Package 'idealstan'

May 12, 2026

Type Package

Title Robust Measurement with 'Stan'

Version 1.0

Date 2026-04-21

BugReports https://github.com/saudiwin/idealstan/issues

Description Offers item-response theory (IRT) ideal-
point measurement modeling for diverse distributions, missing data, and over-
time variation. Full and approximate Bayesian sampling with 'Stan' (<https://mc-stan.org/>).

License MIT + file LICENSE

Depends methods, R (>= 4.0.0)

Imports dplyr, svDialogs, tidyr, stringr, bayesplot, posterior,
gghighlight, ggrepel, ggplot2, tidybayes, lazyeval, rlang,
ggthemes, forcats, ordbetareg, scales, shiny, tibble

Suggests cmdstanr, quarto, knitr, rmarkdown, loo, lubridate, rstan,
shinystan, pscl, tinytable, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

LazyData true

NeedsCompilation no

SystemRequirements GNU make

Additional_repositories https://stan-dev.r-universe.dev

VignetteBuilder quarto

RoxygenNote 7.3.2

Author Robert Kubinec [aut, cre]

Maintainer Robert Kubinec <bobkubinec@gmail.com>

Repository CRAN

Date/Publication 2026-05-12 19:20:18 UTC

Contents

`delaware` *Rollcall vote data for Delaware State Legislature*

Description

This data frame contains the rollcall voting data for the Delaware state legislature from 1995 to present. The data is in long format so that each row is one vote cast by a legislator. It includes a column, `group_id`, that lists a party for each legislator (D=Democrat, R=Republican,X=Independent).

Usage

```
delaware
```

Format

A long data frame with one row for every vote cast by a legislator.

Details

The original data come from Boris Shor and Nolan McCarty (2002), "The Ideological Mapping of American Legislatures", American Political Science Review.

Source

https://www.cambridge.org/core/journals/american-political-science-review/article/ideological-mapping-of-american-legislatures/8E1192C22AA0B9F9B56167998A41CAB0

---

derive_chain *Helper Function for* loo *calculation*

Description

This function accepts a log-likelihood matrix produced by `id_post_pred` and extracts the IDs of the MCMC chains. It is necessary to use this function as the second argument to the `loo` function along with an exponentiated log-likelihood matrix. See the package vignette How to Evaluate Models for more details.

Usage

```
derive_chain(ll_matrix = NULL)
```

Arguments

`ll_matrix`	A log-likelihood matrix as produced by the `id_post_pred()` function

Value

An integer vector of MCMC chain IDs extracted from the log-likelihood matrix, for use as the `chain_id` argument to `relative_eff`.

Examples

```
sim <- id_sim_gen()
est <- id_estimate(sim, model_type=1, fixtype='vb_full',
                   use_method="pathfinder", nchains=2, ncores=2)
ll <- id_post_pred(est, type='log_lik')
chain_ids <- derive_chain(ll)
# use with loo
loo::loo(exp(ll), r_eff=loo::relative_eff(exp(ll), chain_id=chain_ids))
```

idealdata-class *Data and Identification for* idealdata

Description

idealdata objects contain the relevant legislator/bill (person/item) matrix of data along with slots containing information about the kind of identification used in the estimation.

See Also

id_make to create an idealdata object suitable for estimation with id_estimate.

idealstan-class *Results of* id_estimate *function*

Description

The idealstan objects store the results of estimations carried out by the id_estimate function. These objects include the full results of Bayesian sampling performed by the stan function in the **rstan** package.

id_estimate *Estimate an* idealstan *model*

Description

This function will take a pre-processed idealdata vote/score dataframe and run one of the available IRT/latent space ideal point models on the data using Stan's MCMC engine.

Usage

```
id_estimate(
  idealdata = NULL,
  model_type = 2,
  inflate_zero = FALSE,
  vary_ideal_pts = "none",
  seed = NULL,
  keep_param = NULL,
  grainsize = 1,
  mpi_export = NULL,
  use_subset = FALSE,
  sample_it = FALSE,
  subset_group = NULL,
```

```
  subset_person = NULL,
  sample_size = 20,
  nchains = 4,
  niters = 1000,
  use_method = "mcmc",
  ignore_db = NULL,
  restrict_ind_high = NULL,
  fix_high = 2,
  fix_low = (-2),
  restrict_ind_low = NULL,
  num_restrict_high = 1,
  num_restrict_low = 1,
  fixtype = "prefix",
  const_type = "persons",
  id_refresh = 0,
  prior_only = FALSE,
  warmup = 1000,
  ncores = 4,
  use_groups = FALSE,
  discrim_reg_upb = 1,
  discrim_reg_lb = -1,
  discrim_miss_upb = 1,
  discrim_miss_lb = -1,
  discrim_reg_scale = 2,
  discrim_reg_shape = 2,
  discrim_miss_scale = 2,
  discrim_miss_shape = 2,
  person_sd = 3,
  time_fix_sd = 0.1,
  time_var = 10,
  spline_knots = NULL,
  spline_degree = 2,
  ar1_up = 1,
  ar1_down = -1,
  boundary_prior = NULL,
  time_center_cutoff = 50,
  restrict_var = FALSE,
  ar_prior = c(2, 2),
  diff_reg_sd = 3,
  diff_miss_sd = 3,
  restrict_sd_high = NULL,
  restrict_sd_low = NULL,
  restrict_N_high = 1000,
  restrict_N_low = 1000,
  ordbeta_phi_mean = 1,
  ordbeta_cut_alpha = c(1, 1, 1),
  ordbeta_cut_phi = 0,
  gp_nugget = 0.1,
```

```
  gp_rho = 0.5,
  gp_alpha = 0.5,
  cmdstan_path_user = NULL,
  map_over_id = "persons",
  save_files = NULL,
  compile_optim = FALSE,
  debug = FALSE,
  init_pathfinder = TRUE,
  debug_mode = 0,
  ...
)
```

Arguments

idealdata	An object produced by the id_make() containing a score/vote matrix for use for estimation & plotting
model_type	An integer reflecting the kind of model to be estimated. See below.
inflate_zero	If the outcome is distributed as Poisson (count/unbounded integer), setting this to TRUE will fit a traditional zero-inflated model.
vary_ideal_pts	Default 'none'. If 'random_walk', 'AR1', 'GP', or 'splines', a time-varying ideal point model will be fit with either a random-walk process, an AR1 process, a Gaussian process or a spline. Note that the spline is the easiest time-varying model to fit so long as the number of knots (option spline_knots) is significantly less than the number of time points in the data. See documentation for more info.
seed	The integer seed passed on to the estimation engine (including the algorithm used to obtain starting values)
keep_param	A list with logical values for different categories of paremeters which should/should not be kept following estimation. Can be any/all of person_int for the person-level intercepts (static ideal points), person_vary for person-varying ideal points, item for observed item parameters (discriminations/intercepts), item_miss for missing item parameters (discriminations/intercepts), and extra for other parameters (hierarchical covariates, ordinal intercepts, etc.). Takes the form list(person_int=TRUE,perso If any are missing in the list, it is assumed that those parameters will be excluded. If NULL (default), will save all parameters in output.
grainsize	The grainsize parameter for the reduce_sum function used for within-chain parallelization. The default is 1, which means 1 chunk (item or person) per core. Set to -1. to use
mpi_export	If within_chains="mpi", this parameter should refer to the directory where the necessary data and Stan code will be exported to. If missing, an interactive dialogue will prompt the user for a directory.
use_subset	Whether a subset of the legislators/persons should be used instead of the full response matrix
sample_it	Whether or not to use a random subsample of the response matrix. Useful for testing.

subset_group
: If person/legislative data was included in the id_make() function, then you can subset by any value in the $group column of that data if use_subset is TRUE.

subset_person
: A list of character values of names of persons/legislators to use to subset if use_subset is TRUE and person/legislative data was included in the id_make() function with the required $person.names column

sample_size
: If sample_it is TRUE, this value reflects how many legislators/persons will be sampled from the response matrix

nchains
: The number of chains to use in Stan's sampler. Minimum is one. See rstan::stan() for more info. If use_method="pathfinder", this parameter will determine the number of Pathfinder paths to estimate.

niters
: The number of iterations to run Stan's sampler. Shouldn't be set much lower than 500. See rstan::stan() for more info.

use_method
: The type of estimation to use to estimate the posterior distribution of the parameters. The default is "mcmc", which uses Stan's Hamiltonian Markov Chain Monte Carlo inference. The other options, "pathfinder" and "laplace", are variational algorithms that derive an easier/faster to compute also used by default for finding initial starting values for full HMC sampling.

ignore_db
: If there are multiple time periods (particularly when there are very many time periods), you can pass in a data frame (or tibble) with one row per person per time period and an indicator column ignore that is equal to 1 for periods that should be considered in sample and 0 for periods for periods that should be considered out of sample. This is useful for excluding time periods from estimation for persons when they could not be present, i.e. such as before entrance into an organization or following death. If ignore equals 0, the person's ideal point is estimated as a standard Normal draw rather than an auto-correlated parameter, reducing computational burden substantially. Note that there can only be one pre-sample period of 0s, one in-sample period of 1s, and one post-sample period of 0s. Multiple in-sample periods cannot be interspersed with out of sample periods. The columns must be labeled as person_id, time_id and ignore and must match the formatting of the columns fed to the id_make function.

restrict_ind_high
: If fixtype is not "vb_full", a vector of character values or integer indices of a legislator/person or bill/item to pin to a high value (default +1).

fix_high
: A vector of length restrict_ind_high with values that the high fixed person ideal point(s) should be fixed to. Default is +2. Does not apply when const_type="items"; in that case, use restrict_sd/restrict_N parameters (see below).

fix_low
: A vector of length restrict_ind_low with values that the high fixed person ideal point(s) should be fixed to. Default is -2. Does not apply when const_type="items"; in that case, use restrict_sd/restrict_N parameters (see below).

restrict_ind_low
: If fixtype is not "vb_full", a vector of character values or integer indices of a legislator/person or bill/item to pin to a low value (default -1).

num_restrict_high
: If using variational inference for identification (fixtype="vb_full"), how many parameters to constraint to positive values? Default is 1.

num_restrict_low
: If using variational inference for identification (ixtype="vb_full"), how many parameters to constraint to positive negative values? Default is 1.

fixtype
: Sets the particular kind of identification used on the model, could be either 'vb_full' (identification provided exclusively by running a variational identification model with no prior info), or 'prefix' (two indices of ideal points or items to fix are provided to options restrict_ind_high and restrict_ind_low). See details for more information.

const_type
: Whether "persons" are the parameters to be fixed for identification (the default) or "items". Each of these pinned parameters should be specified to fix_high and fix_low if fixtype equals "prefix", otherwise the model will select the parameters to pin to fixed values.

id_refresh
: The number of times to report iterations from the variational run used to identify models. Default is 0 (nothing output to console).

prior_only
: Whether to only sample from priors as opposed to the full model with likelihood (the default). Useful for doing posterior predictive checks.

warmup
: The number of iterations to use to calibrate Stan's sampler on a given model. Shouldn't be less than 100. See rstan::stan() for more info.

ncores
: The number of cores in your computer to use for parallel processing in the Stan engine. See rstan::stan() for more info. If within_chain is set to "threads", this parameter will determine the number of threads (independent processes) used for within-chain parallelization.

use_groups
: If TRUE, group parameters from the person/legis data given in id_make() will be estimated instead of individual parameters.

discrim_reg_upb
: Upper bound of the rescaled Beta distribution for observed discrimination parameters (default is +1)

discrim_reg_lb
: Lower bound of the rescaled Beta distribution for observed discrimination parameters (default is -1). Set to 0 for conventional IRT.

discrim_miss_upb
: Upper bound of the rescaled Beta distribution for missing discrimination parameters (default is +1)

discrim_miss_lb
: Lower bound of the rescaled Beta distribution for missing discrimination parameters (default is -1). Set to 0 for conventional IRT.

discrim_reg_scale
: Set the scale parameter for the rescaled Beta distribution of the discrimination parameters.

discrim_reg_shape
: Set the shape parameter for the rescaled Beta distribution of the discrimination parameters.

discrim_miss_scale
: Set the scale parameter for the rescaled Beta distribution of the missingness discrimination parameters.

