Package 'bgms'

May 7, 2026

Type Package

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

Version 0.1.6.3

Date 2026-02-14

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.

Copyright Includes datasets 'ADHD' and 'Boredom', which are licensed under CC-BY 4. See individual data documentation for license and citation.

License GPL (>= 2)

URL https://Bayesian-Graphical-Modelling-Lab.github.io/bgms/, https://github.com/Bayesian-Graphical-Modelling-Lab/bgms

BugReports https://github.com/Bayesian-Graphical-Modelling-Lab/bgms/issues

Imports Rcpp (>= 1.0.7), RcppParallel, Rdpack, methods, coda, lifecycle

RdMacros Rdpack

LinkingTo Rcpp, RcppArmadillo, RcppParallel, dqrng, BH

RoxygenNote 7.3.3

Depends R (>= 3.5)

LazyData true

Encoding UTF-8

Suggests covr, ggplot2, knitr, parallel, 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>),
Don van den Bergh [aut] (ORCID:
<https://orcid.org/0000-0002-9838-7308>),
Nikola Sekulovski [ctb] (ORCID:
<https://orcid.org/0000-0001-7032-1684>),
Giuseppe Arena [ctb] (ORCID: <https://orcid.org/0000-0001-5204-3326>),
Laura Groot [ctb],
Gali Geller [ctb]

Repository CRAN

Date/Publication 2026-02-14 16:50:09 UTC

Contents

ADHD *ADHD Symptom Checklist for Children Aged 6–8 Years*

Description

This dataset includes ADHD symptom ratings for 355 children aged 6 to 8 years from the Children's Attention Project (CAP) cohort (Silk et al. 2019). The sample consists of 146 children diagnosed with ADHD and 209 without a diagnosis. Symptoms were assessed through structured interviews with parents using the NIMH Diagnostic Interview Schedule for Children IV (DISC-IV) (Shaffer et al. 2000). The checklist includes 18 items: 9 Inattentive (I) and 9 Hyperactive/Impulsive (HI). Each item is binary (1 = present, 0 = absent).

Usage

```
data("ADHD")
```

Format

A matrix with 355 rows and 19 columns.

group ADHD diagnosis: 1 = diagnosed, 0 = not diagnosed

avoid Often avoids, dislikes, or is reluctant to engage in tasks that require sustained mental effort (I)

closeatt Often fails to give close attention to details or makes careless mistakes in schoolwork, work, or other activities (I)

distract Is often easily distracted by extraneous stimuli (I)

forget Is often forgetful in daily activities (I)

instruct Often does not follow through on instructions and fails to finish schoolwork, chores, or duties in the workplace (I)

listen Often does not seem to listen when spoken to directly (I)

loses Often loses things necessary for tasks or activities (I)

org Often has difficulty organizing tasks and activities (I)

susatt Often has difficulty sustaining attention in tasks or play activities (I)

blurts Often blurts out answers before questions have been completed (HI)

fidget Often fidgets with hands or feet or squirms in seat (HI)

interrupt Often interrupts or intrudes on others (HI)

motor Is often "on the go" or often acts as if "driven by a motor" (HI)

quiet Often has difficulty playing or engaging in leisure activities quietly (HI)

runs Often runs about or climbs excessively in situations in which it is inappropriate (HI)

seat Often leaves seat in classroom or in other situations in which remaining seated is expected (HI)

talks Often talks excessively (HI)

turn Often has difficulty awaiting turn (HI)

Source

Silk et al. (2019). Data retrieved from doi:10.1371/journal.pone.0211053.s004. Licensed under the CC-BY 4.0: https://creativecommons.org/licenses/by/4.0/

References

Shaffer D, Fisher P, Lucas CP, Dulcan MK, Schwab-Stone ME (2000). "NIMH Diagnostic Interview Schedule for Children Version IV (NIMH DISC-IV): description, differences from previous versions, and reliability of some common diagnoses." *Journal of the American Academy of Child & Adolescent Psychiatry*, **39**, 28–38. doi:10.1097/00004583200001000000014, PMID: 10638065.

Silk TJ, Malpas CB, Beare R, Efron D, Anderson V, Hazell P, Jongeling B, Nicholson JM, Sciberras E (2019). "A network analysis approach to ADHD symptoms: More than the sum of its parts." *PLOS ONE*, **14**(1), e0211053. doi:10.1371/journal.pone.0211053.

bgm *Bayesian Estimation or Edge Selection for Markov Random Fields*

Description

The bgm function estimates the pseudoposterior distribution of category thresholds (main effects) and pairwise interaction parameters of a Markov Random Field (MRF) model for binary and/or ordinal variables. Optionally, it performs Bayesian edge selection using spike-and-slab priors to infer the network structure.

Usage

```
bgm(
  x,
  variable_type = "ordinal",
  baseline_category,
  iter = 1000,
  warmup = 1000,
  pairwise_scale = 2.5,
  main_alpha = 0.5,
  main_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,
  beta_bernoulli_alpha_between = 1,
  beta_bernoulli_beta_between = 1,
  dirichlet_alpha = 1,
  lambda = 1,
  na_action = c("listwise", "impute"),
  update_method = c("nuts", "adaptive-metropolis", "hamiltonian-mc"),
  target_accept,
  hmc_num_leapfrogs = 100,
  nuts_max_depth = 10,
  learn_mass_matrix = TRUE,
  chains = 4,
  cores = parallel::detectCores(),
  display_progress = c("per-chain", "total", "none"),
  seed = NULL,
  standardize = FALSE,
  verbose = getOption("bgms.verbose", TRUE),
  interaction_scale,
  burnin,
  save,
  threshold_alpha,
  threshold_beta
```

)

