Package ‘hbamr’

June 23, 2026

Title Hierarchical Bayesian Aldrich-McKelvey Scaling via 'Stan'

Version 2.4.7

Description Perform hierarchical Bayesian Aldrich-McKelvey scaling using Hamiltonian Monte Carlo via 'Stan'. Aldrich-McKelvey ('AM') scaling is a method for estimating the latent positions of survey respondents and external objects on a common scale using positional survey data. The hierarchical versions of the Bayesian 'AM' model included in this package outperform other versions both in terms of yielding meaningful posterior distributions for respondent positions and in terms of recovering true respondent positions in simulations. The package contains functions for preparing data, fitting models, extracting estimates, plotting key results, and comparing models using cross-validation. The original version of the default model is described in Bølstad (2024) <doi:10.1017/pan.2023.18>.

License GPL (>= 3)

URL https://jbolstad.github.io/hbamr/

BugReports https://github.com/jbolstad/hbamr/issues

Depends R (>= 3.4.0)

Imports colorspace, dplyr, future, future.apply, ggplot2, loo, matrixStats, methods, parallel, plyr, progressr, RColorBrewer, Rcpp (>= 1.0.7), RcppParallel (>= 5.1.4), rlang, rstan (>= 2.26.1), rstantools (>= 2.2.0), stats, tidyr

Suggests data.table, knitr, rmarkdown

LinkingTo BH (>= 1.66.0), Rcpp (>= 1.0.7), RcppEigen (>= 0.3.3.9.1), RcppParallel (>= 5.1.4), rstan (>= 2.26.1), StanHeaders (>= 2.26.22)

Biarch true

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

SystemRequirements GNU make

VignetteBuilder knitr

NeedsCompilation yes

Author Jørgen Bølstad [aut, cre] (ORCID:
<https://orcid.org/0000-0002-7623-5741>)

Maintainer Jørgen Bølstad <jorgen.bolstad@stv.uio.no>

Repository CRAN

Date/Publication 2026-06-23 02:40:03 UTC

Contents

hbamr-package 2
fbam 3
get_est 5
get_plot_data 6
hbam 7
hbam_cv 12
LC1980 15
LC2012 15
plot_by_group 16
plot_over_self 17
plot_respondents 18
plot_stimuli 19
prep_data 20
prep_data_cv 21

Index **23**

`hbamr-package` *Hierarchical Bayesian Aldrich-McKelvey Scaling via Stan*

Description

Fit hierarchical Bayesian Aldrich-McKelvey (HBAM) models using a form of Hamiltonian Monte Carlo via Stan. Aldrich-McKelvey (AM) scaling is a method for estimating the latent positions of survey respondents and external objects on a common scale using positional survey data. The hierarchical versions of the Bayesian AM model included in this package outperform other versions both in terms of yielding meaningful posterior distributions for respondent positions and in terms of recovering true respondent positions in simulations. The package contains functions for preparing data, fitting models, extracting estimates, plotting key results, and comparing models using cross-validation.

Author(s)

Jørgen Bølstad

References

- Bølstad, Jørgen (2024). Hierarchical Bayesian Aldrich-McKelvey Scaling. *Political Analysis*. 32(1): 50–64. doi:10.1017/pan.2023.18.
- Stan Development Team (2024). RStan: the R interface to Stan. https://mc-stan.org.

See Also

- https://jbolstad.github.io/hbamr/

`fbam` *Fit an FBAM model using optimization*

Description

Fit a simplified Bayesian Aldrich-McKelvey model with fixed hyperparameters using optimization via `rstan`. Users may replace the default priors by supplying their own values for the hyperparameters.

Usage

```
fbam(
  self = NULL,
  stimuli = NULL,
  model = "FBAM",
  allow_miss = 2,
  req_valid = NA,
  req_unique = 2,
  group_id = NULL,
  data = NULL,
  seed = sample.int(.Machine$integer.max, 1),
  sigma_alpha = NULL,
  sigma_beta = 0.3,
  sigma_mu_alpha = NULL,
  sigma_mu_beta = 0.2,
  ...
)
```

Arguments

`self`	An optional numerical vector of N ideological self-placements. Any missing data must be coded as NA. If this argument is not supplied (either here or in a previous call to `prep_data()`), respondent positions will not be estimated. If the data have been prepared in advance via the `prep_data()` function, the argument supplied here will be ignored.
`stimuli`	An N × J matrix of numerical stimulus placements, where J is the number of stimuli. Any missing data must be coded as NA. This argument will not be used if the data have been prepared in advance via the `prep_data()` function.

`model` Character: Name of the model to be used. Defaults to FBAM. The available options are the three models with "FBAM" in their name. See the documentation for the `hbam()` function for descriptions of the models.

`allow_miss` Integer specifying how many missing stimulus positions should be accepted for an individual still to be included in the analysis. This argument will not be used if the data have been prepared in advance via the `prep_data()` function. Defaults to 2.

