Package 'bistablehistory'

January 14, 2022

Title Cumulative History Analysis for Bistable Perception Time Series

Version 1.1.0

Description Estimates cumulative history for time-series for continuously viewed bistable perceptual rivalry displays. Computes cumulative history via a homogeneous first order differential process. I.e., it assumes exponential growth/decay of the history as a function time and perceptually dominant state, Pastukhov & Braun (2011) <doi:10.1167/11.10.12>. Supports Gamma, log normal, and normal distribution families. Provides a method to compute history directly and example of using the computation on a custom Stan code.

License GPL (>= 3)

URL https://github.com/alexander-pastukhov/bistablehistory

BugReports https://github.com/alexander-pastukhov/bistablehistory/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

VignetteBuilder knitr

Biarch true

Depends R (>= 3.5.0), loo, rlang, rstantools (>= 2.1.1)

Imports methods, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), dplyr, tibble, magrittr, glue, boot, future, purrr, tidyr

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements GNU make

Suggests testthat, knitr, rmarkdown, ggplot2

NeedsCompilation yes

Author Alexander Pastukhov [aut, cre] (<https://orcid.org/0000-0002-8738-8591>)

Maintainer Alexander Pastukhov <pastukhov.alexander@gmail.com>

Repository CRAN

Date/Publication 2022-01-13 23:12:41 UTC

R topics documented:

`bistablehistory-package` *Cumulative History Analysis for Bistable Perception Time Series*

Description

Estimates cumulative history for time-series for continuously viewed bistable perceptual rivalry displays. Computes cumulative history via a homogeneous first order differential process. I.e., it assumes exponential growth/decay of the history as a function time and perceptually dominant state, Pastukhov & Braun (2011) doi: 10.1167/11.10.12. Supports Gamma, log normal, and normal distribution families. Provides a method to compute history directly and example of using the computation on a custom Stan code.

References

Stan Development Team (2020). RStan: the R interface to Stan. R package version 2.21.2. https://mc-stan.org

See Also

`vignette("cumulative-history",package = "bistablehistory")` `vignette("usage-examples",package = "bistablehistory")` `vignette("writing-stan-code",package = "bistablehistory")`

`bayes_R2` *Computes R-squared using Bayesian R-squared approach.*

Description

For detail refer to: Andrew Gelman, Ben Goodrich, Jonah Gabry, and Aki Vehtari (2018). R-squared for Bayesian regression models. The American Statistician doi: 10.1080/00031305.2018.1549100 and https://avehtari.github.io/bayes_R2/bayes_R2.html

Usage

```
## S3 method for class 'cumhist'
bayes_R2(object, summary = TRUE, probs = c(0.055, 0.945), ...)
```

Arguments

`object`	An object of class cumhist
`summary`	Whether summary statistics should be returned instead of raw sample values. Defaults to `TRUE`
`probs`	The percentiles used to compute summary, defaults to 89% credible interval.
`...`	Unused.

Value

vector of values or a data.frame with summary

Examples

```
br_fit <- fit_cumhist(br_singleblock, state = "State", duration = "Duration")
bayes_R2(br_fit)
```

`br` *Binocular rivalry data*

Description

Dataset on binocular rivalry for eight participants.

Usage

```
br
```

Format

A data frame with 3769 rows and 6 variables:

Observer Participant ID.

Display Display, all rows contain `"BR"`

Block Run / block index.

Time Time relative to the run onset in *seconds*

State Factor with levels `"Left"`, `"Right"` (clear states), and `"Mixed"`.

Duration Duration of a dominance phase in *seconds*. Note that the duration for the last dominance phase is curtailed and, therefore, set to zero.

Source

doi: 10.1167/11.10.12

br_contrast *Binocular rivalry, variable contrast*

Description

Dataset on binocular rivalry with variable but equal contrast for six participants.

Usage

```
br_contrast
```

Format

A data frame with 4616 rows and 6 variables:

Observer Participant ID.

Block Run / block index.

Contrast Contrast on scale from 0 to 1.

Time Time relative to the run onset in *seconds*

State Factor with levels `"Left"`, `"Right"` (clear states), and `"Mixed"`.

Duration Duration of a dominance phase in *seconds*. Note that the duration for the last dominance phase is curtailed and, therefore, set to zero.

br_singleblock *Single run for binocular rivalry stimulus*

Description

A single subject / single run dataset for binocular rivalry.

