Package 'bgms'

July 22, 2025

Type Package

Title Bayesian Analysis of Networks of Binary and/or Ordinal Variables

Version 0.1.4.2

Date 2024-12-03

Maintainer Maarten Marsman <m.marsman@uva.nl>

Description Bayesian variable selection methods for analyzing the structure of a Markov Random Field model for a network of binary and/or ordinal variables. Details of the implemented methods can be found in: Marsman, van den Bergh, and Haslbeck (in press) <doi:10.31234/osf.io/ukwrf>.

License GPL (>= 2)

URL https://maartenmarsman.github.io/bgms/

BugReports https://github.com/MaartenMarsman/bgms/issues

Imports Rcpp (>= 1.0.7), Rdpack, methods

RdMacros Rdpack

LinkingTo Rcpp, RcppProgress

RoxygenNote 7.3.2

Depends R (>= 2.10)

LazyData true

Encoding UTF-8

Suggests knitr, qgraph, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website tidyverse/tidytemplate

NeedsCompilation yes

Author Maarten Marsman [aut, cre] (ORCID: <https://orcid.org/0000-0001-5309-7502>), Karoline Huth [ctb] (ORCID: <https://orcid.org/0000-0002-0662-1591>), Nikola Sekulovski [ctb] (ORCID: <https://orcid.org/0000-0001-7032-1684>),

Don van den Bergh [ctb] (ORCID:
<https://orcid.org/0000-0002-9838-7308>)

Repository CRAN

Date/Publication 2024-12-05 10:20:06 UTC

Contents

`bgm` *Bayesian edge selection or Bayesian estimation for a Markov random field model for binary and/or ordinal variables.*

Description

The function `bgm` explores the joint pseudoposterior distribution of parameters and possibly edge indicators for a Markov Random Field model for mixed binary and ordinal variables.

Usage

```
bgm(
  x,
  variable_type = "ordinal",
  reference_category,
  iter = 10000,
  burnin = 500,
  interaction_scale = 2.5,
  threshold_alpha = 0.5,
  threshold_beta = 0.5,
  edge_selection = TRUE,
  edge_prior = c("Bernoulli", "Beta-Bernoulli", "Stochastic-Block"),
  inclusion_probability = 0.5,
  beta_bernoulli_alpha = 1,
  beta_bernoulli_beta = 1,
  dirichlet_alpha = 1,
  na.action = c("listwise", "impute"),
  save = FALSE,
  display_progress = TRUE
)
```

Arguments

x	A data frame or matrix with n rows and p columns containing binary and ordinal variables for n independent observations and p variables in the network. Regular binary and ordinal variables are recoded as non-negative integers (0, 1, ..., m) if not already done. Unobserved categories are collapsed into other categories after recoding (i.e., if category 1 is unobserved, the data are recoded from (0, 2) to (0, 1)). Blume-Capel ordinal variables are also coded as non-negative integers if not already done. However, since "distance" to the reference category plays an important role in this model, unobserved categories are not collapsed after recoding.
variable_type	What kind of variables are there in x? Can be a single character string specifying the variable type of all p variables at once or a vector of character strings of length p specifying the type for each variable in x separately. Currently, bgm supports "ordinal" and "blume-capel". Binary variables are automatically treated as "ordinal". Defaults to variable_type = "ordinal".
reference_category	The reference category in the Blume-Capel model. Should be an integer within the range of integer scores observed for the "blume-capel" variable. Can be a single number specifying the reference category for all Blume-Capel variables at once, or a vector of length p where the i-th element contains the reference category for variable i if it is Blume-Capel, and bgm ignores its elements for other variable types. The value of the reference category is also recoded when bgm recodes the corresponding observations. Only required if there is at least one variable of type "blume-capel".
iter	How many iterations should the Gibbs sampler run? The default of 1e4 is for illustrative purposes. For stable estimates, it is recommended to run the Gibbs sampler for at least 1e5 iterations.
burnin	The number of iterations of the Gibbs sampler before saving its output. Since it may take some time for the Gibbs sampler to converge to the posterior distribution, it is recommended not to set this number too low. When edge_selection = TRUE, the bgm function will perform 2 * burnin iterations, first burnin iterations without edge selection, then burnin iterations with edge selection. This helps ensure that the Markov chain used for estimation starts with good parameter values and that the adaptive MH proposals are properly calibrated.
interaction_scale	The scale of the Cauchy distribution that is used as a prior for the pairwise interaction parameters. Defaults to 2.5.
threshold_alpha, threshold_beta	The shape parameters of the beta-prime prior density for the threshold parameters. Must be positive values. If the two values are equal, the prior density is symmetric about zero. If threshold_beta is greater than threshold_alpha, the distribution is skewed to the left, and if threshold_beta is less than threshold_alpha, it is skewed to the right. Smaller values tend to lead to more diffuse prior distributions.
edge_selection	Should the function perform Bayesian edge selection on the edges of the MRF in addition to estimating its parameters (edge_selection = TRUE), or should it just

