Package: pumBayes (via r-universe)

October 7, 2026

Type Package

Title Bayesian Estimation of Probit Unfolding Models for Binary Preference Data

Version 1.0.2

Maintainer Skylar Shi <dshi98@uw.edu>

Description Bayesian estimation and analysis methods for Probit Unfolding Models (PUMs), a novel class of scaling models designed for binary preference data. These models allow for both monotonic and non-monotonic response functions. The package supports Bayesian inference for both static and dynamic PUMs using Markov chain Monte Carlo (MCMC) algorithms with minimal or no tuning. Key functionalities include posterior sampling, hyperparameter selection, data preprocessing, model fit evaluation, and visualization. The methods are particularly suited to analyzing voting data, such as from the U.S. Congress or Supreme Court, but can also be applied in other contexts where non-monotonic responses are expected. For methodological details, see Shi et al. (2025) <doi:10.48550/arXiv.2504.00423>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.6.0)

Imports Rcpp

Suggests knitr, rmarkdown, pscl, MCMCpack

LinkingTo Rcpp, RcppArmadillo, RcppDist, mvtnorm, RcppTN

URL https://github.com/SkylarShiHub/pumBayes

BugReports https://github.com/SkylarShiHub/pumBayes/issues

Language en

NeedsCompilation yes

Repository https://skylarshihub.r-universe.dev

Date/Publication 2026-02-09 06:19:43 UTC

RemoteUrl https://github.com/skylarshihub/pumbayes

RemoteRef HEAD

RemoteSha 9399409a5a250a4c7d5396545aa3b10fb49d1f55

Contents

calc_waic . . . 2
dtnorm . . . 3
h116 . . . 4
item_char . . . 5
post_rank . . . 6
predict_ideal . . . 6
predict_irt . . . 7
predict_pum . . . 8
preprocess_rollcall . . . 9
sample_pum_dynamic . . . 10
sample_pum_static . . . 12
scotus.1937.2021 . . . 13
tune_hyper . . . 14

Index **15**

`calc_waic` *Calculate a block version of Watanabe-Akaike Information Criterion (WAIC)*

Description

This function is used to get the WAIC value when blocking members

Usage

```
calc_waic(vote_info, years_v = NULL, prob_array)
```

Arguments

`vote_info`	A logical vote matrix (or a rollcall object) in which rows represent members and columns represent issues. The entries should be FALSE ("No"), TRUE ("Yes"), or NA (missing data).
`years_v`	A vector representing the time period for each vote in the model. This is defultly set as 'NULL' for a static model.
`prob_array`	An array of probabilities with three dimensions.

Value

The block WAIC value for a static PUM or a vector of WAIC by time for a dynamic PUM.

Examples

```
# Long-running example
data(h116)
h116.c = preprocess_rollcall(h116)
hyperparams <- list(beta_mean = 0, beta_var = 1, alpha_mean = c(0, 0),
                    alpha_scale = 5, delta_mean = c(-2, 10), delta_scale = sqrt(10))
control <- list(num_iter = 2, burn_in = 0, keep_iter = 1, flip_rate = 0.1)
h116.c.pum <- sample_pum_static(h116.c, hyperparams,
                                control, pos_leg = grep("SCALISE", rownames(h116.c$votes)),
                                    verbose = FALSE, pre_run = NULL, appended = FALSE)
h116.c.pum.predprob = predict_pum(h116.c, years_v = NULL, h116.c.pum)
h116.c.pum.waic = calc_waic(h116.c, prob_array = h116.c.pum.predprob)
```

`dtnorm` *Density Function for Truncated Normal Distribution*

Description

This function calculates the density of a truncated normal distribution at specified points.

Usage

```
dtnorm(x, mean = 0, sd = 1, lower = -Inf, upper = Inf)
```

Arguments

`x`	A numeric vector of quantiles at which to evaluate the density.
`mean`	A numeric value specifying the mean of the normal distribution (default is 0).
`sd`	A numeric value specifying the standard deviation of the normal distribution (default is 1, must be positive).
`lower`	A numeric value specifying the lower bound of truncation (default is -Inf).
`upper`	A numeric value specifying the upper bound of truncation (default is Inf).