Arguments

x	A data frame or matrix with n rows and p columns containing binary and ordinal responses. Variables are automatically recoded to non-negative integers (0, 1, . . . , m). For regular ordinal variables, unobserved categories are collapsed; for Blume–Capel variables, all categories are retained.
variable_type	Character or character vector. Specifies the type of each variable in x. Allowed values: "ordinal" or "blume-capel". Binary variables are automatically treated as "ordinal". Default: "ordinal".
baseline_category	Integer or vector. Baseline category used in Blume–Capel variables. Can be a single integer (applied to all) or a vector of length p. Required if at least one variable is of type "blume-capel".
iter	Integer. Number of post–burn-in iterations (per chain). Default: 1e3.
warmup	Integer. Number of warmup iterations before collecting samples. A minimum of 1000 iterations is enforced, with a warning if a smaller value is requested. Default: 1e3.
pairwise_scale	Double. Scale of the Cauchy prior for pairwise interaction parameters. Default: 2.5.
main_alpha, main_beta	Double. Shape parameters of the beta-prime prior for threshold parameters. Must be positive. If equal, the prior is symmetric. Defaults: main_alpha = 0.5 and main_beta = 0.5.
edge_selection	Logical. Whether to perform Bayesian edge selection. If FALSE, the model estimates all edges. Default: TRUE.
edge_prior	Character. Specifies the prior for edge inclusion. Options: "Bernoulli", "Beta-Bernoulli", or "Stochastic-Block". Default: "Bernoulli".
inclusion_probability	Numeric scalar. Prior inclusion probability of each edge (used with the Bernoulli prior). Default: 0.5.
beta_bernoulli_alpha, beta_bernoulli_beta	Double. Shape parameters for the beta distribution in the Beta–Bernoulli and the Stochastic-Block priors. Must be positive. For the Stochastic-Block prior these are the shape parameters for the within-cluster edge inclusion probabilities. Defaults: beta_bernoulli_alpha = 1 and beta_bernoulli_beta = 1.
beta_bernoulli_alpha_between, beta_bernoulli_beta_between	Double. Shape parameters for the between-cluster edge inclusion probabilities in the Stochastic-Block prior. Must be positive. Default: beta_bernoulli_alpha_between = 1 and beta_bernoulli_beta_between = 1
dirichlet_alpha	Double. Concentration parameter of the Dirichlet prior on block assignments (used with the Stochastic Block model). Default: 1.
lambda	Double. Rate of the zero-truncated Poisson prior on the number of clusters in the Stochastic Block Model. Default: 1.

`na_action` Character. Specifies missing data handling. Either `"listwise"` (drop rows with missing values) or `"impute"` (perform single imputation during sampling). Default: `"listwise"`.

`update_method` Character. Specifies how the MCMC sampler updates the model parameters:

`"adaptive-metropolis"` Componentwise adaptive Metropolis–Hastings with Robbins–Monro proposal adaptation.

`"hamiltonian-mc"` Hamiltonian Monte Carlo with fixed path length (number of leapfrog steps set by `hmc_num_leapfrogs`).

`"nuts"` The No-U-Turn Sampler, an adaptive form of HMC with dynamically chosen trajectory lengths.

Default: `"nuts"`.

`target_accept` Numeric between 0 and 1. Target acceptance rate for the sampler. Defaults are set automatically if not supplied: `0.44` for adaptive Metropolis, `0.65` for HMC, and `0.80` for NUTS.

`hmc_num_leapfrogs` Integer. Number of leapfrog steps for Hamiltonian Monte Carlo. Must be positive. Default: `100`.

`nuts_max_depth` Integer. Maximum tree depth in NUTS. Must be positive. Default: `10`.

`learn_mass_matrix` Logical. If `TRUE`, adapt a diagonal mass matrix during warmup (HMC/NUTS only). If `FALSE`, use the identity matrix. Default: `TRUE`.

`chains` Integer. Number of parallel chains to run. Default: `4`.

`cores` Integer. Number of CPU cores for parallel execution. Default: `parallel::detectCores()`.

`display_progress` Character. Controls progress reporting during sampling. Options: `"per-chain"` (separate bar per chain), `"total"` (single combined bar), or `"none"` (no progress). Default: `"per-chain"`.

`seed` Optional integer. Random seed for reproducibility. Must be a single non-negative integer.

`standardize` Logical. If `TRUE`, the Cauchy prior scale for each pairwise interaction is adjusted based on the range of response scores. Variables with more response categories have larger score products $x_i \cdot x_j$, which typically correspond to smaller interaction effects σ_{ij}. Without standardization, a fixed prior scale is relatively wide for these smaller effects, resulting in less shrinkage for high-category pairs and more shrinkage for low-category pairs. Standardization scales the prior proportionally to the maximum score product, ensuring equivalent relative shrinkage across all pairs. After internal recoding, regular ordinal variables have scores $0, 1, \ldots, m$. The adjusted scale for the interaction between variables i and j is `pairwise_scale * m_i * m_j`, so that `pairwise_scale` itself applies to the unit interval case (binary variables where $m_i = m_j = 1$). For Blume-Capel variables with reference category b, scores are centered as $-b, \ldots, m - b$, and the adjustment uses the maximum absolute product of the score endpoints. For mixed pairs, ordinal variables use raw score endpoints $(0, m)$ and Blume-Capel variables use centered score endpoints $(-b, m - b)$. Default: `FALSE`.

verbose — Logical. If TRUE, prints informational messages during data processing (e.g., missing data handling, variable recoding). Defaults to getOption("bgms.verbose", TRUE). Set options(bgms.verbose = FALSE) to suppress messages globally.

interaction_scale, burnin, save, threshold_alpha, threshold_beta — ‘r lifecycle::badge("deprecated")‘ Deprecated arguments as of **bgms 0.1.6.0**. Use ‘pairwise_scale‘, ‘warmup‘, ‘main_alpha‘, and ‘main_beta‘ instead.

Details

This function models the joint distribution of binary and ordinal variables using a Markov Random Field, with support for edge selection through Bayesian variable selection. The statistical foundation of the model is described in Marsman et al. (2025), where the ordinal MRF model and its Bayesian estimation procedure were first introduced. While the implementation in **bgms** has since been extended and updated (e.g., alternative priors, parallel chains, HMC/NUTS warmup), it builds on that original framework.

Key components of the model are described in the sections below.

