Hierarchical Bayesian Modeling of the Delay Discounting Task using the following parameters: "r" (exponential discounting rate) & "beta" (inverse temp.).

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

Usage

```
dd_exp(data = "choose", niter = 3000, nwarmup = 1000, nchain = 4,
  ncore = 1, 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", "delay_later", "amount_later", "delay_sooner", "amount_sooner", and "choice". 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 the Delay Discounting Task, there should be six columns of data with the labels "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", and "choice". 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.

`"delay_later"` An integer representing the delayed days for the later option within the given trial. (e.g., 1 6 15 28 85 170).

`"amount_later"` A floating number representing the amount for the later option within the given trial. (e.g., 10.5 38.3 13.4 31.4 30.9, etc.).

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

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

`"choice"` An integer value representing the chosen option within the given trial (e.g., 0=instant amount, 1=delayed amount)

*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 (`"dd_exp"`).

`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

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

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 <- dd_exp(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)
```

---

dd_hyperbolic *Delay Discounting Task*

Description

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

MODEL: Hyperbolic Model (Mazur, 1987)

Usage

```
dd_hyperbolic(data = "choose", niter = 3000, nwarmup = 1000, nchain = 4,
  ncore = 1, 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", "delay_later", "amount_later", "delay_sooner", "amount_sooner", and "choice". 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 the Delay Discounting Task, there should be six columns of data with the labels "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", and "choice". 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.

`"delay_later"` An integer representing the delayed days for the later option within the given trial. (e.g., 1 6 15 28 85 170).

`"amount_later"` A floating number representing the amount for the later option within the given trial. (e.g., 10.5 38.3 13.4 31.4 30.9, etc.).

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

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

`"choice"` An integer value representing the chosen option within the given trial (e.g., 0=instant amount, 1=delayed amount)

*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 (`"dd_hyperbolic"`).

`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

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

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 <- dd_hyperbolic(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)
```

---

dd_hyperbolic_single *Delay Discounting Task (Ebert & Prelec, 2007)*

Description

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

MODEL: Hyperbolic

Usage

```
dd_hyperbolic_single(data = "choose", niter = 3000, nwarmup = 1000,
  nchain = 4, ncore = 1, 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", "delay_later", "amount_later", "delay_sooner", "amount_sooner", and "choice". 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 the Delay Discounting Task, there should be six columns of data with the labels "subjID", "delay_later", "amount_later", "delay_sooner", "amount_sooner", and "choice". 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.

`"delay_later"` An integer representing the delayed days for the later option within the given trial. (e.g., 1 6 15 28 85 170).

`"amount_later"` A floating number representing the amount for the later option within the given trial. (e.g., 10.5 38.3 13.4 31.4 30.9, etc.).

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

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

`"choice"` An integer value representing the chosen option within the given trial (e.g., 0=instant amount, 1=delayed amount)

*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.

Value

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

`model` Character string with the name of the model ("dd_hyperbolic_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.

Examples

```
## Not run:
# Run the model and store results in "output"
output <- dd_hyperbolic_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)
```

`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", core = 2)
```

Arguments

`modelData`	Object returned by `'hBayesDM'` model function
`ic`	Information Criterion. 'looic', 'waic', or 'both'
`core`	Number of cores to use for leave-one-out estimation

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 Orthogonolized Go/Nogo Task using the following parameters: "xi" (noise), "ep" (learning rate), and "rho" (effective size).

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

Usage

```
gng_m1(data = "choose", niter = 5000, nwarmup = 2000, nchain = 4,
  ncore = 1, 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", "cue", "keyPressed", and "outcome". 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 (Q(Go), Q(NoGo))? TRUE or FALSE.
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 the Go/No-Go Task, there should be four columns of data with the labels "subjID", "cue", "keyPressed", and "outcome". 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.

"cue" A nominal integer that specifies the cue shown within the given trial (e.g. 1, 2, 3, or 4 in the GNG).

"keyPressed" A binary value representing whether or not the participant responded on the given trial (1 == Press; 0 == No press).

"outcome" A 1, 0, or -1 for each given trial (1 == Positive Feedback; 0 == Neutral Feedback; -1 == Negative Feedback).

*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 (`"gng_m1"`).

`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

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

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 <- gng_m1(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)
```

