Package ‘bgms’

August 4, 2026

Type Package

Title Bayesian Analysis of Graphical Models

Version 0.2.0.0

Date 2026-08-03

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

Description Bayesian estimation and edge selection for graphical models of mixed binary, ordinal, and continuous variables. The variable types determine the model: an ordinal Markov random field for discrete data, a Gaussian graphical model for continuous data, or a mixed Markov random field combining both. Edge inclusion is determined through spike-and-slab priors, yielding posterior inclusion probabilities for each edge. Supports multi-group comparison via 'bgmCompare()', simulation, prediction, and missing data imputation.

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, S7, methods, lifecycle, graphics, grDevices, parallel, stats, utils

RdMacros Rdpack

LinkingTo Rcpp, RcppArmadillo, RcppParallel, dqrng, BH

Depends R (>= 3.5)

LazyData true

Encoding UTF-8

Suggests coda, covr, knitr, MASS, qgraph, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 8.0.0

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 2026-08-04 03:00:02 UTC

Contents

ADHD . . . 3
bernoulli_prior . . . 5
beta_bernoulli_prior . . . 5
beta_prime_prior . . . 6
bgm . . . 7
bgmCompare . . . 16
Boredom . . . 22
calibration_check . . . 23
cauchy_prior . . . 25
coef.bgmCompare . . . 26
coef.bgms . . . 27
exponential_prior . . . 28
extract_arguments . . . 29
extract_category_thresholds . . . 29
extract_centrality . . . 30
extract_ess . . . 32
extract_group_params . . . 33
extract_inclusion_bf . . . 34
extract_indicators . . . 35
extract_indicator_priors . . . 36
extract_log_odds . . . 37
extract_main_effects . . . 38
extract_pairwise_interactions . . . 39
extract_partial_correlations . . . 40
extract_posterior_inclusion_probabilities . . . 41
extract_precision . . . 42
extract_prior_inclusion_probabilities . . . 43
extract_rhat . . . 45
extract_sbm . . . 46

gamma_prior . . . 46
normal_prior . . . 47
plot.bgmCompare . . . 48
plot.bgms . . . 51
plot.bgms_calibration . . . 53
plot.bgms_centrality . . . 54
plot.bgms_prior_sensitivity . . . 55
plot_edge_posterior . . . 56
predict.bgmCompare . . . 58
predict.bgms . . . 59
print.bgmCompare . . . 62
print.bgms . . . 62
print.bgms_calibration . . . 63
print.bgms_prior_sensitivity . . . 64
print.bgms_verdicts . . . 65
prior_sensitivity_check . . . 65
sample_ggm_prior . . . 69
sample_graph_prior . . . 72
sample_sbm_prior . . . 75
sbm_prior . . . 76
simulate.bgmCompare . . . 77
simulate.bgms . . . 78
simulate_mrf . . . 80
summarize_zratio_gauge . . . 83
summary.bgmCompare . . . 87
summary.bgms . . . 88
summary.bgms_centrality . . . 89
verdicts . . . 90
Wenchuan . . . 92
$.bgmCompare . . . 93
$.bgms . . . 94

Index **95**

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 data frame 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.

bernoulli_prior *Bernoulli Prior for Inclusion Indicators*

Description

Specifies a Bernoulli prior for inclusion indicators with a fixed inclusion probability. Used for edge selection in `bgm` and difference selection in `bgmCompare`.

Usage

```
bernoulli_prior(inclusion_probability = 0.5)
```

Arguments

`inclusion_probability`
: Numeric scalar or symmetric matrix. Prior probability of each edge being included. A scalar applies to all edges; a matrix allows edge-specific probabilities. Must be in (0, 1). Default: `0.5`.

Value

An object of class `"bgms_indicator_prior"` with `family = "Bernoulli"`.

See Also

`beta_bernoulli_prior`, `sbm_prior`, `bgm`

Other prior-constructors: `beta_bernoulli_prior()`, `beta_prime_prior()`, `cauchy_prior()`, `exponential_prior()`, `gamma_prior()`, `normal_prior()`, `sbm_prior()`

Examples

```
bernoulli_prior()
bernoulli_prior(inclusion_probability = 0.25)
```

beta_bernoulli_prior *Beta-Bernoulli Prior for Inclusion Indicators*

Description

Specifies a Beta-Bernoulli prior for inclusion indicators. The inclusion probability is drawn from a Beta(α, β) distribution and shared across all edges.

Usage

```
beta_bernoulli_prior(alpha = 1, beta = 1)
```

Arguments

alpha	Positive numeric. First shape parameter of the Beta distribution. Default: 1.
beta	Positive numeric. Second shape parameter of the Beta distribution. Default: 1.

Value

An object of class "bgms_indicator_prior" with family = "Beta-Bernoulli".

See Also

bernoulli_prior, sbm_prior, bgm

Other prior-constructors: bernoulli_prior(), beta_prime_prior(), cauchy_prior(), exponential_prior(), gamma_prior(), normal_prior(), sbm_prior()

Examples

```
beta_bernoulli_prior()
beta_bernoulli_prior(alpha = 2, beta = 5)
```

---

beta_prime_prior *Beta-Prime Prior for Model Parameters*

Description

Specifies a beta-prime prior on model parameters. The parameterization follows the logistic transformation: $\sigma(\mu) \sim \text{Beta}(\alpha, \beta)$, so $\mu = \text{logit}(Y)$ where $Y \sim \text{Beta}(\alpha, \beta)$.

Usage

```
beta_prime_prior(alpha = 0.5, beta = 0.5)
```

Arguments

alpha	Positive numeric. First shape parameter. Default: 0.5.
beta	Positive numeric. Second shape parameter. Default: 0.5.

Value

An object of class "bgms_parameter_prior" with family = "beta-prime".

See Also

cauchy_prior, normal_prior, bgm

Other prior-constructors: bernoulli_prior(), beta_bernoulli_prior(), cauchy_prior(), exponential_prior(), gamma_prior(), normal_prior(), sbm_prior()

Examples

```
beta_prime_prior()
beta_prime_prior(alpha = 1, beta = 1)
```

---

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

Description

The bgm function estimates the pseudoposterior distribution of the parameters of a Markov Random Field (MRF) for binary, ordinal, continuous, or mixed (discrete and continuous) variables. Optionally, it performs Bayesian edge selection using discrete spike-and-slab priors to infer the network structure.

Usage

```
bgm(
  x,
  variable_type = "ordinal",
  baseline_category,
  iter = 2000,
  warmup = 2000,
  interaction_prior = normal_prior(scale = 1),
  threshold_prior = beta_prime_prior(alpha = 0.5, beta = 0.5),
  means_prior = normal_prior(scale = 1),
  precision_scale_prior = exponential_prior(eta = 1),
  delta = NULL,
  edge_selection = TRUE,
  edge_prior = bernoulli_prior(0.5),
  precision_graph_prior = c("hierarchical", "joint"),
  na_action = c("listwise", "impute"),
  update_method = c("nuts", "adaptive-metropolis", "gibbs"),
  target_accept,
  nuts_max_depth = 10,
  learn_mass_matrix = TRUE,
  chains = 4,
  cores = parallel::detectCores(),
  display_progress = c("per-chain", "total", "none"),
  seed = NULL,
  verbose = getOption("bgms.verbose", TRUE),
  progress_callback = NULL,
  pairwise_scale,
  main_alpha,
  main_beta,
  inclusion_probability,
  beta_bernoulli_alpha,
```

```
  beta_bernoulli_beta,
  beta_bernoulli_alpha_between,
  beta_bernoulli_beta_between,
  dirichlet_alpha,
  lambda,
  interaction_scale,
  burnin,
  save,
  threshold_alpha,
  threshold_beta,
  standardize
)
```

Arguments

`x` A data frame or matrix with `n` rows and `p` columns. Columns may contain binary, ordinal, or continuous variables (see `variable_type`). Discrete variables are automatically recoded to non-negative integers (`0, 1, ..., m`); for regular ordinal variables, unobserved categories are collapsed, while Blume–Capel variables retain all categories. Continuous variables are column-centered internally so that the GGM likelihood is formulated with a zero-mean assumption.

`variable_type` Character or character vector. Specifies the type of each variable in `x`. Allowed values: `"ordinal"`, `"blume-capel"`, or `"continuous"`. A single string applies to all variables. A per-variable vector that mixes discrete (`"ordinal"` / `"blume-capel"`) and `"continuous"` types fits a mixed MRF. 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: `2e3`.

`warmup` Integer. Number of warmup iterations before collecting samples. Short warmups trigger progressive warnings (NUTS only); see `validate_sampler()` for the thresholds. With `update_method = "gibbs"` and edge selection, the first 15\ the warmup runs the full model (all edges included) so the precision matrix settles, and edge selection is active for the remaining 85\ both windows scale with the warmup budget. Default: `2e3`.

`interaction_prior` A prior specification object for pairwise interaction parameters, created by one of the prior constructor functions:

- `normal_prior()`: Normal(0, scale) prior (default).
- `cauchy_prior()`: Cauchy(0, scale) prior.
- `beta_prime_prior()`: Beta-prime prior.

Default: `normal_prior(scale = 1)`.

`threshold_prior` A prior specification object for threshold (main effect) parameters, created by one of the prior constructor functions:

- `beta_prime_prior()`: Beta-prime prior (default).
- `cauchy_prior()`: Cauchy(0, scale) prior.
- `normal_prior()`: Normal(0, scale) prior.

Default: `beta_prime_prior(alpha = 0.5, beta = 0.5)`.

`means_prior` A prior specification object for continuous variable means (mixed MRF models only), created by one of the prior constructor functions:

- `normal_prior()`: Normal(0, scale) prior (default).
- `cauchy_prior()`: Cauchy(0, scale) prior.
- `beta_prime_prior()`: Beta-prime prior.

Only used when the model includes continuous variables. Ignored for pure ordinal or pure continuous (GGM) models. Default: `normal_prior(scale = 1)`.

`precision_scale_prior`
A prior specification object for the diagonal elements of the precision matrix, created by one of:

- `exponential_prior()`: Exponential prior (default).
- `gamma_prior()`: Gamma prior.

Both constructors accept the rate in one of two frames: `rate` (raw) or `eta` (standardized). `eta` is the rate on the diagonal in the standardized frame, the coordinate in which the pairwise (slab) prior has unit scale; it fixes the scale of the diagonal relative to the slab, and the raw rate is derived at fit time as `eta / s`, where `s` is the scale of the `interaction_prior`. At fixed `eta`, graph and partial-correlation inference is invariant to the slab scale, so the standardized frame keeps the prior geometry fixed when the slab scale changes. It requires an `interaction_prior` with a scale parameter (`cauchy_prior()` or `normal_prior()`). Only used for models with continuous variables (GGM and mixed MRF). Ignored for pure ordinal models. Default: `exponential_prior(eta = 1)`.

`delta` Non-negative numeric, or `NULL` for the dimension- adaptive default. Determinant-tilt exponent on the continuous-block precision matrix K (GGM) or K_{yy} (mixed MRF): multiplies the prior by $|K|^{\delta}$, softly repelling the chain from the positive-definite cone boundary. `delta = NULL` (default) auto-resolves to $0.5\log(p)$ where p is the dimension of the continuous precision matrix (the number of variables for GGM, the number of continuous variables for mixed MRF). The rule is the simple form of the dimension-adaptive scaling $\delta(p) = c\log p$ with $c \in (0.3, 0.6)$ (Marsman et al., in preparation). Pass `delta = 0` for the untilted prior or a non-negative numeric to override. Both NUTS and adaptive-Metropolis update paths apply the tilt. Not allowed for pure ordinal models (no precision matrix to tilt).

`edge_selection` Logical. Whether to perform Bayesian edge selection. If `FALSE`, the model estimates all edges. Default: `TRUE`.

`edge_prior` An edge prior specification object, or a character string (deprecated). Specifies the prior for edge inclusion. Preferred: pass an object from one of:

- `bernoulli_prior()`: Fixed inclusion probability (default).
- `beta_bernoulli_prior()`: Beta-distributed inclusion.
- `sbm_prior()`: Stochastic Block Model.

Legacy character strings "Bernoulli", "Beta-Bernoulli", "Stochastic-Block" are still accepted but deprecated. Default: bernoulli_prior(0.5).

For continuous (GGM) and mixed models with beta_bernoulli_prior() or sbm_prior() under precision_graph_prior = "joint", the hyperparameter updates carry a normalizing-constant correction: under the determinant-tilted precision prior, the plain conjugate updates target the wrong marginals for the inclusion probability and the block structure. In mixed models the tilt acts on the continuous precision block, so the table is built for the continuous variables and the block-structure corrections read continuous-continuous edges only; with fewer than two continuous variables no edge is tilted and the plain conjugate updates apply unchanged. The correction table is built from the prior distribution at the first fit of a model configuration (a one-time cost of the order of minutes, announced when verbose = TRUE) and cached on disk via tools::R_user_dir("bgms", "cache"), so later fits of the same configuration skip the build. The default precision_graph_prior = "hierarchical" tracks the normalizer in the edge moves instead, so it needs no correction table and builds none. With beta_bernoulli_prior() the sampled inclusion probability is returned per chain in fit$inclusion_parameter_samples.

precision_graph_prior

Character. How the precision prior composes with the edge prior under edge selection for continuous (GGM) data:

"joint" The un-normalised joint specification $p(K, \Gamma) \propto \mathrm{slab}(K) \cdot \mathrm{diag}(K) \cdot |K|^{\delta} \cdot \mathbf{1}\{K \in \mathcal{M}^{+}(\Gamma)\} \cdot \pi(\Gamma)$. The graph marginal is $\pi(\Gamma) \cdot Z(\Gamma)$; with beta_bernoulli_prior() or sbm_prior() the hyperparameter updates carry the normalizing-constant correction described under edge_prior. The realized edge-inclusion prior therefore differs from the nominal edge prior for every edge prior: at three variables a uniform beta_bernoulli_prior(1, 1) realizes about 0.37, and a fixed bernoulli_prior(0.5) at $\delta = 0$ realizes about 0.27. The correction keeps the learned inclusion probability coherent with the joint model; it does not restore the nominal prior, and with a fixed inclusion probability nothing absorbs the tilt. A message reports this when verbose = TRUE; extract_prior_inclusion_probabilities returns the realized prior.