Value

A list of class "bgms" with posterior summaries, posterior mean matrices, and access to raw MCMC draws. The object can be passed to print(), summary(), and coef().

Main components include:

- posterior_summary_main: Data frame with posterior summaries (mean, sd, MCSE, ESS, Rhat) for category threshold parameters.
- posterior_summary_pairwise: Data frame with posterior summaries for pairwise interaction parameters.
- posterior_summary_indicator: Data frame with posterior summaries for edge inclusion indicators (if edge_selection = TRUE).
- posterior_mean_main: Matrix of posterior mean thresholds (rows = variables, cols = categories or parameters).
- posterior_mean_pairwise: Symmetric matrix of posterior mean pairwise interaction strengths.
- posterior_mean_indicator: Symmetric matrix of posterior mean inclusion probabilities (if edge selection was enabled).
- Additional summaries returned when edge_prior = "Stochastic-Block". For more details about this prior see Sekulovski et al. (2025).
  - posterior_summary_pairwise_allocations: Data frame with posterior summaries (mean, sd, MCSE, ESS, Rhat) for the pairwise cluster co-occurrence of the nodes. This serves to indicate whether the estimated posterior allocations,co-clustering matrix and posterior cluster probabilities (see blow) have converged.
  - posterior_coclustering_matrix: a symmetric matrix of pairwise proportions of occurrence of every variable. This matrix can be plotted to visually inspect the estimated number of clusters and visually inspect nodes that tend to switch clusters.
  - posterior_mean_allocations: A vector with the posterior mean of the cluster allocations of the nodes. This is calculated using the method proposed in Dahl (2009).

  - `posterior_mode_allocations`: A vector with the posterior mode of the cluster allocations of the nodes.
  - `posterior_num_blocks`: A data frame with the estimated posterior inclusion probabilities for all the possible number of clusters.
- `raw_samples`: A list of raw MCMC draws per chain:

  `main` List of main effect samples.

  `pairwise` List of pairwise effect samples.

  `indicator` List of indicator samples (if edge selection enabled).

  `allocations` List of cluster allocations (if SBM prior used).

  `nchains` Number of chains.

  `niter` Number of post–warmup iterations per chain.

  `parameter_names` Named lists of parameter labels.
- `arguments`: A list of function call arguments and metadata (e.g., number of variables, warmup, sampler settings, package version).

The `summary()` method prints formatted posterior summaries, and `coef()` extracts posterior mean matrices.

NUTS diagnostics (tree depth, divergences, energy, E-BFMI) are included in `fit$nuts_diag` if `update_method = "nuts"`.

Ordinal Variables

The function supports two types of ordinal variables:

Regular ordinal variables: Assigns a category threshold parameter to each response category except the lowest. The model imposes no additional constraints on the distribution of category responses.

Blume-Capel ordinal variables: Assume a baseline category (e.g., a "neutral" response) and score responses by distance from this baseline. Category thresholds are modeled as:

$$\mu_c = \alpha \cdot (c - b) + \beta \cdot (c - b)^2$$

where:

- μ_c: category threshold for category c
- α: linear trend across categories
- β: preference toward or away from the baseline
  - If $\beta < 0$, the model favors responses near the baseline category;
  - if $\beta > 0$, it favors responses farther away (i.e., extremes).
- b: baseline category

Accordingly, pairwise interactions between Blume-Capel variables are modeled in terms of $c - b$ scores.

Edge Selection

When `edge_selection = TRUE`, the function performs Bayesian variable selection on the pairwise interactions (edges) in the MRF using spike-and-slab priors.

Supported priors for edge inclusion:

- **Bernoulli**: Fixed inclusion probability across edges.
- **Beta-Bernoulli**: Inclusion probability is assigned a Beta prior distribution.
- **Stochastic-Block**: Cluster-based edge priors with Beta, Dirichlet, and Poisson hyperpriors.

All priors operate via binary indicator variables controlling the inclusion or exclusion of each edge in the MRF.

Prior Distributions

- **Pairwise effects**: Modeled with a Cauchy (slab) prior.
- **Main effects**: Modeled using a beta-prime distribution.
- **Edge indicators**: Use either a Bernoulli, Beta-Bernoulli, or Stochastic-Block prior (as above).

Sampling Algorithms and Warmup

Parameters are updated within a Gibbs framework, but the conditional updates can be carried out using different algorithms:

- **Adaptive Metropolis–Hastings**: Componentwise random–walk updates for main effects and pairwise effects. Proposal standard deviations are adapted during burn–in via Robbins–Monro updates toward a target acceptance rate.
- **Hamiltonian Monte Carlo (HMC)**: Joint updates of all parameters using fixed–length leapfrog trajectories. Step size is tuned during warmup via dual–averaging; the diagonal mass matrix can also be adapted if `learn_mass_matrix = TRUE`.
- **No–U–Turn Sampler (NUTS)**: An adaptive extension of HMC that dynamically chooses trajectory lengths. Warmup uses a staged adaptation schedule (fast–slow–fast) to stabilize step size and, if enabled, the mass matrix.

When `edge_selection = TRUE`, updates of edge–inclusion indicators are carried out with Metropolis–Hastings steps. These are switched on after the core warmup phase, ensuring that graph updates occur only once the samplers' tuning parameters (step size, mass matrix, proposal SDs) have stabilized.

After warmup, adaptation is disabled. Step size and mass matrix are fixed at their learned values, and proposal SDs remain constant.

Warmup and Adaptation

The warmup procedure in `bgm` uses a multi-stage adaptation schedule (Stan Development Team 2023). Warmup iterations are split into several phases:

- **Stage 1 (fast adaptation)**: A short initial interval where only step size (for HMC/NUTS) is adapted, allowing the chain to move quickly toward the typical set.