---

gng_m2 *Orthogonalized Go/Nogo Task*

Description

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

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

Usage

```
gng_m2(data = "choose", niter = 5000, nwarmup = 2000, nchain = 4,
  ncore = 1, 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", "cue", "keyPressed", and "outcome". 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 (Q(Go), Q(NoGo))? TRUE or FALSE.
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 the Go/No-Go Task, there should be four columns of data with the labels "subjID", "cue", "keyPressed", and "outcome". 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.

"cue" A nominal integer that specifies the cue shown within the given trial (e.g. 1, 2, 3, or 4 in the GNG).

"keyPressed" A binary value representing whether or not the participant responded on the given trial (1 == Press; 0 == No press).

"outcome" A 1, 0, or -1 for each given trial (1 == Positive Feedback; 0 == Neutral Feedback; -1 == Negative Feedback).

*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 (`"gng_m2"`).

`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

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

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 <- gng_m2(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)
```

---

gng_m3 *Orthogonalized Go/Nogo Task*

Description

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

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

Usage

```
gng_m3(data = "choose", niter = 5000, nwarmup = 2000, nchain = 4,
  ncore = 1, nthin = 1, inits = "fixed", 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", "cue", "keyPressed", and "outcome". 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 (Q(Go), Q(NoGo))? TRUE or FALSE.
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 the Go/No-Go Task, there should be four columns of data with the labels "subjID", "cue", "keyPressed", and "outcome". 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.

"cue" A nominal integer that specifies the cue shown within the given trial (e.g. 1, 2, 3, or 4 in the GNG).

"keyPressed" A binary value representing whether or not the participant responded on the given trial (1 == Press; 0 == No press).

"outcome" A 1, 0, or -1 for each given trial (1 == Positive Feedback; 0 == Neutral Feedback; -1 == Negative Feedback).

*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 (`"gng_m3"`).

`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

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

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 <- gng_m3(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)
```

---

gng_m4 *Orthogonalized Go/Nogo Task*

Description

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

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

Usage

```
gng_m4(data = "choose", niter = 5000, nwarmup = 2000, nchain = 4,
  ncore = 1, nthin = 1, inits = "fixed", 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", "cue", "keyPressed", and "outcome". 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 (Q(Go), Q(NoGo))? TRUE or FALSE.
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 the Go/No-Go Task, there should be four columns of data with the labels "subjID", "cue", "keyPressed", and "outcome". 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.

"cue" A nominal integer that specifies the cue shown within the given trial (e.g. 1, 2, 3, or 4 in the GNG).

"keyPressed" A binary value representing whether or not the participant responded on the given trial (1 == Press; 0 == No press).

"outcome" A 1, 0, or -1 for each given trial (1 == Positive Feedback; 0 == Neutral Feedback; -1 == Negative Feedback).

*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 (`"gng_m4"`).

`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

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

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 <- gng_m4(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)
```

`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_pvl_decay` *Iowa Gambling Task*

Description

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

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

Usage

```
igt_pvl_decay(data = "choose", niter = 3000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "random", indPars = "mean",
  payscale = 100, 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", "gain", and "loss". 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".
payscale	Raw payoffs within data are divided by this number. Used for scaling data. Defaults to 100
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 the Iowa Gambling Task, there should be four columns of data with the labels "subjID", "choice", "gain", and "loss". 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"` A nominal integer representing which deck was chosen within the given trial (e.g. A, B, C, or D == 1, 2, 3, or 4 in the IGT).

`"gain"` A floating number representing the amount of currency won on the given trial (e.g. 50, 50, 100).

`"loss"` A floating number representing the amount of currency lost on the given trial (e.g. 0, -50).

*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 ("igt_pvl_decay").

`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

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

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 <- igt_pvl_decay(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)
```

`igt_pvl_delta` *Iowa Gambling Task (Ahn et al., 2008)*

Description

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

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

Usage

```
igt_pvl_delta(data = "choose", niter = 3000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "random", indPars = "mean",
  payscale = 100, 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", "gain", and "loss". 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".
`payscale`	Raw payoffs within data are divided by this number. Used for scaling data. Defaults to 100
`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 the Iowa Gambling Task, there should be four columns of data with the labels "subjID", "choice", "gain", and "loss". 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"` A nominal integer representing which deck was chosen within the given trial (e.g. A, B, C, or D == 1, 2, 3, or 4 in the IGT).

`"gain"` A floating number representing the amount of currency won on the given trial (e.g. 50, 50, 100).

`"loss"` A floating number representing the amount of currency lost on the given trial (e.g. 0, -50).

*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 ("igt_pvl_delta").

`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

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

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 <- igt_pvl_delta(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)
```

igt_vpp *Iowa Gambling Task*

Description

Hierarchical Bayesian Modeling of the Iowa Gambling Task using the following parameters: "A" (learning rate), "alpha" (outcome sensitivity), "cons" (response consistency), "lambda" (loss aversion), "epP" (gain impact), "epN" (loss impact), "K" (decay rate), and "w" (RL weight).