estimate the parameters (edge_selection = FALSE)? The default is edge_selection = TRUE.

edge_prior The inclusion or exclusion of individual edges in the network is modeled with binary indicator variables that capture the structure of the network. The argument edge_prior is used to set a prior distribution for the edge indicator variables, i.e., the structure of the network. Currently, three options are implemented: The Bernoulli model edge_prior = "Bernoulli" assumes that the probability that an edge between two variables is included is equal to inclusion_probability and independent of other edges or variables. When inclusion_probability = 0.5, this means that each possible network structure is given the same prior weight. The Beta-Bernoulli model edge_prior = "Beta-Bernoulli" assumes a beta prior for the unknown inclusion probability with shape parameters beta_bernoulli_alpha and beta_bernoulli_beta. If beta_bernoulli_alpha = 1 and beta_bernoulli_beta = 1, this means that networks with the same complexity (number of edges) get the same prior weight. The Stochastic Block model edge_prior = "Stochastic-Block" assumes that nodes can be organized into blocks or clusters. In principle, the assignment of nodes to such clusters is unknown, and the model as implemented here considers all possible options (i.e., specifies a Dirichlet process on the node to block

allocation as described by Geng et al. 2019). This model is advantageous when nodes are expected to fall into distinct clusters. The inclusion probabilities for the edges are defined at the level of the clusters, with a beta prior for the unknown inclusion probability with shape parameters beta_bernoulli_alpha and beta_bernoulli_beta. The default is edge_prior = "Bernoulli".

inclusion_probability

The prior edge inclusion probability for the Bernoulli model. Can be a single probability, or a matrix of p rows and p columns specifying an inclusion probability for each edge pair. The default is inclusion_probability = 0.5.

beta_bernoulli_alpha, beta_bernoulli_beta

The two shape parameters of the Beta prior density for the Bernoulli inclusion probability. Must be positive numbers. Defaults to beta_bernoulli_alpha = 1 and beta_bernoulli_beta = 1.

dirichlet_alpha

The shape of the Dirichlet prior on the node-to-block allocation parameters for the Stochastic Block model.

na.action How do you want the function to handle missing data? If na.action = "listwise", listwise deletion is used. If na.action = "impute", missing data are imputed iteratively during the MCMC procedure. Since imputation of missing data can have a negative impact on the convergence speed of the MCMC procedure, it is recommended to run the MCMC for more iterations. Also, since the numerical routines that search for the mode of the posterior do not have an imputation option, the bgm function will automatically switch to interaction_prior = "Cauchy" and adaptive = TRUE.

save Should the function collect and return all samples from the Gibbs sampler (save = TRUE)? Or should it only return the (model-averaged) posterior means (save = FALSE)? Defaults to FALSE.

display_progress
: Should the function show a progress bar (display_progress = TRUE)? Or not (display_progress = FALSE)? The default is TRUE.

Details

Currently, bgm supports two types of ordinal variables. The regular, default, ordinal variable type has no restrictions on its distribution. Every response category except the first receives its own threshold parameter. The Blume-Capel ordinal variable assumes that there is a specific reference category, such as the "neutral" in a Likert scale, and responses are scored in terms of their distance to this reference category. Specifically, the Blume-Capel model specifies the following quadratic model for the threshold parameters:

$$\mu_{\mathrm{c}} = \alpha \times \mathrm{c} + \beta \times (\mathrm{c} - \mathrm{r})^2,$$

where μ_{c} is the threshold for category c. The parameter α models a linear trend across categories, such that $\alpha > 0$ leads to an increasing number of observations in higher response categories and $\alpha < 0$ leads to a decreasing number of observations in higher response categories. The parameter β models the response style in terms of an offset with respect to the reference category r; if $\beta < 0$ there is a preference to respond in the reference category (i.e., the model introduces a penalty for responding in a category further away from the reference_category category r), while if $\beta > 0$ there is preference to score in the extreme categories further away from the reference_category category.