"hierarchical" (default) The hierarchical specification $p(\Gamma)\, p(K \mid \Gamma)$ with $p(K \mid \Gamma)$ normalized per graph, so the graph marginal is exactly the edge prior $\pi(\Gamma)$. Each edge move evaluates the normalizer ratio with a fast local approximation: a theta-independent absolute-moment surface built once at the start of the analysis from block-Gibbs anchors on components of up to 80 variables. Within that anchored range the surface tracks a block-Gibbs reference to about 0.003 nats. It is scored against that reference at five gamma_prior() diagonal shapes – 0.5, 1, 2, 3 and 5 – and deploys on the range those points span up to shape 10, with the interior interpolated rather than measured. At shape 10 the guarantee is of a different kind: the diagonal has concentrated far enough that the whole mediated correction is bounded by 0.00028 nats over the scored blocks at a standardized rate (eta) of 2 or below, so there is little left to approximate rather than a tight approximation of something large. That bound is set at the top of the scored rate range, since the correction grows with the rate, and does not extend above it. Above shape 10 that bound is what the fit deploys: the mediating correc-

tion is switched off and every edge is served the isolated-edge ratio, which is exact for an edge with no mediating structure, so the whole error is the mediation it drops – at most 0.00028 nats, measured at shapes 12, 15 and 20. The alternative there is not a coarser correction but a broken one: the additive kernel returns zero on a large common-neighbour block and discards the whole ratio. A shape below 0.5 keeps that additive kernel, and shares its limitation: on a fully connected common-neighbour block the additive ratio collapses to zero above a size that depends on the cell and on `delta`, measured between about 12 and 32 variables. Because `delta` defaults to $0.5 \log p$, small models are not exposed at all – the largest possible block sits below the boundary – and the collapse becomes reachable from roughly 20 to 30 variables upward. A fit reports it (`n_collapsed`). A mediating block beyond the anchored size range is predicted by continuing the surface along its own boundary slope, measured at blocks of 90 to 150 variables at a median of 0.0006 nats and at most 0.0011 for common-neighbour blocks, and a median of 0.0043 and at most 0.0060 for bipartite ones: bounded and far tighter than the alternatives, but outside the in-range figure, and reported by a note when a fit relies on it. A fit reaches that regime only with a dense posterior on many variables; sparse graphs never do. A trust gauge audits the approximation after sampling on two channels: the rate at which the chain's edge decisions would differ under the exact calculation, and the projected distortion of the inclusion probabilities from the measured error under the edge prior's feedback (`summarize_zratio_gauge`). The summary is returned as `fit$zratio_diag` and issues print like other sampler warnings. The gauge runs by default; `options(bgms.zratio_gauge_sweeps = 0L)` turns it off, and `fit$zratio_diag` is then `NULL`. Its cost is fixed per chain rather than proportional to `iter`: zero on a sparse posterior, where no mediating block is non-trivial and the ratio is exact, and about 5-10 seconds per chain on a dense posterior at 100-200 variables. That is negligible on a production-length fit and noticeable on a short exploratory one, which is what the off switch is for. Requires a `normal_prior()` or `cauchy_prior()` interaction prior; a `beta_prime_prior()` slab is rejected. On mixed data the normalizer lives on the continuous block K_{yy}, so the Z-ratio enters the continuous-continuous edge moves only, with the mediating-block counts read off the continuous subgraph; discrete and cross edges are unchanged.

Default: `"hierarchical"`, so a continuous fit targets the nominal edge prior out of the box and pays for the approximation and its trust gauge; `"joint"` is the un-approximated alternative and remains fully supported.

The two specifications differ only in how $p(K \mid \Gamma)$ is normalized across graphs, so they differ only where the sampler moves between graphs. Where it does not, `"hierarchical"` is accepted and the fit is the same under either value:

- With `edge_selection = FALSE` the graph is fixed, there is nothing to normalize across, and the two specifications coincide exactly. The correction surface and the trust gauge are skipped, so the fit does not pay to build them, and `fit$zratio_diag` is `NULL`.
- With no continuous precision block — an ordinal model, or mixed data with fewer than two continuous variables — there is no K for the argument to refer to. A message reports this when the argument was named and

verbose = TRUE; inheriting the default is silent, since the default is not a request.

The slab is checked only where the choice is meaningful, so a beta_prime_prior() is rejected on a continuous block under edge selection and tolerated where the argument has no referent.

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.

"nuts" The No-U-Turn Sampler, a gradient-based sampler available for all variable types, including under edge selection. Continuous precision blocks use a free-element Cholesky parameterization that keeps the precision matrix positive-definite and encodes excluded edges through per-column null-space coordinates.

"gibbs" A Gibbs sampler for the Gaussian graphical model, with a conjugate row-block draw of the precision matrix. Available only for all-continuous data with a Normal or Cauchy (slab) interaction prior and a Gamma scale prior on the precision diagonal. Edge selection is supported for both slabs; the graph is updated by a full-conditional birth/death between-step (a Cauchy slab uses its scale-mixture representation).

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 and 0.80 for NUTS.

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 (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. Sampling uses min(cores, chains); some computations outside of sampling (such as building the edge-selection prior correction table) use all cores. 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. On Windows with RcppParallel >= 6.0.0, a fixed seed reproduces a run only at the same cores setting: under the oneTBB 2022 scheduler, runs at different core counts are statistically equivalent but not bit-for-bit identical. Results are bit-reproducible for a given installed binary; they are not portable across compilers or optimization settings — a rebuilt binary may produce different draws from the same seed, with any differences concentrated where the posterior is weakly identified.

`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.

`progress_callback` An optional R function with signature `function(completed, total)` that is called at regular intervals during sampling, where `completed` is the number of iterations completed across all chains and `total` is the total number of iterations. Useful for external front-ends (e.g., JASP) that supply their own progress reporting. When `NULL` (the default), no callback is invoked.

`pairwise_scale` **[Deprecated]** Double. Scale of the Cauchy prior for pairwise interaction parameters. Use `interaction_prior` instead. Default: `1`.

`main_alpha`, `main_beta` **[Deprecated]** Double. Shape parameters of the beta-prime prior for threshold parameters. Use `threshold_prior` instead. Defaults: `main_alpha = 0.5` and `main_beta = 0.5`.

`inclusion_probability` **[Deprecated]** Numeric scalar. Use `edge_prior = bernoulli_prior(inclusion_probability)` instead. Default: `0.5`.

`beta_bernoulli_alpha`, `beta_bernoulli_beta` **[Deprecated]** Double. Use `edge_prior = beta_bernoulli_prior(alpha, beta)` instead. Defaults: `1`.

`beta_bernoulli_alpha_between`, `beta_bernoulli_beta_between` **[Deprecated]** Double. Use `edge_prior = sbm_prior(alpha_between, beta_between)` instead. Defaults: `1`.

`dirichlet_alpha` **[Deprecated]** Double. Use `edge_prior = sbm_prior(dirichlet_alpha = ...)` instead. Default: `1`.

`lambda` **[Deprecated]** Double. Use `edge_prior = sbm_prior(lambda = ...)` instead. Default: `1`.

`interaction_scale`, `burnin`, `save`, `threshold_alpha`, `threshold_beta` **[Deprecated]** Deprecated arguments as of **bgms 0.1.6.0**. Use `interaction_prior`, `warmup`, and `threshold_prior` instead.

`standardize` **[Deprecated]** Logical. Deprecated as of **bgms 0.2.0.0**. Through 0.1.6.3, `TRUE` adjusted each pair's interaction prior scale by the product of the two variables' maximum scores. Pairwise interactions are now on the association scale and share one prior scale, so the per-pair adjustment is gone: `standardize = FALSE` (the old default) warns and proceeds, while `standardize = TRUE` errors and points to setting the scale directly through `interaction_prior`.

Details

Depending on the variable types, the model is an ordinal MRF, a Gaussian graphical model (GGM), or a mixed MRF. Both regular ordinal variables and Blume–Capel ordinal variables (with a baseline category) are supported.

Edge selection uses spike-and-slab priors with Bernoulli, Beta-Bernoulli, or Stochastic-Block priors on the edge inclusion indicators. Parameters are sampled with NUTS (default) or adaptive

Metropolis–Hastings, with a multi-stage warmup schedule. Missing data can be handled via listwise deletion or Gibbs imputation.

For full details on model specification, prior choices, warmup, and output interpretation, see the package website at https://bayesian-graphical-modelling-lab.github.io/bgms-docs/.

Value

An S7 object of class bgms with posterior summaries, posterior mean matrices, and access to raw MCMC draws. Its fields are accessible with $ and [[for backward compatibility, and it 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 main-effect parameters. For OMRF models these are category thresholds; for mixed MRF models these are discrete thresholds and continuous means. NULL for GGM models (no main effects).
- posterior_summary_quadratic: Data frame with posterior summaries for the residual variance parameters (GGM and mixed MRF). NULL for OMRF models.
- posterior_summary_pairwise: Data frame with posterior summaries for partial association parameters.
- posterior_summary_indicator: Data frame with posterior summaries for edge inclusion indicators (if edge_selection = TRUE).
- posterior_mean_main: Posterior mean of main-effect parameters. NULL for GGM models. For OMRF: a matrix (p x max_categories) of category thresholds. For mixed MRF: a list with $discrete (threshold matrix) and $continuous (q x 1 matrix of means).
- posterior_mean_pairwise: Symmetric matrix of posterior mean partial associations (zero diagonal). For continuous variables these are half the negated precision off-diagonal, $\omega_{jk} = -\Theta_{jk}/2$; for discrete variables these are half the log adjacent-category odds ratio. Use extract_precision(), extract_partial_correlations(), or extract_log_odds() to convert to interpretable scales.
- posterior_mean_residual_variance: Named numeric vector of posterior mean residual variances $1/\Theta_{ii}$. Present for GGM and mixed MRF models; NULL for OMRF.
- 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_mean_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"`.

References

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

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.

See Also

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

Other model-fitting: `bgmCompare()`

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_family = c("Normal", "Cauchy"),
  difference_prior = bernoulli_prior(0.5),
  difference_probability,
  interaction_prior = normal_prior(scale = 1),
  threshold_prior = beta_prime_prior(alpha = 0.5, beta = 0.5),
  iter = 2000,
  warmup = 2000,
  na_action = c("listwise", "impute"),
  update_method = c("nuts", "adaptive-metropolis"),
  target_accept,
  nuts_max_depth = 10,
  learn_mass_matrix = TRUE,
  chains = 4,
  cores = parallel::detectCores(),
  display_progress = c("per-chain", "total", "none"),
  seed = NULL,
  verbose = getOption("bgms.verbose", TRUE),
  progress_callback = NULL,
  pairwise_scale,
  main_alpha,
  main_beta,
  beta_bernoulli_alpha,
  beta_bernoulli_beta,
  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,
  standardize
)
```

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. See the *Categories across groups* section for how the category codes of an ordinal variable are treated when the groups do not observe the same ones.

`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. Selection is per variable, not per category: one indicator gates a variable's entire block of category-threshold differences across all group contrasts, so the inclusion Bayes factor for a main-effect difference is a block test of whether that variable's thresholds differ between groups at all — a single category's difference is never selected on its own. 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 prior for difference parameters. Default: `1`.

difference_family
: Character. Distributional family of the prior on difference parameters, one of "Normal" (default) or "Cauchy". Governs both the pairwise-interaction differences and the main-effect (threshold) differences; under difference_selection = TRUE it is the slab of the spike-and-slab. Independent of interaction_prior, which governs the baseline interactions.

difference_prior
: An indicator prior specification object for difference selection, created by one of:

- bernoulli_prior(): Fixed inclusion probability (default).
- beta_bernoulli_prior(): Beta-distributed inclusion.
- sbm_prior(): Stochastic Block Model.

Legacy character strings "Bernoulli" and "Beta-Bernoulli" are still accepted but deprecated. Default: bernoulli_prior(0.5).

difference_probability
: **[Deprecated]** Numeric. Use difference_prior = bernoulli_prior(probability) instead. Default: 0.5.

interaction_prior
: A prior specification object for baseline pairwise interaction parameters, created by one of the prior constructor functions:

- normal_prior(): Normal(0, scale) prior (default).
- cauchy_prior(): Cauchy(0, scale) prior.

When supplied, overrides pairwise_scale. Default: normal_prior(scale = 1), matching bgm. Governs the baseline pairwise interactions only; the group differences are governed by difference_family and difference_scale.

threshold_prior
: A prior specification object for threshold (main effect) parameters, created by one of the prior constructor functions:

- beta_prime_prior(): Beta-prime prior (default).
- normal_prior(): Normal(0, scale) prior.

When supplied, overrides main_alpha and main_beta. Default: beta_prime_prior(alpha = 0.5, beta = 0.5).

iter
: Integer. Number of post–warmup iterations per chain. Default: 2e3.

warmup
: Integer. Number of warmup iterations before sampling. Default: 2e3.

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" or "nuts". Default: "nuts".

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

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

learn_mass_matrix
: Logical. If TRUE, adapts a diagonal mass matrix during warmup (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. Results are bit-reproducible for a given installed binary: the same seed on the same build yields identical draws. They are not portable across compilers or optimization settings — a rebuilt binary may produce different draws from the same seed, with any differences concentrated where the posterior is weakly identified.

`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.

`progress_callback`
An optional R function with signature `function(completed, total)` that is called at regular intervals during sampling, where `completed` is the number of iterations completed across all chains and `total` is the total number of iterations. Useful for external front-ends (e.g., JASP) that supply their own progress reporting. When `NULL` (the default), no callback is invoked.

`pairwise_scale` **[Deprecated]** Double. Scale of the baseline pairwise interaction prior. Retained for backward compatibility, it sets a Cauchy prior at that scale, which is not the current default. Use `interaction_prior` instead.

`main_alpha`, `main_beta`
[Deprecated] Doubles. Shape parameters of the beta-prime prior for baseline threshold parameters. Use `threshold_prior = beta_prime_prior(alpha, beta)` instead.

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

`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`
[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`, `interaction_prior`, `threshold_prior`, and `warmup` instead.

`standardize` **[Deprecated]** Logical. Deprecated as of **bgms 0.2.0.0**. Through 0.1.6.3, `TRUE` adjusted each pair's baseline and difference prior scale by the product of the two variables' maximum scores. Pairwise interactions are now on the association scale and share one prior scale, so the per-pair adjustment is gone: `standardize = FALSE` (the old default) warns and proceeds, while `standardize = TRUE` errors and points to setting the scales directly through `interaction_prior` and `difference_scale`.

Details

Group-specific parameters are decomposed into a shared baseline plus group differences that sum to zero. Difference selection uses spike-and-slab priors (Bernoulli or Beta-Bernoulli). Parameters are sampled with NUTS (default) or adaptive Metropolis–Hastings, using the same multi-stage warmup schedule as `bgm`.

Groups are numbered `1, 2, ...` in the order they first appear in `group_indicator`, whatever that vector's storage type: the first row's group is group 1, the first row belonging to some other group is group 2, and so on. (Note that this is first appearance, not sorted order – an indicator reading `c("fr", "fr", "en", ...)` makes `"fr"` group 1.) With `x` and `y` instead, `x` is group 1 and `y` is group 2. Every output keys on these numbers, and the extractor column names (`group1`, `group2`) stay numeric. The original labels are carried into the fit and shown on the displays a person reads – the `print` and `summary` headers, plot panel titles, calibration panel titles, and centrality labels – so that, for example, group 2 prints as `"group 2 (en)"`. Fits made with earlier versions of **bgms** carry no labels and display the bare numbers.

For full details on model specification, prior choices, and output interpretation, see the package website at `https://bayesian-graphical-modelling-lab.github.io/bgms-docs/`.

Value

An S7 object of class `bgmCompare` supporting list-style `$` / `[[` access for backward compatibility, containing posterior summaries, posterior mean matrices, and raw MCMC samples. Some `posterior_summary_*` fields are computed lazily on first access:

- `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"`.

Categories across groups

Groups being compared often differ in which categories of an ordinal variable they actually use — a clinical group may never give the lowest answer, a control group never the highest. `bgmCompare()` models the **union** of the categories observed across the groups: a category that any group uses is kept for all of them. Only a category value that *no* group uses is dropped, after which the remaining categories are renumbered contiguously from 0. That renumbering is reported with a `message()`; no category anyone observed is ever merged into another.

Keeping a category that some group never uses has a consequence worth knowing. That group contributes no observations to the category, so its data say nothing about where its threshold for that category lies, and the estimated difference for that group-by-category combination is determined by the prior rather than by the data: it will be large and very uncertain, and it is not evidence of a group difference. An unused *reference* category goes further: every threshold is measured relative to category 0, so a group that never used it has no data fixing the level of its threshold vector at all, and every one of that variable's threshold differences for that group rests on the prior. `bgmCompare()` raises a `warning()` naming every variable, category, and group this affects, and records the per-group category counts in the fitted object (`extract_arguments(fit)$category_support`) so they can be checked afterwards. The affected rows are also marked with a `*` in the printed summary; see `summary.bgmCompare()`. Under `main_difference_selection = TRUE` such a cell's difference is never tested alone: the inclusion indicator is per variable, so the block's Bayes factor pools the prior-driven cell with the variable's identified categories (see `main_difference_selection`). Only the category thresholds are affected this way; the pairwise (edge) parameters and their differences are estimated from all the data and are not.

Blume–Capel variables are exempt from all of this. Their two parameters are functions of the numeric category *score*, so renumbering the categories would change the model rather than relabel it, and a score no one happened to observe is still a meaningful point on the scale. Their categories are therefore always retained exactly as supplied.

References

There are no references for Rd macro `\insertAllCites` on this help page.

See Also

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

Other model-fitting: `bgm()`

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 data frame 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.

calibration_check *Calibration Check*

Description

Reliability diagrams of the model's conditional predictions, one per variable, with the consistency band a calibrated model would wander inside.

Usage