`req_valid` Integer specifying how many valid observations to require for a respondent to be included in the analysis. The default is `req_valid = J - allow_miss`, but if specified, `req_valid` takes precedence. This argument will not be used if the data have been prepared in advance via the `prep_data()` function.

`req_unique` Integer specifying how many unique positions on the ideological scale each respondent is required to have used when placing the stimuli in order to be included in the analysis. The default is `req_unique = 2`. This argument will not be used if the data have been prepared in advance via the `prep_data()` function.

`group_id` Vector of length N identifying which group each respondent belongs to. The format can be factor, character, integer, or numeric. Respondents with NAs on `group_id` will be dropped when `group_id` is supplied. These data are only required by models with `"MULTI"` in their name and will be ignored when fitting other models.

`data` List of data that have been prepared in advance via the `prep_data()` function. Not required if the arguments `self` and `stimuli` are provided.

`seed` A positive integer specifying an optional seed for reproducibility. If this argument is not supplied, a random seed will be generated and the function will produce slightly different results on each run.

`sigma_alpha` A positive numeric value specifying the standard deviation of the prior on the shift parameters in the FBAM model, or the standard deviation of the parameters' deviation from the group-means in FBAM_MULTI models. (This argument will be ignored by HBAM models.) Defaults to B / 5, where B measures the length of the survey scale as the number of possible placements on one side of the center.

`sigma_beta` A positive numeric value specifying the standard deviation of the prior on the logged stretch parameters in the FBAM model, or the standard deviation of the logged parameters' deviation from the group-means in FBAM_MULTI models. (This argument will be ignored by HBAM models.) Defaults to .3.

`sigma_mu_alpha` A positive numeric value specifying the standard deviation of the prior on the group-means of the shift parameters in MULTI-type models. Defaults to B / 10.

`sigma_mu_beta` A positive numeric value specifying the standard deviation of the prior on the group-means of the logged stretch parameters in MULTI-type models. Defaults to .2.

`...` Arguments passed to `rstan::optimizing()`.

Value

A list produced by `rstan::optimizing()`.

Examples

```
# Loading ANES 2012 data:
data(LC2012)

# Making a small subset of the data for illustration:
self <- LC2012[1:1000, 2]
stimuli <- LC2012[1:1000, -c(1:2)]

# Fitting the FBAM model:
fit_fbam <- fbam(self, stimuli)

# Obtaining point estimates for the latent stimulus positions:
theta_est <- get_est(fit_fbam, par = "theta")
```

`get_est` *Extract point estimates or other summaries of marginal posterior distributions*

Description

For objects produced by `hbam()`, this function is a wrapper for `rstan::summary()`. For objects produced by `fbam()` it offers a way to extract point estimates.

Usage

```
get_est(
  object,
  par = "theta",
  format_orig = FALSE,
  probs = c(0.025, 0.5, 0.975),
  simplify = TRUE,
  ...
)
```

Arguments

`object`	An instance of class `stanfit` produced by `hbam()`, or a list produced by `fbam()`.
`par`	Character: Name of the parameter type to be extracted. Typically `"theta"` (stimulus positions) or `"chi"` (respondent positions). Descriptive aliases are also accepted: `"stimuli"` for `"theta"`, `"respondents"` for `"chi"`, `"shift"` for `"alpha"`, and `"stretch"` for `"beta"`.
`format_orig`	Logical: Should individual-level parameters be mapped to the original dataset by returning rows of NAs for respondents who were not included in the analysis? Defaults to `FALSE`.
`probs`	A numeric vector of quantiles of interest for summarizing `stanfit` objects.

`simplify`	Logical: Should the returned object be simplified by dropping the Monte Carlo standard error and the posterior standard deviation? Defaults to `TRUE`.
`...`	Other arguments are passed on to `rstan::summary()` when summarizing `stanfit` objects.

Value

A tibble containing summaries of marginal posterior distributions. For objects produced by `fbam()`, only maximum a posteriori estimates are returned.

`get_plot_data` *Extract data for plotting results from an HBAM model*

Description

Extract data for plotting results from an HBAM model.

Usage

```
get_plot_data(object, n_draws = 15, seed = 1)
```

Arguments

`object`	An instance of class `stanfit` produced by `hbam()` or a list produced by `fbam()`.
`n_draws`	Integer specifying the number of posterior draws to use when illustrating the uncertainty of the population distribution. This only applies for `stanfit` objects.
`seed`	A positive integer specifying an optional seed for reproducibility. The seed is used to select respondent position draws for illustrating uncertainty. This only applies for `stanfit` objects.

Value

A list of three `tibbles`: The first element contains the posterior mean stimulus positions, as well as the x- and y-values of the posterior modes (which can be useful for labeling the distributions). The second element contains the posterior draws for the stimulus positions (which can be used to calculate marginal posterior densities). The third element contains the selected number of posterior draws for each respondent (which form the key ingredient for `plot_respondents`).

hbam *Fit an HBAM model*

Description

Fit a Hierarchical Bayesian Aldrich-McKelvey model using automatically tuned Hamiltonian Monte Carlo sampling (NUTS) via `rstan`.