The Bayesian estimation procedure (edge_selection = FALSE) simply estimates the threshold and pairwise interaction parameters of the ordinal MRF, while the Bayesian edge selection procedure (edge_selection = TRUE) also models the probability that individual edges should be included or excluded from the model. Bayesian edge selection imposes a discrete spike and slab prior distribution on the pairwise interactions. By formulating it as a mixture of mutually singular distributions, the function can use a combination of Metropolis-Hastings and Gibbs sampling to create a Markov chain that has the joint posterior distribution as an invariant. The current option for the slab distribution is a Cauchy with an optional scaling parameter. The slab distribution is also used as the prior for the interaction parameters for Bayesian estimation. A beta-prime distribution is used for the exponent of the category parameters. For Bayesian edge selection, two prior distributions are implemented for the edge inclusion variables (i.e., the prior probability that an edge is included); the Bernoulli prior and the Beta-Bernoulli prior.

Value

If save = FALSE (the default), the result is a list of class "bgms" containing the following matrices with model-averaged quantities:

- indicator: A matrix with p rows and p columns, containing the posterior inclusion probabilities of individual edges.
- interactions: A matrix with p rows and p columns, containing model-averaged posterior means of the pairwise associations.
- thresholds: A matrix with p rows and max(m) columns, containing model-averaged category thresholds. In the case of "blume-capel" variables, the first entry is the parameter for the linear effect and the second entry is the parameter for the quadratic effect, which models the offset to the reference category.

If save = TRUE, the result is a list of class "bgms" containing:

- `indicator`: A matrix with `iter` rows and `p * (p - 1) / 2` columns, containing the edge inclusion indicators from every iteration of the Gibbs sampler.
- `interactions`: A matrix with `iter` rows and `p * (p - 1) / 2` columns, containing parameter states from every iteration of the Gibbs sampler for the pairwise associations.
- `thresholds`: A matrix with `iter` rows and `sum(m)` columns, containing parameter states from every iteration of the Gibbs sampler for the category thresholds.

Column averages of these matrices provide the model-averaged posterior means.

In addition to the analysis results, the bgm output lists some of the arguments of its call. This is useful for post-processing the results.

References

Geng J, Bhattacharya A, Pati D (2019). "Probabilistic community detection with unknown number of communities." *Journal of the American Statistical Association*, **114**, 893–905. doi:10.1080/01621459.2018.1458618.

Examples

```
#Store user par() settings
op <- par(no.readonly = TRUE)

##Analyse the Wenchuan dataset

# Here, we use 1e4 iterations, for an actual analysis please use at least
# 1e5 iterations.
fit = bgm(x = Wenchuan)


#-----------------------------------------------------------------------------|
# INCLUSION - EDGE WEIGHT PLOT
#-----------------------------------------------------------------------------|

par(mar = c(6, 5, 1, 1))
plot(x = fit$interactions[lower.tri(fit$interactions)],
     y = fit$indicator[lower.tri(fit$indicator)], ylim = c(0, 1),
     xlab = "", ylab = "", axes = FALSE, pch = 21, bg = "gray", cex = 1.3)
abline(h = 0, lty = 2, col = "gray")
abline(h = 1, lty = 2, col = "gray")
abline(h = .5, lty = 2, col = "gray")
mtext("Posterior Mode Edge Weight", side = 1, line = 3, cex = 1.7)
mtext("Posterior Inclusion Probability", side = 2, line = 3, cex = 1.7)
axis(1)
axis(2, las = 1)


#-----------------------------------------------------------------------------|
# EVIDENCE - EDGE WEIGHT PLOT
#-----------------------------------------------------------------------------|

#For the default choice of the structure prior, the prior odds equal one:
```

```
prior.odds = 1
posterior.inclusion = fit$indicator[lower.tri(fit$indicator)]
posterior.odds = posterior.inclusion / (1 - posterior.inclusion)
log.bayesfactor = log(posterior.odds / prior.odds)
log.bayesfactor[log.bayesfactor > 5] = 5

par(mar = c(5, 5, 1, 1) + 0.1)
plot(fit$interactions[lower.tri(fit$interactions)], log.bayesfactor, pch = 21, bg = "#bfbfbf",
     cex = 1.3, axes = FALSE, xlab = "", ylab = "", ylim = c(-5, 5.5),
     xlim = c(-0.5, 1.5))
axis(1)
axis(2, las = 1)
abline(h = log(1/10), lwd = 2, col = "#bfbfbf")
abline(h = log(10), lwd = 2, col = "#bfbfbf")

text(x = 1, y = log(1 / 10), labels = "Evidence for Exclusion", pos = 1,
     cex = 1.7)
text(x = 1, y = log(10), labels = "Evidence for Inclusion", pos = 3, cex = 1.7)
text(x = 1, y = 0, labels = "Absence of Evidence", cex = 1.7)
mtext("Log-Inclusion Bayes Factor", side = 2, line = 3, cex = 1.5, las = 0)
mtext("Posterior Mean Interactions ", side = 1, line = 3.7, cex = 1.5, las = 0)


#---------------------------------------------------------------------------|
# THE MEDIAN PROBABILITY NETWORK
#---------------------------------------------------------------------------|

tmp = fit$interactions[lower.tri(fit$interactions)]
tmp[posterior.inclusion < 0.5] = 0

median.prob.model = matrix(0, nrow = ncol(Wenchuan), ncol = ncol(Wenchuan))
median.prob.model[lower.tri(median.prob.model)] = tmp
median.prob.model = median.prob.model + t(median.prob.model)

rownames(median.prob.model) = colnames(Wenchuan)
colnames(median.prob.model) = colnames(Wenchuan)

library(qgraph)
qgraph(median.prob.model,
       theme = "TeamFortress",
       maximum = .5,
       fade = FALSE,
       color = c("#f0ae0e"), vsize = 10, repulsion = .9,
       label.cex = 1.1, label.scale = "FALSE",
       labels = colnames(Wenchuan))

#Restore user par() settings
par(op)
```