`discrim_miss_shape`
: Set the shape parameter for the rescaled Beta distribution of the missingness discrimination parameters.

`person_sd`
: The standard deviation of the Normal distribution prior for persons (all non-constrained person ideal point parameters). Default is weakly informative (3) on the logit scale.

`time_fix_sd`
: The variance of the over-time component of the first person/legislator is fixed to this value as a reference. Default is 0.1.

`time_var`
: The mean of the exponential distribution for over-time variances for ideal point parameters. Default (10) is weakly informative on the logit scale.

`spline_knots`
: Number of knots (essentially, number of points at which to calculate time-varying ideal points given T time points). Default is NULL, which means that the spline is equivalent to polynomial time trend of degree `spline_degree`. Note that the spline number (if not null) must be equal or less than the number of time points–and there is no reason to have it equal to the number of time points as that will likely over-fit the data.

`spline_degree`
: The degree of the spline polynomial. The default is 2 which is a quadratic polynomial. A value of 1 will result in independent knots (essentially pooled across time points T). A higher value will result in wigglier time series. There is no "correct" value but lower values are likely more stable and easier to identify.

`ar1_up`
: The upper bound of the AR(1) parameter, default is +1.

`ar1_down`
: The lower bound of the AR(1) parameter, default is 0. Set to -1 to allow for inverse responses to time shocks.

`boundary_prior`
: If your time series has very low variance (change over time), you may want to use this option to put a boundary-avoiding inverse gamma prior on the time series variance parameters if your model has a lot of divergent transitions. To do so, pass a list with a element called `beta` that signifies the rate parameter of the inverse-gamma distribution. For example, try `boundary_prior=list(beta=1)`. Increasing the value of `beta` will increase the "push" away from zero. Setting it too high will result in time series that exhibit a lot of "wiggle" without much need.

`time_center_cutoff`
: The number of time points above which the model will employ a centered time series approach for AR(1) and random walk models. Below this number the model will employ a non-centered approach. The default is 50 time points, which is relatively arbitrary and higher values may be better if sampling quality is poor above the threshold.

`restrict_var`
: Whether to fix the variance parameter for the first person trajectory. Default is FALSE (usually not necessary).

`ar_prior`
: If an AR(1) model is used, this 2-length vector sets the shape (alpha) and scale (beta) of the Beta prior on the AR-1 adjustment parameters (often denoted phi). The first element of the vector is alpha and the second is beta. See Stan documentation of the Beta distribution for more info.

`diff_reg_sd`
: Set the prior standard deviation for the bill (item) intercepts for the non-inflated model.

diff_miss_sd Set the prior standard deviation for the bill (item) intercepts for the inflated model.

restrict_sd_high
Set the level of tightness for high fixed parameters (top/positive end of scale). If NULL, the default, will set to .1 if const_type="persons" and 10 if const_type="items". For const_type="persons", value is the SD of normal distribution centered around fix_high. For const_type="items", parameter is equal to the prior shape for high pinned parameters (divide by restrict_N_high + restrict_sd_high) to get expected value.

restrict_sd_low
Set the level of tightness for low fixed parameters (low/negative end of scale). If NULL, the default, will set to .1 if const_type="persons" and 10 if const_type="items". For const_type="persons", value is the SD of normal distribution centered around fix_low. For const_type="items", parameter is equal to the prior shape for high pinned parameters (divide by restrict_N_low + restrict_sd_low) to get expected value.

restrict_N_high
Set the prior scale for high/positive pinned parameters. Default is 1000 (equivalent to 1,000 observations of the pinned value). Higher values make the pin stronger (for example if there is a lot of data).

restrict_N_low Set the prior shape for low/negative pinned parameters. Default is 1000 (equivalent to 1,000 observations of the pinned value). Higher values make the pin stronger (for example if there is a lot of data).

ordbeta_phi_mean
The mean of the prior for phi, the dispersion parameter in the ordered beta distribution. Value of this parameter (default is 1) is given as the mean of the exponential distribution for prior values of phi.

ordbeta_cut_alpha
A length 2 vector of positive continuous values for alpha in the induced dirichlet distribution. This distribution is used for the cutpoints of the ordered beta distribution. Default is c(1,1), which is uninformative.

ordbeta_cut_phi
A value for the phi paremeter of the induced dirichlet distribution used for ordered beta cutpoint priors. Default is 0, which is weakly informative.

gp_nugget The nugget of the squared-exponential kernel (equals additional variance of the GP-distributed ideal points)

gp_rho The mean of the exponential prior of the rho parameters of the GP squared-exponential kernel

gp_alpha The mean of the exponential prior of the alpha parameters of the GP squared-exponential kernel

cmdstan_path_user
Default is NULL, and so will default to whatever is set in cmdstanr package. Specify a file path here to use a different cmdtstan installation.

map_over_id This parameter identifies which ID variable to use to construct the shards for within-chain parallelization. It defaults to "persons" but can also take a value of "items". It is recommended to select whichever variable has more distinct values to improve parallelization.

save_files	The location to save CSV files with MCMC draws from cmdstanr. The default is NULL, which will use a folder in the package directory.
compile_optim	Whether to use Stan compile optimization flags (off by default)
debug	For debugging purposes, turns off threading to enable more informative error messages from Stan. Also recompiles model objects.
init_pathfinder	Whether to generate initial values from the Pathfinder algorithm (see Stan documentation). If FALSE, will generate random start values..
debug_mode	Whether to debug code by printing values of log-probability statements to the console. A level of 1 will print log-probability before and after likelihood functions are calculated. A level of 2 will also print out the log probability contributions of priors. Default is 0.
...	Additional parameters passed on to Stan's sampling engine. See cmdstanr::sample for more information.

Details

To run an IRT ideal point model, you must first pre-process your data using the `id_make()` function. Be sure to specify the correct options for the kind of model you are going to run: if you want to run an unbounded outcome (i.e. Poisson or continuous), the data needs to be processed differently. Also any hierarchical covariates at the person or item level need to be specified in `id_make()`. If they are specified in `id_make()`, than all subsequent models fit by this function will have these covariates.

Note that for static ideal point models, the covariates are only defined for those persons who are not being used as constraints.

As of this version of `idealstan`, the following model types are available. Simply pass the number of the model in the list to the `model_type` option to fit the model.

1. IRT 2-PL (binary response) ideal point model, no missing-data inflation
2. IRT 2-PL ideal point model (binary response) with missing- inflation
3. Ordinal IRT (rating scale) ideal point model no missing-data inflation
4. Ordinal IRT (rating scale) ideal point model with missing-data inflation
5. Ordinal IRT (graded response) ideal point model no missing-data inflation
6. Ordinal IRT (graded response) ideal point model with missing-data inflation
7. Poisson IRT (Wordfish) ideal point model with no missing data inflation
8. Poisson IRT (Wordfish) ideal point model with missing-data inflation
9. unbounded (Gaussian) IRT ideal point model with no missing data
10. unbounded (Gaussian) IRT ideal point model with missing-data inflation
11. Positive-unbounded (Log-normal) IRT ideal point model with no missing data
12. Positive-unbounded (Log-normal) IRT ideal point model with missing-data inflation
13. Latent Space (binary response) ideal point model with no missing data
14. Latent Space (binary response) ideal point model with missing-data inflation
15. Ordered Beta (proportion/percentage) with no missing data
16. Ordered Beta (proportion/percentage) with missing-data inflation

Value

A fitted idealstan() object that contains posterior samples of all parameters either via full Bayesian inference or a variational approximation if use_method is set to "pathfinder" or "laplace". This object can then be passed to the plotting functions for further analysis.

Time-Varying Inference

In addition, each of these models can have time-varying ideal point (person) parameters if a column of dates is fed to the id_make() function. If the option vary_ideal_pts is set to 'random_walk', id_estimate will estimate a random-walk ideal point model where ideal points move in a random direction. If vary_ideal_pts is set to 'AR1', a stationary ideal point model is estimated where ideal points fluctuate around long-term mean. If vary_ideal_pts is set to 'GP', then a semi-parametric Gaussian process time-series prior will be put around the ideal points. If vary_ideal_pts is set to 'splines', then the ideal point trajectories will be a basis spline defined by the parameters spline_knots and spline_degree. Please see the package vignette and associated paper for more detail about these time-varying models.

Missing Data

The inflation model used to account for missing data assumes that missingness is a function of the persons' (legislators') ideal points. In other words,the model will take into account if people with high or low ideal points tend to have more/less missing data on a specific item/bill. Missing data should be coded as NA when it is passed to the id_make function. If there isn't any relationship between missing data and ideal points, then the model assumes that the missingness is ignorable conditional on each item, but it will still adjust the results to reflect these ignorable (random) missing values. The inflation is designed to be general enough to handle a wide array of potential situations where strategic social choices make missing data important to take into account.

To leave missing data out of the model, simply choose a version of the model in the list above that is non-inflated.

Models can be either fit on the person/legislator IDs or on group-level IDs (as specified to the id_make function). If group-level parameters should be fit, set use_groups to TRUE.

Covariates

Covariates are included in the model if they were specified as options to the id_make() function. The covariate plots can be accessed with id_plot_cov() on a fitted idealstan model object.

Identification

Identifying IRT models is challenging, and ideal point models are still more challenging because the discrimination parameters are not constrained. As a result, more care must be taken to obtain estimates that are the same regardless of starting values. The parameter fixtype enables you to change the type of identification used. The default, 'vb_full', does not require any further information from you in order for the model to be fit. In this version of identification, an unidentified model is run using variational Bayesian inference (see rstan::vb()). The function will then select two persons/legislators or items/bills that end up on either end of the ideal point spectrum, and pin their ideal points to those specific values. To control whether persons/legislator or items/bills are constrained, the const_type can be set to either "persons" or "items" respectively. In many

situations, it is prudent to select those persons or items ahead of time to pin to specific values. This allows the analyst to be more specific about what type of latent dimension is to be estimated. To do so, the `fixtype` option should be set to `"prefix"`. The values of the persons/items to be pinned can be passed as character values to `restrict_ind_high` and `restrict_ind_low` to pin the high/low ends of the latent scale respectively. Note that these should be the actual data values passed to the `id_make` function. If you don't pass any values, you will see a prompt asking you to select certain values of persons/items.