Usage

```
br_singleblock
```

Format

A data frame with 76 rows and 6 variables:

Observer Participant ID, all rows contain *"ap"*

Group Display, all rows contain *"BR"*

Block Run / block index, all rows contain *1*

Time Time relative to the run onset in *seconds*

State Index of a perceptually dominant state, *1*, *2* - clear perceptual state, *3* mixed / transition phase

Duration Duration of a dominance phase in *seconds*. Note that the duration for the last dominance phase is curtailed and, therefore, set to zero.

Source

doi: 10.1167/11.10.12

`br_single_subject` *Single experimental session for binocular rivalry stimulus*

Description

A single subject / multiple runs dataset for binocular rivalry.

Usage

```
br_single_subject
```

Format

A data frame with 76 rows and 6 variables:

Observer Participant ID, all rows contain *"ap"*

Display Display, all rows contain *"BR"*

Block Run / block index

Time Time relative to the run onset in *seconds*

State Index of a perceptually dominant state, *1*, *2* - clear perceptual state, *3* mixed / transition phase

Duration Duration of a dominance phase in *seconds*. Note that the duration for the last dominance phase is curtailed and, therefore, set to zero.

Source

doi: 10.1167/11.10.12

`coef.cumhist` *Extract Model Coefficients*

Description

Extracts models population-level coefficients history-specific terms and fixed-effect terms for every modeled distribution parameter.

Usage

```
## S3 method for class 'cumhist'
coef(object, summary = TRUE, probs = c(0.055, 0.945), ...)
```

Arguments

object	An object of class cumhist
summary	Whether summary statistics should be returned instead of raw sample values. Defaults to TRUE
probs	The percentiles used to compute summary, defaults to 89% credible interval.
...	Unused.

Value

data.frame with values or summary

Examples

```
br_fit <- fit_cumhist(br_singleblock,
                      state = "State",
                      duration = "Duration",
                      fixed_effects = "Time")
coef(br_fit)
```

---

compute_history *Computes cumulative history for the time-series*

Description

Computes cumulative history for each state in the time-series.

Usage

```
compute_history(
  data,
  state,
  duration = NULL,
  onset = NULL,
  random_effect = NULL,
  session = NULL,
  run = NULL,
  tau = 1,
  mixed_state = 0.5,
  history_init = 0
)
```

Arguments

`data`	A table with time-series.
`state`	String, the name of the column that specifies perceptual state. The column type should be a factor with two or three levels (the third level is assumed to correspond to a transition/mixed phase) or should be convertible to a two level factor (as it would be impossible to infer the identity of transition/ mixed phase).
`duration`	String, name of the column with duration of individual perceptual dominance phases. Optional, you can specify `onset` instead.
`onset`	String, name of the column with onsets of the perceptual dominance states. Optional, used to compute duration of the dominance phases, if these are not provided explicitly via `duration` parameter.
`random_effect`	String, name of the column that identifies random effect, e.g. individual participants, stimuli for a single participant, etc. If omitted, no random effect is assumed. If specified and there is more than one level (participant, stimulus, etc.), it is used in a hierarchical model.
`session`	String, name of the column that identifies unique experimental session for which a mean dominance phase duration will be computed (see `norm_tau` parameter). Code assumes that session IDs are different within a participant but can be the same between them. If omitted, a single mean dominance duration based on the entire time series is used.
`run`	String, name of the column that identifies unique runs/blocks. If omitted, the data is assumed to belong to a single time series. Code assumes that run IDs are different within an experimental session but can be the same between the session. E.g. session A, runs 1, 2, 3.. and session B, runs 1, 2, 3 but not session A, runs 1, 2, 1.
`tau`	Time constant of exponential growth/decay normalized to the mean duration of clear percepts within each `session`. Can be 1) a single positive number (>0) that is used for all participants and runs, 2) `NULL` (default) - a *single* value will be fitted for all participants and runs, 3) `"random"` - an independent tau is fitted for each random cluster, 4) `"1|random"`- a tau for a random cluster is sampled from a population distribution, i.e., pooled parameter values via a multilevel model.
`mixed_state`	Specifies an activation level during transition/mixed phases (state #3, see `state`). Either a single number (range 0..1) that will be used as a fixed level or a vector of two numbers `c(mu,kappa)` that specifies, correspondingly, mean (range 0..1) and precision (>0) of beta proportion distribution, it should be sampled from. Defaults to a fixed value of `0.5`.
`history_init`	Initial value for cumulative history computation. Either a numeric scalar in 0..1 range or a vector of two numbers in 0..1 range. In the latter case, two histories will start at different levels.