bgmCompare *Bayesian variable selection or Bayesian estimation for differences in the Markov random field model for binary and/or ordinal variables in two independent samples.*

Description

The bgmCompare function estimates the pseudoposterior distribution of the parameters of a Markov Random Field model for mixed binary and ordinal variables, and the differences in pairwise interactions and category thresholds between two groups. The groups are assumed to be two independent samples.

Usage

```
bgmCompare(
  x,
  y,
  difference_selection = TRUE,
  main_difference_model = c("Free", "Collapse", "Constrain"),
  variable_type = "ordinal",
  reference_category,
  pairwise_difference_scale = 1,
  main_difference_scale = 1,
  pairwise_difference_prior = c("Bernoulli", "Beta-Bernoulli"),
  main_difference_prior = c("Bernoulli", "Beta-Bernoulli"),
  pairwise_difference_probability = 0.5,
  main_difference_probability = 0.5,
  pairwise_beta_bernoulli_alpha = 1,
  pairwise_beta_bernoulli_beta = 1,
  main_beta_bernoulli_alpha = 1,
  main_beta_bernoulli_beta = 1,
  interaction_scale = 2.5,
  threshold_alpha = 0.5,
  threshold_beta = 0.5,
  iter = 10000,
  burnin = 500,
  na.action = c("listwise", "impute"),
  save = FALSE,
  display_progress = TRUE
)
```

Arguments

x A data frame or matrix with n_1 rows and p columns containing binary and ordinal responses for the first group. Regular ordinal variables are recoded as non-negative integers (0, 1, ..., m) if not already done. Unobserved categories are collapsed into other categories after recoding (i.e., if category 1 is unobserved, the data are recoded from (0, 2) to (0, 1)). Blume-Capel ordinal variables are also

coded as non-negative integers if not already done. However, since "distance" from the reference category plays an important role in this model, unobserved categories are not collapsed after recoding.

`y` A data frame or matrix with n_2 rows and `p` columns containing binary and ordinal responses for the second group. The variables or columns in `y` must match the variables or columns in `x`. In the paired samples design, the rows in `x` must match the rows in `y`. Note that `x` and `y` are recoded independently, although the function checks that the number of different responses observed matches between `x` and `y`.

`difference_selection`
Logical. If `TRUE`, `bgmCompare` will model the inclusion or exclusion of the between samples parameter differences; if `FALSE`, it will estimate all between-sample parameter differences. Default is `TRUE`.