- **Stage 2 (slow windows)**: A sequence of expanding, memoryless windows where both step size and, if `learn_mass_matrix = TRUE`, the diagonal mass matrix are adapted. Each window ends with a reset of the dual–averaging scheme for improved stability.
- **Stage 3a (final fast interval)**: A short interval at the end of the core warmup where the step size is adapted one final time.
- **Stage 3b (proposal–SD tuning)**: Only active when `edge_selection = TRUE` under HMC/NUTS. In this phase, Robbins–Monro adaptation of proposal standard deviations is performed for the Metropolis steps used in edge–selection moves.
- **Stage 3c (graph selection warmup)**: Also only relevant when `edge_selection = TRUE`. At the start of this phase, a random graph structure is initialized, and Metropolis–Hastings updates for edge inclusion indicators are switched on.

When `edge_selection = FALSE`, the total number of warmup iterations equals the user–specified `burnin`. When `edge_selection = TRUE` and `update_method` is `"nuts"` or `"hamiltonian-mc"`, the schedule automatically appends additional Stage-3b and Stage-3c intervals, so the total warmup is strictly greater than the requested `burnin`.

After all warmup phases, the sampler transitions to the sampling phase with adaptation disabled. Step size and mass matrix (for HMC/NUTS) are fixed at their learned values, and proposal SDs remain constant.

This staged design improves stability of proposals and ensures that both local parameters (step size) and global parameters (mass matrix, proposal SDs) are tuned before collecting posterior samples.

For adaptive Metropolis–Hastings runs, step size and mass matrix adaptation are not relevant. Proposal SDs are tuned continuously during burn–in using Robbins–Monro updates, without staged fast/slow intervals.

Missing Data

If `na_action = "listwise"`, observations with missing values are removed. If `na_action = "impute"`, missing values are imputed during Gibbs sampling.

References

Dahl DB (2009). "Modal clustering in a class of product partition models." *Bayesian Analysis*, **4**(2), 243–264. doi:10.1214/09BA409.

Marsman M, van den Bergh D, Haslbeck JMB (2025). "Bayesian analysis of the ordinal Markov random field." *Psychometrika*, **90**(1), 146–182. doi:10.1017/psy.2024.4.

Sekulovski N, Arena G, Haslbeck JMB, Huth KBS, Friel N, Marsman M (2025). "A Stochastic Block Prior for Clustering in Graphical Models." *Retrieved from* https://osf.io/preprints/psyarxiv/29p3m_v1. OSF preprint.

Stan Development Team (2023). *Stan Modeling Language Users Guide and Reference Manual*. Version 2.33, https://mc-stan.org/docs/.

See Also

`vignette("intro", package = "bgms")` for a worked example.

Examples

```
# Run bgm on subset of the Wenchuan dataset
fit = bgm(x = Wenchuan[, 1:5], chains = 2)

# Posterior inclusion probabilities
summary(fit)$indicator

# Posterior pairwise effects
summary(fit)$pairwise
```

---

bgmCompare *Bayesian Estimation and Variable Selection for Group Differences in Markov Random Fields*

Description

The bgmCompare function estimates group differences in category threshold parameters (main effects) and pairwise interactions (pairwise effects) of a Markov Random Field (MRF) for binary and ordinal variables. Groups can be defined either by supplying two separate datasets (x and y) or by a group membership vector. Optionally, Bayesian variable selection can be applied to identify differences across groups.

Usage

```
bgmCompare(
  x,
  y,
  group_indicator,
  difference_selection = TRUE,
  main_difference_selection = FALSE,
  variable_type = "ordinal",
  baseline_category,
  difference_scale = 1,
  difference_prior = c("Bernoulli", "Beta-Bernoulli"),
  difference_probability = 0.5,
  beta_bernoulli_alpha = 1,
  beta_bernoulli_beta = 1,
  pairwise_scale = 2.5,
  main_alpha = 0.5,
  main_beta = 0.5,
  iter = 1000,
  warmup = 1000,
  na_action = c("listwise", "impute"),
  update_method = c("nuts", "adaptive-metropolis", "hamiltonian-mc"),
  target_accept,
```

```
  hmc_num_leapfrogs = 100,
  nuts_max_depth = 10,
  learn_mass_matrix = TRUE,
  chains = 4,
  cores = parallel::detectCores(),
  display_progress = c("per-chain", "total", "none"),
  seed = NULL,
  standardize = FALSE,
  verbose = getOption("bgms.verbose", TRUE),
  main_difference_model,
  reference_category,
  main_difference_scale,
  pairwise_difference_scale,
  pairwise_difference_prior,
  main_difference_prior,
  pairwise_difference_probability,
  main_difference_probability,
  pairwise_beta_bernoulli_alpha,
  pairwise_beta_bernoulli_beta,
  main_beta_bernoulli_alpha,
  main_beta_bernoulli_beta,
  interaction_scale,
  threshold_alpha,
  threshold_beta,
  burnin,
  save
)
```

Arguments

x	A data frame or matrix of binary and ordinal responses for Group 1. Variables should be coded as nonnegative integers starting at 0. For ordinal variables, unused categories are collapsed; for Blume–Capel variables, all categories are retained.
y	Optional data frame or matrix for Group 2 (two-group designs). Must have the same variables (columns) as x.
group_indicator	Optional integer vector of group memberships for rows of x (multi-group designs). Ignored if y is supplied.
difference_selection	Logical. If TRUE, spike-and-slab priors are applied to difference parameters. Default: TRUE.
main_difference_selection	Logical. If TRUE, apply spike-and-slab selection to main effect (threshold) differences. If FALSE, main effect differences are always included (no selection). Since main effects are often nuisance parameters and their selection can interfere with pairwise selection under the Beta-Bernoulli prior, the default is FALSE. Only used when difference_selection = TRUE.

`variable_type` Character vector specifying type of each variable: `"ordinal"` (default) or `"blume-capel"`.

`baseline_category`
Integer or vector giving the baseline category for Blume–Capel variables.

`difference_scale`
Double. Scale of the Cauchy prior for difference parameters. Default: `1`.

`difference_prior`
Character. Prior for difference inclusion: `"Bernoulli"` or `"Beta-Bernoulli"`. Default: `"Bernoulli"`.