Value

A matrix `nrow(data)` × 2 with computed history values

Examples

```
df <- compute_history(br_singleblock, state = "State",
                      duration = "Duration", tau = 1,
                      mixed_state = 0.5, history_init = 0)
```

---

cumhist-class *Class* cumhist.

Description

Cumulative history model fitted to time-series data.

Details

See `methods(class = "cumhist")` for an overview of available methods.

Slots

`family` A `string` with distribution family.

`data` A `list` with preprocessed data.

`stanfit` a `stanfit` object.

See Also

`fit_cumhist`

---

extract_history *Computes history for a fitted model*

Description

Computes history for a fitted model, uses only mean values for each history parameter. Uses values for each random cluster, if `"random"` or `"1|random"` parametrisation was used.

Usage

```
extract_history(object)
```

Arguments

`object`	An object of class cumhist

Value

A matrix of cumulative history values for each state

Examples

```
br_fit <- fit_cumhist(br_singleblock, state = "State", duration = "Duration")
extract_history(br_fit)
```

`extract_history_parameter` *Extracts a history parameter as a matrix*

Description

Extracts a history parameter as a matrix with `samplesN` rows and randomN (found in `object$data$randomN`) columns.

Usage

```
extract_history_parameter(
  object,
  param_name,
  samplesN = NULL,
  link_function = NULL
)
```

Arguments

`object`	A cumhist object
`param_name`	String, a name of the parameter
`samplesN`	Number of samples, if NULL is computed from rstan (but it is cheaper to do this once).
`link_function`	A link function to use (exp or inv.logit) or `NULL` for identity.

Value

Matrix with `samplesN` rows and randomN (found in `object$data$randomN`) columns

Examples

```
br_fit <- fit_cumhist(br_singleblock, state="State", duration="Duration")
extract_history_parameter(br_fit, "tau", link_function = exp)
```

extract_replicate_term_to_matrix

Extract a term and replicates it randomN times for each linear model

Description

Extract a term and replicates it randomN times for each linear model. Used for population mean or variance terms.

Usage

```
extract_replicate_term_to_matrix(object, term)
```

Arguments

object	An object of class cumhist
term	String, term name

Value

Matrix

Examples

```
br_fit <- fit_cumhist(br_singleblock, state = "State", duration = "Duration")
bH_mu <- extract_replicate_term_to_matrix(br_fit, "bH_mu")
```

extract_term_to_matrix

Extracts a term with one column per fixed or random-level into a matrix

Description

Extracts a 3D array for a term with sample, linear-model, random/fixed-effect order and returns a matrix with samples as rows and columns in order 1) all random/fixed effects for lm1, 2) all random/fixed effects for lm2, etc.

Usage

```
extract_term_to_matrix(object, term)
```

Arguments

object	An object of class cumhist
term	String, term name

Value

Matrix

Examples

```
br_fit <- fit_cumhist(br_singleblock, state = "State", duration = "Duration")
a <- extract_term_to_matrix(br_fit, "a")
```

---

fast_history_compute *Computes cumulative history*

Description

Computes cumulative history based on common history values and normalized_tau and mixed_state that are defined for each random cluster / individual.

Usage

```
fast_history_compute(df, normalized_tau, mixed_state, history_init)
```

Arguments

df	DataFrame with "state" (integer, 1 and 2 clear state, 3 - mixed state), "duration" (double), "irandom" (integer, 1-based index of a random cluster), "run_start" (integer, 1 for the first entry of the run, 0 otherwise), "session_tmean" (double)
normalized_tau	DoubleVector A normalized tau value for each random cluster / individual. Thus, its length must be equal to the number of unique indexes in df["irandom"].
mixed_state	DoubleVector A values used for the mixed state for each random cluster / individual. Thus, its length must be equal to the number of unique indexes in df["irandom"].
history_init	DoubleVector, size 2. Initial values of history for a run.

Value

NumericMatrix, size df.nrows() × 2. Computed history values for each state.

Examples

```
df <- preprocess_data(br_singleblock, state="State", duration="Duration")
fast_history_compute(df, 1, 0.5, c(0, 0))
```

`fit_cumhist` *Fits cumulative history for bistable perceptual rivalry displays.*

Description

Fits a generalized linear model using cumulative history and specified fixed effects.