Value

A numeric vector of density values corresponding to the input ‘x‘. The values are normalized to ensure the total probability within the truncation bounds equals 1. Values outside the truncation bounds are set to 0.

Examples

```
dtnorm(c(-1, 0, 1), mean = 0, sd = 1, lower = -1, upper = 1)
```

h116 *116th U.S. House of Representatives Roll Call Votes*

Description

This dataset contains roll call voting records from the 116th U.S. House of Representatives. The data was obtained using the ‘readKH()‘ function from Voteview, which reads roll call vote data from the Voteview database.

Usage

```
data(h116)
```

Format

A list with 8 elements:

votes A ‘452 × 952‘ matrix of roll call votes, where each row represents a legislator and each column represents a vote.

codes A list containing vote codes:

- **yea** Codes representing ’Yes’ votes.
- **nay** Codes representing ’No’ votes.
- **notInLegis** Codes for members not in the legislature.
- **missing** Codes for missing votes.

n Integer, number of legislators (452).

m Integer, number of votes (952).

legis.data A data frame (‘452 × 6‘) containing legislator information:

- **state** State abbreviation of each legislator.
- **icpsrState** ICPSR state code.
- **cd** Congressional district.
- **icpsrLegis** ICPSR legislator ID.
- **party** Party affiliation (‘"D"‘, ‘"R"‘, ‘"I"‘.).
- **partyCode** Numerical party code (‘200‘ for Democrats, ‘100‘ for Republicans, ‘328‘ for Independents.).

vote.data Currently NULL (reserved for additional vote metadata).

desc Description: ‘"116th U.S. House of Representatives"‘.

source URL for the original data source.

Source

Jeffrey B. Lewis, Keith Poole, Howard Rosenthal, Adam Boche, Aaron Rudkin, and Luke Sonnet. Voteview: Congressional roll-call votes database. https://voteview.com/, 2024. Accessed: 2024-07-15.

Examples

```
data(h116)
str(h116)
```

---

item_char *Generate Data for Item Characteristic Curves*

Description

This function calculates the data needed to plot the item characteristic curve for a specific issue based on posterior samples.

Usage

```
item_char(vote_num, x = NULL, post_samples)
```

Arguments

vote_num	The vote number of the issue to be reviewed. This refers to numbers in the column names of the input vote matrix, not the clerk session vote number.
x	A vector showing the range of beta in the x axis.
post_samples	A list of posterior samples of parameters obtained from 'sample_pum_static' in 'pumBayes'.

Value

A data frame containing 'beta_samples', mean probabilities ('means'), and confidence intervals ('ci_lower' and 'ci_upper') for the input issue, which can be used to plot the item characteristic curve.

Examples

```
data(h116)
h116.c = preprocess_rollcall(h116)
hyperparams <- list(beta_mean = 0, beta_var = 1, alpha_mean = c(0, 0),
                    alpha_scale = 5, delta_mean = c(-2, 10), delta_scale = sqrt(10))
control <- list(num_iter = 2, burn_in = 0, keep_iter = 1, flip_rate = 0.1)
h116.c.pum <- sample_pum_static(h116.c, hyperparams,
                             control, pos_leg = grep("SCALISE", rownames(h116.c$votes)),
                                  verbose = FALSE, pre_run = NULL, appended = FALSE)
item_data <- item_char(vote_num = 5, x = c(-4,2), post_samples = h116.c.pum)
```

post_rank *Generate Quantile Ranks for Legislators*

Description

This function calculates quantile ranks for each legislator based on posterior samples of beta parameters from MCMC. The function can handle any specified quantiles, such as median (0.5), and is flexible to support other quantiles provided as input.

Usage

```
post_rank(beta, quantiles = c(0.5))
```

Arguments

beta	A matrix of posterior samples of beta obtained from MCMC, with columns representing legislators.
quantiles	A numeric vector specifying the quantiles to be calculated for the ranks (default is ‘c(0.5)‘ for median rank).

Value

A data frame containing the legislators' names, party affiliations, states, and their ranks at each specified quantile. If the median is included, it will be named ‘median‘ in the output. The output data frame is sorted in ascending order based on the values in the median column.