`difference_probability`
Numeric. Prior inclusion probability for differences (Bernoulli prior). Default: `0.5`.

`beta_bernoulli_alpha`, `beta_bernoulli_beta`
Doubles. Shape parameters of the Beta prior for inclusion probabilities in the Beta–Bernoulli model. Defaults: `1`.

`pairwise_scale` Double. Scale of the Cauchy prior for baseline pairwise interactions. Default: `2.5`.

`main_alpha`, `main_beta`
Doubles. Shape parameters of the beta-prime prior for baseline threshold parameters. Defaults: `0.5`.

`iter` Integer. Number of post–warmup iterations per chain. Default: `1e3`.

`warmup` Integer. Number of warmup iterations before sampling. Default: `1e3`.

`na_action` Character. How to handle missing data: `"listwise"` (drop rows) or `"impute"` (impute within Gibbs). Default: `"listwise"`.

`update_method` Character. Sampling algorithm: `"adaptive-metropolis"`, `"hamiltonian-mc"`, or `"nuts"`. Default: `"nuts"`.

`target_accept` Numeric between 0 and 1. Target acceptance rate. Defaults: 0.44 (Metropolis), 0.65 (HMC), 0.80 (NUTS).

`hmc_num_leapfrogs`
Integer. Leapfrog steps for HMC. Default: `100`.

`nuts_max_depth` Integer. Maximum tree depth for NUTS. Default: `10`.

`learn_mass_matrix`
Logical. If `TRUE`, adapts a diagonal mass matrix during warmup (HMC/NUTS only). Default: `TRUE`.

`chains` Integer. Number of parallel chains. Default: `4`.

`cores` Integer. Number of CPU cores. Default: `parallel::detectCores()`.

`display_progress`
Character. Controls progress reporting: `"per-chain"`, `"total"`, or `"none"`. Default: `"per-chain"`.

`seed` Optional integer. Random seed for reproducibility.

`standardize` Logical. If `TRUE`, the Cauchy prior scale for each pairwise interaction (both baseline and difference) is adjusted based on the range of response scores. Without standardization, pairs with more response categories experience less shrinkage because their naturally smaller interaction effects make a fixed prior relatively wide. Standardization equalizes relative shrinkage across all pairs, with `pairwise_scale` itself applying to the unit interval (binary) case. See `bgm` for details on the adjustment. Default: `FALSE`.

verbose Logical. If TRUE, prints informational messages during data processing (e.g., missing data handling, variable recoding). Defaults to getOption("bgms.verbose", TRUE). Set options(bgms.verbose = FALSE) to suppress messages globally.

main_difference_model, reference_category, pairwise_difference_scale, main_difference_scale, pairwise_difference_prior, main_difference_prior, pairwise_difference_probability, main_difference_probability, pairwise_beta_bernoulli_alpha, pairwise_beta_bernoulli_beta, main_beta_bernoulli_alpha, main_beta_bernoulli_beta, interaction_scale, threshold_alpha, threshold_beta, burnin, save
'r lifecycle::badge("deprecated")' Deprecated arguments as of **bgms 0.1.6.0**. Use 'difference_scale', 'difference_prior', 'difference_probability', 'beta_bernoulli_alpha', 'beta_bernoulli_beta', 'baseline_category', 'pairwise_scale', and 'warmup' instead.

Details

This function extends the ordinal MRF framework Marsman et al. (2025) to multiple groups. The basic idea of modeling, analyzing, and testing group differences in MRFs was introduced in Marsman et al. (2025), where two–group comparisons were conducted using adaptive Metropolis sampling. The present implementation generalizes that approach to more than two groups and supports additional samplers (HMC and NUTS) with staged warmup adaptation.

Key components of the model:

Value

A list of class "bgmCompare" containing posterior summaries, posterior mean matrices, and raw MCMC samples:

- posterior_summary_main_baseline, posterior_summary_pairwise_baseline: summaries of baseline thresholds and pairwise interactions.
- posterior_summary_main_differences, posterior_summary_pairwise_differences: summaries of group differences in thresholds and pairwise interactions.
- posterior_summary_indicator: summaries of inclusion indicators (if difference_selection = TRUE).
- posterior_mean_main_baseline, posterior_mean_pairwise_baseline: posterior mean matrices (legacy style).
- raw_samples: list of raw draws per chain for main, pairwise, and indicator parameters.
- arguments: list of function call arguments and metadata.

The summary() method prints formatted summaries, and coef() extracts posterior means.

NUTS diagnostics (tree depth, divergences, energy, E-BFMI) are included in fit$nuts_diag if update_method = "nuts".

Pairwise Interactions

For variables i and j, the group-specific interaction is represented as:

$$\theta_{ij}^{(g)} = \phi_{ij} + \delta_{ij}^{(g)},$$

where ϕ_{ij} is the baseline effect and $\delta_{ij}^{(g)}$ are group differences constrained to sum to zero.

Ordinal Variables

Regular ordinal variables: category thresholds are decomposed into a baseline plus group differences for each category.

Blume–Capel variables: category thresholds are quadratic in the category index, with both the linear and quadratic terms split into a baseline plus group differences.

Variable Selection

When `difference_selection = TRUE`, spike-and-slab priors are applied to difference parameters:

- **Bernoulli**: fixed prior inclusion probability.
- **Beta–Bernoulli**: inclusion probability given a Beta prior.

Sampling Algorithms and Warmup

Parameters are updated within a Gibbs framework, using the same sampling algorithms and staged warmup scheme described in `bgm`:

- **Adaptive Metropolis–Hastings**: componentwise random–walk proposals with Robbins–Monro adaptation of proposal SDs.
- **Hamiltonian Monte Carlo (HMC)**: joint updates with fixed leapfrog trajectories; step size and optionally the mass matrix are adapted during warmup.
- **No–U–Turn Sampler (NUTS)**: an adaptive HMC variant with dynamic trajectory lengths; warmup uses the same staged adaptation schedule as HMC.