```
calibration_check(
  bgms_object,
  newdata = NULL,
  nrep = 200,
  probs = c(0.025, 0.975),
  grid_size = 101,
  seed = NULL,
  ndraws = 500,
  ...
)
```

Arguments

bgms_object	A fitted model object of class bgms (from bgm()) or bgmCompare (from bgmCompare()).
newdata	Optional data to evaluate the predictions on, in the layout the fitting function was given. Defaults to the data the model was fitted to. For a bgmCompare fit the rows must be the fitted cases in the order the data were given, because each case's group is read from the fit.
nrep	Number of resampled datasets behind the consistency band. Default 200.
probs	Numeric of length two; the band's quantiles. Default c(0.025, 0.975).
grid_size	Number of points the curves are read off on. Default 101.
seed	Optional integer seed for the band resampling.
ndraws	Posterior draws in the predictive mixture behind a continuous variable's panel. Ignored when every variable is discrete. Default 500.
...	Passed to methods.

Details

Discrete variables are checked against the categories they fall in. For every case and every category threshold the model issues a cumulative probability and the data record the outcome; isotonic regression (pool-adjacent-violators) estimates the observed frequency as a monotone function of the predicted one, and a calibrated variable tracks the diagonal. The band resamples each case's *category* from its own predicted distribution and refits the curve. It has to be built that way: the cumulative threshold events of one case are nested, so resampling the events independently would

understate the band and make an ordinary wander look like miscalibration. The predictions are the posterior-mean ones.

Continuous variables have a density rather than a distribution over categories, and are checked through the probability integral transform $u_{ij} = F_j(y_{ij} \mid y_{i,-j})$. The panel is the empirical distribution function of the u's against the uniform diagonal. F_j is the predictive mixture over `ndraws` posterior draws, so parameter uncertainty sits inside the distribution the observation is transformed by; the two panel kinds therefore differ in what they condition on, the isotonic one on posterior-mean probabilities and this one on the full predictive distribution. The band is the simultaneous envelope of the empirical distribution functions of n independent uniforms. It is not resampled from the model, because it does not have to be: whatever the conditional density was, a value drawn from it transforms to an exact $\text{Uniform}(0, 1)$, so the null is known and one band serves every continuous variable in the fit.

Both panels live on the unit square with the diagonal as the calibrated reference, so a mixed fit produces one figure and one summary table, with the `kind` column recording which construction produced each row.

Group comparisons. On a `bgmCompare()` fit the check runs per group: a group's cases are the ones its own parameters predict, so pooling them would let a variable predicted too high in one group cancel against the other, exactly as pooling variables would. Every variable is discrete there, so every panel is isotonic, and the `group` argument of the `bgmCompare` method selects which groups to check (default: all). `predict.bgmCompare()` issues posterior-mean predictions, which is what the isotonic curve conditions on anyway, so `ndraws` has no effect.

Evaluated on the fitted data the check is in-sample, and the band is the reference a model that is calibrated by construction produces on the same data. In-sample the observation also entered the parameters it is judged against, so the transform is mildly under-dispersed and the check is conservative. Supplying held-out `newdata` removes that and makes the check strictly harder to pass.

Calibrated conditional predictions do not imply that the model reproduces the joint distribution: a model can predict each variable well from the others and still understate how strongly they depend on one another. That second question is answered by a display built on `simulate()`, not by this check.

Value

An object of class `bgms_calibration`, a list with:

curves A data frame with one row per variable and grid point: `variable`, the panel `kind` (`"pav"` or `"pit"`), `grid`, the fitted `curve`, and the band bounds `lower` and `upper`.

summary One row per variable, worst first: `kind`, `mean_dev` and `max_dev`, the mean and maximum absolute distance of the curve from the diagonal, and `share_outside_band`, the proportion of the grid points at which the curve lies outside its consistency band, on a 0 to 1 scale rather than a percentage.

nrep, probs, grid, ndraws The settings the check ran under.

For a `bgmCompare()` fit both tables carry an extra `group` column and the object a `groups` element.

See Also

predict.bgms() for the conditional predictions themselves, simulate.bgms() for replicated datasets to build a joint-level display on

Other diagnostics: plot.bgms_calibration(), plot.bgms_prior_sensitivity(), print.bgms_calibration(), print.bgms_prior_sensitivity(), prior_sensitivity_check(), summarize_zratio_gauge()

Examples

```
fit = bgm(Wenchuan[, 1:5], display_progress = "none")
check = calibration_check(fit, nrep = 50)
check
plot(check)
```

---

cauchy_prior *Cauchy Prior for Model Parameters*

Description

Specifies a Cauchy(0, scale) prior on model parameters. This is the default prior for pairwise interactions in bgm and produces heavy-tailed shrinkage toward zero.

Usage

```
cauchy_prior(scale = 1)
```

Arguments

scale	Positive numeric. Scale (half-width at half-maximum) of the Cauchy distribution. Default: 1.

Value

An object of class "bgms_parameter_prior" with family = "cauchy".

See Also

normal_prior, beta_prime_prior, bgm

Other prior-constructors: bernoulli_prior(), beta_bernoulli_prior(), beta_prime_prior(), exponential_prior(), gamma_prior(), normal_prior(), sbm_prior()

Examples

```
cauchy_prior()
cauchy_prior(scale = 2.5)
```

`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.

See Also

`bgmCompare()`, `print.bgmCompare()`, `summary.bgmCompare()`

Other posterior-methods: `coef.bgms()`, `plot.bgmCompare()`, `plot.bgms()`, `plot_edge_posterior()`, `print.bgmCompare()`, `print.bgms()`, `print.bgms_verdicts()`, `summary.bgmCompare()`, `summary.bgms()`, `verdicts()`

Examples

```
# See ?bgmCompare for a full example
```

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

Description

Returns the posterior mean main effects, 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 main-effect parameters. `NULL` for GGM models (no main effects). For OMRF models this is a numeric matrix (p x max_categories) of category thresholds. For mixed MRF models this is a list with `$discrete` (p x max_categories matrix) and `$continuous` (q x 1 matrix of means).

pairwise Posterior mean of the partial association matrix (zero diagonal). Use `extract_precision()` for the full precision matrix.

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

mean_allocations Posterior mean block allocations. Present only for Stochastic-Block edge-prior fits.

mode_allocations Posterior mode block allocations. Present only for Stochastic-Block edge-prior fits.

num_blocks Data frame of the posterior distribution over the number of blocks. Present only for Stochastic-Block edge-prior fits.

See Also

`bgm()`, `print.bgms()`, `summary.bgms()`

Other posterior-methods: `coef.bgmCompare()`, `plot.bgmCompare()`, `plot.bgms()`, `plot_edge_posterior()`, `print.bgmCompare()`, `print.bgms()`, `print.bgms_verdicts()`, `summary.bgmCompare()`, `summary.bgms()`, `verdicts()`

Examples

```
fit = bgm(x = Wenchuan[, 1:3])
coef(fit)
```

`exponential_prior` *Exponential Prior for Scale Parameters*

Description

Specifies an Exponential prior for positive scale parameters. This is a convenience function equivalent to `gamma_prior(shape = 1)` with the same rate argument. As in `gamma_prior`, the rate can be given in the raw frame (`rate`) or the standardized frame (`eta`, the rate on the standardized diagonal at unit slab scale); supply one of the two, not both. With neither, the default is `eta = 1`.

Usage

```
exponential_prior(rate = NULL, eta = NULL)
```

Arguments

`rate`	Positive numeric. Rate parameter of the Exponential distribution in the raw frame. Mutually exclusive with `eta`.
`eta`	Positive numeric. Rate parameter of the Exponential distribution on the standardized diagonal (unit slab scale). Mutually exclusive with `rate`. Default when neither is supplied: `1`.

Value

An object of class `"bgms_scale_prior"` with `family = "exponential"`.

See Also

`gamma_prior`, `bgm`

Other prior-constructors: `bernoulli_prior()`, `beta_bernoulli_prior()`, `beta_prime_prior()`, `cauchy_prior()`, `gamma_prior()`, `normal_prior()`, `sbm_prior()`

Examples

```
exponential_prior() # standardized frame, eta = 1
exponential_prior(rate = 2) # raw frame
```

extract_arguments *Extract Model Arguments*

Description

Retrieves the arguments used when fitting a model with bgm() or bgmCompare().

Usage

```
extract_arguments(bgms_object)
```

Arguments

bgms_object	A fitted model object of class bgms (from bgm()) or bgmCompare (from bgmCompare()).

Value

A named list containing all arguments passed to the fitting function, including data dimensions, prior settings, and MCMC configuration.

For bgmCompare fits the list additionally carries main_effect_indices: an integer matrix with one row per variable and two columns giving the zero-based first and last column of that variable's block in the baseline main-effect parameters returned by extract_main_effects(). Variables do not occupy a fixed number of columns – an ordinal variable contributes one per category and a Blume-Capel variable two – so this layout is what maps parameter columns back to variables. It is not available for bgms fits, whose main effects are already returned as one row per variable.

See Also

bgm(), bgmCompare(), summary.bgms(), summary.bgmCompare()

Other extractors: extract_category_thresholds(), extract_centrality(), extract_ess(), extract_group_params(), extract_inclusion_bf(), extract_indicator_priors(), extract_indicators(), extract_log_odds(), extract_main_effects(), extract_pairwise_interactions(), extract_partial_correlatio extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabilitie extract_rhat(), extract_sbm(), plot.bgms_centrality(), summary.bgms_centrality()

extract_category_thresholds *Extract Category Threshold Estimates*

Description

[Deprecated]

extract_category_thresholds() was renamed to extract_main_effects() to reflect that main effects include continuous means and precision diagonal (mixed MRF), not only category thresholds.

Usage

```
extract_category_thresholds(bgms_object)
```

Arguments

bgms_object	A fitted model object of class bgms (from bgm()) or bgmCompare (from bgmCompare()).

Value

See extract_main_effects() for details.

See Also

extract_main_effects()

Other extractors: extract_arguments(), extract_centrality(), extract_ess(), extract_group_params(), extract_inclusion_bf(), extract_indicator_priors(), extract_indicators(), extract_log_odds(), extract_main_effects(), extract_pairwise_interactions(), extract_partial_correlations(), extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabiliti extract_rhat(), extract_sbm(), plot.bgms_centrality(), summary.bgms_centrality()

---

extract_centrality *Extract Posterior Centrality*

Description

Evaluates a node centrality on every posterior draw of the network, giving the posterior distribution of each node's centrality.

Usage

```
extract_centrality(bgms_object, measure = "strength", group = 1, ...)
```

Arguments

bgms_object	A fitted model object of class bgms (from bgm()) or bgmCompare (from bgmCompare()).
measure	Character; the centrality to evaluate. Currently "strength" (default), the sum of the absolute weights of a node's edges.
group	For a bgmCompare fit: a single group index, giving that group's centrality, or two indices, giving the difference in centrality between them (the first minus the second). Default 1. Ignored for a bgm fit, which has one network.
...	Passed to methods.

Details

The centrality is evaluated on the model-averaged pairwise draws, in which an edge excluded at a given iteration contributes exactly zero. Structural uncertainty therefore propagates into the centrality without any extra step: a node whose edges are themselves uncertain gets a wide centrality posterior, which is the honest summary. Conditioning on the included-only draws instead would report each node's centrality in the models where its edges happen to be present, which is a different and generally larger quantity.

Strength centrality sums the absolute edge weights, so positive and negative associations both add to a node's total involvement rather than cancelling.

For a `bgmCompare()` fit each group's network is rebuilt on every draw as `baseline + (P %*% differences)`, with the fit's own contrast projection `P`, rather than from posterior means. That is what carries the uncertainty through: with `group = c(1, 2)` the credible interval is the interval of the *difference* in a node's centrality, which answers whether the groups differ in it directly, where two separately drawn intervals do not. A draw in which a difference indicator is zero gives both groups the same edge weight and so contributes exactly zero to the difference, which is why this is computed per draw and not from summaries.

Read a centrality difference with care, and as numbers rather than as a picture (`summary()`; there is no plot method for it): strength sums absolute weights, so a difference of zero can mean identical networks or compensating edge differences, and its sign says nothing about which edges moved. `verdicts()` is the per-difference evidence.

Value

A numeric matrix of class `bgms_centrality` with one row per posterior draw and one column per variable, carrying the `measure` as an attribute. Use `summary()` for posterior means, credible intervals, and the probability of being the most central node, and `plot()` for the ordered interval display.

See Also

`extract_pairwise_interactions()` for the draws this is computed from, `verdicts()` for which of those edges the data settle

Other extractors: `extract_arguments()`, `extract_category_thresholds()`, `extract_ess()`, `extract_group_params()`, `extract_inclusion_bf()`, `extract_indicator_priors()`, `extract_indicators()`, `extract_log_odds()`, `extract_main_effects()`, `extract_pairwise_interactions()`, `extract_partial_correlatio`
`extract_posterior_inclusion_probabilities()`, `extract_precision()`, `extract_prior_inclusion_probabilitie`
`extract_rhat()`, `extract_sbm()`, `plot.bgms_centrality()`, `summary.bgms_centrality()`

Examples

```
fit = bgm(Wenchuan[, 1:5], display_progress = "none")
strength = extract_centrality(fit)
summary(strength)
```

extract_ess *Extract Effective Sample Size*

Description

Retrieves effective sample size estimates for all parameters from a model fitted with bgm() or bgmCompare().

Usage