Examples

```
data(h116)
h116.c = preprocess_rollcall(h116)
hyperparams <- list(beta_mean = 0, beta_var = 1, alpha_mean = c(0, 0),
                    alpha_scale = 5, delta_mean = c(-2, 10), delta_scale = sqrt(10))
control <- list(num_iter = 2, burn_in = 0, keep_iter = 1, flip_rate = 0.1)
h116.c.pum <- sample_pum_static(h116.c, hyperparams,
                                control, pos_leg = grep("SCALISE", rownames(h116.c$votes)),
                                     verbose = FALSE, pre_run = NULL, appended = FALSE)
h116.c.beta.pum.rank = post_rank(beta = h116.c.pum$beta, quantiles = c(0.5))
```

predict_ideal *Calculate Probabilities for the IDEAL Model*

Description

This function computes the probability matrix for the IDEAL Model. Specifically, it calculates the probabilities of voting "Yea" for each legislator (member), issue, (and time period) based on the posterior samples of model parameters.

Usage

```
predict_ideal(vote_info, post_samples)
```

Arguments

vote_info	A logical vote matrix (or a rollcall object) in which rows represent members and columns represent issues. The entries should be FALSE ("No"), TRUE ("Yes"), or NA (missing data).
post_samples	Posterior samples obtained from function 'ideal' in 'pscl' package.

Value

An array of probabilities with three dimensions. The first one represents to members, the second one refers to issues, and the third one refers to MCMC iterations.

Examples

```
# Long-running example
data(h116)
h116.c = preprocess_rollcall(h116)
require(pscl)
cl = constrain.legis(h116.c, x = list("CLYBURN" = -1, "SCALISE" = 1),
                     d = 1)
h116.c.ideal = ideal(h116.c, d = 1, priors = cl, startvals = cl,
                     maxiter = 2, thin = 1, burnin = 0,
                     store.item = TRUE)
h116.c.ideal.predprob = predict_ideal(h116.c, h116.c.ideal)
```

---

predict_irt *Calculate Probabilities for Dynamic Item Response Theory Model*

Description

This function computes the probability matrix for a dynamic item response theory (IRT) model. Specifically, it calculates the probabilities of voting "Yea" for each legislator (member), issue, and time period based on the posterior samples of model parameters.

Usage

```
predict_irt(vote_info, years_v, post_samples)
```

Arguments

vote_info	A logical vote matrix where rows represent members and columns represent issues. The entries should be FALSE ("No"), TRUE ("Yes"), or NA (missing data).
years_v	A vector representing the time period for each vote in the model.
post_samples	MCMC results obtained from 'wnominate' function in 'wnominate' package.

Value

An array of probabilities with three dimensions. The first one represents to members, the second one refers to issues, and the third one refers to MCMC iterations.

Examples

```
# Long-running example
data(scotus.1937.2021)
library(MCMCpack)
special_judge_ind = sapply(c("HLBlack", "PStewart", "WHRehnquist"),
                           function(name){grep(name, rownames(mqVotes))})
e0_v = rep(0, nrow(mqVotes))
E0_v = rep(1, nrow(mqVotes))
e0_v[special_judge_ind] = c(-2, 1, 3)
E0_v[special_judge_ind] = c(10, 10, 10)
theta.start = rep(0, nrow(mqVotes))
indices = c(2, 5, 8, 9, 12, 22, 23, 24, 25, 29, 30, 33, 36, 39,
            42, 43, 44)
values = c(1, 1, -1, -2, -2, 1, -1, 1, 1, -1, 1, 3, 3, 3, 1, 1, -1)
theta.start[indices] = values
data(scotus.1937.2021)
scotus.MQ = MCMCdynamicIRT1d(mqVotes, mqTime, mcmc = 2,
                             burnin = 0, thin = 1, tau2.start = 0.1,
                         theta.start = theta.start, a0 = 0, A0 = 1, b0 = 0, B0 = 1, c0 = -10,
                             d0 = -2, e0 = e0_v, E0 = E0_v,
                       theta.constraints=list(CThomas = "+", SAAlito = "+", WJBrennan = "-",
                                              WODouglas = "-", CEWhittaker = "+"))
scotus.MQ.predprob = predict_irt(mqVotes, mqTime, scotus.MQ)
```

---

predict_pum *Calculate Probabilities for Probit Unfolding Models*

Description

This function computes the probability matrix for both static and dynamic Probit Unfolding Models. Specifically, it calculates the probabilities of voting "Yea" for each legislator (member), issue, (and time period) based on the posterior samples of model parameters.