Usage

```
hbam(
  self = NULL,
  stimuli = NULL,
  model = "HBAM",
  allow_miss = 2,
  req_valid = NA,
  req_unique = 2,
  prefs = NULL,
  group_id = NULL,
  data = NULL,
  pars = c("alpha", "beta", "chi", "lambda", "theta"),
  extra_pars = NULL,
  include = TRUE,
  chains = 4,
  cores = parallel::detectCores(logical = FALSE),
  warmup = 1000,
  iter = 2000,
  seed = sample.int(.Machine$integer.max, 1),
  control = list(max_treedepth = 7),
  sigma_alpha = NULL,
  sigma_beta = 0.3,
  sigma_mu_alpha = NULL,
  sigma_mu_beta = 0.2,
  ...
)
```

Arguments

`self`	An optional numerical vector of N ideological self-placements. Any missing data must be coded as NA. If this argument is not supplied (either here or in a previous call to `prep_data()`), respondent positions will not be estimated. If the data have been prepared in advance via the `prep_data()` function, the argument supplied here will be ignored.
`stimuli`	An N × J matrix of numerical stimulus placements, where J is the number of stimuli. Any missing data must be coded as NA. This argument will not be used if the data have been prepared in advance via the `prep_data()` function.

`model` Character: Name of the model to be used. Defaults to HBAM. The available models are described under Details.

`allow_miss` Integer specifying how many missing stimulus positions should be accepted for an individual still to be included in the analysis. This argument will not be used if the data have been prepared in advance via the `prep_data()` function. Defaults to 2.

`req_valid` Integer specifying how many valid observations to require for a respondent to be included in the analysis. The default is `req_valid = J - allow_miss`, but if specified, `req_valid` takes precedence. This argument will not be used if the data have been prepared in advance via the `prep_data()` function.

`req_unique` Integer specifying how many unique positions on the ideological scale each respondent is required to have used when placing the stimuli in order to be included in the analysis. The default is `req_unique = 2`. This argument will not be used if the data have been prepared in advance via the `prep_data()` function.

`prefs` An N × J matrix of numerical stimulus ratings or preference scores. These data are only required by the HBAM_R_MINI model and will be ignored when fitting other models.

`group_id` Vector of length N identifying which group each respondent belongs to. The format can be factor, character, integer, or numeric. Respondents with NAs on `group_id` will be dropped when `group_id` is supplied. These data are only required by models with `"MULTI"` in their name and will be ignored when fitting other models.

`data` List of data that have been prepared in advance via the `prep_data()` function. Not required if the arguments `self` and `stimuli` are provided.

`pars` A vector of character strings specifying parameters of interest. If `include = TRUE`, only samples for parameters named in pars are stored in the fitted results. Conversely, if `include = FALSE`, samples for all parameters except those named in pars are stored in the fitted results. The default is to store results for "alpha", "beta", "chi", "lambda", and "theta".

`extra_pars` A vector of character strings specifying parameters or generated quantities to be added to `pars`. This makes it easy to add one or several additional parameters of interest, without having to repeat the default `pars` vector. The default is `NULL`.

`include` Logical scalar defaulting to `TRUE` indicating whether to include or exclude the parameters given by the pars argument. If `FALSE`, only entire multidimensional parameters can be excluded, rather than particular elements of them.

`chains` A positive integer specifying the number of Markov chains. Defaults to 4.

`cores` The number of cores to use when executing the Markov chains in parallel. By default, all detected physical cores will be used if `chains` is equal to or higher than the number of cores.

`warmup` A positive integer specifying the number of warmup (aka burn-in) iterations per chain. If step-size adaptation is on (which it is by default), this also controls the number of iterations for which adaptation is run (and hence these warmup samples should not be used for inference). The number of warmup iterations should be smaller than `iter`.

`iter` A positive integer specifying the number of iterations for each chain (including warmup).

`seed` A positive integer specifying an optional seed for reproducibility. If this argument is not supplied, a random seed will be generated and the function will produce slightly different results on each run.

`control` A named list of parameters to control the sampler's behavior. See the documentation for `rstan::stan` for more details.

`sigma_alpha` A positive numeric value specifying the standard deviation of the prior on the shift parameters in the FBAM model, or the standard deviation of the parameters' deviation from the group-means in FBAM_MULTI models. (This argument will be ignored by HBAM models.) Defaults to B / 5, where B measures the length of the survey scale as the number of possible placements on one side of the center.

`sigma_beta` A positive numeric value specifying the standard deviation of the prior on the logged stretch parameters in the FBAM model, or the standard deviation of the logged parameters' deviation from the group-means in FBAM_MULTI models. (This argument will be ignored by HBAM models.) Defaults to .3.