```
extract_ess(bgms_object, estimator = c("rb", "mixt"))
```

Arguments

bgms_object	A fitted model object of class bgms (from bgm()) or bgmCompare (from bgmCompare()).
estimator	Character; which effective sample size to return for the edge (or difference) indicators. "rb" (default) returns the ESS of the Rao-Blackwellized inclusion draws. "mixt" **[Deprecated]** returns the indicator chain's transition-based ESS. Ignored for all other parameter types.

Details

The indicator element is the effective sample size of the inclusion inference, and also the n_eff column of the fit summary's inclusion table (summary(fit)$indicator): the continuous ESS of the Rao-Blackwellized inclusion draws, and therefore the ESS of the inclusion probability the fit reports (see extract_posterior_inclusion_probabilities(), which is Rao-Blackwellized by default). It is NA for indicators whose Rao-Blackwellized draws are constant to double precision, where the inclusion probability is at its numerical bound.

estimator = "mixt", the transition-based ESS of the binary indicator chain, is deprecated and no longer a summary column; a call recomputes it from the raw indicator draws and warns. It converts flip counts to an effective sample size through a two-state first-order Markov model that a substantial share of edge chains violate, and the precision of the inclusion probability and its Bayes factor is carried by the Rao-Blackwellized ESS and Monte Carlo standard error. The directional flip counts themselves (n0->1, n1->0) remain in the inclusion table.

Fits made with bgms < 0.2.0.0 carry no Rao-Blackwellized draws; a default call falls back to the transition ESS without warning, while an explicit estimator = "rb" errors.

Value

A named list with ESS values for each parameter type present in the model (e.g., main, pairwise, indicator).

See Also

bgm(), bgmCompare(), extract_rhat() for the Rao-Blackwellized indicator R-hat, extract_posterior_inclusion_prob

Other extractors: extract_arguments(), extract_category_thresholds(), extract_centrality(), extract_group_params(), extract_inclusion_bf(), extract_indicator_priors(), extract_indicators(), extract_log_odds(), extract_main_effects(), extract_pairwise_interactions(), extract_partial_correlatio extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabilitie extract_rhat(), extract_sbm(), plot.bgms_centrality(), summary.bgms_centrality()

---

extract_group_params *Extract Group-Specific Parameters*

Description

Computes group-specific parameter estimates by combining baseline parameters and group differences from a model fitted with bgmCompare().

Usage

```
extract_group_params(bgms_object)
```

Arguments

bgms_object	A fitted model object of class bgmCompare (from bgmCompare()).

Value

A list with elements main_effects_groups (main effects per group) and pairwise_effects_groups (pairwise effects per group).

See Also

bgmCompare(), extract_pairwise_interactions(), extract_main_effects()

Other extractors: extract_arguments(), extract_category_thresholds(), extract_centrality(), extract_ess(), extract_inclusion_bf(), extract_indicator_priors(), extract_indicators(), extract_log_odds(), extract_main_effects(), extract_pairwise_interactions(), extract_partial_correlatio extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabilitie extract_rhat(), extract_sbm(), plot.bgms_centrality(), summary.bgms_centrality()

extract_inclusion_bf *Extract Rao-Blackwellized Inclusion Bayes Factors*

Description

Computes inclusion Bayes factors from a model fitted with bgm() (edge inclusion) or bgmCompare() (difference inclusion), using the Rao-Blackwellized odds accumulators recorded during sampling. For each indicator the sampler sums the birth/death acceptance probability on the acceptance-probability scale, so the posterior inclusion odds follow from the exact identity

$$\frac{\bar{J}}{1-\bar{J}} = \frac{n_{01} + n_1 - n_{10}}{n_0 - n_{01} + n_{10}},$$

where n_{01} and n_{10} sum the acceptance probabilities of birth and death proposals and n_0, n_1 count them. Because $1 - \alpha$ is never formed per draw, the odds stay finite down to log acceptances of about -745, so edges that saturate the naive average of the RB draws (which rounds to 0 or 1 near the boundary) still receive a finite Bayes factor here.

The prior inclusion odds are removed edge by edge, so the returned value is the inclusion Bayes factor rather than the posterior odds: the two coincide only at a prior inclusion probability of $1/2$ (the default). For bgm() fits the prior odds come from extract_prior_inclusion_probabilities(); for continuous or stochastic-block models that call may run and cache a short prior-only chain. For bgmCompare() fits the exchangeable difference prior supplies a single prior inclusion probability (Bernoulli or Beta-Bernoulli); a stochastic-block difference prior has no single marginal, so the result is posterior odds there. The log argument applies to that return unchanged.

The accumulators are exact on the log scale everywhere, while the Bayes factor scale saturates at double precision: an entry whose log exceeds about 709.78 nats (a Bayes factor beyond about 1.8e308) is +Inf under log = FALSE even though its log-scale value is finite. Use log = TRUE for workflows that must separate such extreme evidence.

Usage

```
extract_inclusion_bf(bgms_object, log = FALSE)
```

Arguments

bgms_object	A fitted model object of class bgms (from bgm()) or bgmCompare (from bgmCompare()).
log	Logical. If FALSE (default), return inclusion Bayes factors; if TRUE, return their natural logarithm.

Value

A symmetric p x p matrix of inclusion Bayes factors, or of their natural logarithms when log = TRUE, with variable names as row and column names. Entries are NA for indicators that were never updated, +Inf when no exclusion evidence remains (denominator exactly zero), and -Inf when no inclusion evidence remains. On the Bayes factor scale the latter is 0, and +Inf also covers entries that overflow double precision. For bgms the diagonal is NA; for bgmCompare the diagonal holds main-effect difference Bayes factors.

"Never updated" means never *proposed*, which is narrower than it sounds. An edge indicator in bgm() that stays included for the whole run is still proposed at every iteration, so the accumulators see its conditional inclusion odds and it gets a finite (possibly very large) Bayes factor. An unselected main-effect difference in bgmCompare() (main_difference_selection = FALSE) is never proposed at all, so no Rao-Blackwellized quantity exists for it and NA is the honest entry rather than a lost number.

See Also

extract_posterior_inclusion_probabilities(), extract_prior_inclusion_probabilities()

Other extractors: extract_arguments(), extract_category_thresholds(), extract_centrality(), extract_ess(), extract_group_params(), extract_indicator_priors(), extract_indicators(), extract_log_odds(), extract_main_effects(), extract_pairwise_interactions(), extract_partial_correlatio extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabilitie extract_rhat(), extract_sbm(), plot.bgms_centrality(), summary.bgms_centrality()

Examples

```
fit = bgm(x = Wenchuan[, 1:3])
extract_inclusion_bf(fit)

# log = TRUE keeps evidence that saturates the Bayes factor scale readable.
extract_inclusion_bf(fit, log = TRUE)
```

---

extract_indicators *Extract Indicator Samples*

Description

Retrieves posterior samples of inclusion indicators from a model fitted with bgm() (edge inclusion indicators) or bgmCompare() (difference indicators).

Usage

```
extract_indicators(bgms_object)
```

Arguments

bgms_object	A fitted model object of class bgms (from bgm()) or bgmCompare (from bgmCompare()).

Value

A matrix with one row per post-warmup iteration and one column per indicator, containing binary (0/1) samples.

bgms One column per edge. Requires edge_selection = TRUE.

bgmCompare Columns for main-effect and pairwise difference indicators. Requires difference_selection = TRUE.

See Also

bgm(), bgmCompare(), extract_posterior_inclusion_probabilities()

Other extractors: extract_arguments(), extract_category_thresholds(), extract_centrality(), extract_ess(), extract_group_params(), extract_inclusion_bf(), extract_indicator_priors(), extract_log_odds(), extract_main_effects(), extract_pairwise_interactions(), extract_partial_correlatio extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabilitie extract_rhat(), extract_sbm(), plot.bgms_centrality(), summary.bgms_centrality()

---

extract_indicator_priors

Extract Indicator Prior Structure

Description

Retrieves the prior specification used for inclusion indicators in a model fitted with bgm() (edge indicators) or bgmCompare() (difference indicators).

Usage

```
extract_indicator_priors(bgms_object)
```

Arguments

bgms_object	A fitted model object of class bgms (from bgm()) or bgmCompare (from bgmCompare()).

Value

A named list describing the prior structure, including the prior type and any hyperparameters.

bgms Requires edge_selection = TRUE. Returns a list with the prior type ("Bernoulli", "Beta-Bernoulli", or "Stochastic-Block") and associated hyperparameters.

bgmCompare Requires difference_selection = TRUE. Returns the difference prior specification.

See Also

bgm(), bgmCompare(), extract_indicators()

Other extractors: extract_arguments(), extract_category_thresholds(), extract_centrality(), extract_ess(), extract_group_params(), extract_inclusion_bf(), extract_indicators(), extract_log_odds(), extract_main_effects(), extract_pairwise_interactions(), extract_partial_correlatio extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabilitie extract_rhat(), extract_sbm(), plot.bgms_centrality(), summary.bgms_centrality()

extract_log_odds *Extract Posterior Mean Log-Odds (Pairwise Interactions)*

Description

Retrieves the posterior mean pairwise interaction matrix for discrete variables from a model fitted with bgm(). These are the log-odds parameters of the discrete (Markov random field) block. GGM models have no discrete variables and return NULL.

Usage

```
extract_log_odds(bgms_object)
```

Arguments

bgms_object A fitted model object of class bgms (from bgm()).

Value

A named numeric matrix of posterior mean log-odds interactions, or NULL for GGM models.

OMRF A symmetric matrix with zero diagonal and one row and column per variable.

Mixed MRF A symmetric matrix with zero diagonal and one row and column per discrete variable.

GGM NULL (invisibly).

See Also

bgm(), extract_pairwise_interactions(), extract_precision()

Other extractors: extract_arguments(), extract_category_thresholds(), extract_centrality(), extract_ess(), extract_group_params(), extract_inclusion_bf(), extract_indicator_priors(), extract_indicators(), extract_main_effects(), extract_pairwise_interactions(), extract_partial_correlat extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabiliti extract_rhat(), extract_sbm(), plot.bgms_centrality(), summary.bgms_centrality()

Examples

```
fit = bgm(x = Wenchuan[, 1:3])
extract_log_odds(fit)
```

extract_main_effects *Extract Main Effect Estimates*

Description

Retrieves main-effect parameters from a model fitted with bgm() (posterior means) or bgmCompare() (posterior samples of baseline main effects). For OMRF models these are category thresholds; for mixed MRF models these include discrete thresholds and continuous means. GGM models have no main effects and return NULL.

Usage

```
extract_main_effects(bgms_object)
```

Arguments

bgms_object	A fitted model object of class bgms (from bgm()) or bgmCompare (from bgmCompare()).

Value

The structure depends on the model type:

GGM (bgms) NULL (invisibly). GGM models have no main effects; use extract_precision() to obtain the precision matrix.

OMRF (bgms) A numeric matrix with one row per variable and one column per category threshold, containing posterior means. Columns beyond the number of categories for a variable are NA.

Mixed MRF (bgms) A list with two elements:

- **discrete** A numeric matrix (p rows x max_categories columns) of posterior mean thresholds for discrete variables.
- **continuous** A numeric matrix (q rows x 1 column) of posterior mean continuous variable means.

bgmCompare A matrix with one row per post-warmup iteration, containing posterior samples of baseline main-effect parameters.

See Also

bgm(), bgmCompare(), extract_pairwise_interactions(), extract_category_thresholds()

Other extractors: extract_arguments(), extract_category_thresholds(), extract_centrality(), extract_ess(), extract_group_params(), extract_inclusion_bf(), extract_indicator_priors(), extract_indicators(), extract_log_odds(), extract_pairwise_interactions(), extract_partial_correlations extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabiliti extract_rhat(), extract_sbm(), plot.bgms_centrality(), summary.bgms_centrality()

Examples

```
fit = bgm(x = Wenchuan[, 1:3])
extract_main_effects(fit)
```

---

extract_pairwise_interactions

Extract Pairwise Interaction Samples

Description

Retrieves posterior samples of pairwise interaction parameters from a model fitted with bgm() or bgmCompare().

Usage

```
extract_pairwise_interactions(bgms_object)
```

Arguments

bgms_object	A fitted model object of class bgms (from bgm()) or bgmCompare (from bgmCompare()).

Value

A matrix with one row per post-warmup iteration and one column per edge, containing posterior samples of interaction strengths.

bgms Columns correspond to all unique variable pairs.

bgmCompare Columns correspond to the baseline pairwise interaction parameters.

See Also

bgm(), bgmCompare(), extract_main_effects()

Other extractors: extract_arguments(), extract_category_thresholds(), extract_centrality(), extract_ess(), extract_group_params(), extract_inclusion_bf(), extract_indicator_priors(), extract_indicators(), extract_log_odds(), extract_main_effects(), extract_partial_correlations(), extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabiliti
extract_rhat(), extract_sbm(), plot.bgms_centrality(), summary.bgms_centrality()

extract_partial_correlations

Extract Posterior Mean Partial Correlations

Description

Computes the posterior mean partial correlation matrix from a model fitted with bgm(). For GGM models this is the full matrix. For mixed MRF models this is the matrix for the continuous block. OMRF models have no partial correlations and return NULL.

Partial correlations are computed from the precision matrix as $\rho_{ij} = -\Theta_{ij}/\sqrt{\Theta_{ii}\Theta_{jj}}$.

Usage

```
extract_partial_correlations(bgms_object)
```

Arguments

bgms_object	A fitted model object of class bgms (from bgm()).

Value

A named numeric matrix containing posterior mean partial correlations, or NULL for OMRF models.

GGM A symmetric matrix with ones on the diagonal and one row and column per variable.

Mixed MRF A symmetric matrix with ones on the diagonal and one row and column per continuous variable.

OMRF NULL (invisibly).

See Also

bgm(), extract_precision()

Other extractors: extract_arguments(), extract_category_thresholds(), extract_centrality(), extract_ess(), extract_group_params(), extract_inclusion_bf(), extract_indicator_priors(), extract_indicators(), extract_log_odds(), extract_main_effects(), extract_pairwise_interactions(), extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabiliti
extract_rhat(), extract_sbm(), plot.bgms_centrality(), summary.bgms_centrality()

Examples

```
fit = bgm(
  x = Wenchuan[, 1:3],
  variable_type = rep("continuous", 3)
)
extract_partial_correlations(fit)
```

extract_posterior_inclusion_probabilities
Extract Posterior Inclusion Probabilities

Description

Computes posterior inclusion probabilities from a model fitted with bgm() (edge inclusion) or bgmCompare() (difference inclusion).

Two estimators of the same posterior inclusion probability are available through estimator. The default "rb" (Rao-Blackwellized) averages the one-step draw $J_t = \gamma_t + (1 - 2\gamma_t)\,\alpha_t$, where γ_t is the indicator state before the move and α_t is the acceptance probability of the birth/death proposal; "raw" instead averages the indicator draws. Averaging J_t is a lower-variance estimator and in exact arithmetic lies strictly inside $(0, 1)$, so even indicators whose raw average saturates at 0 or 1 receive an interior estimate. In double precision, however, the average of the stored J_t draws still rounds to exactly 0 or 1 for edges with overwhelming per-iteration evidence, because $1 - \alpha_t$ underflows once α_t drops below about 1e-16. For inclusion Bayes factors, use extract_inclusion_bf(), which accumulates the odds on the acceptance-probability scale and stays finite far beyond that ceiling; its log = TRUE return carries evidence beyond what the Bayes factor scale can represent in double precision. The "rb" estimator changes only the summary, not the sampler; it inherits the chain's mixing, does not rescue a chain that has failed to explore the model space, and requires a fit from bgms >= 0.2.0.0. Because the RB draw is continuous, the standard MCSE/ESS/split-R-hat machinery applies to it, and the fit summary's inclusion table reports mcse, n_eff, and Rhat on the RB draws. They quantify precision *conditional on exploration*: a stuck chain can show a beautifully converged J chain with a high n_eff, so read them beside the per-direction flip counts (n0->1, n1->0), which record the exploration itself and whose asymmetry no symmetric summary recovers. The RB draws vary on almost every edge, including edges whose indicator never flipped; the three columns are NA only where the RB draws are constant to double precision, which places the inclusion probability at its numerical bound and the verdict beyond any threshold. All of this is in the fit summary's inclusion table, summary(fit)$indicator.

Usage