Usage

```
fit_cumhist(
  data,
  state,
  duration = NULL,
  onset = NULL,
  random_effect = NULL,
  session = NULL,
  run = NULL,
  fixed_effects = NULL,
  tau = NULL,
  mixed_state = 0.5,
  history_init = 0,
  family = "gamma",
  history_priors = NULL,
  intercept_priors = NULL,
  history_effect_prior = NULL,
  fixed_effects_priors = NULL,
  chains = 1,
  cores = NULL,
  ...
)
```

Arguments

`data`	A table with time-series.
`state`	String, the name of the column that specifies perceptual state. The column type should be a factor with two or three levels (the third level is assumed to correspond to a transition/mixed phase) or should be convertible to a two level factor (as it would be impossible to infer the identity of transition/ mixed phase).
`duration`	String, name of the column with duration of individual perceptual dominance phases. Optional, you can specify `onset` instead.
`onset`	String, name of the column with onsets of the perceptual dominance states. Optional, used to compute duration of the dominance phases, if these are not provided explicitly via `duration` parameter.
`random_effect`	String, name of the column that identifies random effect, e.g. individual participants, stimuli for a single participant, etc. If omitted, no random effect is assumed. If specified and there is more than one level (participant, stimulus, etc.), it is used in a hierarchical model.

`session` String, name of the column that identifies unique experimental session for which a mean dominance phase duration will be computed (see `norm_tau` parameter). Code assumes that session IDs are different within a participant but can be the same between them. If omitted, a single mean dominance duration based on the entire time series is used.

`run` String, name of the column that identifies unique runs/blocks. If omitted, the data is assumed to belong to a single time series. Code assumes that run IDs are different within an experimental session but can be the same between the session. E.g. session A, runs 1, 2, 3.. and session B, runs 1, 2, 3 but not session A, runs 1, 2, 1.

`fixed_effects` String or vector of strings. Name of column(s) with values to be used for fitting an additional fixed effect(s). E.g., contrast in binocular rivalry, rotation speed for kinetic-depth effect, etc.

`tau` Time constant of exponential growth/decay normalized to the mean duration of clear percepts within each `session`. Can be 1) a single positive number (>0) that is used for all participants and runs, 2) `NULL` (default) - a *single* value will be fitted for all participants and runs, 3) `"random"` - an independent tau is fitted for each random cluster, 4) `"1|random"`- a tau for a random cluster is sampled from a population distribution, i.e., pooled parameter values via a multilevel model.

`mixed_state` Specifies an activation level during transition/mixed phases (state #3, see `state`). Either a single number (range 0..1) that will be used as a fixed level or a vector of two numbers `c(mu,kappa)` that specifies, correspondingly, mean (range 0..1) and precision (>0) of beta proportion distribution, it should be sampled from. Defaults to a fixed value of `0.5`.

`history_init` Initial value for cumulative history computation. Either a numeric scalar in 0..1 range or a vector of two numbers in 0..1 range. In the latter case, two histories will start at different levels.

`family` String, distribution used to fit duration of perceptual dominance phases. Options include `"gamma"` (default), `"lognormal"`, and `"normal"`.

`history_priors` Named list of optional priors for population-level cumulative history parameters. Must follow the format `list("tau"=c(1,0.15))` with values coding mean and standard deviation of the normal distribution.

`intercept_priors`
A vector of optional priors for population-level intercept parameter. Should be `c(<shape-mean>,<shape-sd>,<scale-mean>,<scale-sd>)` format for Gamma family, `c(<mean>,<sd>)` for normal and lognormal families. The values code mean and standard deviation of the normal distribution.

`history_effect_prior`
A vector of options priors for population-level slope of history effect. The values code mean and standard deviation of the normal distribution. Defaults to mu=0, sigma=1.

`fixed_effects_priors`
A named list of optional priors for fixed effects. Must follow the format `list("<name-of-variable>"=c`
where `<mu>` and `<sigma>` are mean and standard deviation of a normal distribution. Defaults to mu=0, sigma=1.

chains	Number of chains for sampling.
cores	Number of CPU cores to use for sampling. If omitted, All cores are used.
...	Additional arguments passed to rstan::sampling() function.