`sigma_mu_alpha` A positive numeric value specifying the standard deviation of the prior on the group-means of the shift parameters in MULTI-type models. Defaults to B / 10.

`sigma_mu_beta` A positive numeric value specifying the standard deviation of the prior on the group-means of the logged stretch parameters in MULTI-type models. Defaults to .2.

`...` Arguments passed to `rstan::sampling()`.

Details

This package provides several alternative models that can be selected using the names below. Users who are unsure which model to use are advised to use the default HBAM model. If speed or sampling diagnostics are an issue, HBAM_MINI may provide a useful alternative.

HBAM is the default model, which allows for scale flipping and employs hierarchical priors on the shift and stretch parameters. It also models heteroskedastic errors that vary by both individual and stimuli. Compared to the model in Bølstad (2024), this version has been slightly revised to provide faster sampling. A key difference from the original model is that the respondent positions are not treated as parameters, but rather calculated as a function of self-placements, individual-level parameters, and simulated errors. This makes the model considerably faster, while yielding very similar results. The model simulates errors in the self-placements of the same magnitude as that with which the respondent in question places the stimulus with the smallest errors. All models in the package use this approach.

HBAM_MULTI is a version that models differences between groups defined by the user. It requires a vector identifying the groups to be supplied as the argument `group_id`. The model gives each group separate hyperparameters for the locations of the prior distributions for the shift and stretch parameters. Rather than shrinking the estimates toward the mode for the whole dataset, this model shrinks the estimates toward the mode for the group. The vectors of hyperparameters are called `mu_alpha` and `mu_beta` and are constructed to have means of 0. The scales of the priors on these hyperparameters can be set by the user via the arguments `sigma_mu_alpha` and `sigma_mu_beta`. The default values are B / 10 and .2, respectively. (Here, B measures the length of the survey scale

as the number of possible placements on one side of the center.) One potential use for this model is to supply self-placements as `group_id`, and thus give each self-placement group its own prior distribution for the shift and stretch parameters.

HBAM_NF (formerly HBAM_0) is a version of the HBAM model that does not allow for scale flipping. This may be useful if there are truly zero cases of scale flipping in the data. Such scenarios can be created artificially, but may also arise in real data. For example, expert surveys appear unlikely to contain many instances of scale flipping. For data that contain zero cases of flipping, models that allow for flipping contain superfluous parameters that lead to inefficient sampling. Models that do not allow for flipping will sample faster and typically yield slightly more accurate estimates. Such models are therefore usually preferable when no flipping is present.

HBAM_MULTI_NF is a version of the HBAM_MULTI model that does not allow for scale flipping.

HBAM_MINI is a version of the HBAM model that assumes the prediction errors in the stimuli placements to be homoskedastic. This model tends to sample faster than the standard HBAM model while yielding very similar point estimates. For large datasets, this model may provide a reasonable compromise between model complexity and estimation speed.

FBAM is a version of the HBAM model with fixed hyperparameters to allow fitting via optimization rather than MCMC – which can be useful for large data sets. This model allows the user to specify the scales of the priors for the shift and (logged) stretch parameters via the arguments `sigma_alpha` and `sigma_beta`. The default values are B / 5 and .3, respectively. These defaults are intended to be realistic and moderately informative. Users who want to control the degree of shrinkage of the individual-level parameters may find it useful to fit this model – or other FBAM models – via either MCMC or optimization.

FBAM_MULTI is a version of the FBAM model that shares the group-modeling features of the HBAM_MULTI model. It allows the user to set the scales of the priors for the shift and stretch parameters via the arguments `sigma_alpha` and `sigma_beta`, and set the scales of the priors on `mu_alpha` and `mu_beta` via the arguments `sigma_mu_alpha` and `sigma_mu_beta`.

FBAM_MULTI_NF is a version of the FBAM_MULTI model that does not allow for scale flipping.

HBAM_R_MINI is a version of the HBAM_MINI model that incorporates the rationalization component of the ISR model by Bølstad (2020). This model requires additional data to be supplied as the argument `prefs`: An N × J matrix of stimuli ratings from the respondents. The rationalization part of the model is simplified relative to the original ISR model: The direction in which respondents move disfavored stimuli is estimated as a common expectation for each possible self-placement on the scale.

BAM is an unpooled model with wide uniform priors on the shift and stretch parameters. It is similar to the JAGS version introduced by Hare et al. (2015), although the version included here has been adjusted to yield stretch parameters with an average of approximately one (and thus produce a scale similar to those of the other models in the package). Like the other models, this version of the BAM model also simulates errors in the self-placements to yield a realistic level of uncertainty. While this model is simple and fast, it tends to overfit the data and produce invalid posterior distributions for some respondent positions (see Bølstad 2024). However, it could potentially be useful as a baseline for model comparisons in situations where respondent positions are not of interest.

HBAM_2 has been replaced by the more general HBAM_MULTI model.