```
extract_posterior_inclusion_probabilities(
  bgms_object,
  estimator = c("rb", "raw")
)
```

Arguments

bgms_object	A fitted model object of class bgms (from bgm()) or bgmCompare (from bgmCompare()).
estimator	Character; which estimator of the posterior inclusion probability to return. "rb" (default) returns the lower-variance Rao-Blackwellized average; "raw" averages the indicator draws.

Value

A symmetric p x p matrix of posterior inclusion probabilities, with variable names as row and column names.

bgms Off-diagonal entries are edge inclusion probabilities. Requires `edge_selection = TRUE`.

bgmCompare Diagonal entries are main-effect inclusion probabilities; off-diagonal entries are pairwise difference inclusion probabilities. Requires `difference_selection = TRUE`. With `estimator = "rb"`, indicators that were not selected (e.g. main-effect differences when `main_difference_selection = FALSE`) are returned as `NA`.

See Also

`extract_inclusion_bf()`, `bgm()`, `bgmCompare()`, `extract_indicators()`

Other extractors: `extract_arguments()`, `extract_category_thresholds()`, `extract_centrality()`, `extract_ess()`, `extract_group_params()`, `extract_inclusion_bf()`, `extract_indicator_priors()`, `extract_indicators()`, `extract_log_odds()`, `extract_main_effects()`, `extract_pairwise_interactions()`, `extract_partial_correlations()`, `extract_precision()`, `extract_prior_inclusion_probabilities()`, `extract_rhat()`, `extract_sbm()`, `plot.bgms_centrality()`, `summary.bgms_centrality()`

`extract_precision` *Extract Posterior Mean Precision Matrix*

Description

Retrieves the posterior mean precision matrix from a model fitted with `bgm()`. For GGM models this is the full precision matrix. For mixed MRF models this is the precision matrix of the continuous (Gaussian) block. OMRF models have no precision matrix and return `NULL`.

For mixed MRF models the precision matrix is reconstructed from the internal association-scale parameterization.

Usage

```
extract_precision(bgms_object)
```

Arguments

`bgms_object`	A fitted model object of class `bgms` (from `bgm()`).

Value

A named numeric matrix containing the posterior mean precision matrix, or `NULL` for OMRF models.

GGM A symmetric matrix with one row and column per variable.

Mixed MRF A symmetric matrix with one row and column per continuous variable.

OMRF `NULL` (invisibly).

See Also

bgm(), coef.bgms(), extract_partial_correlations()

Other extractors: extract_arguments(), extract_category_thresholds(), extract_centrality(), extract_ess(), extract_group_params(), extract_inclusion_bf(), extract_indicator_priors(), extract_indicators(), extract_log_odds(), extract_main_effects(), extract_pairwise_interactions(), extract_partial_correlations(), extract_posterior_inclusion_probabilities(), extract_prior_inclusion_p extract_rhat(), extract_sbm(), plot.bgms_centrality(), summary.bgms_centrality()

Examples

```
fit = bgm(
  x = Wenchuan[, 1:3],
  variable_type = rep("continuous", 3)
)
extract_precision(fit)
```

---

extract_prior_inclusion_probabilities

Extract Prior Inclusion Probabilities

Description

Returns the prior edge-inclusion probabilities of a model fitted with bgm() with edge_selection = TRUE, as a symmetric matrix in the same shape and orientation as extract_posterior_inclusion_probabilities(), so the two can be combined element-wise into prior and posterior inclusion odds.

Usage

```
extract_prior_inclusion_probabilities(
  bgms_object,
  iter = 4000L,
  warmup = 1000L,
  recompute = FALSE
)
```

Arguments

bgms_object	A fitted model object of class bgms from bgm() run with edge_selection = TRUE.
iter	Integer. Post-warmup iterations for the prior-only chain, when one is needed. Default 4000.
warmup	Integer. Warmup iterations for the prior-only chain. Default 1000.
recompute	Logical. Re-run the prior-only chain even when a cached estimate is present. Default FALSE.

Details

For models without a continuous block of at least two variables (ordinal MRFs, mixed models with a single continuous variable) the prior inclusion probability is the edge prior's marginal: the fixed probability for `bernoulli_prior()`, `alpha / (alpha + beta)` for `beta_bernoulli_prior()`, and the exchangeable partition mixture of the within- and between-block means for `sbm_prior()`.

Under `precision_graph_prior = "joint"` the spike-and-slab prior on a continuous block leaves $p(K, \Gamma)$ un-normalized, so the graph marginal is reweighted by the per-graph normalizer — the positive-definite-cone mass of the slab under the edge pattern, further shaped by the determinant tilt — so the prior inclusion probability of a continuous-continuous edge differs from the edge-prior marginal at any `delta`, including `delta = 0`. For `bernoulli_prior()` and `beta_bernoulli_prior()` it is read from the same cached normalizing-constant table that corrects the fit's hyperparameter updates (see `bgm()`); a fit whose slab family has no table (beta-prime) falls back to a prior-only chain. For `sbm_prior()` the reweighting also shifts the partition, and the probabilities are estimated by a prior-only chain run with the fit's own prior and correction settings; the estimate is cached on the fit, and `recompute = TRUE` re-runs it.

Under `precision_graph_prior = "hierarchical"` the conditional $p(K \mid \Gamma)$ is normalized per graph, so integrating K out returns $\pi(\Gamma)$ exactly and the continuous block carries the same edge-prior marginal as every other class. The tilted table and the prior-only chain do not apply, and neither is built.

In mixed models only continuous-continuous edges live in the joint block, so the matrix carries up to three distinct values: discrete-discrete, continuous-continuous, and cross edges. Variables are a priori exchangeable within type, which makes the probabilities constant within these classes.

Value

A symmetric matrix of prior inclusion probabilities with the variable names as row and column names and a zero diagonal, matching `extract_posterior_inclusion_probabilities()`.

See Also

`extract_posterior_inclusion_probabilities()`, `bgm()`

Other extractors: `extract_arguments()`, `extract_category_thresholds()`, `extract_centrality()`, `extract_ess()`, `extract_group_params()`, `extract_inclusion_bf()`, `extract_indicator_priors()`, `extract_indicators()`, `extract_log_odds()`, `extract_main_effects()`, `extract_pairwise_interactions()`, `extract_partial_correlations()`, `extract_posterior_inclusion_probabilities()`, `extract_precision()`, `extract_rhat()`, `extract_sbm()`, `plot.bgms_centrality()`, `summary.bgms_centrality()`

Examples

```
fit = bgm(x = Wenchuan[, 1:3])

# An ordinal MRF has no continuous block, so every edge carries the edge
# prior's own marginal -- 0.5 under the default bernoulli_prior(0.5).
extract_prior_inclusion_probabilities(fit)

# These are the odds extract_inclusion_bf() divides out.
extract_posterior_inclusion_probabilities(fit)
```

extract_rhat *Extract R-hat Convergence Diagnostics*

Description

Retrieves R-hat convergence diagnostics for all parameters from a model fitted with bgm() or bgmCompare().

Usage

```
extract_rhat(bgms_object)
```

Arguments

bgms_object	A fitted model object of class bgms (from bgm()) or bgmCompare (from bgmCompare()).

Details

The indicator element is the split-R-hat of the Rao-Blackwellized inclusion draws, matching the default of extract_ess(); the indicator chain's transition-based effective sample size remains available as extract_ess(fit, estimator = "mixt") (deprecated).

One failure mode these diagnostics may miss: chains can agree that an edge is included while disagreeing about the magnitude of its weight, for example when the weight's posterior has more than one mode and different chains settle in different ones. The indicator diagnostics cannot see this by construction, and the pooled-draw Rhat responds only weakly when inclusion is intermittent. When it matters, compare each edge's weight across chains using only the draws in which the edge is included; a dedicated diagnostic for this is planned for a future release.

Value

A named list with R-hat values for each parameter type present in the model (e.g., main, pairwise, indicator).

See Also

bgm(), bgmCompare(), extract_ess()

Other extractors: extract_arguments(), extract_category_thresholds(), extract_centrality(), extract_ess(), extract_group_params(), extract_inclusion_bf(), extract_indicator_priors(), extract_indicators(), extract_log_odds(), extract_main_effects(), extract_pairwise_interactions(), extract_partial_correlations(), extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabilities(), extract_sbm(), plot.bgms_centrality(), summary.bgms_centrality

extract_sbm *Extract Stochastic Block Model Summaries*

Description

Retrieves posterior summaries from a model fitted with the Stochastic Block prior. Works on both bgms fits (where SBM governs edge inclusion) and bgmCompare fits (where SBM governs the off-diagonal pairwise difference inclusions).

Usage

```
extract_sbm(bgms_object)
```

Arguments

bgms_object	A fitted model object of class bgms (from bgm()) or bgmCompare (from bgmCompare()).

Value

A list with elements posterior_num_blocks, posterior_mean_allocations, posterior_mode_allocations, and posterior_mean_coclustering_matrix. For bgms, requires edge_selection = TRUE and edge_prior = sbm_prior(...). For bgmCompare, requires difference_selection = TRUE and difference_prior = sbm_prior(...).

See Also

bgm(), bgmCompare(), extract_indicators(), extract_posterior_inclusion_probabilities()

Other extractors: extract_arguments(), extract_category_thresholds(), extract_centrality(), extract_ess(), extract_group_params(), extract_inclusion_bf(), extract_indicator_priors(), extract_indicators(), extract_log_odds(), extract_main_effects(), extract_pairwise_interactions(), extract_partial_correlations(), extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabilities(), extract_rhat(), plot.bgms_centrality(), summary.bgms_centrali

gamma_prior *Gamma Prior for Scale Parameters*

Description

Specifies a Gamma prior for positive scale parameters such as the diagonal elements of the precision matrix. The rate can be given in one of two frames:

- rate: the raw frame; the Gamma(shape, rate) prior applies to the diagonal as-is.
- eta: the standardized frame; eta is the Gamma rate on the standardized diagonal, the coordinate in which the pairwise (slab) prior has unit scale. eta fixes the scale of the diagonal relative to the slab, and the raw rate is derived at fit time as eta / s, where s is the scale of the interaction prior. At fixed eta, graph and partial-correlation inference is invariant to the slab scale s.

Supply either `rate` or `eta`, not both; with neither, the default is the standardized frame with `eta = 1`. Small `eta` places the prior mass well inside the positive-definite cone; large `eta` places mass near the cone boundary.

Usage

```
gamma_prior(shape = 1, rate = NULL, eta = NULL)
```

Arguments

`shape`	Positive numeric. Shape parameter of the Gamma distribution. Default: `1`.
`rate`	Positive numeric. Rate parameter of the Gamma distribution in the raw frame. Mutually exclusive with `eta`.
`eta`	Positive numeric. Rate parameter of the Gamma distribution on the standardized diagonal (unit slab scale). Mutually exclusive with `rate`. Default when neither is supplied: `1`.

Value

An object of class `"bgms_scale_prior"` with `family = "gamma"`.

See Also

`exponential_prior`, `bgm`

Other prior-constructors: `bernoulli_prior()`, `beta_bernoulli_prior()`, `beta_prime_prior()`, `cauchy_prior()`, `exponential_prior()`, `normal_prior()`, `sbm_prior()`

Examples

```
gamma_prior() # standardized frame, eta = 1
gamma_prior(shape = 2, eta = 0.5)
gamma_prior(shape = 2, rate = 0.5) # raw frame
```

`normal_prior` *Normal Prior for Model Parameters*

Description

Specifies a Normal(0, scale) prior on model parameters. Produces lighter-tailed shrinkage than the Cauchy prior and is better suited for simulation-based calibration (SBC) studies. Can be used for interactions, thresholds, or continuous means.

Usage

```
normal_prior(scale = 1)
```

Arguments

scale	Positive numeric. Standard deviation of the normal distribution. Default: 1.

Value

An object of class "bgms_parameter_prior" with family = "normal".

See Also

cauchy_prior, beta_prime_prior, bgm

Other prior-constructors: bernoulli_prior(), beta_bernoulli_prior(), beta_prime_prior(), cauchy_prior(), exponential_prior(), gamma_prior(), sbm_prior()

Examples

```
normal_prior()
normal_prior(scale = 0.5)
```

---

plot.bgmCompare *Plot a Fitted bgmCompare Model*

Description

Draws the group differences as three panels split by what the data settle about each pair, or the groups' own networks on a shared layout.

Usage

```
## S3 method for class 'bgmCompare'
plot(
  x,
  type = c("difference", "groups", "centrality"),
  evidence_threshold = 10,
  group = 1,
  layout = "spring",
  max_panels = 3L,
  page = 1L,
  ...
)
```

Arguments

x	A fitted model object of class bgmCompare, from bgmCompare().
type	Character; which display to draw. "difference" (default) is the difference evidence plot; "groups" draws each group's own network on one shared layout; "centrality" is the posterior strength centrality of extract_centrality().

`evidence_threshold`
: Numeric > 1; the inclusion Bayes factor separating evidence of a difference from undecided, as in `verdicts()`. Default `10`.

`group`
: For `type = "centrality"`: a single group index, passed to `extract_centrality()` for that group's centrality. A difference in centrality (two indices) can be extracted and summarized but has no plot; see `extract_centrality()` for the interpretation caveat. Default `1`.

`layout`
: Layout passed to `qgraph::qgraph()`. Default `"spring"`.

`max_panels`
: For `type = "groups"`: how many group networks are drawn at once. A fit with more groups than this is drawn a page at a time. Default `3`.

`page`
: For `type = "groups"`: which page of `max_panels` networks to draw. Default `1`.

`...`
: Passed to `qgraph::qgraph()`, or to `plot.bgms_centrality()` for `type = "centrality"`.

Details

The difference evidence plot. The default picture is about differences, because differences are what `bgmCompare()` parameterizes, and it is the display `plot.bgms()` uses, read for differences: three panels on one shared layout – the pairs whose difference the data support, the pairs whose difference the data rule out, and the pairs the data cannot decide. Each panel is titled with what it holds and how many pairs are in it, and with the rule that put them there, stated as a Bayes factor rather than its logarithm. The classification is the one `verdicts(fit)` reports at the same `evidence_threshold`.

Only the first panel is weighted: line width is the posterior mean difference and colour carries its sign – blue for a positive difference, vermillion for a negative one, the Okabe-Ito pair, so the sign survives common forms of colour-vision deficiency. Which group a positive difference favours follows the contrast coding, which `extract_group_params()` reports per group. The other two are drawn at uniform width, dashed for a difference the data rule out and dotted for undecided, because for those pairs the classification is the result.

The figure carries no key: the panel titles name the evidence class and the rule that defines it, and the sign convention is documented here.

"The groups do not differ anywhere" is a common and correct finding, and it is what a filled second panel and an empty first panel say.

Device size. Three networks side by side need a wide device. R's default 7 by 7 inches gives each panel a little over two inches of width, and at that size the node labels and the panel titles crowd. Open a wide device before plotting – `width = 13, height = 5` is a good starting point – or pass the same to whichever device the output is going to:

```
dev.new(width = 13, height = 5)   # or pdf(f, width = 13, height = 5)
plot(fit)
```

The single-panel and `type = "groups"` displays are content with the default device.

More than two groups. `bgmCompare()` gives each pair a single inclusion indicator shared across all `K - 1` contrasts, so the three-way split is exactly as well defined for `K > 2` as it is for two groups and the panels are the same. What changes is the width channel: a pair then has `K - 1` posterior mean differences rather than one, and no single number is "the" difference. The first panel is therefore drawn unweighted for `K > 2` – uniform width, plain ink, no sign colour – because there the

classification is the whole of what the panel reports. Read the magnitudes where they are per group: `plot(fit, type = "groups")` for the picture, `extract_group_params()` for the numbers.

The panels need the split to exist, and more than two groups *without* `difference_selection` is the one case where it does not. Selection off means weights are the display – as it does for a `bgm()` fit – and beyond two groups the weights of a pair are `K - 1` numbers rather than one, whose honest weighted picture is the groups themselves. `plot()` therefore draws the `type = "groups"` panels in that case, with the same layout, the same `max_panels` paging and the same passthrough to `qgraph::qgraph()`. `extract_group_params()` remains the way to read the differences as numbers.

Main-effect differences are not edges. When `main_difference_selection = TRUE` gave them their own indicators, their evidence is carried on the nodes: each node wears a ring filled to its difference indicator's posterior inclusion probability – a full ring is probability 1, half a ring 0.5 – coloured by the verdict (accented for presence, grey for undecided, faint for absence). The fill fraction carries the number, so the encoding does not rest on colour alone. Under the default `main_difference_selection = FALSE` those indicators do not exist and no ring is drawn. `verdicts()` remains the place to read main-effect differences precisely; the rings are a summary of it.

Without difference selection there is no indicator and so no inclusion Bayes factor to split the pairs by. `plot()` then draws one panel with every pair on it, width the posterior mean difference and colour its sign, titled `"Difference weights"` so that the channel the figure is drawn in is never in doubt. On more than two groups a pair has `K - 1` differences and no one of them is the weight to draw, so `plot()` draws the groups' own networks instead – the `type = "groups"` display, reached without asking for it, because with selection off that is what the weighted picture of such a fit is.

`type = "groups"` draws each group's own posterior mean network on the layout the difference display uses, so a node sits in the same place throughout and a reader compares by position. Those panels are estimates, not evidence, so every pair is drawn with its weight. A fit with more groups than `max_panels` is paged rather than squeezed. The difference evidence is the default display and is not repeated here.

Drawing needs the suggested package qgraph.

Value

`x`, invisibly. Called for the side effect of drawing.

See Also

The **easybgm** package for more extensive plotting and summary options around bgms fits; `verdicts()` for the table the panels encode, `extract_centrality()`, `prior_sensitivity_check()`

Other posterior-methods: `coef.bgmCompare()`, `coef.bgms()`, `plot.bgms()`, `plot_edge_posterior()`, `print.bgmCompare()`, `print.bgms()`, `print.bgms_verdicts()`, `summary.bgmCompare()`, `summary.bgms()`, `verdicts()`

Examples

```
fit = bgmCompare(
  x = Wenchuan[, 1:5],
  group_indicator = rep(1:2, length.out = nrow(Wenchuan)),
  display_progress = "none"
```

```
)
plot(fit)
plot(fit, type = "groups")
```

---

plot.bgms *Plot a Fitted bgms Model*

Description

Draws the network as three panels split by what the data settle about each pair, or one of the other standard displays.

Usage

```
## S3 method for class 'bgms'
plot(
  x,
  type = c("network", "centrality"),
  evidence_threshold = 10,
  layout = "spring",
  ...
)
```

Arguments

x	A fitted model object of class bgms, from bgm().
type	Character; which display to draw. "network" (default) is the edge evidence plot; "centrality" is the posterior strength centrality of extract_centrality().
evidence_threshold	Numeric > 1; the inclusion Bayes factor separating evidence of presence from undecided, as in verdicts(). Default 10.
layout	Layout passed to qgraph::qgraph(). Default "spring".
...	Passed to qgraph::qgraph() for type = "network", and to plot.bgms_centrality() otherwise.

Details

The edge evidence plot. A single network drawing has to make every pair either an edge or a blank, and a blank cannot say whether the data ruled the pair out or simply had too little to say about it. bgm() returns an inclusion Bayes factor for every pair, so that choice does not have to be made: the network is drawn as three panels on one shared layout – the pairs the data support, the pairs the data rule out, and the pairs the data cannot decide. Each panel is titled with what it holds and how many pairs are in it, and with the rule that put them there, stated as a Bayes factor rather than its logarithm. The classification is the one verdicts() reports at the same evidence_threshold.

Only the first panel is weighted. There, line width is the posterior mean pairwise association and colour carries its sign – blue for a positive association, vermillion for a negative one, the Okabe-Ito pair, so the sign survives common forms of colour-vision deficiency – because that is where the effect sizes are. The other two panels are drawn at uniform width, dashed for evidence of absence and dotted for undecided: for those pairs the classification is the result, and a width would suggest an effect size that the data have either ruled out or not established.

The figure carries no key. Each panel is titled with the evidence class it holds and the rule that defines it, which is what a key would otherwise repeat; the sign convention is documented here rather than reprinted on every figure.

The layout is computed once from every pair and reused, so a node sits in the same place in all three panels and a reader compares them by position.

Device size. Three networks side by side need a wide device. R's default 7 by 7 inches gives each panel a little over two inches of width, and at that size the node labels and the panel titles crowd. Open a wide device before plotting – `width = 13, height = 5` is a good starting point – or pass the same to whichever device the output is going to:

```
dev.new(width = 13, height = 5)   # or pdf(f, width = 13, height = 5)
plot(fit)
```

The single-panel display (no edge selection) is content with the default device.

A fit with no edge left in a panel is a result, not a failure. An all-absence fit fills the second panel and leaves the first empty, which is the honest picture of it.

Without edge selection there is no indicator and so no inclusion Bayes factor, and nothing to split the pairs by: the analysis is an estimation one. `plot()` then draws one panel with every pair on it, width the posterior mean association and colour its sign, and titles it `"Edge weights"` so that the channel the figure is drawn in is never in doubt.

Drawing the network requires the suggested package qgraph. The structure plots and the other network displays live in the easybgm package, which builds on these fits.

Value

x, invisibly. Called for the side effect of drawing.

See Also

The **easybgm** package for more extensive plotting and summary options around bgms fits; `verdicts()` for the table the panels encode, `extract_centrality()`, `plot_edge_posterior()` for one edge in detail

Other posterior-methods: `coef.bgmCompare()`, `coef.bgms()`, `plot.bgmCompare()`, `plot_edge_posterior()`, `print.bgmCompare()`, `print.bgms()`, `print.bgms_verdicts()`, `summary.bgmCompare()`, `summary.bgms()`, `verdicts()`

Examples

```
fit = bgm(Wenchuan[, 1:5], display_progress = "none")
plot(fit)
plot(fit, type = "centrality")
```

plot.bgms_calibration *Plot a Calibration Check*

Description

Reliability diagrams as small multiples, one panel per variable: the isotonic fit of observed frequency on predicted probability, its consistency band, and the diagonal a calibrated model tracks.

Usage

```
## S3 method for class 'bgms_calibration'
plot(x, variables = NULL, max_panels = 9L, page = 1L, ...)
```

Arguments

x	An object of class bgms_calibration, from calibration_check().
variables	Optional character vector selecting which variables to draw. Defaults to all, worst departure first.
max_panels	Number of panels drawn at once. A fit with more variables than this is drawn one page at a time, worst departure first. Default 9.
page	Which page of max_panels panels to draw. Default 1.
...	Ignored.

Details

Seventeen variables at once leave each panel too small to read, so the default draws the nine worst and reports how to reach the rest. variables selects panels by name; page walks through them a screen at a time.

Value

x, invisibly. Called for the side effect of drawing.

See Also

calibration_check()

Other diagnostics: calibration_check(), plot.bgms_prior_sensitivity(), print.bgms_calibration(), print.bgms_prior_sensitivity(), prior_sensitivity_check(), summarize_zratio_gauge()

Examples

```
fit = bgm(Wenchuan[, 1:5], display_progress = "none")
check = calibration_check(fit, nrep = 50)
plot(check)

# One variable at a time.
plot(check, variables = "intrusion")
```

---

plot.bgms_centrality *Plot Posterior Centrality*

Description

Draws each node's posterior mean centrality with its credible interval, nodes ordered by mean.

Usage

```
## S3 method for class 'bgms_centrality'
plot(x, probs = c(0.025, 0.975), ...)
```

Arguments

x	An object of class bgms_centrality, from extract_centrality().
probs	Numeric of length two; the credible-interval quantiles. Default c(0.025, 0.975).
...	Ignored.

Value

x, invisibly. Called for the side effect of drawing.

See Also

extract_centrality()

Other extractors: extract_arguments(), extract_category_thresholds(), extract_centrality(), extract_ess(), extract_group_params(), extract_inclusion_bf(), extract_indicator_priors(), extract_indicators(), extract_log_odds(), extract_main_effects(), extract_pairwise_interactions(), extract_partial_correlations(), extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabilities(), extract_rhat(), extract_sbm(), summary.bgms_centrality()

Examples

```
fit = bgm(Wenchuan[, 1:5], display_progress = "none")
plot(extract_centrality(fit))
```

plot.bgms_prior_sensitivity

Plot a Prior Sensitivity Check

Description

One panel, answer first: the label above the panel states how many edge verdicts depend on the slab scale. Each edge's natural log inclusion-Bayes-factor curve (the evidence for the edge) is drawn across the anchored scale range, with dots at the anchor scales; curve points masked for low importance ESS leave visible gaps, and an edge that saturates at some scale is capped at a large finite Bayes factor rather than running off to infinity. Edges whose verdict genuinely depends on the scale are colored and labeled by name; all other edges are the muted background. The shaded band is the undecided zone between the evidence thresholds; the zones are labeled at the left edge.

Usage

```
## S3 method for class 'bgms_prior_sensitivity'
plot(x, max_labels = 10L, ...)
```

Arguments

`x`	A `bgms_prior_sensitivity` object.
`max_labels`	Integer. Maximum scale-dependent edges to color and label by name; the rest are counted in a corner note. Default: `10`.
`...`	Ignored.

Details

The drawn curves clamp the pooled inclusion probability at $1-10^{-6}$, which caps a plotted log Bayes factor at $\ln(10^6) \approx 13.8$. A curve running flat along 13.8 has reached that display cap; it is not evidence levelling off. The uncapped value at the chosen scale is `x$edges$chosen_scale_log_bf`, which `print()` reports and which can be far larger. `x$edges$saturated` does not mark capped curves: it records that the edge's inclusion indicator never flipped in the chain, which is a different condition.

Value

`x`, invisibly. Called for the side effect of drawing.

See Also

`prior_sensitivity_check()`

Other diagnostics: `calibration_check()`, `plot.bgms_calibration()`, `print.bgms_calibration()`, `print.bgms_prior_sensitivity()`, `prior_sensitivity_check()`, `summarize_zratio_gauge()`

plot_edge_posterior *Plot the Posterior of One Edge Weight*

Description

Draws one edge's weight the way JASP draws a parameter: the posterior density against the prior it was updated from, with the evidence for the edge as a filled probability wheel and the estimate printed beside it.

Usage

```
plot_edge_posterior(
  bgms_object,
  variable1,
  variable2,
  binwidth = lifecycle::deprecated(),
  ...
)
```

Arguments

`bgms_object`	A fitted model object of class `bgms`, from `bgm()`.
`variable1, variable2`	The two variables naming the edge. Either names or column positions.
`binwidth`	**[Deprecated]** The panel no longer expresses the weight as probability per bin, so this has nothing to set; it is warned about and ignored.
`...`	Ignored.

Details

The panel shows two curves on one density scale. The solid accented curve is the posterior of the edge weight; the dashed grey curve is the prior it was updated from, computed in closed form from the fit's own `interaction_prior` rather than assumed, so a panel drawn from a `cauchy_prior(scale = 2.5)` fit and one drawn from the `normal_prior(scale = 1)` default do not look alike.

What the two curves are, and how the evidence is read off them, follows the model the fit actually used.

With edge selection (the default), the posterior of the weight is a spike-and-slab: a point mass at zero and a continuous part. The panel draws the continuous part *conditional on the edge being included*, which is a genuine density integrating to one, so the y axis carries numbers. The mass at zero is not drawn as a stem competing with that density; it is carried by the probability wheel, whose accented share is the posterior inclusion probability and whose pale share is the probability that the edge is absent. The evidence is printed as the natural log of the inclusion Bayes factor from `extract_inclusion_bf()`, which `bgms` estimates by Rao-Blackwellizing the indicator draws. It is *not* a ratio of densities at zero, so the panel does not draw Savage-Dickey ordinates: dots at zero would assert an estimator the package does not use.

Without edge selection there is no indicator and no point mass; the posterior of the weight is continuous and the Savage-Dickey density ratio is the licensed estimator of the inclusion Bayes factor. The panel then draws the JASP figure: both ordinates at zero are marked with grey dots, and their ratio – prior over posterior – is the Bayes factor for the edge, printed on the same natural-log scale. The wheel is filled by $BF/(1+BF)$, which is the posterior probability that the edge is there when the two hypotheses are equally likely before seeing the data; the filled share is that probability and the pale share its complement. JASP labels those two shares `data|H1` and `data|H0`; this panel does not, because the notation cannot be read without knowing the convention, and says it here instead.

The prior ordinate at zero is exact. The posterior ordinate is estimated from the draws by a Gaussian kernel density with the Sheather-Jones bandwidth (`stats::density(bw = "SJ")`), evaluated at zero; JASP's own implementations use a logspline fit, which `bgms` does not adopt because it would add a dependency for one number. The two estimators agree closely where the posterior is smooth near zero and diverge where it is not, which is the regime in which a Savage-Dickey Bayes factor is unreliable whatever fits its density.

On a block of continuous variables the slab sits on an entry of a precision matrix, whose joint prior is that slab times a prior on the diagonal, restricted to the positive-definite cone. The curve drawn is the slab – the density the sampler evaluates for that edge, and the one the prior sensitivity machinery reweights on – which is the marginal prior of the entry only up to that restriction. The restriction pulls the true marginal in, so on a continuous block the drawn prior is the wider of the two. On a discrete block the parameters are unconstrained and the slab is the marginal prior exactly.

The median and the 95% credible interval printed at the top right are of the same posterior the density shows – conditional on inclusion under edge selection, unconditional without it – and the interval is repeated as a bar above the curve.

An edge the data rule out is still drawn. When no retained draw included it there is no conditional posterior to show, so the panel draws the prior, the wheel (nearly all pale) and the evidence, and the missing accented curve is itself the statement: decisive absence is a result, not a failure.

The panel follows the package's plotting conventions (see the internal `R/plot_style.R`): offset axes, no box, large type, no headline title. It carries no caption; what each mark means is stated above rather than reprinted under every figure the panel draws. The edge is named as an annotation, where the rest of the numbers are.

No verdict word is printed. What the panel shows is the evidence – the wheel, the inclusion probability and the log Bayes factor – and the reading those license is the reader's to make at a threshold they choose; `verdicts()` is where the package states verdicts, and it names the threshold it used.

Value

`bgms_object`, invisibly. Called for the side effect of drawing.

See Also

`verdicts()`, `extract_pairwise_interactions()`, `extract_inclusion_bf()`

Other posterior-methods: `coef.bgmCompare()`, `coef.bgms()`, `plot.bgmCompare()`, `plot.bgms()`, `print.bgmCompare()`, `print.bgms()`, `print.bgms_verdicts()`, `summary.bgmCompare()`, `summary.bgms()`, `verdicts()`

Examples