`main_difference_model`
A string specifying options for how the bgmCompare function should handle the comparison of threshold parameters when the observed categories in the samples do not match. The "Collapse" option tells bgmCompare to collapse the two categories into one (for the data set where both categories were observed). The "Constrain" option sets the difference between the category thresholds in the two data sets to zero if the category is not observed in one of the two data sets. The "Free" option tells bgmCompare to estimate a separate set of thresholds in the two samples and to not model their differences.

`variable_type` A string or vector specifying the type of variables in `x` (and `y`). Supported types are "ordinal" and "blume-capel", with binary variables treated as "ordinal". Default is "ordinal".

`reference_category`
The reference category in the Blume-Capel model. Should be an integer within the range of integer values observed for the "blume-capel" variable. Can be a single number that sets the reference category for all Blume-Capel variables at once, or a vector of length `p`, where the `i`-th element is the reference category for the `i` variable if it is a Blume-Capel variable, and elements for other variable types are ignored. The value of the reference category is also recoded when 'bgmCompare' recodes the corresponding observations. Required only if there is at least one variable of type "blume-capel".

`pairwise_difference_scale`
The scale of the Cauchy distribution that is used as the prior for the pairwise difference parameters. Defaults to `1`.

`main_difference_scale`
The scale of the Cauchy distribution that is used as the prior for the threshold difference parameters. Defaults to `1`.

`pairwise_difference_prior`
A character string that specifies the model to use for the inclusion probability of pairwise differences. Options are "Bernoulli" or "Beta-Bernoulli". Default is "Bernoulli".

`main_difference_prior`
A character string that specifies the model to use for the inclusion probability of threshold differences. Options are "Bernoulli" or "Beta-Bernoulli". Default is "Bernoulli".

pairwise_difference_probability
: The inclusion probability for a pairwise difference in the Bernoulli model. Can be a single probability or a matrix of p rows and p columns specifying the probability of a difference for each edge pair. Defaults to 0.5.

main_difference_probability
: The inclusion probability for a threshold difference in the Bernoulli model. Defaults to 0.5, implying no prior preference. Can be a single probability or a vector of length p specifying the probability of a difference between the category thresholds for each variable. Defaults to 0.5.

pairwise_beta_bernoulli_alpha
: The alpha parameter of the beta distribution for the Beta-Bernoulli model for group differences in pairwise interactions. Default is 1.

pairwise_beta_bernoulli_beta
: The beta parameter of the beta distribution for the Beta-Bernoulli model for group differences in pairwise interactions. Default is 1.

main_beta_bernoulli_alpha
: The alpha parameter of the beta distribution for the Beta-Bernoulli model for group differences in category thresholds. Default is 1.

main_beta_bernoulli_beta
: The beta parameter of the beta distribution for the Beta-Bernoulli model for group differences in category thresholds. Default is 1.

interaction_scale
: The scale of the Cauchy distribution that is used as a prior for the nuisance pairwise interaction parameters. Defaults to 2.5.

threshold_alpha, threshold_beta
: The shape parameters of the beta-prime prior density for the nuisance threshold parameters. Must be positive values. If the two values are equal, the prior density is symmetric about zero. If threshold_beta is greater than threshold_alpha, the distribution is left-skewed, and if threshold_beta is less than threshold_alpha, it is right-skewed. Smaller values tend to result in more diffuse prior distributions.

iter
: The function uses a Gibbs sampler to sample from the posterior distribution of the model parameters and indicator variables. How many iterations should this Gibbs sampler run? The default of 1e4 is for illustrative purposes. For stable estimates, it is recommended to run the Gibbs sampler for at least 1e5 iterations.

burnin
: The number of iterations of the Gibbs sampler before saving its output. Since it may take some time for the Gibbs sampler to converge to the posterior distribution, it is recommended not to set this number too low. When difference_selection = TRUE, the bgm function will perform 2 * burnin iterations, first burnin iterations without difference selection, then burnin iterations with difference selection. This helps ensure that the Markov chain used for estimation starts with good parameter values and that the adaptive MH proposals are properly calibrated.

na.action
: How do you want the function to handle missing data? If na.action = "listwise", listwise deletion is used. If na.action = "impute", missing data will be imputed iteratively during Gibbs sampling. Since imputation of missing data can

have a negative impact on the convergence speed of the Gibbs sampling procedure, it is recommended to run the procedure for more iterations.

`save` Should the function collect and return all samples from the Gibbs sampler (`save = TRUE`)? Or should it only return the (model-averaged) posterior means (`save = FALSE`)? Defaults to `FALSE`.