The pinned values for persons/items are set by default to +1/-1, though this can be changed using the `fix_high` and `fix_low` options. This pinned range is sufficient to identify all of the models implemented in idealstan, though fiddling with some parameters may be necessary in difficult cases. For time-series models, one of the person ideal point over-time variances is also fixed to .1, a value that can be changed using the option `time_fix_sd`.

References

1. Clinton, J., Jackman, S., & Rivers, D. (2004). The Statistical Analysis of Roll Call Data. *The American Political Science Review*, 98(2), 355-370. doi:10.1017/S0003055404001194
2. Bafumi, J., Gelman, A., Park, D., & Kaplan, N. (2005). Practical Issues in Implementing and Understanding Bayesian Ideal Point Estimation. *Political Analysis*, 13(2), 171-187. doi:10.1093/pan/mpi010
3. Kubinec, R. "Generalized Ideal Point Models for Time-Varying and Missing-Data Inference". Working Paper.
4. Betancourt, Michael. "Robust Gaussian Processes in Stan". (October 2017). Case Study.

See Also

`id_make()` for pre-processing data, `id_plot_persons()` for plotting results, `summary()` for obtaining posterior quantiles, `id_post_pred()` for producing predictive replications.

Examples

```
# First we can simulate data for an IRT 2-PL model that is inflated for missing data
library(ggplot2)
library(dplyr)

# This code will take at least a few minutes to run

bin_irt_2pl_abs_sim <- id_sim_gen(model_type='binary',inflate=TRUE)

# Now we can put that directly into the id_estimate function
# to get full Bayesian posterior estimates
# We will constrain discrimination parameters
# for identification purposes based on the true simulated values

bin_irt_2pl_abs_est <- id_estimate(bin_irt_2pl_abs_sim,
                       model_type=2,
                       use_method="pathfinder",
                       restrict_ind_high =
                       sort(bin_irt_2pl_abs_sim@simul_data$true_person,
```

```
                        decreasing=TRUE,
                        index=TRUE)$ix[1],
                        restrict_ind_low =
                        sort(bin_irt_2pl_abs_sim@simul_data$true_person,
                        decreasing=FALSE,
                        index=TRUE)$ix[1],
                        fixtype='prefix',
                        ncores=2,
                        nchains=2)

# We can now see how well the model recovered the true parameters

id_sim_coverage(bin_irt_2pl_abs_est) %>%
         bind_rows(.id='Parameter') %>%
         ggplot(aes(y=avg,x=Parameter)) +
           stat_summary(fun.args=list(mult=1.96)) +
           theme_minimal()


# In most cases, we will use pre-existing data
# and we will need to use the id_make function first
# We will use the full rollcall voting data
# from the 114th Senate as a rollcall object



data('senate114')

# Running this model will take at least a few minutes, even with
# variational inference (use_method="pathfinder") turned on

# first convert absences to NA values

senate114$cast_code[senate114$cast_code=="NA"] <- NA

# reset factor levels

senate114$cast_code <- factor(senate114$cast_code,
                              levels=c("No","Yes"))

to_idealstan <-   id_make(score_data = senate114,
outcome_disc = 'cast_code',
person_id = 'bioname',
item_id = 'rollnumber',
group_id= 'party_code',
time_id='date')

sen_est <- id_estimate(to_idealstan,
model_type = 2,
use_method = "pathfinder",
fixtype='prefix',
restrict_ind_high = "BARRASSO, John A.",
restrict_ind_low = "WARREN, Elizabeth")
```

```
# After running the model, we can plot
# the results of the person/legislator ideal points

id_plot_persons(sen_est)
```

`id_extract` *Generic Method for Extracting Posterior Samples*

Description

This is a generic function.

Usage

```
id_extract(object, ...)
```

Arguments

`object`	A fitted `idealstan` object
`...`	Other arguments passed on to underlying functions

Details

This generic will extract the full `stan` posterior samples from `idealstan` objects.

See the corresponding method definition for more information about what you can acccess with this generic.

Value

A `tibble` of posterior draws, with rows as draws and columns as parameters.

Examples

```
data('senate114')
senate114$cast_code <- ifelse(senate114$cast_code=="Absent", NA,
                              as.integer(senate114$cast_code) - 1L)
sen_data <- id_make(senate114, outcome_disc='cast_code',
                    person_id='bioname', item_id='rollnumber',
                    group_id='party_code')
sen_est <- id_estimate(sen_data, model_type=1, fixtype='vb_full', use_method="pathfinder", ncores=4)
id_extract(sen_est, extract_type='persons')
```

id_extract,idealstan-method *Extract* stan *joint posterior distribution from* idealstan *object*

Description

This convenience function allows you to extract the underlying rstan posterior estimates for the full parameters estimates of the idealstan model object. See extract for the underlying function and more options.

You can use this function to access a matrix or array of the full posterior estimates of each of the parameters in an idealstan object. There are available options to pick certain parameters of the model, such as the person (legislator) ideal points or item (bill) discrimination scores. Alternatively, you can leave the extract_type option blank and receive a list of all of the available parameters. Please note that the list of parameters do not have particularly informative names.

All parameters are returned in the order in which they were input into the id_make function.

Usage

```
## S4 method for signature 'idealstan'
id_extract(object, extract_type = "persons", ...)
```

Arguments

object	A fitted idealstan object (see id_estimate)
extract_type	Can be one of 'persons' for person/legislator ideal points, 'obs_discrim' for non-inflated item (bill) discrimination scores, 'obs_diff' for non-inflated item (bill) difficulty scores, 'miss_discrim' for inflated item (bill) discrimination scores, and 'miss_diff' for inflated item (bill) difficulty scores.
...	Any additional arguments passed on to the extract function.

Value

A tibble of posterior draws, with rows as draws and columns as parameters.

Examples

```
data('senate114')
senate114$cast_code <- ifelse(senate114$cast_code=="Absent", NA,
                              as.integer(senate114$cast_code) - 1L)
sen_data <- id_make(senate114, outcome_disc='cast_code',
                    person_id='bioname', item_id='rollnumber',
                    group_id='party_code')
sen_est <- id_estimate(sen_data, model_type=1, use_method="pathfinder", fixtype='vb_full', ncores=4)

# will launch interactive browser
if(interactive()) {
  launch_shinystan(sen_est)
```

```
}
```

`id_make` *Create data to run IRT model*

Description

To run an IRT model using `idealstan`, you must first process your data using the `id_make` function.

Usage

```
id_make(
  score_data = NULL,
  outcome_disc = "outcome_disc",
  outcome_cont = "outcome_cont",
  person_id = "person_id",
  item_id = "item_id",
  time_id = "time_id",
  group_id = "group_id",
  model_id = "model_id",
  ordered_id = "ordered_id",
  ignore_id = "ignore_id",
  simul_data = NULL,
  person_cov = NULL,
  item_cov = NULL,
  item_cov_miss = NULL,
  remove_cov_int = FALSE,
  unbounded = FALSE,
  exclude_level = NA,
  simulation = FALSE
)
```

Arguments

`score_data`	A data frame in long form, i.e., one row in the data for each measured score or vote in the data or a `rollcall` data object from package `pscl`.
`outcome_disc`	Column name of the outcome with discrete values in `score_data`, default is `"outcome_disc"`
`outcome_cont`	Column name of the outcome with discrete values in `score_data`, default is `"outcome_disc"`
`person_id`	Column name of the person/legislator ID index in `score_data`, default is `'person_id'`. Should be integer, character or factor.
`item_id`	Column name of the item/bill ID index in `score_data`, default is `'item_id'`. Should be integer, character or factor.

time_id	Column name of the time values in score_data: optional, default is 'time_id'. Should be a date or date-time class, but can be an integer (i.e., years in whole numbers).
group_id	Optional column name of a person/legislator group IDs (i.e., parties) in score_data. Optional, default is 'group_id'. Should be integer, character or factor.
model_id	Column name of the model/response types in the data. Default is "model_id". Only necessary if a model with multiple response types (i.e., binary + continuous outcomes). Must be a column with a series of integers matching the model types in id_estimate() showing which row of the data matches which outcome.
ordered_id	Column name of the variable showing the count of categories for ordinal/categorical items (must be at least 3 categories)
ignore_id	Optional column for identifying observations that should not be modeled (i.e., not just treated as missing, rather removed during estimation). Should be a binary vector (0 for remove and 1 for include). Useful for time-varying models where persons may not be present during particular periods and missing data is ignorable.
simul_data	Optionally, data that has been generated by the id_sim_gen() function.
person_cov	A one-sided formula that specifies the covariates in score_data that will be used to hierarchically model the person/legislator ideal points
item_cov	A one-sided formula that specifies the covariates in score_data that will be used to hierarchically model the item/bill discrimination parameters for the regular model
item_cov_miss	A one-sided formula that specifies the covariates in the dataset that will be used to hierarchically model the item/bill discrimination parameters for the missing data model.
remove_cov_int	Whether to remove constituent terms from hierarchical covariates that interact covariates with IDs like person_id or item_id. Set to TRUE if including these constituent terms would cause multi-collinearity with other terms in the model (such as running a group-level model with a group-level interaction or a person-level model with a person-level interaction).
unbounded	Whether or not the outcome/response is unbounded (i.e., continuous or Poisson). If it is, missing value is recoded as the maximum of the outcome + 1.
exclude_level	A vector of any values that should be treated as NA in the response matrix. Unlike missing values, these values will be dropped from the data before estimation rather than modeled explicitly.
simulation	If TRUE, simulated values are saved in the idealdata object for later plotting with the id_show_trues() function

Details

This function accepts a long data frame where one row equals one item-person (bill-legislator) observation with associated continuous or discrete outcomes/responses. You either need to include columns with specific names as required by the id_make function such as person_id for person IDs and item_id for item IDs or specify the names of the columns containing the IDs to the id_make function for each column name (see examples). The only required columns are the item/bill ID

and the person/legislator ID along with an outcome column, `outcome_disc` for discrete variables and `outcome_cont` for continuous variables. If both columns are included, then any value can be included for `outcome_disc` if there are values for `outcome_cont` and vice versa.

If items of multiple types are included, a column `model_id` must be included with the model type (see `id_estimate` function documentation for list of model IDs) for the response distribution, such as 1 for binary non-inflated, etc. If an ordinal outcome is included, an additional column `ordered_id` must be included that has the total count of categories for that ordinal variable (i.e., 3 for 3 categories).

For discrete data, it is recommended to include a numeric variable that starts at 0, such as values of 0 and 1 for binary data and 0,1,2 for ordinal/categorical data. For continuous (unbounded) data, it is recommended to standardize the outcome to improve model convergence and fit.

Missing data should be passed as `NA` values in either `outcome_disc` or `outcome_cont` and will be processed internally.

Value

A `idealdata` object that can then be used in the `id_estimate()` function to fit a model.

Time-Varying Models

To run a time-varying model, you need to include the name of a column with dates (or integers) that is passed to the `time_id` option.

Continuous Outcomes

If the outcome is continuous, you need to pass a dataframe with one column named "outcome_disc" or pass the name of the column with the continuous data to the `outcome_disc` argument.