For details on the staged adaptation schedule (fast–slow–fast phases), see `bgm`. In addition, when `difference_selection = TRUE`, updates of inclusion indicators are delayed until late warmup. In HMC/NUTS, this appends two extra phases (Stage-3b and Stage-3c), so that the total number of warmup iterations exceeds the user-specified `warmup`.

After warmup, adaptation is disabled: step size and mass matrix are fixed at their learned values, and proposal SDs remain constant.

References

Marsman M, Waldorp LJ, Sekulovski N, Haslbeck JMB (2025). "Bayes factor tests for group differences in ordinal and binary graphical models." *Psychometrika*, **90**(5), 1809–1842. doi:10.1017/psy.2025.10060.

Marsman M, van den Bergh D, Haslbeck JMB (2025). "Bayesian analysis of the ordinal Markov random field." *Psychometrika*, **90**(1), 146–182. doi:10.1017/psy.2024.4.

See Also

`vignette("comparison", package = "bgms")` for a worked example.

Examples

```
## Not run:
# Run bgmCompare on subset of the Boredom dataset
x = Boredom[Boredom$language == "fr", 2:6]
y = Boredom[Boredom$language != "fr", 2:6]

fit <- bgmCompare(x, y, chains = 2)

# Posterior inclusion probabilities
summary(fit)$indicator

# Bayesian model averaged main effects for the groups
coef(fit)$main_effects_groups

# Bayesian model averaged pairwise effects for the groups
coef(fit)$pairwise_effects_groups

## End(Not run)
```

Boredom *Short Boredom Proneness Scale Responses*

Description

This dataset includes responses to the 8-item Short Boredom Proneness Scale (SBPS), a self-report measure of an individual's susceptibility to boredom (Martarelli et al. 2023). Items were rated on a 7-point Likert scale ranging from 1 ("strongly disagree") to 7 ("strongly agree"). The scale was administered in either English (Struk et al. 2015) or French (translated by (Martarelli et al. 2023)).

Usage

```
data("Boredom")
```

Format

A matrix with 986 rows and 9 columns. Each row corresponds to a respondent.

language Language in which the SBPS was administered: "en" = English, "fr" = French

loose_ends I often find myself at "loose ends," not knowing what to do.

entertain I find it hard to entertain myself.

repetitive Many things I have to do are repetitive and monotonous.

stimulation It takes more stimulation to get me going than most people.

motivated I don't feel motivated by most things that I do.

keep_interest In most situations, it is hard for me to find something to do or see to keep me interested.

sit_around Much of the time, I just sit around doing nothing.

half_dead_dull Unless I am doing something exciting, even dangerous, I feel half-dead and dull.

Source

Martarelli et al. (2023). Data retrieved from https://osf.io/qhux8. Licensed under the CC-BY 4.0: https://creativecommons.org/licenses/by/4.0/

References

Martarelli CS, Baillifard A, Audrin C (2023). "A Trait-Based Network Perspective on the Validation of the French Short Boredom Proneness Scale." *European Journal of Psychological Assessment*, **39**(6), 390–399. doi:10.1027/10155759/a000718.

Struk AA, Carriere JSA, Cheyne JA, Danckert J (2015). "A Short Boredom Proneness Scale: Development and Psychometric Properties." *Assessment*, **24**(3), 346–359. doi:10.1177/1073191115609996.

---

coef.bgmCompare *Extract Coefficients from a bgmCompare Object*

Description

Returns posterior means for raw parameters (baseline + differences) and group-specific effects from a bgmCompare fit, as well as inclusion indicators.

Usage

```
## S3 method for class 'bgmCompare'
coef(object, ...)
```

Arguments

object	An object of class bgmCompare.
...	Ignored.

Value

A list with components:

main_effects_raw Posterior means of the raw main-effect parameters (variables x [baseline + differences]).

pairwise_effects_raw Posterior means of the raw pairwise-effect parameters (pairs x [baseline + differences]).

main_effects_groups Posterior means of group-specific main effects (variables x groups), computed as baseline plus projected differences.

pairwise_effects_groups Posterior means of group-specific pairwise effects (pairs x groups), computed as baseline plus projected differences.

indicators Posterior mean inclusion probabilities as a symmetric matrix, with diagonals corresponding to main effects and off-diagonals to pairwise effects.

`coef.bgms` *Extract Coefficients from a bgms Object*

Description

Returns the posterior mean thresholds, pairwise effects, and edge inclusion indicators from a `bgms` model fit.

Usage

```
## S3 method for class 'bgms'
coef(object, ...)
```

Arguments

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

Value

A list with the following components:

main Posterior mean of the category threshold parameters.

pairwise Posterior mean of the pairwise interaction matrix.

indicator Posterior mean of the edge inclusion indicators (if available).

`predict.bgmCompare` *Predict Conditional Probabilities from a Fitted bgmCompare Model*

Description

Computes conditional probability distributions for one or more variables given the observed values of other variables in the data, using group-specific parameters from a `bgmCompare` model.

Usage

```
## S3 method for class 'bgmCompare'
predict(
  object,
  newdata,
  group,
  variables = NULL,
  type = c("probabilities", "response"),
  method = c("posterior-mean"),
  ...
)
```

Arguments

object	An object of class bgmCompare.
newdata	A matrix or data frame with n rows and p columns containing the observed data. Must have the same variables (columns) as the original data used to fit the model.
group	Integer specifying which group's parameters to use for prediction (1 to number of groups). Required argument.
variables	Which variables to predict. Can be: • A character vector of variable names • An integer vector of column indices • NULL (default) to predict all variables
type	Character string specifying the type of prediction: "probabilities" Return the full conditional probability distribution for each variable and observation. "response" Return the predicted category (mode of the conditional distribution).
method	Character string specifying which parameter estimates to use: "posterior-mean" Use posterior mean parameters.
...	Additional arguments (currently ignored).

Details

Group-specific parameters are obtained by applying the projection matrix to convert baseline parameters and differences into group-level estimates. The function then computes the conditional distribution of target variables given the observed values of all other variables.