Value

An object of class cumhist

Examples

```
data(br_singleblock)
gamma_fit <- fit_cumhist(br_singleblock, state = "State", duration = "Duration")
```

---

fixef *Extract the fixed-effects estimates*

Description

Extracts models fixed-effect terms for every modeled distribution parameter.

Usage

```
fixef(object, summary = TRUE, probs = c(0.055, 0.945))
```

Arguments

object	An object of class cumhist
summary	Whether summary statistics should be returned instead of raw sample values. Defaults to TRUE
probs	The percentiles used to compute summary, defaults to 89% credible interval.

Value

tibble with values or summary, NULL if not fixed effects were used.

Examples

```
br_fit <- fit_cumhist(br_singleblock,
                      state = "State",
                      duration = "Duration",
                      fixed_effects = "Time")
fixef(br_fit)
```

historyef *Extract the history-effects estimates*

Description

Extracts models population-level coefficients history-specific terms for every modeled distribution parameter.

Usage

```
historyef(object, summary = TRUE, probs = c(0.055, 0.945))
```

Arguments

object	An object of class cumhist
summary	Whether summary statistics should be returned instead of raw sample values. Defaults to TRUE
probs	The percentiles used to compute summary, defaults to 89% credible interval.

Value

data.frame with values or summary

Examples

```
br_fit <- fit_cumhist(br_singleblock, state="State", duration="Duration")
historyef(br_fit)
```

history_mixed_state *Extract values of used or fitted history parameter mixed_state*

Description

A short-cut for `history_parameter(object,"mixed_state",...)`.

Usage

```
history_mixed_state(
  object,
  summary = TRUE,
  probs = c(0.055, 0.945),
  includePopulationLevel = TRUE
)
```

Arguments

`object`	An object of class cumhist
`summary`	Whether summary statistics should be returned instead of raw sample values. Defaults to `TRUE`
`probs`	The percentiles used to compute summary, defaults to 89% credible interval.
`includePopulationLevel`	Logical, for pooled random effect only. Whether to include population mean as a separate `"_population"` level, default to `TRUE`.

Value

A single value, if fixed value was used. A vector or a tibble, depending on the option used (single intercept, independent or random intercepts), and whether summary was requested.

Examples

```
br_fit <- fit_cumhist(br_singleblock, state="State", duration="Duration")
history_tau(br_fit)
```

`history_parameter` *Extract values of used or fitted history parameter*

Description

Extract values of used or fitted history parameter

Usage

```
history_parameter(
  object,
  param,
  summary = TRUE,
  probs = c(0.055, 0.945),
  includePopulationLevel = TRUE
)
```

Arguments

`object`	An object of class cumhist
`param`	Parameter name: `"tau"` or `"mixed_state"`
`summary`	Whether summary statistics should be returned instead of raw sample values. Defaults to `TRUE`
`probs`	The percentiles used to compute summary, defaults to 89% credible interval.
`includePopulationLevel`	Logical, for pooled random effect only. Whether to include population mean as a separate `"_population"` level, default to `TRUE`.

Value

A vector, if summary was not requested. Or a tibble with a summary or if a fixed value was used.

Examples

```
br_fit <- fit_cumhist(br_singleblock, state="State", duration="Duration")
history_parameter(br_fit, "tau")
```

`history_tau` *Extract values of used or fitted history parameter tau*

Description

A short-cut for `history_parameter(object,"tau",...)`.

Usage

```
history_tau(
  object,
  summary = TRUE,
  probs = c(0.055, 0.945),
  includePopulationLevel = TRUE
)
```

Arguments

`object`	An object of class cumhist
`summary`	Whether summary statistics should be returned instead of raw sample values. Defaults to `TRUE`
`probs`	The percentiles used to compute summary, defaults to 89% credible interval.
`includePopulationLevel`	Logical, for pooled random effect only. Whether to include population mean as a separate `"_population"` level, default to `TRUE`.

Value

A single value, if fixed value was used. A vector or a tibble, depending on the option used (single intercept, independent or random intercepts), and whether summary was requested.

Examples

```
br_fit <- fit_cumhist(br_singleblock, state="State", duration="Duration")
history_tau(br_fit)
```

kde *Kinetic-depth effect data*

Description

Dataset on kinetic-depth effect for eleven participants.

Usage

```
kde
```

Format

A data frame with 38698 rows and 6 variables:

Observer Participant ID.

Display Display, all rows contain `"KD"`

Block Run / block index.