```
fit = bgm(Wenchuan[, 1:5], display_progress = "none")
plot_edge_posterior(fit, "intrusion", "dreams")
```

`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.

Other prediction: `predict.bgms()`, `simulate.bgmCompare()`, `simulate.bgms()`, `simulate_mrf()`

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. Supports ordinal, Blume-Capel, continuous (GGM), and mixed MRF models.

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.

For GGM (continuous) models, the conditional distribution of $X_j|X_{-j}$ is Gaussian with mean $-\omega_{jj}^{-1}\sum_{k\neq j}\omega_{jk}x_k$ and variance ω_{jj}^{-1}, where Ω is the precision matrix.

Value

Ordinal models:

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.

GGM (continuous) models:

For `type = "probabilities"`: A named list with one element per predicted variable. Each element is a matrix with `n` rows and 2 columns (`"mean"` and `"sd"`) containing the conditional Gaussian parameters $E(X_j|X_{-j})$ and $\mathrm{SD}(X_j|X_{-j})$.

For `type = "response"`: A matrix with `n` rows and `length(variables)` columns containing conditional means.

When `method = "posterior-sample"`, conditional parameters are averaged over posterior draws, and an attribute `"sd"` is included.

Mixed MRF models:

For mixed models, the return list contains elements for both discrete and continuous predicted variables. Discrete variables return probability matrices (as in ordinal models); continuous variables return conditional mean and SD matrices (as in GGM models).

See Also

`simulate.bgms` for generating new data from the model.

Other prediction: `predict.bgmCompare()`, `simulate.bgmCompare()`, `simulate.bgms()`, `simulate_mrf()`

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.

See Also

bgmCompare(), summary.bgmCompare(), coef.bgmCompare()

Other posterior-methods: coef.bgmCompare(), coef.bgms(), plot.bgmCompare(), plot.bgms(), plot_edge_posterior(), print.bgms(), print.bgms_verdicts(), summary.bgmCompare(), summary.bgms(), verdicts()

Examples

```
# See ?bgmCompare for a full example
```

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.

See Also

bgm(), summary.bgms(), coef.bgms()

Other posterior-methods: coef.bgmCompare(), coef.bgms(), plot.bgmCompare(), plot.bgms(), plot_edge_posterior(), print.bgmCompare(), print.bgms_verdicts(), summary.bgmCompare(), summary.bgms(), verdicts()

Examples

```
fit = bgm(x = Wenchuan[, 1:3])
print(fit)
```

---

print.bgms_calibration *Print a Calibration Check*

Description

Prints the per-variable departure from the diagonal, worst variable first.

Usage

```
## S3 method for class 'bgms_calibration'
print(x, digits = 3, max_rows = 10L, ...)
```

Arguments

x	An object of class bgms_calibration, from calibration_check().
digits	Number of digits for the printed columns. Default 3.
max_rows	Number of variables to print. Default 10.
...	Ignored.

Value

x, invisibly.

See Also

calibration_check()

Other diagnostics: calibration_check(), plot.bgms_calibration(), plot.bgms_prior_sensitivity(), print.bgms_prior_sensitivity(), prior_sensitivity_check(), summarize_zratio_gauge()

---

print.bgms_prior_sensitivity

Print a Prior Sensitivity Check

Description

Prints the answer first: how many edge verdicts are robust to the slab scale, which edges genuinely depend on it (by name), which cannot be certified, the verdict counts per scale, and how the chosen scale compares with the size of the estimated interactions. Machinery (sampler, refit cost, the run-to-run noise band) is confined to a closing Details line.

Usage

```
## S3 method for class 'bgms_prior_sensitivity'
print(x, max_rows = 10L, ...)
```

Arguments

x	A bgms_prior_sensitivity object.
max_rows	Integer. Maximum edges to name in the scale-dependent table; the rest are counted and left to $edges. Default: 10.
...	Ignored.

Value

x, invisibly.

See Also

prior_sensitivity_check()

Other diagnostics: calibration_check(), plot.bgms_calibration(), plot.bgms_prior_sensitivity(), print.bgms_calibration(), prior_sensitivity_check(), summarize_zratio_gauge()

print.bgms_verdicts *Print Edge Verdicts*

Description

Prints the verdict tally, the leading rows of the verdict table, and a warning line when any verdict is Monte-Carlo fragile.

Usage

```
## S3 method for class 'bgms_verdicts'
print(x, digits = 3, max_rows = 10L, ...)
```

Arguments

x	An object of class bgms_verdicts, from verdicts().
digits	Number of digits for the printed numeric columns. Default 3.
max_rows	Number of rows to print. Default 10.
...	Ignored.

Value

x, invisibly.

See Also

verdicts()

Other posterior-methods: coef.bgmCompare(), coef.bgms(), plot.bgmCompare(), plot.bgms(), plot_edge_posterior(), print.bgmCompare(), print.bgms(), summary.bgmCompare(), summary.bgms(), verdicts()

prior_sensitivity_check *Prior Sensitivity of Inclusion Verdicts*

Description

Recovers each indicator's continuous inclusion-Bayes-factor curve $\mathrm{BF}_e(s)$ across the scale s of the slab that gates it, and classifies every verdict trajectory along it. The curve is anchored at a handful of fixed-scale fits — the chosen-scale anchor is the original fit itself — and filled in between anchors by importance reweighting. A built-in Monte Carlo noise guard, calibrated from a repeated refit at one anchor, keeps the check from crying wolf on boundary cases.

On a bgm() fit with edge selection the unit is the edge indicator and the swept prior is the interaction slab. On a bgmCompare() fit with difference selection the unit is the difference indicator and the swept prior is the difference slab (difference_scale), which covers the pairwise and the main-effect difference families alike, since bgmCompare() gives them one scale.

Usage

```
prior_sensitivity_check(
  bgms_object,
  anchors = c(0.4, 0.63, 1, 1.6, 2.5),
  evidence_threshold = 10,
  vary = c("auto", "slab", "slab-and-diagonal"),
  refit_sampler = "same-as-fit",
  iter = NULL,
  warmup = NULL,
  tolerance = 0.5 * log(10),
  ess_floor = 400,
  include_preferred_scale = FALSE,
  cores = NULL,
  seed = 1L,
  keep_fits = FALSE,
  verbose = FALSE
)
```

Arguments

`bgms_object` A fitted bgms object from `bgm()` run with `edge_selection = TRUE`, or a `bgmCompare` object from `bgmCompare()` run with `difference_selection = TRUE`.

`anchors` Numeric vector of positive anchor multipliers of the chosen scale. Default `c(0.4, 0.63, 1, 1.6, 2.5)` (log-spaced, so adjacent anchors' usable reweighting radii overlap). The multiplier `1` is always included and is the original fit; each other anchor is one warm refit.

`evidence_threshold` Positive numeric. Inclusion Bayes factor threshold for a presence verdict; `1 / evidence_threshold` is the absence threshold. Default: `10`.

`vary` One of `"auto"` (default), `"slab"`, or `"slab-and-diagonal"`. Only relevant for models with a prior on the precision diagonal (continuous and mixed); ignored for discrete ones, which have none. The slab scale s and the diagonal rate are tied through the standardized frame (raw rate $= \eta/s$), so a sweep of s has to hold one of the two fixed. `"slab"` holds the raw diagonal rate at the fitted value and moves the interaction prior alone, answering how much the verdicts depend on how wide an edge is allowed to be. `"slab-and-diagonal"` holds η fixed and lets the raw rate follow, answering how much they depend on the overall prior scale with its shape held fixed. `"auto"` follows the frame the fit itself used: `"slab-and-diagonal"` when the diagonal prior was given as `eta`, `"slab"` when it was given as a raw `rate`. The resolved mode is named in the printed report.

`refit_sampler` One of `"same-as-fit"` (default; inherit the original fit's update method) or an explicit `"nuts"`, `"adaptive-metropolis"`, or `"gibbs"`. NUTS refits of an ordinal fit carry the adapted metric and run fastest; when an inherited slower sampler makes the check cost more than about a minute, a message suggests the switch. Refitting with a different sampler than the original fit is sound because every comparison the check makes runs refit-against-refit under one identical pipeline; the original fit never enters a comparison.

`iter`, `warmup` Integer sampling and warmup iterations per refit, or `NULL` (default) to use the validated short schedule for warm NUTS refits and inherit the original fit's schedule otherwise.

`tolerance` Numeric. Minimum natural log BF change across scales for a verdict flip to count as a move, before the MCSE and noise floors. Default: `0.5 * log(10)` (about 1.15, the same decision boundary the former 0.5 carried in $\log_{10}$ units).

`ess_floor` Positive numeric. Minimum pooled importance effective sample size for a curve point to be reported; points below it are `NA`. Default: `400`.

`include_preferred_scale` Logical. Add an extra anchor at the data-preferred scale $\hat{s}$. Default: `FALSE`.

`cores` Integer thread count for each refit's chains. Default: the original fit's core count.

`seed` Integer base seed for the refits. Default: `1`.

`keep_fits` Logical. Retain the full refit objects in the result (for power users); the default keeps only per-scale summaries. Default: `FALSE`.

`verbose` Logical. If `TRUE`, print each internal refit's raw sampler notes (energy, tree-depth, dropped chains) live as it runs. By default these are captured, not printed: the convergence gate adjudicates them, a failed anchor is reported once in plain language, and the raw text stays available in `$refit_diagnostics`. Default: `FALSE`.

Details

The anchored curve. The model is refit at the non-unit `anchors` (multipliers of the chosen scale s_0; the `1x` anchor is the original fit and is never refit). A fit at fixed anchor scale s_a is reweighted to a nearby scale s with per-draw slab-density ratios over the currently included edges; the likelihood cancels, so no refit is needed between anchors. The posterior-density-ratio reweighting identity of Bartos et al. (2026) is used locally around each refit anchor; the anchored construction itself is specific to **bgms**. Each point of a dense log-spaced display grid pools every anchor that clears `ess_floor` there, weighting each anchor's inclusion-probability estimate by its inverse variance ($\mathrm{ESS}/(p(1-p))$); the pooling is on the inclusion-probability scale and is then transformed to the natural log Bayes factor, which keeps the curve continuous across anchor hand-offs and finite at capped edges. Points where no anchor clears the floor are `NA` rather than extrapolated, and non-overlapping anchor radii trigger a warning to add anchors; log-spaced default anchors make the radii overlap. Exactness is kept off the pooled curve and on the anchor fits themselves: the per-anchor verdict columns and every chosen-scale quantity are read straight from each fit's own Rao-Blackwellized statistics, so the `1x` column is exactly the original fit's reported analysis. Between the anchors the curve is therefore importance-reweighted rather than refit, and can deviate from a refit at that scale by up to roughly `0.01` in inclusion probability at the extrapolation ends; the anchors themselves – including the `1x` anchor, which is the user's own fit – are exact.

Warm starts. For ordinal (omrf) fits each refit starts from the original fit's per-chain final state, and a NUTS refit additionally carries the adapted step size and diagonal mass matrix, so a short warmup suffices and the whole check costs about one original fit. Because the warm starts sit near the chosen-scale posterior, cross-chain dispersion is reduced by construction, which weakens split-$\hat{R}$ as a between-chain diagnostic; the refit gate therefore leans on per-chain verdict agreement and the indicator transition ESS, not on $\hat{R}$ alone. Continuous (GGM) and mixed-MRF fits refit cold (full warmup), costing about one fit per scale.

The mover rule. An edge is flagged scale-sensitive only if its verdict differs somewhere along the curve *and* its natural log Bayes-factor change across scales exceeds `max(tolerance, 2 * MCSE, noise)`, with the per-point MCSE from the chain-level spread of the reweighted estimate (so it carries both the between-chain and the importance-sampling uncertainty) and the noise band the 95th percentile of the spread between one anchor refit and its repeat, over threshold-relevant edges ($|\log \mathrm{BF}| \leq 3 \log 10 \approx 6.91$; near-saturated edges would inflate it). An edge that is threshold-relevant in one of the two refits and saturated in the other has a censored rather than an infinite spread, and is left out of that percentile; the printed report counts them. When no threshold-relevant edge has a measurable spread the noise band is `NA` and the mover rule falls back to `max(tolerance, 2 * MCSE)`. The `$edges$mover` column stores `stable`, `indistinguishable-from-wobble`, or `moved-beyond-wobble`; the printed report shows the same categories in plain language ("robust", "changed, within run-to-run noise", "changed, beyond run-to-run noise", with edges failing the sufficiency check below printed as "not certifiable"). A bare verdict flip inside the replicate noise is never reported as a move.

Edge-level sufficiency. An edge whose per-chain verdicts disagree, or whose between-chain-inflated log-BF band straddles a verdict threshold, is marked `insufficient` at that scale: the refit cannot certify its verdict. This errs toward caution — disagreement widens the band rather than vanishing into a pooled estimate.

Two-level gate. Each refit passes a refit-level gate (continuous split-$\hat{R}$ below 1.01, bulk RB-inclusion median $\hat{R}$ below 1.01, E-BFMI and energy variance-ratio for NUTS) before its verdicts are used; a refit that fails is reported as unusable rather than silently pooled.

Value

An object of class `"bgms_prior_sensitivity"`: a list with the per-edge `edges` table (chosen-scale verdict from the original fit, per-anchor verdict columns, dense-grid stability range, mover category, an `insufficient` flag with its two subcauses `insufficient_noisy` (Bayes factor within Monte Carlo error of a threshold) and `insufficient_disagree` (chains disagree on the verdict)), a `grid` data frame (one row per anchor fit and the replicate, with convergence gates), the `multipliers` display grid with the `log_bf`, `log_bf_mcse`, and `verdict` curve matrices (grid-by-edge, `NA` where masked), the `curve` bookkeeping (per-point importance ESS, anchor used, `ess_floor`, per-point chain unanimity), the `wobble` noise yardstick, `refit_diagnostics` (the captured raw sampler notes per anchor and the replicate), the data-`preferred_scale` (`NA` where the swept prior is not the one on the pairwise interactions), the `unit` the check reported on, the resolved `vary` mode, and the settings used.

References

Bartoš F, Wagenmakers E, Marsman M, van den Bergh D (2026). "Efficient Bayes factor sensitivity analysis." *arXiv preprint*. arXiv:2604.21596.

See Also

`bgm()`, `extract_posterior_inclusion_probabilities()`

Other diagnostics: `calibration_check()`, `plot.bgms_calibration()`, `plot.bgms_prior_sensitivity()`, `print.bgms_calibration()`, `print.bgms_prior_sensitivity()`, `summarize_zratio_gauge()`

Examples