Value

For type = "probabilities": A named list with one element per predicted variable. Each element is a matrix with n rows and num_categories + 1 columns containing $P(X_j = c|X_{-j})$ for each observation and category.

For type = "response": A matrix with n rows and length(variables) columns containing predicted categories.

See Also

predict.bgms for predicting from single-group models, simulate.bgmCompare for simulating from group-comparison models.

Examples

```
# Fit a comparison model
x <- Boredom[Boredom$language == "fr", 2:6]
y <- Boredom[Boredom$language != "fr", 2:6]
fit <- bgmCompare(x, y, chains = 2)

# Predict conditional probabilities using group 1 parameters
probs_g1 <- predict(fit, newdata = x[1:10, ], group = 1)

# Predict responses using group 2 parameters
pred_g2 <- predict(fit, newdata = y[1:10, ], group = 2, type = "response")
```

---

predict.bgms *Predict Conditional Probabilities from a Fitted bgms Model*

Description

Computes conditional probability distributions for one or more variables given the observed values of other variables in the data.

Usage

```
## S3 method for class 'bgms'
predict(
  object,
  newdata,
  variables = NULL,
  type = c("probabilities", "response"),
  method = c("posterior-mean", "posterior-sample"),
  ndraws = NULL,
  seed = NULL,
  ...
)
```

Arguments

object	An object of class bgms.
newdata	A matrix or data frame with n rows and p columns containing the observed data. Must have the same variables (columns) as the original data used to fit the model.
variables	Which variables to predict. Can be:

- A character vector of variable names
- An integer vector of column indices
- NULL (default) to predict all variables

type — Character string specifying the type of prediction:

"probabilities" Return the full conditional probability distribution for each variable and observation.

"response" Return the predicted category (mode of the conditional distribution).

method — Character string specifying which parameter estimates to use:

"posterior-mean" Use posterior mean parameters.

"posterior-sample" Average predictions over posterior draws.

ndraws — Number of posterior draws to use when method = "posterior-sample". If NULL, uses all available draws.

seed — Optional random seed for reproducibility when method = "posterior-sample".

... — Additional arguments (currently ignored).

Details

For each observation, the function computes the conditional distribution of the target variable(s) given the observed values of all other variables. This is the same conditional distribution used internally by the Gibbs sampler.

Value

For type = "probabilities": A named list with one element per predicted variable. Each element is a matrix with n rows and num_categories + 1 columns containing $P(X_j = c|X_{-j})$ for each observation and category.

For type = "response": A matrix with n rows and length(variables) columns containing predicted categories.

When method = "posterior-sample", probabilities are averaged over posterior draws, and an attribute "sd" is included containing the standard deviation across draws.

See Also

simulate.bgms for generating new data from the model.

Examples

```
# Fit a model
fit <- bgm(x = Wenchuan[, 1:5], chains = 2)

# Compute conditional probabilities for all variables
probs <- predict(fit, newdata = Wenchuan[1:10, 1:5])

# Predict the first variable only
probs_v1 <- predict(fit, newdata = Wenchuan[1:10, 1:5], variables = 1)
```

```
# Get predicted categories
pred_class <- predict(fit, newdata = Wenchuan[1:10, 1:5], type = "response")
```

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

Description

Minimal console output for 'bgmCompare' fit objects.

Usage

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

Arguments

`x`	An object of class 'bgmCompare'.
`...`	Ignored.

Value

Invisibly returns 'x'.

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

Description

Minimal console output for 'bgms' fit objects.

Usage

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

Arguments

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

Value

Invisibly returns 'x'.

simulate.bgmCompare *Simulate Data from a Fitted bgmCompare Model*

Description

Generates new observations from the Markov Random Field model for a specified group using the estimated parameters from a fitted bgmCompare object.

Usage

```
## S3 method for class 'bgmCompare'
simulate(
  object,
  nsim = 500,
  seed = NULL,
  group,
  method = c("posterior-mean"),
  iter = 1000,
  ...
)
```

Arguments

object	An object of class bgmCompare.
nsim	Number of observations to simulate. Default: 500.
seed	Optional random seed for reproducibility.
group	Integer specifying which group to simulate from (1 to number of groups). Required argument.
method	Character string specifying which parameter estimates to use:
	"posterior-mean" Use posterior mean parameters (faster, single simulation).
iter	Number of Gibbs iterations for equilibration before collecting samples. Default: 1000.
...	Additional arguments (currently ignored).

Details

Group-specific parameters are obtained by applying the projection matrix to convert baseline parameters and differences into group-level estimates: group_param = baseline + projection[group,] %*% differences.

The function then uses these group-specific interaction and threshold parameters to generate new data via Gibbs sampling.

Value

A matrix with nsim rows and p columns containing simulated observations for the specified group.

See Also

simulate.bgms for simulating from single-group models, predict.bgmCompare for computing conditional probabilities.

Examples

```
# Fit a comparison model
x <- Boredom[Boredom$language == "fr", 2:6]
y <- Boredom[Boredom$language != "fr", 2:6]
fit <- bgmCompare(x, y, chains = 2)

# Simulate 100 observations from group 1
new_data_g1 <- simulate(fit, nsim = 100, group = 1)

# Simulate 100 observations from group 2
new_data_g2 <- simulate(fit, nsim = 100, group = 2)
```

---

simulate.bgms *Simulate Data from a Fitted bgms Model*

Description

Generates new observations from the Markov Random Field model using the estimated parameters from a fitted bgms object.

Usage

```
## S3 method for class 'bgms'
simulate(
  object,
  nsim = 500,
  seed = NULL,
  method = c("posterior-mean", "posterior-sample"),
  ndraws = NULL,
  iter = 1000,
  cores = parallel::detectCores(),
  display_progress = c("per-chain", "total", "none"),
  ...
)
```

Arguments

object	An object of class bgms.
nsim	Number of observations to simulate. Default: 500.
seed	Optional random seed for reproducibility.