Hierarchical Covariates

Covariates can be fit on the person-level ideal point parameters as well as item discrimination parameters for either the inflated (missing) or non-inflated (observed) models. These covariates must be columns that were included with the data fed to the `id_make()` function. The covariate relationships are specified as one-sided formulas, i.e. `~cov1 + cov2 + cov1*cov2`. To interact covariates with the person-level ideal points you can use `~cov1 + person_id + cov1*person_id` and for group-level ideal poins you can use `~cov1 + group_id + cov1*group_id` where `group_id` or `person_id` is the same name as the name of the column for these options that you passed to `id_make` (i.e., the names of the columns in the original data). If you are also going to model these intercepts–i.e. you are interacting the covariate with `person_id` and the model is estimating ideal points at the person level–then set `remove_cov_int` to TRUE to avoid multicollinearity with the ideal point intercepts.

Examples

```
library(dplyr)

# Basic usage: 114th Senate rollcall votes as a discrete binary outcome.
# cast_code is recoded to 0/1 (No/Yes) with absences as NA.
```

```
data('senate114')
senate114$cast_code <- ifelse(senate114$cast_code == "Absent", NA,
                              senate114$cast_code)
senate114$cast_code <- as.integer(senate114$cast_code) - 1L

senate_data <- id_make(score_data = senate114,
                       outcome_disc = 'cast_code',
                       person_id = 'bioname',
                       item_id = 'rollnumber',
                       group_id = 'party_code',
                       time_id = 'date')

# Mixed discrete and continuous outcomes.
# When items have different response distributions, add a model_id column
# with the integer model type for each row (see ?id_estimate for the full
# list).  Discrete values go in outcome_disc; continuous values go in
# outcome_cont.  Rows that belong to the other type should be NA.
#
# Here, roll calls <= 200 are kept as binary (model_id = 1) and roll calls
# > 200 use the NOMINATE probability as a standardised continuous outcome
# (model_id = 9, unbounded normal).

binary_part <- filter(senate114, rollnumber <= 200) %>%
  mutate(outcome_disc = cast_code,
         outcome_cont = NA_real_,
         model_id     = 1L)

cont_part <- filter(senate114, rollnumber > 200) %>%
  mutate(outcome_disc = NA_integer_,
         outcome_cont = as.numeric(scale(prob)),
         model_id     = 9L)

mixed_data <- bind_rows(binary_part, cont_part)

mixed_idealdata <- id_make(score_data  = mixed_data,
                           outcome_disc = 'outcome_disc',
                           outcome_cont = 'outcome_cont',
                           model_id     = 'model_id',
                           person_id    = 'bioname',
                           item_id      = 'rollnumber',
                           group_id     = 'party_code')

# Person- and item-level covariates.
# Pass one-sided formulas to person_cov and item_cov.  Covariates must be
# columns already present in score_data.  Center continuous covariates
# (here, age in units of 10 years) to improve mixing.
#
# person_cov hierarchically predicts each senator's ideal point.
# item_cov hierarchically predicts each roll call's discrimination parameter.

senate114$age <- (2018 - senate114$born)
senate114$age <- (senate114$age - mean(senate114$age)) / 10
```

```
senate_cov_data <- id_make(score_data = senate114,
                           outcome_disc = 'cast_code',
                           person_id    = 'bioname',
                           item_id      = 'rollnumber',
                           group_id     = 'party_code',
                           time_id      = 'date',
                           person_cov   = ~party_code + age,
                           item_cov     = ~session)
```

`id_me` *Calculate ideal point marginal effects*

Description

This function allows you to calculate ideal point marginal effects for a given person-level hierarchical covariate.

Usage

```
id_me(object, ...)
```

Arguments

`object`	A fitted `idealstan` model
`...`	Other values passed on to methods.

Details

This function will calculate item-level ideal point marginal effects for a given covariate that was passed to the `id_make` function using the `person_cov` option. The function will iterate over all items in the model and use numerical differentiation to calculate responses in the scale of the outcome for each item. Note: if the covariate is binary (i.e., only has two values), then the function will calculate the difference between these two values instead of using numerical differentation.

Value

Returns a tibble that has one row per posterior draw per item-specific marginal effect in the scale of the outcome.

Examples

```
data('senate114')
senate114$cast_code <- ifelse(senate114$cast_code=="Absent", NA,
                              as.integer(senate114$cast_code) - 1L)
senate114$age <- (2018 - senate114$born - mean(2018 - senate114$born)) / 10
sen_cov <- id_make(senate114, outcome_disc='cast_code',
                   person_id='bioname', item_id='rollnumber',
                   group_id='party_code', person_cov=~party_code+age)
```

```
sen_cov_est <- id_estimate(sen_cov, model_type=1, fixtype='vb_full',
                use_method="pathfinder",  ncores=4)
me <- id_me(sen_cov_est, covariate='age', draws=50)
me$sum_ideal_effects
```

`id_me,idealstan-method`

Calculate ideal point marginal effects

Description

This function allows you to calculate ideal point marginal effects for a given person-level hierarchical covariate.

Usage

```
## S4 method for signature 'idealstan'
id_me(
  object,
  covariate = NULL,
  group_effects = NULL,
  pred_outcome = NULL,
  eps = 1e-04,
  draws = 100,
  cores = 1,
  lb = 0.025,
  upb = 0.975,
  ...
)
```

Arguments

`object`	A fitted `idealstan` model
`covariate`	The character value for a covariate passed to the `id_make` function before model fitting. Only one covariate can be processed at a time.
`group_effects`	character value of a covariate included in the formula passed to `id_make` for which marginal effect summaries should be grouped by. Useful when looking at the marginal effect of an interaction. Note that grouping by a covariate with many values will result in slow performance.
`pred_outcome`	Numeric value for level of outcome to predict for ordinal responses. Defaults to top level.
`eps`	Parameter for numerical differentiation (usually does not need to be changed) passed on to id_post_pred
`draws`	The total number of draws to use when calculating the marginal effects. Defaults to 100. Use option "all" to use all available MCMC draws.

cores	The total number of cores to use when calculating the marginal effects. Defaults to 1.
lb	The quantile for the lower bound of the aggregated effects (default is 0.025 for a 95% interval)
upb	The quantile for the upper bound of the aggregated effects (default is 0.975 for a 95% interval)
...	Additional arguments passed on to id_post_pred

Details

This function will calculate item-level ideal point marginal effects for a given covariate that was passed to the id_make function using the person_cov option. The function will iterate over all items in the model and use numerical differentiation to calculate responses in the scale of the outcome for each item. Note: if the covariate is binary (i.e., only has two values), then the function will calculate the difference between these two values instead of using numerical differentation.

Value

A list with two objects, ideal_effects with one estimate of the marginal effect per item and posterior draw and sum_ideal_effects with one row per item with that item's median ideal point marginal effect with the quantiles defined by the upb and lb parameters.

Examples

```
data('senate114')
senate114$cast_code <- ifelse(senate114$cast_code=="Absent", NA,
                              as.integer(senate114$cast_code) - 1L)
senate114$age <- (2018 - senate114$born -
                  mean(2018 - senate114$born)) / 10
sen_cov <- id_make(senate114, outcome_disc='cast_code',
                   person_id='bioname', item_id='rollnumber',
                   group_id='party_code', person_cov=~party_code+age)
sen_cov_est <- id_estimate(sen_cov, model_type=1, fixtype='vb_full',
                           use_method="pathfinder", ncores=4)
me <- id_me(sen_cov_est, covariate='age', draws=50)
me$sum_ideal_effects
```

---

id_plot_all_hist *Density plots of Posterior Parameters*

Description

This function produces density plots of the different types of parameters in an idealstan model: item (bill) difficulty and discrimination parameters, and person (legislator) ideal points.

Usage

```
id_plot_all_hist(
  object,
  params = "person",
  param_labels = NULL,
  dens_type = "all",
  return_data = FALSE,
  func = median,
  ...
)
```

Arguments

object	A fitted idealstan object
params	Select the type of parameter from the model to plot. 'person' for person/legislator ideal points, 'miss_diff' and 'miss_discrim' for difficulty and discrimination parameters from the missing/inflated item/bill parameters, and 'obs_diff' and 'obs_discrim' for difficulty and discrimination parameters from the non-missing/non-inflated item/bill parameters.
param_labels	A vector of labels equal to the number of parameters. Primarily useful if return_data is TRUE.
dens_type	Can be 'all' for showing 90% HPD high, 10% HPD low and median posterior values. Or to show one of those posterior estimates at a time, use 'high' for 90% high HPD posterior estimate, 'low' for 10% low HPD posterior estimate, and 'function' for the whatever function is specificied in func (median by default).
return_data	Whether or not to return the plot as a ggplot2 object and the data together in a list instead of plotting.
func	The function to use if 'dens_type' is set to 'function'.
...	Other options passed on to the plotting function, currently ignored.

Value

A ggplot2 object showing density or histogram plots of the selected model parameters. If return_data=TRUE, a list with elements plot and data.

Examples

```
sim <- id_sim_gen()
est <- id_estimate(sim, model_type=1, fixtype='vb_full',
                   use_method="pathfinder", nchains=2, ncores=2)
# plot posterior distribution of person ideal points
id_plot_all_hist(est, params='person')
# plot item discrimination parameters
id_plot_all_hist(est, params='obs_discrim')
```

`id_plot_compare` *Function to compare two fitted idealstan models by plotting ideal points. Assumes that underlying data is the same for both models.*

Description

Function to compare two fitted idealstan models by plotting ideal points. Assumes that underlying data is the same for both models.

Usage

```
id_plot_compare(
  model1 = NULL,
  model2 = NULL,
  scale_flip = FALSE,
  return_data = FALSE,
  labels = NULL,
  hjust = -0.1,
  palette = "Set1",
  color_direction = 1,
  text_size_label = 2,
  rescale = FALSE
)
```

Arguments

`model1`	The first model to compare
`model2`	The second model to compare
`scale_flip`	This parameter is set to true if you have two models that are reflected around the ideal point axis. This can happen as a result of identification and is harmless.
`return_data`	Whether to return the underlying data
`labels`	`TRUE` or `FALSE`, whether to use labels for points
`hjust`	The horizontal adjustment of point labels
`palette`	`colorbrewer` palette name
`color_direction`	Whether to reverse the color scale
`text_size_label`	Size of point labels
`rescale`	Whether to rescale the estimates from two models so they will match regardless of arbitrary scale shifts in the ideal points

Value

A `ggplot2` object comparing ideal points across two `idealstan` models. If `return_data=TRUE`, a list with elements `plot` and `data`.

Examples

```
# Fit the same data under two different model types and compare ideal points
sim <- id_sim_gen()
est1 <- id_estimate(sim, model_type=1, fixtype='vb_full',
                    use_method="pathfinder", nchains=2, ncores=2)
est2 <- id_estimate(sim, model_type=2, fixtype='vb_full',
                    use_method="pathfinder", nchains=2, ncores=2)
id_plot_compare(est1, est2)
```

---

id_plot_cov *Marginal Effects Plot for Hierarchical Covariates*

Description

This function will calculate and plot the ideal point marginal effects, or the first derivative of the IRT/ideal point model with respect to the hierarchical covariate, for each item in the model. The function `id_me()` is used to first calculate the ideal point marginal effects.