Time Time relative to the run onset in *seconds*

State Factor with levels `"Left"`, `"Right"` (clear states), and `"Mixed"`.

Duration Duration of a dominance phase in *seconds*. Note that the duration for the last dominance phase is curtailed and, therefore, set to zero.

Source

doi: 10.1167/11.10.12

kde_two_observers *Multirun data for two participants, kinetic-depth effect display*

Description

Multirun data for two participants, kinetic-depth effect display

Usage

```
kde_two_observers
```

Format

A data frame with 1186 rows and 5 variables:

Observer Participant ID

Block Run / block index

State Factor variable for state with levels `-1` and `1` coding two clear perceptual states and `-2` the mixed / transition phase

Time Time relative to the run onset in *seconds*

Duration Duration of a dominance phase in *seconds*. Note that the duration for the last dominance phase is curtailed and, therefore, set to zero.

Source

doi: 10.1167/11.10.12

`loo.cumhist` *Computes an efficient approximate leave-one-out cross-validation via loo library. It can be used for a model comparison via loo::loo_compare() function.*

Description

Computes an efficient approximate leave-one-out cross-validation via loo library. It can be used for a model comparison via loo::loo_compare() function.

Usage

```
## S3 method for class 'cumhist'
loo(x, ...)
```

Arguments

`x`	A cumhist object
`...`	unused

Value

A named list, see `loo::loo()` for details.

Examples

```
data(br_singleblock)

gamma_fit <- fit_cumhist(br_singleblock, state="State", duration="Duration")
loo_gamma <- loo(gamma_fit)
```

nc *Necker cube data*

Description

Dataset on Necker cube for five participants.

Usage

```
nc
```

Format

A data frame with 3464 rows and 6 variables:

Observer Participant ID.

Display Display, all rows contain `"NC"`

Block Run / block index.

Time Time relative to the run onset in *seconds*

State Factor with levels `"Left"`, `"Right"` (clear states), and `"Mixed"`.

Duration Duration of a dominance phase in *seconds*. Note that the duration for the last dominance phase is curtailed and, therefore, set to zero.

Source

doi: 10.1167/11.10.12

predict.cumhist *Computes predicted dominance phase durations using posterior predictive distribution.*

Description

Computes predicted dominance phase durations using fitted model. Returns predicted values only for the dominance phases that were marked for use. I.e., excluding first and last dominance phases, mixed phases, etc. See `preprocess_data()`.

Usage

```
## S3 method for class 'cumhist'
predict(object, summary = TRUE, probs = NULL, full_length = TRUE, ...)
```

Arguments

object	An object of class cumhist
summary	Whether summary statistics should be returned instead of raw sample values. Defaults to TRUE
probs	The percentiles used to compute summary, defaults to NULL (no CI).
full_length	Only for summary = TRUE, whether the summary table should include rows with no predictions. I.e., rows with mixed phases, first/last dominance phase in the run, etc. See preprocess_data(). Defaults to TRUE.
...	Unused

Value

If summary=FALSE, a numeric matrix iterationsN x clearN. If summary=TRUE but probs=NULL a vector of mean predicted durations. If summary=TRUE and probs is not NULL, a data.frame with a column *"Predicted"* (mean) and a column for each specified quantile.

See Also

fit_cumhist

Examples

```
br_fit <- fit_cumhist(br_singleblock, state = "State", duration = "Duration")
predict(br_fit)

# full posterior prediction samples
predictions_samples <- predict(br_fit, summary=FALSE)
```

---

predict_samples *Computes prediction for a each sample.*

Description

Computing prediction for each sample, recomputing cumulative history and uses fitted parameter values.

Usage

```
predict_samples(
  family,
  fixedN,
  randomN,
  lmN,
  istate,
```

```
  duration,
  is_used,
  run_start,
  session_tmean,
  irandom,
  fixed,
  tau_ind,
  mixed_state_ind,
  history_init,
  a,
  bH,
  bF,
  sigma
)
```