Usage

```
predict_pum(vote_info, years_v = NULL, post_samples)
```

Arguments

vote_info	A logical vote matrix (or a rollcall object) in which rows represent members and columns represent issues. The entries should be FALSE ("No"), TRUE ("Yes"), or NA (missing data).

years_v	A vector representing the time period for each vote in the model. This is defultly set as 'NULL' for a static model.
post_samples	A list of posterior samples of parameters obtained from MCMC.

Value

An array of probabilities with three dimensions. The first one represents to members, the second one refers to issues, and the third one refers to MCMC iterations.

Examples

```
# Long-running example
data(h116)
h116.c = preprocess_rollcall(h116)
hyperparams <- list(beta_mean = 0, beta_var = 1, alpha_mean = c(0, 0),
                    alpha_scale = 5, delta_mean = c(-2, 10), delta_scale = sqrt(10))
control <- list(num_iter = 2, burn_in = 0, keep_iter = 1, flip_rate = 0.1)
h116.c.pum <- sample_pum_static(h116.c, hyperparams,
                             control, pos_leg = grep("SCALISE", rownames(h116.c$votes)),
                                 verbose = FALSE, pre_run = NULL, appended = FALSE)
h116.c.pum.predprob = predict_pum(h116.c, years_v = NULL, h116.c.pum)
```

---

preprocess_rollcall *Preprocess Roll Call Data*

Description

This function is used to preprocess roll call data for analysis. It allows users to remove legislators, combine legislators with specified indices, exclude lopsided votes based on minority voting proportions, and filter out legislators with excessive missing votes.

Usage

```
preprocess_rollcall(
  x,
 data_preprocess = list(leg_rm = NULL, combine_leg_index = NULL, combine_leg_party =
    NULL, lop_leg = 0.6, lop_issue = 0)
)
```

Arguments

x	A roll call object.
data_preprocess	A list of parameters for preprocessing data: • 'leg_rm' (default = NULL): A vector of indices specifying legislators to be removed.

- 'combine_leg_index' (default = NULL): A list of vectors where each vector specifies the indices of legislators to be combined.
- 'combine_leg_party' (default = NULL): A vector specifying the party affiliations for combined legislators.
- 'lop_leg' (default = 0.6): A threshold indicating the maximum allowable proportion of missing votes for each legislator. Legislators with a proportion of missing votes greater than this value are removed.
- 'lop_issue' (default = 0): A threshold for the proportion of non-missing votes on the minority side. Voting issues with a minority proportion lower than this value are excluded.

Value

A roll call object that has been processed.

Examples

```
data(h116)
h116.c = preprocess_rollcall(h116)
```

`sample_pum_dynamic` *Generate posterior samples from the dynamic probit unfolding model*

Description

This function generates posterior samples for all parameters based on the dynamic probit unfolding model.

Usage

```
sample_pum_dynamic(
  vote_info,
  years_v,
  hyperparams,
  control,
  sign_refs,
  verbose = FALSE,
  pre_run = NULL,
  appended = FALSE
)
```

Arguments

`vote_info`	A logical vote matrix where rows represent members and columns represent issues. The entries should be FALSE ("No"), TRUE ("Yes"), or NA (missing data).
`years_v`	A vector representing the time period for each vote in the model.