Usage

```
id_plot_cov(
  object,
  calc_param = NULL,
  label_high = "High",
  label_low = "Low",
  group_effects = NULL,
  plot_model_id = NULL,
  pred_outcome = NULL,
  lb = 0.05,
  upb = 0.95,
  facet_ncol = 2,
  cov_type = "person_cov",
  ...
)
```

Arguments

`object`	A fitted `idealstan` object
`calc_param`	Whether to calculate ideal point marginal effects for a given covariate. If NULL, the default, the function will instead produce a plot of the raw coefficients from the ideal point model. If passing the name of a covariate, should be a character value of a column in the data passed to the `id_make` function.
`label_high`	What label to use on the plot for the high end of the latent scale
`label_low`	What label to use on the plot for the low end of the latent scale

group_effects	Character value for name of column in data by which to subset the data. Must be a column passed to the id_make function
plot_model_id	The integer of the model ID to plot. If NULL and there are multiple model types, facet_wrap will be used to produce multiple plots with one for each model type.
pred_outcome	For discrete models with more than 2 categories, or binary models with missing data, which outcome to predict. This should be the value that matches what the outcome was coded as in the data passed to id_make().
lb	The lower limit of the posterior density to use for calculating credible intervals
upb	The upper limit of the posterior density to use for calculating credible intervals
facet_ncol	If facetting by multiple models or grouped factors, sets the number of columns in the multiple plots
cov_type	Either 'person_cov' for person or group-level hierarchical parameters, 'discrim_reg_cov' for bill/item discrimination parameters from regular (non-inflated) model, and 'discrim_infl_cov' for bill/item discrimination parameters from inflated model.
...	Additional argument passed on to id_me

Details

The ends of the latent variable can be specified via the label_low and label_high options, which will use those labels for item discrimination.

Note that the function produces a ggplot2 object, which can be further modified with ggplot2 functions.

Value

A ggplot2 plot that can be further customized with ggplot2 functions if need be.

Examples

```
# Fit a model with a person-level covariate
data('senate114')
senate114$cast_code <- ifelse(senate114$cast_code=="Absent", NA,
                              as.integer(senate114$cast_code) - 1L)
senate114$age <- (2018 - senate114$born - mean(2018 - senate114$born)) / 10
sen_cov <- id_make(senate114, outcome_disc='cast_code',
                   person_id='bioname', item_id='rollnumber',
                   group_id='party_code', person_cov=~party_code+age)
sen_cov_est <- id_estimate(sen_cov, model_type=1, fixtype='vb_full',
                use_method="pathfinder", nchains=2, ncores=4)

# Plot marginal effects of age on ideal points
id_plot_cov(sen_cov_est, calc_param='age')

# Plot the raw person-level covariate coefficients
id_plot_cov(sen_cov_est)
```

`id_plot_gbeta_prior` *Launch the Generalized Beta Distribution Explorer Shiny App*

Description

This function starts an interactive Shiny application to visualize a generalized Beta distribution on a symmetrical interval from -scale to +scale in order to calculate the distribution paramters alpha (the `restrict_sd_high`/`restrict_N_low` parameter to `id_estimate`) and beta (the `restrict_N_high`/`restrict_sd_low` parameter to `id_estimate`). This function is useful for understanding what values to use to pin item discrimination parameters in `id_estimate` given a specific prior mean `initial_y` and a relative level of precision `prior_sample_size` (also known as `phi`). Higher values of `prior_sample_size` will imply a tighter prior around the pinned discrimination parameter.

Usage

```
id_plot_gbeta_prior(initial_y = 0, prior_sample_size = 200, limits = c(-1, 1))
```

Arguments

`initial_y`	Initial observed variate `y` (default: 0)
`prior_sample_size`	Relative level of precision/tightness around `y` (default: 200). Can be thought of as the relative sample size used to estimate `y`.
`limits`	Limits of prior. Default is `[-1,1]`. Set to `[0,1]` for a conventional IRT with all positive discrimination parameters (i.e. as in a test-taking scenario).

Details

The function is also useful for calculating the prior for all non-constrained discrimination parameters as well (`discrim_reg_shape` and `discrim_reg_scale`). These parameters are denoted in the Shiny app output for easy cut and paste to the `id_estimate` function call.

Value

A `ggplot2` object showing the generalized Beta prior distribution for the selected parameter values, or an interactive Shiny application if run interactively.

Examples

```
# Launch interactive Shiny app to explore the generalized Beta prior
if(interactive()) {

id_plot_gbeta_prior(initial_y=0.5, prior_sample_size=200)

}
```

id_plot_items *Plot Item Discrimination or Difficulty Parameters*

Description

This function plots item-level parameters (discrimination or difficulty) from a fitted idealstan object. It can display both observed (non-inflated) and missing (inflated) item parameters.

Usage

```
id_plot_items(
  object,
  param_type = "discrimination",
  item_type = "both",
  include = NULL,
  high_limit = 0.95,
  low_limit = 0.05,
  text_size_label = 2,
  point_size = 1,
  item_labels = TRUE,
  order_by = "median",
  return_data = FALSE,
  show_true = FALSE,
  use_chain = NULL,
  ...
)
```

Arguments

object	A fitted idealstan object
param_type	Which parameter to plot: "discrimination" (default) or "difficulty"
item_type	Which item parameters to show: "both" (default), "observed", or "missing". "observed" shows the non-inflated parameters (gamma/beta), "missing" shows the inflated/absence parameters (nu/omega).
include	A character vector of item IDs to include in the plot (all others excluded). If NULL (default), all items are shown.
high_limit	The quantile (number between 0 and 1) for the high end of posterior uncertainty to show in plot. Default is 0.95.
low_limit	The quantile (number between 0 and 1) for the low end of posterior uncertainty to show in plot. Default is 0.05.
text_size_label	The size of text labels for item names. Default is 2.
point_size	The size of points in the plot. Default is 1.
item_labels	If TRUE (default), show item name labels on the plot.

order_by	How to order items: "median" (default) orders by posterior median, "name" orders alphabetically by item name, "uncertainty" orders by width of posterior interval.
return_data	If TRUE, return the data used for plotting along with the plot as a list. Default is FALSE.
show_true	If the model was fitted on simulated data, show the true parameter values. Default is FALSE.
use_chain	Which MCMC chains to use. Default is NULL (use all chains).
...	Other arguments (currently ignored)

Value

A ggplot2 object, or if return_data=TRUE, a list with elements plot and data.

Examples

```
to_idealstan <-   id_make(score_data = senate114,
outcome_disc = 'cast_code',
person_id = 'bioname',
item_id = 'rollnumber',
time_id='date')

fitted_model <- id_estimate(to_idealstan,
model_type = 2,
use_method = "pathfinder",
fixtype='vb_partial',
restrict_ind_high = "BARRASSO, John A.",
restrict_ind_low = "WARREN, Elizabeth")

# After fitting a model:
# Plot discrimination parameters for all items
id_plot_items(fitted_model)

# Plot only difficulty parameters
id_plot_items(fitted_model, param_type = "difficulty")

# Plot only observed (non-inflated) discrimination
id_plot_items(fitted_model, item_type = "observed")

# Plot specific items only
id_plot_items(fitted_model, include = c("224", "368", "391"))
```

`id_plot_legis_var` *Plot Legislator/Person Over-time Variances*

Description

This function can be used on a fitted `idealstan` object to plot the over-time variances (average rates of change in ideal points) for all the persons/legislators in the model.

Usage

```
id_plot_legis_var(
  object,
  return_data = FALSE,
  include = NULL,
  high_limit = 0.95,
  low_limit = 0.05,
  text_size_label = 2,
  text_size_group = 2.5,
  point_size = 1,
  hjust_length = -0.7,
  person_labels = TRUE,
  group_labels = FALSE,
  person_ci_alpha = 0.1,
  group_color = TRUE,
  ...
)
```

Arguments

`object`	A fitted `idealstan` object
`return_data`	If true, the calculated legislator/bill data is returned along with the plot in a list
`include`	Specify a list of person/legislator IDs to include in the plot (all others excluded)
`high_limit`	The quantile (number between 0 and 1) for the high end of posterior uncertainty to show in plot
`low_limit`	The quantile (number between 0 and 1) for the low end of posterior uncertainty to show in plot
`text_size_label`	ggplot2 text size for legislator labels
`text_size_group`	ggplot2 text size for group text used for points
`point_size`	If `person_labels` and `group_labels` are set to `FALSE`, controls the size of the points plotted.
`hjust_length`	horizontal adjustment of the legislator labels
`person_labels`	if `TRUE`, use the person_id column to plot labels for the person (legislator) ideal points

group_labels — if TRUE, use the group column to plot text markers for the group (parties) from the person/legislator data

person_ci_alpha — The transparency level of the dot plot and confidence bars for the person ideal points

group_color — If TRUE, give each group/bloc a different color

... — Other options passed on to plotting function, currently ignored

Details

This function will plot the person/legislator over-time variances as a vertical dot plot with associated high-density posterior interval (can be changed with high_limit and low_limit options).

Value

A ggplot2 object showing person/legislator over-time variance in ideal points as a dot plot with posterior intervals. If return_data=TRUE, a list with elements plot and data.

Examples

```
data('senate114')
senate114$cast_code <- ifelse(senate114$cast_code=="Absent", NA,
                              as.integer(senate114$cast_code) - 1L)
senate_data <- id_make(senate114, outcome_disc='cast_code',
                       person_id='bioname', item_id='rollnumber',
                       group_id='party_code', time_id='date')
senate_time_fit <- id_estimate(senate_data,
                               model_type=2, fixtype='vb_full',
                               vary_ideal_pts='random_walk',
                               use_method="pathfinder",
                               restrict_ind_high="WARREN, Elizabeth",
                               restrict_ind_low="BARRASSO, John A.",
                               nchains=2, ncores=4)
id_plot_legis_var(senate_time_fit)
```

---

id_plot_persons *Plot Legislator/Person and Bill/Item Ideal Points*

Description

This function can be used on a fitted idealstan object to plot the relative positions and uncertainties of legislator/persons and bills/items.