`display_progress` Should the function show a progress bar (`display_progress = TRUE`)? Or not (`display_progress = FALSE`)? The default is `TRUE`.

Details

In the first group, the pairwise interactions between the variables i and j are modeled as

$$\sigma_{ij} = \theta_{ij} + \delta_{ij}/2,$$

and in the second group as

$$\sigma_{ij} = \theta_{ij} - \delta_{ij}/2,$$

The pairwise interaction parameter θ_{ij} denotes an overall effect that is considered nuisance, and attention is focused on the pairwise difference parameter δ_{ij}, which reflects the difference in the pairwise interaction between the two groups.

The `bgmCompare` function supports two types of ordinal variables, which can be mixed. The default ordinal variable introduces a threshold parameter for each category except the lowest category. For this variable type, the threshold parameter for variable i, category c, is modeled as

$$\mu_{ic} = \tau_{ic} + \epsilon_{ic}/2,$$

in the first group and in the second group as

$$\mu_{ic} = \tau_{ic} - \epsilon_{ic}/2,$$

The category threshold parameter τ_{ic} denotes an overall effect that is considered nuisance, and attention is focused on the threshold difference parameter ϵ_{ic}, which reflects the difference in threshold of for variable i, category c between the two groups.

The Blume-Capel ordinal variable assumes that there is a specific reference category, such as "neutral" in a Likert scale, and responses are scored according to their distance from this reference category. In the first group, the threshold parameters are modelled as

$$\mu_{ic} = (\tau_{i1} + \epsilon_{i1}/2) \times c + (\tau_{i2} + \epsilon_{i2}/2) \times (c - r)^2,$$

and in the second groups as

$$\mu_{ic} = (\tau_{i1} - \epsilon_{i1}/2) \times c + (\tau_{i2} - \epsilon_{i2}/2) \times (c - r)^2.$$

The linear and quadratic category threshold parameters τ_{i1} and τ_{i2} denote overall effects that are considered nuisance, and attention is focused on the two threshold difference parameters ϵ_{i1} and ϵ_{i2}, which reflect the differences in the quadratic model for the variable i between the two groups.

Bayesian variable selection is used to model the presence or absence of the difference parameters δ and ϵ, which allow us to assess parameter differences between the two groups. Independent spike and slab priors are specified for these difference parameters. The spike and slab priors use binary indicator variables to select the difference parameters, assigning them a diffuse Cauchy prior with an optional scaling parameter if selected, or setting the difference parameter to zero if not selected.

The function offers two models for the probabilistic inclusion of parameter differences:

- **Bernoulli Model**: This model assigns a fixed probability of selecting a parameter difference, treating them as independent events. A probability of 0.5 indicates no preference, giving equal prior weight to all configurations.
- **Beta-Bernoulli Model**: Introduces a beta distribution prior for the inclusion probability that models the complexity of the configuration of the difference indicators. When the alpha and beta shape parameters of the beta distribution are set to 1, the model assigns the same prior weight to the number of differences present (i.e., a configuration with two differences or with four differences is a priori equally likely).

Inclusion probabilities can be specified for pairwise interactions with `pairwise_difference_probability` and for category thresholds with `threshold_difference_probability`.

The pairwise interaction parameters θ, the category threshold parameters τ, and, in the not yet implemented paired-samples designs, the between-sample interactions ω are considered nuisance parameters that are common to all models. The pairwise interaction parameters θ and the between-sample interactions ω are assigned a diffuse Cauchy prior with an optional scaling parameter. The exponent of the category threshold parameters τ are assigned beta-prime distribution with optional scale values.

Value

If `save = FALSE` (the default), the result is a list of class "bgmCompare" containing the following matrices:

- `indicator`: A matrix with `p` rows and `p` columns containing the posterior inclusion probabilities of the differences in pairwise interactions on the off-diagonal and the posterior inclusion probabilities of the differences in category thresholds on the diagonal.
- `difference_pairwise`: A matrix with `p` rows and `p` columns, containing model-averaged posterior means of the differences in pairwise interactions.
- `difference_threshold`: A matrix with `p` rows and `max(m)` columns, containing model-averaged posterior means of the differences in category thresholds.
- `interactions`: A matrix with `p` rows and `p` columns, containing posterior means of the nuisance pairwise interactions.
- `thresholds`: A matrix with `p` rows and `max(m)` columns containing the posterior means of the nuisance category thresholds. In the case of "blume-capel" variables, the first entry is the parameter for the linear effect and the second entry is the parameter for the quadratic effect, which models the offset to the reference category.