hyperparams	A list of hyperparameter values including: - ‘beta_mean‘: Prior mean of beta. - ‘beta_var‘: Prior variance of beta. - ‘alpha_mean‘: A vector of 2 values for the prior means of alpha1 and alpha2. - ‘alpha_scale‘: Scale parameter for alpha1 and alpha2. - ‘delta_mean‘: A vector of 2 values for the prior means of delta1 and delta2. - ‘delta_scale‘: Scale parameter for delta1 and delta2. - ‘rho_mean‘: Prior mean of the autocorrelation parameter ‘rho‘. - ‘rho_sigma‘: Standard deviation of the prior for ‘rho‘.
control	A list specifying the MCMC configurations, including: - ‘num_iter‘: Total number of iterations. - ‘burn_in‘: The number of initial iterations to discard as part of the burn-in period before retaining samples. - ‘keep_iter‘: Interval at which samples are retained. - ‘flip_rate‘: Probability of directly flipping signs in the M-H step, rather than resampling from the priors. - ‘sd_prop_rho‘: Proposal standard deviation for ‘rho‘ in the Metropolis-Hastings step.
sign_refs	A list containing sign constraints, including: - ‘pos_inds‘: Indices of members constrained to have positive values. - ‘neg_inds‘: Indices of members constrained to have negative values. - ‘pos_year_inds‘: List of years corresponding to each ‘pos_ind‘. - ‘neg_year_inds‘: List of years corresponding to each ‘neg_ind‘.
verbose	Logical. If ‘TRUE‘, prints progress and additional information during the sampling process.
pre_run	A list containing the output from a previous run of the function. If provided, the last iteration of the previous run will be used as the initial point of the new run. Defaults to ‘NULL‘.
appended	Logical. If ‘TRUE‘, the new samples will be appended to the samples from the previous run. Defaults to ‘FALSE‘.

Value

A list containing: - ‘beta‘: A data frame of posterior samples for beta. - ‘alpha1‘: A data frame of posterior samples for alpha1. - ‘alpha2‘: A data frame of posterior samples for alpha2. - ‘delta1‘: A data frame of posterior samples for delta1. - ‘delta2‘: A data frame of posterior samples for delta2. - ‘rho‘: A data frame of posterior samples for rho.

Examples

```
# Long-running example
data(scotus.1937.2021)
hyperparams = list(alpha_mean = c(0, 0), alpha_scale = 5,
                   delta_mean = c(-2, 10), delta_scale = sqrt(10),
                   rho_mean = 0.9, rho_sigma = 0.04)
control = list(num_iter = 2, burn_in = 0, keep_iter = 1, flip_rate = 0.1, sd_prop_rho = 0.1)
sign_refs = list(pos_inds = c(39, 5), neg_inds = c(12, 29),
                 pos_year_inds = list(1:31, 1), neg_year_inds = list(1:29, 1:24))
scotus.pum = sample_pum_dynamic(mqVotes, mqTime, hyperparams, control, sign_refs,
                                verbose = FALSE, pre_run = NULL, appended = FALSE)
```

`sample_pum_static` *Generate posterior samples from the static probit unfolding model*

Description

This function generates posterior samples of all parameters based on the static probit unfolding model.

Usage

```
sample_pum_static(
  vote_info,
  hyperparams,
  control,
  pos_leg = 0,
  verbose = FALSE,
  pre_run = NULL,
  appended = FALSE
)
```

Arguments

`vote_info`	A logical vote matrix (or a rollcall object) in which rows represent members and columns represent issues.
`hyperparams`	A list of hyperparameter values: - ‘beta_mean‘: Prior mean for beta. - ‘beta_var‘: Variance of beta. - ‘alpha_mean‘: A vector of two components representing the prior means of ‘alpha1‘ and ‘alpha2‘. - ‘alpha_scale‘: Scale parameter for ‘alpha1‘ and ‘alpha2‘. - ‘delta_mean‘: A vector of two components representing the prior means of ‘delta1‘ and ‘delta2‘. - ‘delta_scale‘: Scale parameter for ‘delta1‘ and ‘delta2‘.
`control`	A list of MCMC configurations: - ‘num_iter‘: Total number of iterations. It is recommended to set this to at least 30,000 to ensure reliable results. - ‘burn_in‘: The number of initial iterations to discard as part of the burn-in period before retaining samples. - ‘keep_iter‘: Interval at which iterations are kept for posterior samples. - ‘flip_rate‘: Probability of directly flipping signs in the M-H step, rather than resampling from the priors.
`pos_leg`	Name of the legislator whose position is kept positive.
`verbose`	Logical. If ‘TRUE‘, prints progress and additional information during the sampling process.
`pre_run`	A list containing the output from a previous run of the function. If provided, the last iteration of the previous run will be used as the initial point of the new run. Defaults to ‘NULL‘.
`appended`	Logical. If ‘TRUE‘, the new samples will be appended to the samples from the previous run. Defaults to ‘FALSE‘.