Usage

```
id_plot_persons(
  object,
  return_data = FALSE,
  include = NULL,
  high_limit = 0.95,
  low_limit = 0.05,
  item_plot = NULL,
  item_plot_type = "non-inflated",
  text_size_label = 2,
  text_size_group = 2.5,
  point_size = 1,
  hjust_length = -0.7,
  person_labels = TRUE,
  group_labels = FALSE,
  person_ci_alpha = 0.2,
  show_true = FALSE,
  group_color = TRUE,
  hpd_limit = NULL,
  sample_persons = NULL,
  ...
)
```

Arguments

object	A fitted idealstan object or a named list of idealstan objects to compare across models
return_data	If true, the calculated legislator/bill data is returned along with the plot in a list
include	Specify a list of person/legislator IDs to include in the plot (all others excluded)
high_limit	The quantile (number between 0 and 1) for the high end of posterior uncertainty to show in plot
low_limit	The quantile (number between 0 and 1) for the low end of posterior uncertainty to show in plot
item_plot	The IDs (character vector) of the bill/item midpoints to overlay on the plot
item_plot_type	Whether to show the 'non-inflated' item/bill midpoints, the 'inflated' item/bill midpoints, or produce plots for 'both' kinds of models. Defaults to 'non-inflated' and will only display an item/bill midpoint if one has been specified in item_plot.
text_size_label	ggplot2 text size for legislator labels
text_size_group	ggplot2 text size for group text used for points
point_size	If person_labels and group_labels are set to FALSE, controls the size of the points plotted.
hjust_length	horizontal adjustment of the legislator labels

person_labels	if TRUE, use the person_id column to plot labels for the person (legislator) ideal points
group_labels	if TRUE, use the group column to plot text markers for the group (parties) from the person/legislator data
person_ci_alpha	The transparency level of the dot plot and confidence bars for the person ideal points
show_true	Whether to show the true values of the legislators (if model has been simulated)
group_color	If TRUE, give each group/bloc a different color
hpd_limit	The greatest absolute difference in high-posterior density interval shown for any point. Useful for excluding imprecisely estimated persons/legislators from the plot. Default is NULL if you don't want to exclude any.
sample_persons	If you don't want to use the full number of persons/legislators from the model, enter a proportion (between 0 and 1) to select only a fraction of the persons/legislators.
...	Other options passed on to plotting function, currently ignored

Details

This plot shows the distribution of ideal points for the legislators/persons in the model. It will plot them as a vertical dot plot with associated high-density posterior interval (can be changed with `high_limit` and `low_limit` options). In addition, if item/bill IDs as a character vector is passed to the `item_plot` option, then an item/bill midpoint will be overlain on the ideal point plot, showing the point at which legislators/persons are indifferent to voting/answering on the bill/item. Note that because this is an ideal point model, it is not possible to tell from the midpoint itself which side will be voting which way. For that reason, the legislators/persons are colored by their votes/scores to make it clear.

To compare across multiple `idealstan` models, pass a named list `list(model1=model1,model2=model2,etc)` to the `object` option. Note that these comparisons will done by individual persons/groups, so if there are a lot of persons/groups, consider using the `include` option to only compare a specific set of persons/groups.

Value

A `ggplot2` object showing person/legislator ideal points as a dot plot with posterior intervals. If `return_data=TRUE`, a list with elements `plot` and `data`.

Examples

```
# First create data and run a model

to_idealstan <-   id_make(score_data = senate114,
outcome_disc = 'cast_code',
person_id = 'bioname',
item_id = 'rollnumber',
group_id= 'party_code',
time_id='date')
```

```
sen_est <- id_estimate(to_idealstan,
model_type = 2,
use_method = "pathfinder",
fixtype='vb_partial',
restrict_ind_high = "BARRASSO, John A.",
restrict_ind_low = "WARREN, Elizabeth")

# After running the model, we can plot
# the results of the person/legislator ideal points

id_plot_persons(sen_est)
```

`id_plot_persons_dyn` *Function to plot dynamic ideal point models*

Description

This function can be used on a fitted `idealstan` object to plot the relative positions and uncertainties of legislator/persons and bills/items when the legislator/person ideal points are allowed to vary over time.

Usage

```
id_plot_persons_dyn(
  object,
  return_data = FALSE,
  include = NULL,
  item_plot = NULL,
  text_size_label = 2,
  text_size_group = 2.5,
  high_limit = 0.95,
  low_limit = 0.05,
  line_size = 1,
  highlight = NULL,
  plot_text = TRUE,
  use_ci = TRUE,
  plot_lines = 0,
  draw_line_alpha = 0.5,
  person_line_alpha = 0.3,
  person_ci_alpha = 0.8,
  item_plot_type = "non-inflated",
  show_true = FALSE,
  group_color = TRUE,
  hpd_limit = 10,
  sample_persons = NULL,
```

```
  use_chain = NULL,
  add_cov = TRUE,
  ...
)
```

Arguments

object	A fitted idealstan object or a named list of idealstan objects if the plot is supposed to show a comparison of different fitted idealstan models (see Time Series vignette)
return_data	If true, the calculated legislator/bill data is returned along with the plot in a list
include	Specify a list of person/legislator IDs to include in the plot (all others excluded)
item_plot	The value of the item/bill for which to plot its midpoint (character value)
text_size_label	ggplot2 text size for legislator labels
text_size_group	ggplot2 text size for group text used for points
high_limit	A number between 0 and 1 showing the upper limit to compute the posterior uncertainty interval (defaults to 0.95).
low_limit	A number between 0 and 1 showing the lower limit to compute the posterior uncertainty interval (defaults to 0.05).
line_size	Sets the size of the line of the time-varying ideal points.
highlight	A character referring to one of the persons in person_labels that the plot can highlight relative to other persons
plot_text	If TRUE, will plot person_labels over the lines.
use_ci	Whether or not high-posterior density intervals (credible intervals) should be plotted over the estimates (turn off if the plot is too busy)
plot_lines	The number of lines of actual draws of time-varying ideal points to draw on the plot. Note that these are grouped by persons. Specific draws selected at random from total number of draws of the estimation. Default is 0.
draw_line_alpha	The opacity of lines plotted over the distribution (should be between 0 and 1, default is 0.5).
person_line_alpha	The transparency level of the time-varying ideal point line
person_ci_alpha	The transparency level of ribbon confidence interval around the time-varying ideal points
item_plot_type	Whether to show the 'non-inflated' item/bill midpoints, the 'inflated' item/bill midpoints, or produce plots for 'both' kinds of models. Defaults to 'non-inflated' and will only display an item/bill midpoint if one has been specified in item_plot.
show_true	Whether to show the true values of the legislators (if model has been simulated)
group_color	If TRUE, use the groups instead of individuals to plot colours

hpd_limit	The greatest absolute difference in high-posterior density interval shown for any point. Useful for excluding imprecisely estimated persons/legislators from the plot. Leave NULL if you don't want to exclude any.
sample_persons	If you don't want to use the full number of persons/legislators from the model, enter a proportion (between 0 and 1) to select only a fraction of the persons/legislators. (see id_sim_gen())
use_chain	ID of MCMC chain to use rather than combining all chains. Default is NULL which will use all chains and is recommended.
add_cov	Whether to add values of hierarchical person-level covariates to the time trends (defaults to TRUE).
...	Other options passed on to plotting function, currently ignored

Details

This plot shows the distribution of ideal points for the legislators/persons in the model, and also traces the path of these ideal points over time. It will plot them as a vertical line with associated high-density posterior interval (10\ bill/item from the response matrix is passed to the item_plot option, then an item/bill midpoint will be overlain on the ideal point plot, showing the point at which legislators/persons are indifferent to voting/answering on the bill/item. Note that because this is an ideal point model, it is not possible to tell from the midpoint itself which side will be voting which way. For that reason, the legislators/persons are colored by their votes/scores to make it clear.

Value

A ggplot2 object showing dynamic person/legislator ideal points over time as a line plot with posterior intervals. If return_data=TRUE, a list with elements plot and data.

Examples

```
# First create data and run a model

to_idealstan <-   id_make(score_data = senate114,
outcome_disc = 'cast_code',
person_id = 'bioname',
item_id = 'rollnumber',
group_id= 'party_code',
time_id='date')

sen_est <- id_estimate(to_idealstan,
model_type = 2,
use_method = "pathfinder",
vary_ideal_pts='random_walk',
fixtype='vb_partial',
restrict_ind_high = "BARRASSO, John A.",
restrict_ind_low = "WARREN, Elizabeth")

# After running the model, we can plot
# the results of the person/legislator ideal points
```

```
id_plot_persons_dyn(sen_est)
```

`id_plot_ppc` *Plot Posterior Predictive Distribution for* `idealstan` *Objects*

Description

This function is the generic method for generating posterior distributions from a fitted `idealstan` model. Functions are documented in the actual method.

Usage

```
id_plot_ppc(object, ...)
```

Arguments

`object`	A fitted `idealstan` object
`...`	Other arguments passed on to `bayesplot::ppc_bars()`

Details

This function is a wrapper around `bayesplot::ppc_bars()`, `bayesplot::ppc_dens_overlay()` and `bayesplot::ppc_violin_grouped()` that plots the posterior predictive distribution derived from `id_post_pred()` against the original data. You can also subset the posterior predictions over legislators/persons or bills/item sby specifying the ID of each in the original data as a character vector. Only persons or items can be specified, not both.

If you specify a value for `group` that is either a person ID or a group ID (depending on whether a person or group-level model was fit), then you can see the posterior distributions for those specific persons. Similarly, if an item ID is passed to `item`, you can see how well the model predictions compare to the true values for that specific item.

Value

A `ggplot2` object (from the `bayesplot` package) comparing observed and posterior-predicted outcome distributions.

Examples

```
sim <- id_sim_gen()
est <- id_estimate(sim, model_type=1, fixtype='vb_full',
                   use_method="pathfinder", nchains=2, ncores=2)
ppc <- id_post_pred(est, draws=50)
id_plot_ppc(est, ppc_pred=ppc)
id_plot_ppc(est, ppc_pred=ppc, combine_item=FALSE, prompt_plot=FALSE)
```

id_plot_ppc,idealstan-method *Plot Posterior Predictive Distribution for* idealstan *Objects*

Description

This function is the actual method for generating posterior distributions from a fitted idealstan model.

Usage

```
## S4 method for signature 'idealstan'
id_plot_ppc(
  object,
  ppc_pred = NULL,
  group = NULL,
  item = NULL,
  combine_item = TRUE,
  type = NULL,
  which_mod = NULL,
  prompt_plot = TRUE,
  observed_only = FALSE,
  ...
)
```

Arguments

object	A fitted idealstan object
ppc_pred	The output of the id_post_pred() function on a fitted idealstan object
group	A character vector of the person or group IDs over which to subset the predictive distribution
item	A character vector of the item IDs over which to subset the predictive distribution
combine_item	Whether to combine all items together (TRUE) or create one plot for each item (FALSE)
type	Whether to plot "continuous" or "discrete" responses
which_mod	If you are producing one plot aggregating data across multiple items and you have different item distributions, then you need to specify the item type number to plot (see function documentation in id_estimate()).
prompt_plot	Whether to expect a user prompt for each plot if multiple plots are produced (defaults to TRUE) If NULL (default), will use the type specified in the data. However, if both continuous and discrete items are present, will throw an error if NULL.

observed_only — If the outcome is discrete and has missing data inflation, set to TRUE to only see the observed responses in the plot or FALSE to see all of the responses (missing data category will be the largest).

... — Other arguments passed on to bayesplot::ppc_bars()

Details

This function is a wrapper around bayesplot::ppc_bars(), bayesplot::ppc_dens_overlay() and bayesplot::ppc_violin_grouped() that plots the posterior predictive distribution derived from id_post_pred() against the original data. Because idealstan allows for different distributions for each item, this function can either produce one predictive distribution for all items (the default) or it can produce one distribution for each item (set combine_item to FALSE). The latter is helpful if you have mixed distributions between items, such as continuous and dichotomous values. You can also subset the posterior predictions over legislators/persons or bills/item sby specifying the ID of each in the original data as a character vector. Only persons or items can be specified, not both.

If you specify a value for group that is either a person ID or a group ID (depending on whether a person or group-level model was fit), then you can see the posterior distributions for those specific persons. Similarly, if an item ID is passed to item, you can see how well the model predictions compare to the true values for that specific item.

Value

A ggplot2 object (from the bayesplot package) comparing observed and posterior-predicted outcome distributions.