If `save = TRUE`, the result is a list of class "bgmCompare" containing the following matrices:

- `indicator_pairwise`: A matrix with `iter` rows and `p * (p - 1) / 2` columns containing the inclusion indicators for the differences in pairwise interactions from each iteration of the Gibbs sampler.
- `difference_pairwise`: A matrix with `iter` rows and `p * (p - 1) / 2` columns, containing parameter states for the differences in pairwise interactions from each iteration of the Gibbs sampler.
- `indicator_threshold`: A matrix with `iter` rows and `sum(m)` columns, containing the inclusion indicators for the differences in category thresholds from each iteration of the Gibbs sampler.

- `difference_threshold`: A matrix with `iter` rows and `sum(m)` columns, containing the parameter states for the differences in category thresholds from each iteration of the Gibbs sampler.
- `interactions`: A matrix with `iter` rows and `p * (p - 1) / 2` columns, containing parameter states for the nuisance pairwise interactions in each iteration of the Gibbs sampler.
- `thresholds`: A matrix with `iter` rows and `sum(m)` columns, containing parameter states for the nuisance category thresholds in each iteration of the Gibbs sampler.

Column averages of these matrices provide the model-averaged posterior means.

In addition to the results of the analysis, the output lists some of the arguments of its call. This is useful for post-processing the results.

`mrfSampler` *Sample observations from the ordinal MRF*

Description

This function samples states from the ordinal MRF using a Gibbs sampler. The Gibbs sampler is initiated with random values from the response options, after which it proceeds by simulating states for each variable from a logistic model using the other variable states as predictor variables.

Usage

```
mrfSampler(
  no_states,
  no_variables,
  no_categories,
  interactions,
  thresholds,
  variable_type = "ordinal",
  reference_category,
  iter = 1000
)
```

Arguments

`no_states`	The number of states of the ordinal MRF to be generated.
`no_variables`	The number of variables in the ordinal MRF.
`no_categories`	Either a positive integer or a vector of positive integers of length `no_variables`. The number of response categories on top of the base category: `no_categories = 1` generates binary states.
`interactions`	A symmetric `no_variables` by `no_variables` matrix of pairwise interactions. Only its off-diagonal elements are used.

`thresholds` A `no_variables` by `max(no_categories)` matrix of category thresholds. The elements in row `i` indicate the thresholds of variable `i`. If `no_categories` is a vector, only the first `no_categories[i]` elements are used in row `i`. If the Blume-Capel model is used for the category thresholds for variable `i`, then row `i` requires two values (details below); the first is α, the linear contribution of the Blume-Capel model and the second is β, the quadratic contribution.

`variable_type` What kind of variables are simulated? Can be a single character string specifying the variable type of all `p` variables at once or a vector of character strings of length `p` specifying the type for each variable separately. Currently, bgm supports "ordinal" and "blume-capel". Binary variables are automatically treated as "ordinal". Defaults to `variable_type = "ordinal"`.

`reference_category` An integer vector of length `no_variables` specifying the reference_category category that is used for the Blume-Capel model (details below). Can be any integer value between `0` and `no_categories` (or `no_categories[i]`).

`iter` The number of iterations used by the Gibbs sampler. The function provides the last state of the Gibbs sampler as output. By default set to `1e3`.

Details

There are two modeling options for the category thresholds. The default option assumes that the category thresholds are free, except that the first threshold is set to zero for identification. The user then only needs to specify the thresholds for the remaining response categories. This option is useful for any type of ordinal variable and gives the user the most freedom in specifying their model.

The Blume-Capel option is specifically designed for ordinal variables that have a special type of reference_category category, such as the neutral category in a Likert scale. The Blume-Capel model specifies the following quadratic model for the threshold parameters:

$$\mu_{\mathrm{c}} = \alpha \times \mathrm{c} + \beta \times (\mathrm{c} - \mathrm{r})^2,$$

where μ_{c} is the threshold for category c (which now includes zero), α offers a linear trend across categories (increasing threshold values if $\alpha > 0$ and decreasing threshold values if $\alpha < 0$), if $\beta < 0$, it offers an increasing penalty for responding in a category further away from the reference_category category r, while $\beta > 0$ suggests a preference for responding in the reference_category category.