`method`	Character string specifying which parameter estimates to use: `"posterior-mean"` Use posterior mean parameters (faster, single simulation). `"posterior-sample"` Sample from posterior draws, producing one dataset per draw (accounts for parameter uncertainty). This method uses parallel processing when `cores` > 1.
`ndraws`	Number of posterior draws to use when `method = "posterior-sample"`. If `NULL`, uses all available draws.
`iter`	Number of Gibbs iterations for equilibration before collecting samples. Default: `1000`.
`cores`	Number of CPU cores for parallel execution when `method = "posterior-sample"`. Default: `parallel::detectCores()`.
`display_progress`	Character string specifying the type of progress bar. Options: `"per-chain"`, `"total"`, `"none"`. Default: `"per-chain"`.
`...`	Additional arguments (currently ignored).

Details

This function uses the estimated interaction and threshold parameters to generate new data via Gibbs sampling. When `method = "posterior-sample"`, parameter uncertainty is propagated to the simulated data by using different posterior draws. Parallel processing is available for this method via the `cores` argument.

Value

If `method = "posterior-mean"`: A matrix with `nsim` rows and `p` columns containing simulated observations.

If `method = "posterior-sample"`: A list of matrices, one per posterior draw, each with `nsim` rows and `p` columns.

See Also

`predict.bgms` for computing conditional probabilities, `simulate_mrf` for simulation with user-specified parameters.

Examples

```
# Fit a model
fit <- bgm(x = Wenchuan[, 1:5], chains = 2)

# Simulate 100 new observations using posterior means
new_data <- simulate(fit, nsim = 100)

# Simulate with parameter uncertainty (10 datasets)
new_data_list <- simulate(fit, nsim = 100, method = "posterior-sample", ndraws = 10)

# Use parallel processing for faster simulation
new_data_list <- simulate(fit, nsim = 100, method = "posterior-sample",
```

```
                    ndraws = 100, cores = 2)
```

`simulate_mrf` *Simulate Observations from an Ordinal MRF*

Description

'simulate_mrf()' generates observations from an ordinal Markov Random Field using Gibbs sampling with user-specified parameters.

Usage

```
simulate_mrf(
  num_states,
  num_variables,
  num_categories,
  pairwise,
  main,
  variable_type = "ordinal",
  baseline_category,
  iter = 1000,
  seed = NULL
)
```

Arguments

`num_states`	The number of states of the ordinal MRF to be generated.
`num_variables`	The number of variables in the ordinal MRF.
`num_categories`	Either a positive integer or a vector of positive integers of length `num_variables`. The number of response categories on top of the base category: `num_categories = 1` generates binary states.
`pairwise`	A symmetric `num_variables` by `num_variables` matrix of pairwise interactions. Only its off-diagonal elements are used.
`main`	A `num_variables` by `max(num_categories)` matrix of category thresholds. The elements in row `i` indicate the thresholds of variable `i`. If `num_categories` is a vector, only the first `num_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"`.

`baseline_category`
: An integer vector of length `num_variables` specifying the baseline_category category that is used for the Blume-Capel model (details below). Can be any integer value between `0` and `num_categories` (or `num_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`.

`seed`
: Optional integer seed for reproducibility. If `NULL`, a seed is generated from R's random number generator (so `set.seed()` can be used before calling this function).

Details

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.

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 baseline_category category, such as the neutral category in a Likert scale. The Blume-Capel model specifies the following quadratic model for the threshold parameters:

$$\mu_c = \alpha \times (c - r) + \beta \times (c - r)^2,$$

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 baseline_category category r, while $\beta > 0$ suggests a preference for responding in the baseline_category category.

Value

A `num_states` by `num_variables` matrix of simulated states of the ordinal MRF.

See Also

`simulate.bgms` for simulating from a fitted model.

Examples

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

Pairwise = matrix(0, nrow = num_variables, ncol = num_variables)
Pairwise[2, 1] = Pairwise[4, 1] = Pairwise[3, 2] =
  Pairwise[5, 2] = Pairwise[5, 4] = .25
Pairwise = Pairwise + t(Pairwise)
```

```
Main = matrix(0, nrow = num_variables, ncol = max(num_categories))

x = simulate_mrf(
  num_states = 1e3,
  num_variables = num_variables,
  num_categories = num_categories,
  pairwise = Pairwise,
  main = Main
)

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

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

Main = matrix(NA, num_variables, num_categories)
Main[, 1] = -1
Main[, 2] = -1
Main[3, ] = sort(-abs(rnorm(4)), decreasing = TRUE)
Main[5, ] = sort(-abs(rnorm(4)), decreasing = TRUE)

x = simulate_mrf(
  num_states = 1e3,
  num_variables = num_variables,
  num_categories = num_categories,
  pairwise = Pairwise,
  main = Main,
  variable_type = c("b", "b", "o", "b", "o"),
  baseline_category = 2
)
```

---

summary.bgmCompare *Summary method for 'bgmCompare' objects*

Description

Returns posterior summaries and diagnostics for a fitted 'bgmCompare' model.

Usage

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

Arguments

object	An object of class ‘bgmCompare‘.
...	Currently ignored.

Value

An object of class ‘summary.bgmCompare‘ with posterior summaries.

---

summary.bgms *Summary method for ‘bgms‘ objects*

Description

Returns posterior summaries and diagnostics for a fitted ‘bgms‘ model.

Usage

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

Arguments

object	An object of class ‘bgms‘.
...	Currently ignored.

Value

An object of class ‘summary.bgms‘ with posterior summaries.

Wenchuan *PTSD Symptoms in Wenchuan Earthquake Survivors Who Lost a Child*

Description

This dataset contains responses to 17 items assessing symptoms of post-traumatic stress disorder (PTSD) in Chinese adults who survived the 2008 Wenchuan earthquake and lost at least one child in the disaster (McNally et al. 2015). Participants completed the civilian version of the Posttraumatic Checklist, with each item corresponding to a DSM-IV PTSD symptom. Items were rated on a 5-point Likert scale from "not at all" to "extremely," indicating the degree to which the symptom bothered the respondent in the past month.

Usage

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

Format

A matrix with 362 rows and 17 columns. Each row represents a participant.

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

https://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