Examples

```
sim <- id_sim_gen()
est <- id_estimate(sim, model_type=1, fixtype='vb_full',
                   use_method="pathfinder", nchains=2, ncores=2)
ppc <- id_post_pred(est, draws=50)
id_plot_ppc(est, ppc_pred=ppc)
# per-item plots
id_plot_ppc(est, ppc_pred=ppc, combine_item=FALSE, prompt_plot=FALSE)
```

---

id_plot_rhats *Plotting Function to Display Rhat Distribution*

Description

This plotting function displays a histogram of the Rhat values of all parameters in an idealstan model.

Usage

```
id_plot_rhats(obj)
```

Arguments

obj	A fitted idealstan object.

Value

A ggplot2 histogram showing the distribution of Rhat convergence statistics for all parameters in the model.

Examples

```
sim <- id_sim_gen()
est <- id_estimate(sim, model_type=1, fixtype='vb_full',
                   use_method="pathfinder", nchains=2, ncores=2)
id_plot_rhats(est)
```

---

id_plot_sims *This function plots the results from a simulation generated by id_sim_gen().*

Description

This function plots the results from a simulation generated by id_sim_gen().

Usage

```
id_plot_sims(sims, type = "RMSE")

id_show_trues(sims, type = "RMSE")
```

Arguments

sims	A fitted idealstan object that has true data generated by id_sim_gen()
type	Type of analysis of true versus fitted values, can be 'RMSE', 'Residuals' or 'Coverage'

Value

A ggplot2 object showing RMSE, residuals, or coverage for simulated parameters compared to their true values.

Examples

```
sim <- id_sim_gen()
est <- id_estimate(sim, model_type=1, fixtype='vb_full',
                   use_method="pathfinder", nchains=2, ncores=2)
id_plot_sims(est)               # RMSE (default)
id_plot_sims(est, type='Residuals')
id_plot_sims(est, type='Coverage')
```

`id_post_pred` *Generic Method for Obtaining Posterior Predictive Distribution from Stan Objects*

Description

This function is a generic that is used to match the functions used with `bayesplot::ppc_bars()` to calculate the posterior predictive distribution of the data given the model.

Usage

```
id_post_pred(object, ...)
```

Arguments

`object`	A fitted `idealstan` object
`...`	All other parameters passed on to the underlying function.

Value

`posterior_predict` methods should return a D by N matrix, where D is the number of draws from the posterior predictive distribution and N is the number of data points being predicted per draw.

An object of class `id_pred_obj` (a list of matrices of posterior-predicted draws, one per model type). If `type="log_lik"`, a draws-by-observations `matrix` of class `log_lik` with a `chain_order` attribute for use with `derive_chain`.

Examples

```
sim <- id_sim_gen()
est <- id_estimate(sim, model_type=1, fixtype='vb_full',
                   use_method="pathfinder", nchains=2, ncores=2)
ppc <- id_post_pred(est, draws=50)
ll  <- id_post_pred(est, type='log_lik')
```

`id_post_pred,idealstan-method`
Posterior Prediction for `idealstan` *objects*

Description

This function will draw from the posterior distribution, whether in terms of the outcome (prediction) or to produce the log-likelihood values.

This function can also produce either distribution of the outcomes (i.e., predictions) or the log-likelihood values of the posterior (set option `type` to `'log_lik'`. For more information, see the package vignette How to Evaluate Models.

You can then use functions such as `id_plot_ppc()` to see how well the model does returning the correct number of categories in the score/vote matrix. Also see `help("posterior_predict", package = "rstanarm")`

Usage

```
## S4 method for signature 'idealstan'
id_post_pred(
  object,
  newdata = NULL,
  draws = 100,
  output = "observed",
  type = "predict",
  covar = "person",
  sample_scores = NULL,
  item_subset = NULL,
  pred_outcome = NULL,
  use_cores = 1,
  use_chain = NULL,
  skip_cov = FALSE,
  ...
)
```

Arguments

`object`	A fitted `idealstan` object
`newdata`	Optional: pass a data frame that must have all of the predictors that were given to the id_make function. Used to generate predictions from person or item covariates on to items.
`draws`	The number of draws to use from the total number of posterior draws (default is 100). Set to "all" to use all draws in the chains. For reproducibility, you can also pass a vector of specific draws to use.
`output`	If the model has an unbounded outcome (Poisson, continuous, etc.), then specify whether to show the `'observed'` data (the default) or the binary output `'missing'` showing whether an observation was predicted as missing or not
`type`	Whether to produce posterior predictive values (`'predict'`, the default), the posterior expected (average) values (`'epred'`), or log-likelihood values (`'log_lik'`). See the How to Evaluate Models vignette for more info.
`covar`	What kind of covariates to include as part of the prediction – either "person" (the default) or "items" if you included predictors for item discriminations.

sample_scores	In addition to reducing the number of posterior draws used to calculate the posterior predictive distribution, which will reduce computational overhead. Only available for calculating predictive distributions, not log-likelihood values.
item_subset	Whether to calculate marginal effects for only a subset of items. Should be item IDs that match the item_id column passed to the id_make function.
pred_outcome	In the case of ordinal responses, the number of the category to predict. Defaults to top category.
use_cores	Number of cores to use for multicore parallel processing with the base R parallel package
use_chain	ID of MCMC chain to use rather than all chains (the default).
skip_cov	Whether to skip adding in hierarchical person-level covariates. Defaults to FALSE.
...	Any other arguments passed on to posterior_predict (currently none available)

Value

An object of class id_pred_obj (a list of matrices of posterior-predicted draws, one per model type). If type="log_lik", a draws-by-observations matrix of class log_lik with a chain_order attribute for use with derive_chain.

Examples

```
sim <- id_sim_gen()
est <- id_estimate(sim, model_type=1, fixtype='vb_full',
                   use_method="pathfinder", nchains=2, ncores=2)
# posterior predictive distribution (100 draws)
ppc <- id_post_pred(est, draws=50)
# log-likelihood matrix (for loo)
ll <- id_post_pred(est, type='log_lik')
```

---

id_sim_coverage *Function that computes how often the true value of the parameter is included within the 95/5 high posterior density interval*

Description

Function that computes how often the true value of the parameter is included within the 95/5 high posterior density interval

Usage

```
id_sim_coverage(obj, rep = 1, quantiles = c(0.95, 0.05))
```

Arguments

obj	A fitted idealstan object with true data generated by id_sim_gen()
rep	How many times the models were fitted on new data, currently can only be 1
quantiles	What the quantile coverage of the high posterior density interval should be

Value

A named list of tibble objects, one per parameter type (Person Ideal Points, Absence Discriminations, Item Discrimations), each with columns avg, high, low, Params, est_type, and iter summarising posterior coverage of the true parameter values.

Examples

```
sim <- id_sim_gen()
est <- id_estimate(sim, model_type=1, fixtype='vb_full',
                   use_method="pathfinder", nchains=2, ncores=2)
cov <- id_sim_coverage(est)
# average coverage across all person ideal points
mean(cov[['Person Ideal Points']]$avg)
```

---

id_sim_gen *Simulate IRT ideal point data*

Description

A function designed to simulate IRT ideal point data.

Usage

```
id_sim_gen(
  num_person = 20,
  num_items = 50,
  cov_effect = NULL,
  person_cov = NULL,
  person_cov_effect = NULL,
  item_cov = NULL,
  item_discrim_cov_effect = NULL,
  item_miss_cov = NULL,
  item_miss_discrim_cov_effect = NULL,
  model_type = "binary",
  latent_space = FALSE,
  absence_discrim_sd = 3,
  absence_diff_mean = 0,
  discrim_reg_upb = 1,
  discrim_reg_lb = -1,
  discrim_miss_upb = 1,
```

```
  discrim_miss_lb = -1,
  discrim_reg_scale = 2,
  discrim_reg_shape = 2,
  discrim_miss_scale = 2,
  discrim_miss_shape = 2,
  diff_sd = 3,
  time_points = 1,
  time_process = "random",
  time_sd = 0.1,
  ideal_pts_sd = 3,
  prior_type = "gaussian",
  ordinal_outcomes = 3,
  inflate = FALSE,
  sigma_sd = 1,
  spline_knots = NULL,
  spline_degree = 2,
  spline_intercept_sd = 0.5,
  spline_basis_sd = 0.5,
  phi = 1,
  gp_rho = 0.5,
  gp_alpha = 0.5,
  gp_nugget = 0.1
)
```

Arguments

num_person	The number of persons/persons
num_items	The number of items (bills in the canonical ideal point model)
cov_effect	The effect of a hierarchical/external covariate on the person ideal points. The covariate will be a uniformly-distributed random variable on the [0,1] scale, so covariate effects in the [-2,2] approximate range would result in noticeable effects on the ideal point scale. This is a legacy parameter; prefer using person_cov and person_cov_effect for more control.
person_cov	Either NULL (no covariate), "continuous" to generate a uniform random covariate, "binary" to generate a binary (0/1) covariate, or a numeric matrix with num_person rows where each column is a covariate. If cov_effect is provided and person_cov is NULL, defaults to "continuous" for backward compatibility.
person_cov_effect	A numeric vector of regression coefficients for person covariates. Length must match the number of columns in person_cov matrix. If person_cov is "continuous" or "binary", a single value should be provided.
item_cov	Either NULL (no covariate), "continuous" to generate a uniform random covariate, "binary" to generate a binary (0/1) covariate, or a numeric matrix with num_items rows where each column is a covariate.
item_discrim_cov_effect	A numeric vector of regression coefficients for the effect of item covariates on item discrimination parameters (gamma). Length must match the number of columns in item_cov matrix.

item_miss_cov Either NULL (no covariate), "continuous" to generate a uniform random covariate, "binary" to generate a binary (0/1) covariate, or a numeric matrix with num_items rows where each column is a covariate. Used for missingness model parameters.

item_miss_discrim_cov_effect
A numeric vector of regression coefficients for the effect of item missing covariates on absence discrimination parameters (nu). Length must match the number of columns in item_miss_cov matrix.

model_type One of 'binary', 'ordinal_rating', 'ordinal_grm', 'poisson' 'normal', or 'lognormal'

latent_space Whether to use the latent space formulation of the ideal point model FALSE by default. NOTE: currently, the package only has estimation for a binary response with the latent space formulation.

absence_discrim_sd
The SD of the discrimination parameters for the inflated model

absence_diff_mean
The mean intercept for the inflated model; increasing it will lower the total number of missing data

discrim_reg_upb
The upper bound of the generalized Beta distribution for the observed discrimination parameters (gamma)

discrim_reg_lb The lower bound of the generalized Beta distribution for the observed discrimination parameters (gamma)

discrim_miss_upb
The upper bound of the generalized Beta distribution for the missingness discrimination parameters (nu)

discrim_miss_lb
The lower bound of the generalized Beta distribution for the missingness discrimination parameters (nu)

discrim_reg_scale
The scale parameter for the generalized Beta distribution for the observed discrimination parameters (gamma)

discrim_reg_shape
The shape parameter for the generalized Beta distribution for the observed discrimination parameters (gamma)

discrim_miss_scale
The scale parameter for the generalized Beta distribution for the missingness discrimination parameters (nu)

discrim_miss_shape
The shape parameter for the generalized Beta distribution for the missingness discrimination parameters (nu)

diff_sd The SD of the difficulty parameters (bill/item intercepts) for both missing and observed parameters (beta and omega)

time_points The number of time points for time-varying legislator/person parameters

`time_process` The process used to generate the ideal points: either `'random'` for a random walk, `'AR'` for an AR1 process, `'GP'` for a Gaussian process, or 'splines' for a spline (see parameters `spline_knots` and `spline_degree`).