Arguments

family	int, distribution family: gamma (1), lognormal(2), or normal (3).
fixedN	int, number of fixed parameters (>= 0).
randomN	int, number of random factors (>= 1).
lmN	int, number of linear models (>= 1).
istate	IntegerVector, zero-based perceptual state 0 or 1, 2 is mixed state.
duration	DoubleVector, duration of a dominance phase.
is_used	IntegerVector, whether dominance phase is used for prediction (1) or not (0).
run_start	IntegerVector, 1 whenever a new run starts.
session_tmean	DoubleVector, average dominance phase duration.
irandom	IntegerVector, zero-based index of a random effect.
fixed	NumericMatrix, matrix with fixed effect values.
tau_ind	NumericMatrix, matrix with samples of tau for each random level.
mixed_state_ind	NumericMatrix, matrix with samples of mixed_state for each random level.
history_init	DoubleVector, Initial values of history for a run
a	NumericMatrix, matrix with samples of a (intercept) for each random level.
bH	NumericMatrix, matrix with sample of bH for each linear model and random level.
bF	NumericMatrix, matrix with sample of bF for each linear model and fixed factor.
sigma	DoubleVector, samples of sigma.

Value

NumericMatrix with predicted durations for each sample.

preprocess_data *Preprocesses time-series data for fitting*

Description

Performs sanity checks (e.g., whether `data` can be used as a data.frame), computes duration of dominance phases (if necessary), assumes a single entry for any missing `session`, `run`, `random_effect`.

Usage

```
preprocess_data(
  data,
  state,
  duration = NULL,
  onset = NULL,
  random_effect = NULL,
  session = NULL,
  run = NULL
)
```

Arguments

`data`	A table with one or many time-series.
`state`	String, the name of the column that specifies perceptual state. The column type should be a factor with two or three levels (the third level is assumed to correspond to a transition/mixed phase) or should be convertible to a two level factor (as it would be impossible to infer the identity of transition/ mixed phase).
`duration`	String, name of the column with duration of individual perceptual dominance phases. Optional, you can specify `onset` instead.
`onset`	String, name of the column with onsets of the perceptual dominance states. Optional, used to compute duration of the dominance phases, if these are not provided explicitly via `duration` parameter.
`random_effect`	String, name of the column that identifies random effect, e.g. individual participants, stimuli for a single participant, etc. If omitted, no random effect is assumed. If specified and there is more than one level (participant, stimulus, etc.), it is used in a hierarchical model.
`session`	String, name of the column that identifies unique experimental session for which a mean dominance phase duration will be computed (see `norm_tau` parameter). Code assumes that session IDs are different within a participant but can be the same between them. If omitted, a single mean dominance duration based on the entire time series is used.
`run`	String, name of the column that identifies unique runs/blocks. If omitted, the data is assumed to belong to a single time series. Code assumes that run IDs are different within an experimental session but can be the same between the session. E.g. session A, runs 1, 2, 3.. and session B, runs 1, 2, 3 but not session A, runs 1, 2, 1.

Value

A tibble with columns

- `state`
- `duration`
- `random`
- `irandom` - integer, index of `random` values,
- `session`
- `run`
- `session_tmean` - numeric, mean duration of clear percepts for every combination of `random` and `session`.
- `is_used` - integer, whether computed history value needs to be used for linear model fitting.
- `run_start` - integer, 1 for the first row of the run time-series.

Examples

```
df <- preprocess_data(br_singleblock, state="State", duration="Duration")
```

`print.cumhist` *Prints out cumhist object*

Description

Prints out cumhist object

Usage

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

Arguments

`x`	A cumhist object
`...`	Unused

Value

Nothing, console output only.

Examples

```
br_fit <- fit_cumhist(br_singleblock, state="State", duration="Duration", fixed_effects="Time")
br_fit
```

summary.cumhist *Summary for a cumhist object*

Description

Summary for a cumhist object

Usage

```
## S3 method for class 'cumhist'
summary(object, ...)
```

Arguments

object	A cumhist object
...	Unused

Value

Nothing, console output only.

Examples

```
br_fit <- fit_cumhist(br_singleblock, state="State", duration="Duration")
summary(br_fit)
```

waic.cumhist *Computes widely applicable information criterion (WAIC).*

Description

Computes widely applicable information criterion via loo library. It can be used for a model comparison via loo::loo_compare() function.

Usage

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

Arguments

x	A cumhist object.
...	Additional arguments (unused)

Value

A named list, see `loo::waic()` for details.

Examples

```
data(br_singleblock)
gamma_fit <- fit_cumhist(br_singleblock, state="State", duration="Duration")
waic_gamma <- waic(gamma_fit)
normal_fit <- fit_cumhist(br_singleblock, state="State", duration="Duration", family="normal")
waic_normal <- waic(normal_fit)
loo::loo_compare(waic_gamma, waic_normal)
```

Index