These models can also be used in situations where self-placements are not available and the only goal is to estimate stimulus positions or respondents' shift and stretch parameters. While the latent

respondent positions will not be estimated, all other parameters are unaffected if the argument `self` is dropped when calling `prep_data()` or `hbam()`.

See the `hbamr` vignette for a table summarizing the key characteristics of the available models.

Value

An object of S4 class `stanfit`.

References

- Bølstad, Jørgen (2024). Hierarchical Bayesian Aldrich-McKelvey Scaling. *Political Analysis*. 32(1): 50–64. doi:10.1017/pan.2023.18.
- Bølstad, Jørgen (2020). Capturing Rationalization Bias and Differential Item Functioning: A Unified Bayesian Scaling Approach. *Political Analysis* 28(3): 340–355.
- Hare, Christopher et al. (2015). Using Bayesian Aldrich-McKelvey Scaling to Study Citizens' Ideological Preferences and Perceptions. *American Journal of Political Science* 59(3): 759–774.

Examples

```
# Loading and re-coding ANES 1980 data:
data(LC1980)
LC1980[LC1980 == 0 | LC1980 == 8 | LC1980 == 9] <- NA

# Making a small subset of the data for illustration:
self <- LC1980[1:100, 1]
stimuli <- LC1980[1:100, -1]

# Fitting the HBAM_MINI model, obtaining 1000 draws:
fit_hbam_mini <- hbam(self, stimuli, model = "HBAM_MINI",
                    warmup = 500, iter = 1000, chains = 2)

# Preparing the data before fitting, requiring complete responses:
dat <- prep_data(self, stimuli, allow_miss = 0)
fit_hbam_mini <- hbam(data = dat, model = "HBAM_MINI",
                    warmup = 500, iter = 1000, chains = 2)

# Obtaining posterior summaries for the latent stimulus positions:
stim_est <- get_est(fit_hbam_mini, "stimuli")

# Obtaining posterior summaries for the latent respondent positions
  # in a format matching the rows in the original dataset:
resp_est <- get_est(fit_hbam_mini, "respondents", format_orig = TRUE)

# Fitting the FBAM_MULTI_NF model with self-placements as group_id:
fit_fbam_multi_nf <- hbam(self, stimuli, group_id = self, model = "FBAM_MULTI_NF",
                        warmup = 500, iter = 1000, chains = 2)
```

hbam_cv *Perform K-fold cross-validation*

Description

This function performs K-fold cross-validation for an HBAM or FBAM model in order to estimate the expected log pointwise predictive density for a new dataset (ELPD). Multiple chains for one or more folds can be run in parallel using the `future` package.

Usage

```
hbam_cv(
  self = NULL,
  stimuli = NULL,
  model = "HBAM",
  allow_miss = 0,
  req_valid = NA,
  req_unique = 2,
  prefs = NULL,
  group_id = NULL,
  prep_data = TRUE,
  data = NULL,
  K = 10,
  chains = 2,
  warmup = 1000,
  iter = 3000,
  seed = 1,
  control = list(max_treedepth = 7),
  sigma_alpha = NULL,
  sigma_beta = 0.35,
  sigma_mu_alpha = NULL,
  sigma_mu_beta = 0.3,
  ...
)
```

Arguments

`self`	A numerical vector of N ideological self-placements. Any missing data must be coded as NA. This argument will not be used if the data have been prepared in advance via the `prep_data()` function.
`stimuli`	An N × J matrix of numerical stimulus placements, where J is the number of stimuli. Any missing data must be coded as NA. This argument will not be used if the data have been prepared in advance via the `prep_data()` function.
`model`	Character: Name of the model to be used. Defaults to HBAM.
`allow_miss`	Integer specifying how many missing stimulus positions should be accepted for an individual still to be included in the analysis. This argument will not be

used if the data have been prepared in advance via the `prep_data()` function. Defaults to 0.

`req_valid` Integer specifying how many valid observations to require for a respondent to be included in the analysis. The default is `req_valid = J - allow_miss`, but if specified, `req_valid` takes precedence. This argument will not be used if the data have been prepared in advance via the `prep_data()` function.

`req_unique` Integer specifying how many unique positions on the ideological scale each respondent is required to have used when placing the stimuli in order to be included in the analysis. The default is `req_unique = 2`. This argument will not be used if the data have been prepared in advance via the `prep_data()` function.

`prefs` An N × J matrix of numerical stimulus ratings or preference scores. These data are only required by the HBAM_R_MINI model and will be ignored when fitting other models.

`group_id` Integer vector of length N identifying which group each respondent belongs to. The supplied vector should range from 1 to the total number of groups in the data, and all integers between these numbers should be represented in the supplied data. These data are only required by models with "MULTI" in their name and will be ignored when fitting other models.

`prep_data` Logical: Should the data be prepared before fitting the model? If the data have already been prepared in advance by first running `prep_data()` and `prep_data_cv()`, set `prep_data = FALSE`. Defaults to `prep_data = TRUE`.