`time_sd` The standard deviation of the change in ideal points over time (should be low relative to `ideal_pts_sd`)

`ideal_pts_sd` The SD for the person/person ideal points

`prior_type` The statistical distribution that generates the data for ideal point parameters (alpha) and difficulty intercepts (beta and omega). Currently only 'gaussian' is supported.

`ordinal_outcomes` If `model` is `'ordinal'`, an integer giving the total number of categories

`inflate` If `TRUE`, an missing-data-inflated dataset is produced.

`sigma_sd` If a normal or log-normal distribution is being fitted, this parameter gives the standard

`spline_knots` Number of knots (essentially, number of points at which to calculate time-varying ideal points given T time points). Default is NULL, which means that the spline is equivalent to polynomial time trend of degree `spline_degree`. Note that the spline number (if not null) must be equal or less than the number of time points.

`spline_degree` The degree of the spline polynomial. The default is 2 which is a quadratic polynomial. A value of 1 will result in independent knots (essentially pooled across time points T). A higher value will result in wigglier time series.

`spline_intercept_sd` The SD of the Normal distribution (centered on 0) used to draw the intercept for the basis spline function

`spline_basis_sd` The SD of the Normal distribution (centered on 0) for the coefficients used to create the simulated ideal points from the spline function

`phi` The phi (dispersion) parameter for the ordered beta distribution deviation of the outcome (i.e. the square root of the variance).

`gp_rho` The rho parameter to the squared exponential kernel function for the GP model

`gp_alpha` The alpha parameter to the squared exponential kernel function for the GP model

`gp_nugget` The nugget of the squared-exponential kernel (equals additional variance of the GP-distributed ideal points)

Details

This function produces simulated data that matches (as closely as possible) the models used in the underlying Stan code. Currently the simulation can produce inflated and non-inflated models with binary, ordinal (GRM and rating-scale), Poisson, Normal and Log-Normal responses.

Value

The results is a `idealdata` object that can be used in the id_estimate function to run a model. It can also be used in the simulation plotting functions.

See Also

id_show_trues() for plotting fitted models versus true values.

Examples

```
# Simulate binary (non-inflated) data for 20 persons and 50 items
sim <- id_sim_gen(num_person=20, num_items=50, inflate=FALSE)
head(sim@score_matrix)

# Simulate an ordinal graded-response model with missing-data inflation
sim_ord <- id_sim_gen(model='ordinal_grm', inflate=TRUE, num_person=10, num_items=20)
```

---

id_sim_resid

Residual function for checking estimated samples compared to true simulation scores Returns a data frame with residuals plus quantiles.

Description

Residual function for checking estimated samples compared to true simulation scores Returns a data frame with residuals plus quantiles.

Usage

```
id_sim_resid(obj, rep = 1)
```

Arguments

obj	A fitted idealstan object with true data from id_sim_gen()
rep	Over how many replicates to calculate residuals? Currently can only be 1

Value

A named list of tibble objects, one per parameter type (Ideal Points, Absence Discrimination, Item Discrimination), each with columns avg, high, low, Params, est_type, and iter summarising posterior residuals relative to true parameter values.

Examples

```
sim <- id_sim_gen()
est <- id_estimate(sim, model_type=1, fixtype='vb_full',
                   use_method="pathfinder", nchains=2, ncores=2)
id_sim_resid(est)
```

id_sim_rmse — *RMSE function for calculating individual RMSE values compared to true simulation scores Returns a data frame with RMSE plus quantiles.*

Description

RMSE function for calculating individual RMSE values compared to true simulation scores Returns a data frame with RMSE plus quantiles.

Usage

```
id_sim_rmse(obj, rep = 1)
```

Arguments

obj	A fitted idealstan object with true data from id_sim_gen()
rep	Over how many replicates to calculate RMSE? Currently can only be 1

Value

A named list of tibble objects, one per parameter type (Ideal Points, Absence Discriminations, Item Discrimations), each with columns avg, high, low, Params, est_type, and iter summarising RMSE relative to true parameter values.

Examples

```
sim <- id_sim_gen()
est <- id_estimate(sim, model_type=1, fixtype='vb_full',
                   use_method="pathfinder", nchains=2, ncores=2)
id_sim_rmse(est)
```

launch_shinystan — *Generic Method to Use* shinystan *with* idealstan

Description

A generic function for launching launch_shinystan.

Usage

```
launch_shinystan(object, ...)
```

Arguments

object	A fitted idealstan object.
...	Other arguments passed on to underlying function

Value

No return value; called for side effects (launches the Shinystan interactive diagnostics application).

`launch_shinystan,idealstan-method`
Function to Launch Shinystan with an `idealstan` *Object*

Description

This wrapper will pull the `rstan` samples out of a fitted `idealstan` model and then launch `launch_shinystan`. This function is useful for examining convergence statistics of the underlying MCMC sampling.

Usage

```
## S4 method for signature 'idealstan'
launch_shinystan(
  object,
 pars = c("L_full", "sigma_reg_full", "sigma_abs_free", "A_int_free", "B_int_free",
    "steps_votes", "steps_votes_grm"),
  ...
)
```

Arguments

`object`	A fitted `idealstan` object
`pars`	A character vector of parameters to select from the underlying `rstan` model object
`...`	Other parameters passed on to `shinystan`

Value

No return value; called for side effects (launches the Shinystan interactive diagnostics application).

See Also

`shinystan`

Examples

```
data('senate114')
senate114$cast_code <- ifelse(senate114$cast_code=="Absent", NA,
                              as.integer(senate114$cast_code) - 1L)
sen_data <- id_make(senate114, outcome_disc='cast_code',
                    person_id='bioname', item_id='rollnumber',
                    group_id='party_code')
sen_est <- id_estimate(sen_data, model_type=1, use_method="pathfinder", fixtype='vb_full', ncores=4)
```

```
# will launch interactive browser
if(interactive()) {
  launch_shinystan(sen_est)
}
```

`release_questions` *Function that provides additional check questions for package release*

Description

Function that provides additional check questions for package release

Usage

```
release_questions()
```

`senate114` *Rollcall vote data for 114th Senate*

Description

This rollcall vote object (see `pscl::rollcall()`) contains voting records for the 114th Senate in the US Congress. Not all rollcalls are included, only those that had a 70-30 or closer split in the vote. The data can be pre-processed via the `id_make()` function for estimation. See package vignette for details.

Usage

```
senate114
```

Format

A long data frame with one row for every vote cast by a Senator.

Source

https://voteview.com/

stan_trace *Plot the MCMC posterior draws by chain*

Description

This function allows you to produce trace plots for assessing the quality and convergence of MCMC chains.

Usage

```
stan_trace(object, ...)
```

Arguments

object	A fitted idealstan model
...	Other options passed on to stan_trace

Details

To use this function, you must pass a fitted idealstan object along with the name of a parameter in the model. To determine these parameter names, use the summary function or obtain the data from a plot by passing the return_data=TRUE option to id_plot_persons or id_plot_persons_dyn to find the name of the parameter in the Stan model.

This function is a simple wrapper around mcmc_trace. Please refer to that function's documentation for further options.

Value

A ggplot2 object (from mcmc_trace) showing MCMC trace plots for the selected parameter.

Examples

```
data('senate114')
senate114$cast_code <- ifelse(senate114$cast_code=="Absent", NA,
                              as.integer(senate114$cast_code) - 1L)
sen_data <- id_make(senate114, outcome_disc='cast_code',
                    person_id='bioname', item_id='rollnumber',
                    group_id='party_code')
sen_est <- id_estimate(sen_data, model_type=1,
           fixtype='vb_full', use_method="pathfinder", ncores=4)
stan_trace(sen_est, par='L_full[1]')
```

stan_trace,idealstan-method

Plot the MCMC posterior draws by chain

Description

This function allows you to produce trace plots for assessing the quality and convergence of MCMC chains.

Usage

```
## S4 method for signature 'idealstan'
stan_trace(object, par = "L_full[1]", ...)
```

Arguments

object	A fitted idealstan model
par	The character string name of a parameter in the model
...	Other options passed on to mcmc_trace

Details

To use this function, you must pass a fitted idealstan object along with the name of a parameter in the model. To determine these parameter names, use the summary function or obtain the data from a plot by passing the return_data=TRUE option to id_plot_persons or id_plot_persons_dyn to find the name of the parameter in the Stan model.

This function is a simple wrapper around mcmc_trace. Please refer to that function's documentation for further options.

Value

A ggplot2 object (from mcmc_trace) showing MCMC trace plots for the selected parameter.

Examples

```
data('senate114')
senate114$cast_code <- ifelse(senate114$cast_code=="Absent", NA,
                              as.integer(senate114$cast_code) - 1L)
sen_data <- id_make(senate114, outcome_disc='cast_code',
                    person_id='bioname', item_id='rollnumber',
                    group_id='party_code')
sen_est <- id_estimate(sen_data, model_type=1,
            fixtype='vb_full', use_method="pathfinder",
            ncores=4)
stan_trace(sen_est, par='L_full[1]')
```

summary,idealstan-method

Posterior Summaries for fitted idealstan *object*

Description

This function produces quantiles and standard deviations for the posterior samples of idealstan objects.

Usage

```
## S4 method for signature 'idealstan'
summary(
  object,
  pars = "ideal_pts",
  high_limit = 0.95,
  low_limit = 0.05,
  aggregated = TRUE,
  use_chain = NULL,
  cores = 1
)
```

Arguments

object	An idealstan object fitted by id_estimate
pars	Either 'ideal_pts' for person ideal points, 'items' for items/bills difficulty and discrimination parameters, and 'all' for all parameters in the model, including incidental parameters.
high_limit	A number between 0 and 1 reflecting the upper limit of the uncertainty interval (defaults to 0.95).
low_limit	A number between 0 and 1 reflecting the lower limit of the uncertainty interval (defaults to 0.05).
aggregated	Whether to return summaries of the posterior values or the full posterior samples. Defaults to TRUE.
use_chain	ID of a specific MCMC chain to use. Default (NULL) is all the chains and is recommended.
cores	Number of cores to use for parallel processing when summarizing item parameters. Defaults to 1 (no parallelization). Values greater than 1 use parallel::mclapply for faster computation with many items. Note that parallelization only works on Unix-like systems (Linux, macOS); on Windows, this parameter is ignored and single-core processing is used.

Value

A tibble data frame with parameters as rows and descriptive statistics as columns

Examples

```
data('senate114')
senate114$cast_code <- ifelse(senate114$cast_code=="Absent", NA,
                              as.integer(senate114$cast_code) - 1L)
sen_data <- id_make(senate114, outcome_disc='cast_code',
                    person_id='bioname', item_id='rollnumber',
                    group_id='party_code')
sen_est <- id_estimate(sen_data, model_type=1, fixtype='vb_full', use_method="pathfinder", ncores=4)
summary(sen_est, pars='ideal_pts')
summary(sen_est, pars='items')
```

Index