Value

A list primarily containing: - ‘beta‘: A matrix of posterior samples for ‘beta‘. - ‘alpha1‘: A matrix of posterior samples for ‘alpha1‘. - ‘alpha2‘: A matrix of posterior samples for ‘alpha2‘. - ‘delta1‘: A matrix of posterior samples for ‘delta1‘. - ‘delta2‘: A matrix of posterior samples for ‘delta2‘. - ‘vote_info‘: The input vote object.

Examples

```
# Long-running example
data(h116)
h116.c = preprocess_rollcall(h116)
hyperparams <- list(beta_mean = 0, beta_var = 1, alpha_mean = c(0, 0),
                    alpha_scale = 5, delta_mean = c(-2, 10), delta_scale = sqrt(10))
control <- list(num_iter = 2, burn_in = 0, keep_iter = 1, flip_rate = 0.1)
h116.c.pum <- sample_pum_static(h116.c, hyperparams,
                                control, pos_leg = grep("SCALISE", rownames(h116.c$votes)),
                                     verbose = FALSE, pre_run = NULL, appended = FALSE)
```

---

scotus.1937.2021 *U.S. Supreme Court Voting Data (1937-2021)*

Description

This dataset contains voting records from the U.S. Supreme Court between 1937 and 2021. Loading ‘data(scotus.1937.2021)‘ will load the following two independent objects into the environment:

Usage

```
data(scotus.1937.2021)
```

Format

The dataset consists of the following two objects:

mqVotes A ‘48 × 6108‘ matrix, where each row represents a judge and each column represents a case. Entries are:

- ‘1‘ (‘TRUE‘): The judge voted to reverse the lower court decision.
- ‘0‘ (‘FALSE‘): The judge voted to uphold the lower court decision.
- ‘NA‘: The judge did not vote on the case.

mqTime A numeric vector of length ‘6108‘, indicating the time period associated with each case.

Source

The data were obtained from the Martin-Quinn Scores Database, maintained by Washington University in St. Louis.

Examples

```
data(scotus.1937.2021)
str(mqVotes)
str(mqTime)
```

---

tune_hyper *Generate Probability Samples for Voting "Yes"*

Description

This function generates probability samples for Voting "Yes". It uses predefined hyperparameters and simulates data based on the specified number of members (‘n_leg‘) and issues (‘n_issue‘).

Usage

```
tune_hyper(hyperparams = hyperparams, n_leg, n_issue)
```

Arguments

hyperparams	A list of hyperparameter values: - ‘beta_mean‘: The prior mean of the ‘beta‘ parameter, representing legislator positions. - ‘beta_var‘: The prior variance of ‘beta‘. - ‘alpha_mean‘: A vector of length two, specifying the prior means of the item discrimination parameters, ‘alpha1‘ and ‘alpha2‘. - ‘alpha_scale‘: The scale parameter for ‘alpha1‘ and ‘alpha2‘. - ‘delta_mean‘: A vector of length two, indicating the prior means of the item difficulty parameters, ‘delta1‘ and ‘delta2‘. - ‘delta_scale‘: The scale parameter for ‘delta1‘ and ‘delta2‘.
n_leg	Integer, representing the number of legislators (members) to be simulated.
n_issue	Integer, indicating the number of issues to be simulated.

Value

A numeric vector containing the simulated probabilities of voting "Yes" for legislators across issues.

Examples

```
hyperparams = list(beta_mean = 0, beta_var = 1, alpha_mean = c(0, 0),
                   alpha_scale = 5, delta_mean = c(-2, 10),
                   delta_scale = sqrt(10))
theta = tune_hyper(hyperparams, n_leg = 10, n_issue = 10)
```

Index

∗ **datasets**
h116, 4
scotus.1937.2021, 13

calc_waic, 2

dtnorm, 3

h116, 4

item_char, 5

mqTime *(*scotus.1937.2021*)*, 13
mqVotes *(*scotus.1937.2021*)*, 13

post_rank, 6
predict_ideal, 6
predict_irt, 7
predict_pum, 8
preprocess_rollcall, 9

sample_pum_dynamic, 10
sample_pum_static, 12
scotus.1937.2021, 13

tune_hyper, 14