`data` A list of data produced by `prep_data()` followed by `prep_data_cv()`.

`K` An integer above 2, specifying the number of folds to use in the analysis. Defaults to 10.

`chains` A positive integer specifying the number of Markov chains to use per fold. Defaults to 2.

`warmup` A positive integer specifying the number of warmup (aka burn-in) iterations per chain. It defaults to 1000. The number of warmup iterations should be smaller than `iter`.

`iter` A positive integer specifying the number of iterations for each chain (including warmup). It defaults to 3000 as running fewer chains for longer is a more efficient way to obtain a certain number of draws (and cross-validation can be computationally expensive).

`seed` An integer passed on to `set.seed` before creating the folds to increase reproducibility and comparability. Defaults to 1 and only applies to fold-creation when the argument `prep_data` is `TRUE`. The supplied `seed` argument is also used to generate seeds for the sampling algorithm.

`control` A named list of parameters to control the sampler's behavior. See the documentation for `rstan::stan` for more details.

`sigma_alpha` A positive numeric value specifying the standard deviation of the prior on the shift parameters in the FBAM model, or the standard deviation of the parameters' deviation from the group-means in FBAM_MULTI models. (This argument will be ignored by HBAM models.) Defaults to B / 4, where B measures the length of the survey scale as the number of possible placements on one side of the center.

sigma_beta	A positive numeric value specifying the standard deviation of the prior on the logged stretch parameters in the FBAM model, or the standard deviation of the logged parameters' deviation from the group-means in FBAM_MULTI models. (This argument will be ignored by HBAM models.) Defaults to .35.
sigma_mu_alpha	A positive numeric value specifying the standard deviation of the prior on the group-means of the shift parameters in MULTI-type models. Defaults to B / 5.
sigma_mu_beta	A positive numeric value specifying the standard deviation of the prior on the group-means of the logged stretch parameters in MULTI-type models. Defaults to .3.
...	Arguments passed to `rstan::sampling()`.

Value

A list of classes `kfold` and `loo`, which contains the following named elements:

- `"estimates"`: A `1x2` matrix containing the ELPD estimate and its standard error. The columns have names `"Estimate"` and `"SE"`.
- `"pointwise"`: A `Nx1` matrix with column name `"elpd_kfold"` containing the pointwise contributions for each data point.

Examples

```
# Loading and re-coding ANES 1980 data:
data(LC1980)
LC1980[LC1980 == 0 | LC1980 == 8 | LC1980 == 9] <- NA

# Making a small subset of the data for illustration:
self <- LC1980[1:50, 1]
stimuli <- LC1980[1:50, -1]

# Preparing to run chains in parallel using 2 cores via the future package:
  # Note: You would normally want to use all physical cores for this.
future::plan(future::multisession, workers = 2)

# Performing 4-fold cross-validation for the HBAM_MINI model:
  # Note: You would typically want to run the chains for more iterations.
cv_hbam_mini <- hbam_cv(self, stimuli, model = "HBAM_MINI", K = 4,
                        chains = 1, warmup = 500, iter = 1000)

# Performing 4-fold cross-validation for the FBAM model:
cv_fbam <- hbam_cv(self, stimuli, model = "FBAM", K = 4,
                        chains = 1, warmup = 500, iter = 1000)

# Comparing the results using the loo package:
loo::loo_compare(list(HBAM_MINI = cv_hbam_mini,
                 FBAM = cv_fbam))

# Stop the cluster of parallel sessions:
future::plan(future::sequential)
```

LC1980 *1980 Liberal-Conservative Scales*

Description

Liberal-Conservative 7-point scales from the 1980 National Election Study. Includes (in order) self-placement, and rankings of Carter, Reagan, Kennedy, Anderson, Republican party, Democratic Party. Stored as a matrix of integers. The numbers 0, 8, and 9 are considered to be missing values.

Usage

```
data(LC1980)
```

Format

An object of class `matrix` (inherits from `array`) with 888 rows and 7 columns.

Source

American National Election Studies.

This dataset was originally part of the `basicspace` package under the same name ("LC1980").

Examples

```
data(LC1980)
LC1980[LC1980 == 0 | LC1980 == 8 | LC1980 == 9] <- NA
head(LC1980)
```

LC2012 *2012 Liberal-Conservative Scales*

Description

Liberal-Conservative 7-point scales from the 2012 National Election Study. Includes (in order) original case id, self-placement, and rankings of Obama, Romney, Democratic Party, Republican party. Missing values are coded as NA.

Usage

```
data(LC2012)
```

Format

An object of class `data.frame` with 5914 rows and 6 columns.

Source

American National Election Studies.

Examples

```
data(LC2012)
head(LC2012)
```

`plot_by_group` *Plot posterior densities of parameter averages by group*

Description

Plot posterior densities of group summaries of individual parameters. The respondents can be grouped by any categorical variable and the function works whether the fitted model is of "MULTI"-type or not.