Value

A `no_states` by `no_variables` matrix of simulated states of the ordinal MRF.

Examples

```
# Generate responses from a network of five binary and ordinal variables.
no_variables = 5
no_categories = sample(1:5, size = no_variables, replace = TRUE)

Interactions = matrix(0, nrow = no_variables, ncol = no_variables)
Interactions[2, 1] = Interactions[4, 1] = Interactions[3, 2] =
  Interactions[5, 2] = Interactions[5, 4] = .25
```

```
Interactions = Interactions + t(Interactions)
Thresholds = matrix(0, nrow = no_variables, ncol = max(no_categories))

x = mrfSampler(no_states = 1e3,
               no_variables = no_variables,
               no_categories = no_categories,
               interactions = Interactions,
               thresholds = Thresholds)

# Generate responses from a network of 2 ordinal and 3 Blume-Capel variables.
no_variables = 5
no_categories = 4

Interactions = matrix(0, nrow = no_variables, ncol = no_variables)
Interactions[2, 1] = Interactions[4, 1] = Interactions[3, 2] =
  Interactions[5, 2] = Interactions[5, 4] = .25
Interactions = Interactions + t(Interactions)

Thresholds = matrix(NA, no_variables, no_categories)
Thresholds[, 1] = -1
Thresholds[, 2] = -1
Thresholds[3, ] = sort(-abs(rnorm(4)), decreasing = TRUE)
Thresholds[5, ] = sort(-abs(rnorm(4)), decreasing = TRUE)

x = mrfSampler(no_states = 1e3,
               no_variables = no_variables,
               no_categories = no_categories,
               interactions = Interactions,
               thresholds = Thresholds,
               variable_type = c("b","b","o","b","o"),
               reference_category = 2)
```

`print.bgmCompare` *Print method for* `bgms` *objects*

Description

Used to prevent bgms output cluttering the console.

Usage

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

Arguments

`x`	An object of class `bgms`.
`...`	Ignored.

print.bgms *Print method for* bgms *objects*

Description

Used to prevent bgms output cluttering the console.

Usage

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

Arguments

x	An object of class bgms.
...	Ignored.

Wenchuan *Post-traumatic stress disorder symptoms of Wenchuan earthquake survivors*

Description

A data set containing items measuring symptoms of posttraumatic stress disorder (PTSD) (McNally et al. 2015). Participants were 362 Chinese adults who survived the Wenchuan earthquake and lost at least one child in the disaster. PTSD symptoms were reported using the civilian version of the Posttraumatic Checklist, which consists of 17 items, each assessing one of the DSM-IV symptoms of PTSD. Participants rated each item on a five-point scale ranging from "not at all" to "extremely" to indicate how much the symptom bothered them in the past month.

Usage

```
data("Wenchuan")
```

Format

A matrix with 362 rows and 17 columns:

intrusion Repeated, disturbing memories, thoughts, or images of a stressful experience from the past?

dreams Repeated, disturbing dreams of a stressful experience from the past?

flash Suddenly acting or feeling as if a stressful experience were happening again (as if you were reliving it)?

upset Feeling very upset when something reminded you of a stressful experience from the past?

physior Having physical reactions (e.g., heart pounding, trouble breathing, sweating) when something reminded you of a stressful experience from the past?

avoidth Avoiding thinking about or talking about a stressful experience from the past or avoiding having feelings related to it?

avoidact Avoiding activities or situations because they reminded you of a stressful experience from the past?

amnesia Trouble remembering important parts of a stressful experience from the past?

lossint Loss of interest in activities that you used to enjoy?

distant Feeling distant or cut off from other people?

numb Feeling emotionally numb or being unable to have loving feelings for those close to you?

future Feeling as if your future will somehow be cut short?

sleep Trouble falling or staying asleep?

anger Feeling irritable or having angry outbursts?

concen Having difficulty concentrating?

hyper Being "super-alert" or watchful or on guard?

startle Feeling jumpy or easily startled?

Source

http://psychosystems.org/wp-content/uploads/2014/10/Wenchuan.csv

References

McNally RJ, Robinaugh DJ, Wu GWY, Wang L, Deserno MK, Borsboom D (2015). "Mental disorders as causal systems: A network approach to posttraumatic stress disorder." *Clinical Psychological Science*, **6**, 836–849. doi:10.1177/2167702614553230.

Index