MODEL: Value-Plus-Perseverance (Worthy et al., 2014, Frontiers in Psychology)

Usage

```
igt_vpp(data = "choose", niter = 3000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "random", indPars = "mean",
  payscale = 100, 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", "gain", and "loss". 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".
payscale	Raw payoffs within data are divided by this number. Used for scaling data. Defaults to 100
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 the Iowa Gambling Task, there should be four columns of data with the labels "subjID", "choice", "gain", and "loss". 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"` A nominal integer representing which deck was chosen within the given trial (e.g. A, B, C, or D == 1, 2, 3, or 4 in the IGT).

`"gain"` A floating number representing the amount of currency won on the given trial (e.g. 50, 50, 100).

`"loss"` A floating number representing the amount of currency lost on the given trial (e.g. 0, -50).

*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 ("igt_vpp").

`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

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.

Worthy, D. A., & Todd Maddox, W. (2014). 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(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)
```

`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 Nature Neuroscience)*

Description

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

Contributor: Harhim Park (https://ccs-lab.github.io/team/harhim-park/)

MODEL: Peer influence task - OCU (other-conferred utility) model

Usage

```
peer_ocu(data = "choose", niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, nthin = 1, inits = "fixed", 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", "gain", "loss", "cert", and "gamble". 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 the Risk Aversion Task, there should be four 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 labelled correctly and contain the information below:

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

"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 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 ("ra_prospect").

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

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(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)
```

`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 and WAIC values) of hBayesDM Models*

Description

Print model-fits (mean LOOIC and WAIC values) of hBayesDM Models

Usage

```
printFit(..., ncore = 2, ic = "looic", roundTo = 3)
```

Arguments

...	Model objects output by hBayesDM functions (e.g. output1, output2, etc.)
ncore	Numer of cores to use when computing leave-one-out cross-validation
ic	Which information criterion to use? 'looic', 'waic', or 'both'
roundTo	Number of digits to the right of the decimal point in the output

Value

modelTable A table with relevant model comparison data

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 (PRL) Task using the following parameters: "phi" (1 - learning rate), "rho" (experience decay factor), and "beta" (inverse temperature).

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

Usage

```
prl_ewa(data = "choice", niter = 3000, nwarmup = 1000, nchain = 1,
  ncore = 1, 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", and "outcome". 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 the Probabilistic Reversal Learning Task, there should be three columns of data with the labels "subjID", "choice", and "outcome". 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 value representing the chosen choice option within the given trial (e.g., 1 or 2 in PRL).

`"outcome"` A 1 or -1 for outcome within each given trial (1 = reward, -1 = loss).

*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 (`"prl_ewa"`).

`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

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.

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(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)
```

`prl_fictitious` *Probabilistic Reversal Learning Task*

Description

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

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

Usage

```
prl_fictitious(data = "choice", niter = 3000, nwarmup = 1000,
  nchain = 1, ncore = 1, 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", and "outcome". 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 the Probabilistic Reversal Learning Task, there should be three columns of data with the labels "subjID", "choice", and "outcome". 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 value representing the chosen choice option within the given trial (e.g., 1 or 2 in PRL).

`"outcome"` A 1 or -1 for outcome within each given trial (1 = reward, -1 = loss).

*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 (`"prl_fictitious"`).

`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

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

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 <- prl_fictitious(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)
```

---

prl_fictitious_multipleB

Probabilistic Reversal Learning Task (Glascher et al, 2009), multiple blocks per subject

---

Description

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

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

Usage

```
prl_fictitious_multipleB(data = "choice", niter = 3000, nwarmup = 1000,
  nchain = 1, ncore = 1, 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", "outcome", adn "block". 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 the Probabilistic Reversal Learning Task, there should be three columns of data with the labels "subjID", "choice", and "rewlos". 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 value representing the chosen choice option within the given trial (e.g., 1 or 2 in PRL).

`"outcome"` A 1 or -1 for outcome within each given trial (1 = reward, -1 = loss).

`"block"` An integer value representing the block number of the current trial (e.g., 1 1 1 2 2 2).

*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.

Value

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

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

`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.

Examples

```
## Not run:
# Run the model and store results in "output"
output <- prl_fictitious_multipleB(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)
```

`prl_fictitious_rp` *Probabilistic Reversal Learning Task*

Description

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

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

Usage

```
prl_fictitious_rp(data = "choice", niter = 3000, nwarmup = 1000,
  nchain = 1, ncore = 1, 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", and "outcome". 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 the Probabilistic Reversal Learning Task, there should be three columns of data with the labels "subjID", "choice", and "outcome". 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 value representing the chosen choice option within the given trial (e.g., 1 or 2 in PRL).