Usage

```
plot_by_group(
  object,
  par = "abs_beta",
  group_id = NULL,
  ascending_means = TRUE,
  fill = "#2166AC",
  color = "#053061",
  alpha = 0.5,
  ncol = max(1, round(length(unique(group_id))/10))
)
```

Arguments

`object`	An instance of class `stanfit` produced by `hbam()`.
`par`	Character: Name of the parameter to be plotted. One of the following: `"alpha"`, `"beta"`, `"abs_beta"`, `"lambda"`, or `"chi"`. Defaults to `"abs_beta"`, which means the absolute values of the draws for beta will be used. Further individual-level parameters like `"eta"` can be specified if these have been passed to `hbam()` via the argument `extra_pars` when fitting the model. (Note that homoskedastic models have no `"eta"` parameters and "NF"-type models have no `"lambda"` or `"kappa"` parameters.)
`group_id`	An optional vector that will be used to split the respondents into groups. The vector must either be as long as the number of rows in the original dataset, or as long as the number of respondents included in the analysis. If a `group_id` was previously supplied to `prep_data()` or `hbam()` and if no `group_id` is supplied here, the default is to use the existing `group_id`. If a `group_id` is supplied here, it will be used instead of any previously supplied vector. The `group_id` supplied here does not have to coincide with the `group_id` used to fit a "MULTI"-type model: Any vector that can be used to group the respondents is allowed.

`ascending_means`
: Logical: Should the groups be placed in ascending order based on their posterior means (`TRUE`) or should they be ordered based on their names (`FALSE`)? Defaults to `TRUE`.

`fill`
: Fill color. Passed on to `ggplot2::geom_density()`.

`color`
: Color of outer lines. Passed on to `ggplot2::geom_density()`.

`alpha`
: Number in [0,1]: Inverse level of transparency.

`ncol`
: Number of columns. The default uses a formula to have approximately ten subplots per column.

Value

A `ggplot` object.

`plot_over_self` *Plot individual parameter estimates over self-placements*

Description

Create a boxplot of individual parameter point estimates from an HBAM model over self-placements

Usage

```
plot_over_self(
  object,
  par = "chi",
  estimate = "median",
  names = NULL,
  parlabel = NULL,
  fill = "#2166AC",
  color = "#053061",
  width = 0.7,
  alpha = 0.5,
  outlier.size = 0.3,
  median_color = "black",
  median_lwd = 0.7
)
```

Arguments

`object`
: An object of class `stanfit` produced by `hbam()`, a list produced by `fbam()`, or a `list` of such objects, which will produce a faceted plot.

`par`
: Character: Name of the parameter to be plotted. One of the following: `"alpha"`, `"beta"`, `"abs_beta"`, `"lambda"`, or `"chi"`. Defaults to `"chi"`. Further individual-level parameters like `"eta"` can be specified if these have been passed to `hbam()` via the argument `extra_pars` when fitting the model. (Note that homoskedastic models have no `"eta"` parameters and "NF"-type models have no `"lambda"` or `"kappa"` parameters.)

estimate	Character: Specifying which type of posterior point estimate to use. One of "median" and "mean". Defaults to "median". This only applies for stanfit objects.
names	An optional character vector of model names of same length as the supplied list of models.
parlabel	An optional character containing an alternative label for the parameter (will be parsed if passed as an expression).
fill	Fill color of boxes. Passed on to ggplot2::geom_boxplot().
color	Color of outer lines. Passed on to ggplot2::geom_boxplot().
width	Width of boxes. Passed on to ggplot2::geom_boxplot().
alpha	Number in [0,1]: Inverse level of transparency for fill color.
outlier.size	Size of dots representing outliers. Passed on to ggplot2::geom_boxplot().
median_color	Color of solid line representing the median.
median_lwd	Thickness of solid line representing the median.

Value

A ggplot object.

---

plot_respondents *Plot estimated respondent positions*

Description

Plot the distribution of estimated respondent positions from an HBAM model.

Usage

```
plot_respondents(
  object,
  inc_stimuli = TRUE,
  n_draws = 15,
  color = "#053061",
  fill = "#2166AC",
  alpha_color = 0.6,
  alpha_fill = 0.7/n_draws,
  seed = 1
)
```

Arguments

`object`	An instance of class `stanfit` produced by `hbam()`.
`inc_stimuli`	Logical: Should estimated stimulus positions also be shown?
`n_draws`	Integer specifying the number of posterior draws to use when illustrating the uncertainty of the population distribution. Defaults to 15.
`color`	Color of lines illustrating uncertainty.
`fill`	Fill color for density plots.
`alpha_color`	Number in [0,1]: Inverse level of transparency for line color.
`alpha_fill`	Number in [0,1]: Inverse level of transparency for fill color.
`seed`	A positive integer specifying an optional seed for reproducibility. The seed is used to select respondent position draws for illustrating uncertainty.

Value

A `ggplot` object.