`"outcome"` A 1 or -1 for outcome within each given trial (1 = reward, -1 = loss).

*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 (`"prl_fictitious"`).

`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

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

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.

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(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)
```

prl_rp *Probabilistic Reversal Learning Task*

Description

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

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

Usage

```
prl_rp(data = "choice", niter = 3000, nwarmup = 1000, nchain = 1,
  ncore = 1, 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", and "outcome". 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 the Probabilistic Reversal Learning Task, there should be three columns of data with the labels "subjID", "choice", and "outcome". 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 value representing the chosen choice option within the given trial (e.g., 1 or 2 in PRL).

`"outcome"` A 1 or -1 for outcome within each given trial (1 = reward, -1 = loss).

*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 ("prl_rp").

`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

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.

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(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)
```

prl_rp_multipleB *Probabilistic Reversal Learning Task, multiple blocks per subject*

Description

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

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

Usage

```
prl_rp_multipleB(data = "choice", niter = 3000, nwarmup = 1000,
  nchain = 1, ncore = 1, 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", "outcome", and "block". 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 the Probabilistic Reversal Learning Task, there should be three columns of data with the labels "subjID", "choice", and "outcome". 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 value representing the chosen choice option within the given trial (e.g., 1 or 2 in PRL).

`"outcome"` A 1 or -1 for outcome within each given trial (1 = reward, -1 = loss).

`"block"` An integer value representing the block number of the current trial (e.g., 1 1 1 2 2 2).

*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 (`"prl_rp_multipleB"`).

`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

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.

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(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)
```

`ra_noLA` *Risk Aversion Task*

Description

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

MODEL: Prospect Theory without a loss aversion (LA) parameter

Usage

```
ra_noLA(data = "choose", niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, 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", "gain", "loss", "cert", and "gamble". 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 the Risk Aversion Task, there should be four columns of data with the labels "subjID", "riskyGain", "riskyLoss", and "safeOption". 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.

`"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 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 (`"ra_prospect"`).

`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

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 <- ra_noLA(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)


# 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) and "tau" (inverse temp).

MODEL: Prospect Theory without a risk aversion (RA) parameter

Usage

```
ra_noRA(data = "choose", niter = 4000, nwarmup = 1000, nchain = 4,
  ncore = 1, 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", "gain", "loss", "cert", and "gamble". 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 the Risk Aversion Task, there should be four columns of data with the labels "subjID", "riskyGain", "riskyLoss", and "safeOption". 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.

`"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 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 (`"ra_prospect"`).

`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

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 <- ra_noRA(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)


# 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), and "tau" (inverse temp).

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",
  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", "gain", "loss", "cert", and "gamble". 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 the Risk Aversion Task, there should be four columns of data with the labels "subjID", "riskyGain", "riskyLoss", and "safeOption". 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.

`"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 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 (`"ra_prospect"`).

`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

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.

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://doi.org/10.2307/40455144?ref=search-gateway:1f452c8925000031ef87ca756455c9e3

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(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)


# 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)
```

`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.

`ug_bayes` *Norm-Training Ultimatum Game*

Description

Hierarchical Bayesian Modeling of the Norm-Training Ultimatum Game using 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 = 3000, nwarmup = 1000, nchain = 4,
  ncore = 1, 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", "offer", and "accept". 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 the Norm-Training Ultimatum Game, there should be three columns of data with the labels "subjID", "offer", and "accept". 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.

`"offer"` An real value representing the offer made within the given trial (e.g., 10, 11, 4, etc..).

"accept" A 1 or 0 indicating an offer was accepted or not (1 = accepted, 0 = rejected).

*Note: The data.txt file may contain other columns of data (e.g. "Reaction_Time", "trial", 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 ("ug_bayes").

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

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.

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(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)
```

---

ug_delta *Norm-Training Ultimatum Game*

Description

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

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

Usage

```
ug_delta(data = "choose", niter = 3000, nwarmup = 1000, nchain = 4,
  ncore = 1, 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", "offer", and "accept". 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 the Norm-Training Ultimatum Game, there should be three columns of data with the labels "subjID", "offer", and "accept". 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.

`"offer"` An real value representing the offer made within the given trial (e.g., 10, 11, 4, etc..).

`"accept"` A 1 or 0 indicating an offer was accepted or not (1 = accepted, 0 = rejected).

*Note: The data.txt file may contain other columns of data (e.g. "Reaction_Time", "trial", 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 (`"ug_delta"`).

`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

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

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 <- ug_delta(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)
```

Index