`plot_stimuli` *Plot estimated stimulus positions*

Description

Plot marginal posterior distributions of stimulus positions from an HBAM model

Usage

```
plot_stimuli(object, rev_color = FALSE, alpha = 0.55)
```

Arguments

`object`	An instance of class `stanfit` produced by `hbam()`.
`rev_color`	Logical: Display low positions as red and high positions as blue.
`alpha`	Number in [0,1]: Inverse level of transparency for fill color.

Value

A `ggplot` object.

`prep_data` *Prepare data to fit an HBAM or FBAM model*

Description

This function prepares data to fit a hierarchical Bayesian Aldrich-McKelvey (HBAM) model. It can be run ahead of fitting the models, or it can be run implicitly as part of a single function call to fit the models using `hbam()` or `fbam()`. It applies a set of inclusion criteria, performs any necessary data transformation, and returns a list of data suited for sampling in `rstan`. The data provided to `prep_data()` can be centered, but they do not have to be: The function will detect uncentered data and attempt to center these automatically, assuming that the highest and lowest observed values in the data mark the extremes of the scale.

Usage

```
prep_data(
  self = NULL,
  stimuli,
  prefs = NULL,
  allow_miss = 2,
  req_valid = NA,
  req_unique = 2,
  B = NULL,
  group_id = NULL,
  quiet = FALSE
)
```

Arguments

`self`	An optional numerical vector of N ideological self-placements. Any missing data must be coded as NA. If this argument is not supplied, respondent positions will not be estimated.
`stimuli`	An N × J matrix of numerical stimulus placements, where J is the number of stimuli. Any missing data must be coded as NA.
`prefs`	An N × J matrix of numerical stimulus ratings or preference scores. These data are only required by the `"HBAM_R_MINI"` model and will be ignored when fitting other models.
`allow_miss`	Integer specifying how many missing stimulus positions should be accepted for an individual still to be included in the analysis. Defaults to 2.
`req_valid`	Integer specifying how many valid observations to require for a respondent to be included in the analysis. The default is `req_valid = J - allow_miss`, but if specified, `req_valid` takes precedence.
`req_unique`	Integer specifying how many unique positions on the ideological scale each respondent is required to have used when placing the stimuli in order to be included in the analysis. The default is `req_unique = 2`.

B	Scalar specifying the upper bound of the survey scale after centering. If not supplied, this information will be inferred from the data.
group_id	Vector of length N identifying which group each respondent belongs to. The format can be factor, character, integer, or numeric. Respondents with NAs on group_id will be dropped when group_id is supplied. These data are only required by models with "MULTI" in their name and will be ignored when fitting other models.
quiet	Logical: Should information about the data be printed to the console? Defaults to FALSE.

Value

A list of data to be used by hbam() or fbam(). The returned list includes the logical vector keep, which identifies the rows in the original data that have been kept for further analysis. The stimuli data are stored in a vector as a long-form sparse matrix. If the stimuli data include column-names, these will be preserved for later use.

Examples

```
# Loading and re-coding ANES 1980 data:
data(LC1980)
LC1980[LC1980 == 0 | LC1980 == 8 | LC1980 == 9] <- NA
self <- LC1980[, 1]
stimuli <- LC1980[, -1]

# Prepare data for model fitting, using defaults:
dat <- prep_data(self, stimuli)

# Prepare data for model fitting, using alternative settings:
dat2 <- prep_data(self, stimuli, allow_miss = 0, req_unique = 3)

# Obtain the data that are included in the analysis:
self2 <- self[dat2$keep]
stimuli2 <- stimuli[dat2$keep, ]
```

---

prep_data_cv *Prepare data for a K-fold cross-validation of an HBAM model*

Description

This function turns data prepared for hbam() into a list of K versions, where each version includes a different vector identifying holdout data.

Usage

```
prep_data_cv(data, K = 10, seed = 1)
```

Arguments

data	A list of data produced by prep_data().
K	An integer above 2, specifying the number of folds to use in the analysis. Defaults to 10.
seed	An integer passed on to set.seed before creating the folds to increase reproducibility. Defaults to 1.

Value

A list of K data objects where each version includes a different vector identifying holdout data.

Examples

```
# Loading and re-coding ANES 1980 data:
data(LC1980)
LC1980[LC1980 == 0 | LC1980 == 8 | LC1980 == 9] <- NA
self <- LC1980[, 1]
stimuli <- LC1980[, -1]
dat <- prep_data(self, stimuli)

# Prepare data for cross-validation:
dat_cv <- prep_data_cv(dat, K = 10)
```

Index

∗ **datasets**
    LC1980, 15
    LC2012, 15

fbam, 3

get_est, 5
get_plot_data, 6

hbam, 7
hbam_cv, 12
hbamr *(*hbamr-package*)*, 2
hbamr-package, 2

LC1980, 15
LC2012, 15

plot_by_group, 16
plot_over_self, 17
plot_respondents, 18
plot_stimuli, 19
prep_data, 20
prep_data_cv, 21