```
fit = bgm(Wenchuan[, 1:6], chains = 2)
ps = prior_sensitivity_check(fit)
ps
plot(ps)
```

`sample_ggm_prior` *Sample from the GGM (Partial-Association) Prior*

Description

Draws from the prior of a Gaussian graphical model. The likelihood is omitted ($n = 0$, $S = 0$), so the chain targets the prior alone. Three specifications are supported via the `spec` argument:

- `"conditional"` (default): fix a graph Γ and sample $K \mid \Gamma$ via the same theta-space NUTS sampler that drives `bgm` for continuous data. The chain targets $p(K \mid \Gamma) \propto \mathrm{slab}(K) \cdot \mathrm{diag}(K) \cdot |K|^{\delta} \cdot \mathbf{1}\{K \in \mathcal{M}^{+}(\Gamma)\}/Z(\Gamma)$.
- `"joint"`: sample (K, Γ) jointly from the un-normalised joint prior $p(K, \Gamma) \propto \mathrm{slab}(K) \cdot \mathrm{diag}(K) \cdot |K|^{\delta} \cdot \mathbf{1}\{K \in \mathcal{M}^{+}(\Gamma)\} \cdot \pi(\Gamma)$. Uses the adaptive-Metropolis MH chain from `bgm` with edge selection on and the likelihood off, so the marginal on Γ is $\pi(\Gamma) \cdot Z(\Gamma)$ (joint specification, not hierarchical). Useful for simulation-based calibration of `bgm`'s default sampler.
- `"hierarchical"`: sample (K, Γ) from the hierarchical specification $p(\Gamma)\, p(K \mid \Gamma)$ with $p(K \mid \Gamma)$ normalized per graph, so the marginal on Γ is exactly the edge prior $\pi(\Gamma)$. The per-graph normalizer ratio in each between-edge move is evaluated by the deterministic local Z-ratio approximation. Requires `normal_prior()` or `cauchy_prior()` interactions.

Usage

```
sample_ggm_prior(
  p,
  n_samples,
  n_warmup = 2000,
  interaction_prior = normal_prior(scale = 1),
  precision_scale_prior = exponential_prior(eta = 1),
  step_size = 0.1,
  max_depth = 10L,
  seed = 1L,
  verbose = TRUE,
  edge_indicators = NULL,
  delta = NULL,
  spec = c("conditional", "joint", "hierarchical"),
  edge_inclusion_prob = 0.5,
  update_method = c("adaptive-metropolis", "gibbs"),
  edge_prior = NULL,
  apply_correction = TRUE,
  zratio_diagnostics = TRUE
)
```

Arguments

`p` Integer. Dimension of the precision matrix ($p \geq 2$).

`n_samples` Integer. Number of post-warmup draws to keep.

`n_warmup` Integer. NUTS warmup iterations. Default `2000`.

`interaction_prior`
A `bgms_parameter_prior` for the partial-association off-diagonals $K_{yy,ij} = -K_{ij}/2$. Use `cauchy_prior()` or `normal_prior()`; `beta_prime_prior()` is not supported here. Default: `normal_prior(scale = 1)`, matching `bgm()`.

`precision_scale_prior`
A `bgms_scale_prior` for $K_{ii}/2$. Use `gamma_prior()` or `exponential_prior()`. Both accept the rate in the raw frame (`rate`) or the standardized frame (`eta`; the raw rate is derived as `eta / s` for interaction-prior scale `s`). Default: `exponential_prior(eta = 1)`; with the default `normal_prior(scale = 1)` interaction prior this resolves to $K_{ii}/2 \sim \text{Exponential}(1)$.

`step_size` Positive numeric. Initial NUTS step size used to seed dual-averaging adaptation. Default `0.1`. Used only for `spec = "conditional"` (NUTS path); ignored for the `"joint"` MH path.

`max_depth` Integer. Maximum NUTS tree depth. Default `10`. Used only for `spec = "conditional"`.

`seed` Integer. RNG seed for the chain. Default `1L`.

`verbose` Logical. If `TRUE` (default), print a progress bar.

`edge_indicators`
Optional integer $p \times p$ matrix with `1` = edge included, `0` = excluded. Must be symmetric with `1`s on the diagonal. Default: full graph (all edges included). Used only for `spec = "conditional"` (the chain samples $K \mid \Gamma$); ignored for `spec = "joint"`.

`delta` Non-negative numeric, or `NULL` for the dimension- adaptive default. Determinant-tilt exponent: multiplies the prior by $|K|^{\delta}$, softly repelling the chain from the positive-definite cone boundary. `delta = NULL` (default) auto-resolves to $0.5 \log(p)$, the simple form of the dimension-adaptive rule $\delta(p) = c \log p$ with $c \in (0.3, 0.6)$ (Marsman et al., in preparation). Pass `delta = 0` for the untilted prior or a non-negative numeric to override.

`spec` One of `"conditional"` (default, sample $K \mid \Gamma$ at fixed Γ), `"joint"` (sample (K, Γ) jointly from the un-normalised joint prior), or `"hierarchical"` (sample (K, Γ) from the per-graph normalized specification via the Z-ratio approximation).

`edge_inclusion_prob`
Probability in $(0, 1)$ for the Bernoulli edge prior used when `spec = "joint"`. Default `0.5`. Ignored when `spec = "conditional"`.

`update_method` One of `"adaptive-metropolis"` (default) or `"gibbs"`. Sampler driving the `spec = "joint"` chain; the Gibbs chain uses the conjugate row and edge updates and needs no proposal tuning. Ignored when `spec = "conditional"` (NUTS).

`edge_prior` An edge prior specification object from `bernoulli_prior()`, `beta_bernoulli_prior()`, or `sbm_prior()`, or `NULL` (default) for a Bernoulli prior with probability `edge_inclusion_prob`. Only for `spec = "joint"`.

`apply_correction`
: Logical. For the hierarchical edge priors (`beta_bernoulli_prior()`, `sbm_prior()`), apply the normalizing-constant correction to the hyperparameter updates (default `TRUE`; the correction table is built from the tilted prior sampler and cached across calls). With `FALSE` the plain conjugate updates are used, whose hyperparameter marginals do not match the hyperpriors under the determinant tilt.

`zratio_diagnostics`
: Logical (default `TRUE`). Only for `spec = "hierarchical"`: run the trust gauge (`summarize_zratio_gauge`) on the returned chain and attach the result; detected issues are printed when `verbose`. The gauge redoes a subset of the chain's edge decisions with the exact calculation and reports two alarms: how often the decision outcome differs (`flip_rate`), and the projected distortion of the mean inclusion probability from the measured error under the edge prior's feedback (`harm_pred`). Evidence-free sampling is the regime where the second alarm matters: a small consistent error can shift the graph marginal without flipping individual decisions. This argument switches the gauge on and off; how precisely it audits is set by `options(bgms.zratio_gauge_sweeps)`, the same option the deployed path reads.

Details

The priors are specified on the partial-association scale $K_{yy} = -K/2$: `interaction_prior` acts on $K_{yy,ij} = -K_{ij}/2$, and `precision_scale_prior` acts on $-K_{yy,ii} = K_{ii}/2$. The same convention is used by `bgm` and by the continuous block of the mixed-MRF model, so a prior argument passed here means the same distribution it would mean there. Output samples are reported as entries of K; convert with $K_{yy} = -K/2$ if you want them on the partial-association scale.

When `spec = "conditional"` and `edge_indicators` is supplied, off-diagonals at excluded positions are constrained to zero throughout the chain. `edge_indicators` is ignored when `spec = "joint"` (the chain samples Γ).

When `spec = "joint"`, the chain is initialized from an ancestral draw of the edge prior (hyperparameters from their prior, then indicators given the hyperparameters), keyed to `seed`. Under a hierarchical edge prior the inclusion parameter and the graph density are coupled, and a full-graph start can pin both near 1 for a large number of sweeps in zero-evidence chains.

Value

A list with components

`K_offdiag` Numeric matrix of size `n_samples` x `p * (p - 1) / 2` containing the upper-triangle off-diagonal entries of K for each draw, in row-major order (the upper triangle traversed by row) $(K_{12}, K_{13}, \ldots, K_{1p}, K_{23}, K_{24}, \ldots, K_{2p}, K_{34}, \ldots)$. Under `spec = "conditional"`, excluded edges are returned as `0`; under `spec = "joint"`, off-diagonals at excluded edges are sampled at `0` per the inclusion indicator.

`K_diag` Numeric matrix of size `n_samples` x `p` containing the diagonal entries $K_{11}, \ldots, K_{pp}$.

`offdiag_names` Character vector of length `p * (p - 1) / 2` naming the columns of `K_offdiag` (e.g. `"K_1_2"`).

`diag_names` Character vector of length `p` naming the columns of `K_diag`.

edge_indicators Under spec = "conditional", the p x p integer matrix of fixed inclusion indicators used (full graph if not supplied). Under spec = "joint", an n_samples x p(p-1)/2 integer matrix of sampled Γ_{ij} indicators (column order matches K_offdiag).

theta Only with beta_bernoulli_prior(): numeric vector of length n_samples with the sampled inclusion probability.

allocations Only with sbm_prior(): integer matrix (n_samples x p) of sampled cluster allocations (1-based).

zratio_diagnostics Only with spec = "hierarchical" and zratio_diagnostics = TRUE: the trust-gauge summary from summarize_zratio_gauge.

See Also

cauchy_prior, normal_prior, gamma_prior, exponential_prior, bgm

Examples

```
# Default Normal(0, 1) off-diagonal, Exponential(1) diagonal, p = 4.
draws = sample_ggm_prior(
  p = 4, n_samples = 200, n_warmup = 200,
  verbose = FALSE
)
dim(draws$K_offdiag) # 200 x 6
colnames(draws$K_offdiag) = draws$offdiag_names
head(draws$K_offdiag)

# Sparser graph: drop the (1, 4) edge.
E = matrix(1L, 4, 4)
E[1, 4] = E[4, 1] = 0L
draws = sample_ggm_prior(
  p = 4, n_samples = 200, n_warmup = 200,
  edge_indicators = E, verbose = FALSE
)
colnames(draws$K_offdiag) = draws$offdiag_names
all(draws$K_offdiag[, "K_1_4"] == 0) # TRUE
```

---

sample_graph_prior *Sample from the Graph Prior*

Description

Draws edge-inclusion indicators, together with any edge-prior hyperparameters, from the graph level of the spike-and-slab prior used by bgm for models with continuous variables. The spec argument selects how the precision prior composes with the graph:

- "hierarchical" (default): the graph marginal is exactly the edge prior, $p(\text{hyper})\,\pi(\Gamma \mid \text{hyper})$. Sampling is ancestral and exact: hyperparameters from their prior, then independent pair flips.

- "joint": the graph marginal is reweighted by the per-graph normalizer of the determinant-tilted precision prior, $q(\Gamma \mid \text{hyper}) \propto Z(\Gamma)\,\pi(\Gamma \mid \text{hyper})$. Sampling runs the zero-data (K, Γ) chain of sample_ggm_prior and discards K; with a Beta-Bernoulli or Stochastic-Block prior the hyperparameter updates apply the normalizing-constant correction, so the first call for a model cell may build the correction table (cached across fits).

Usage

```
sample_graph_prior(
  p,
  n_samples,
  edge_prior = bernoulli_prior(0.5),
  spec = c("hierarchical", "joint"),
  interaction_prior = normal_prior(scale = 1),
  precision_scale_prior = exponential_prior(eta = 1),
  delta = NULL,
  theta = NULL,
  allocations = NULL,
  block_probs = NULL,
  n_warmup = 2000,
  seed = 1L,
  verbose = TRUE
)
```

Arguments

p	Integer. Number of nodes ($p \geq 2$).
n_samples	Integer. Number of prior draws.
edge_prior	An edge prior specification object: bernoulli_prior(), beta_bernoulli_prior(), or sbm_prior(). Default bernoulli_prior(0.5).
spec	One of "hierarchical" (default) or "joint".
interaction_prior	A cauchy_prior() or normal_prior() for the pairwise (slab) part of the precision prior. Used only when spec = "joint". Default: normal_prior(scale = 1), matching bgm().
precision_scale_prior	A gamma_prior() or exponential_prior() for the precision diagonal. Used only when spec = "joint".
delta	Non-negative numeric or NULL (default): determinant tilt exponent; NULL resolves to $0.5\log(p)$. Used only when spec = "joint".
theta	Optional numeric in (0, 1): fix the inclusion probability of a Bernoulli or Beta-Bernoulli edge prior instead of sampling it.
allocations	Optional integer vector of length p with 1-based block labels: fix the Stochastic-Block allocation instead of sampling it. Requires block_probs.
block_probs	Optional symmetric matrix with entries in (0, 1): the block-pair inclusion probabilities that go with allocations.

n_warmup	Integer. Warmup iterations of the zero-data chain. Used only when spec = "joint". Default 2e3.
seed	Integer. Seed for the draw; the caller's RNG state is restored on exit.
verbose	Logical. Print progress of the zero-data chain and of a correction-table build. Default TRUE.

Details

The optional conditioning arguments fix the edge-prior hyperparameters instead of sampling them: theta fixes the inclusion probability of a Bernoulli or Beta-Bernoulli prior, and allocations plus block_probs fix the block structure of a Stochastic-Block prior (reducing it to independent pair flips at the given block probabilities).

Under spec = "joint" the tilted graph law depends on the precision prior through its normalizer, so interaction_prior, precision_scale_prior, and delta are part of the graph law; they are ignored under spec = "hierarchical".

Value

A list with elements:

edge_indicators Integer matrix (n_samples x p(p-1)/2) of edge-inclusion indicators, columns in row-major upper-triangle order (matching sample_ggm_prior()'s K_offdiag).

pair_names Character vector labeling the columns as "i-j".

theta Only with an unconditioned beta_bernoulli_prior(): numeric vector of sampled inclusion probabilities.

allocations Only with an unconditioned sbm_prior(): integer matrix (n_samples x p) of sampled block allocations.

spec, edge_prior, p The specification, edge-prior family, and node count of the draw.

See Also

sample_ggm_prior, sample_sbm_prior, bernoulli_prior, beta_bernoulli_prior, sbm_prior, bgm

Examples

```
# Hierarchical spec: the graph marginal is exactly the edge prior.
g = sample_graph_prior(
  p = 6, n_samples = 200,
  edge_prior = bernoulli_prior(0.3), seed = 11
)
mean(g$edge_indicators) # about 0.3

# Beta-Bernoulli: inclusion probabilities are sampled alongside.
g = sample_graph_prior(
  p = 6, n_samples = 200,
  edge_prior = beta_bernoulli_prior(2, 4), seed = 11
)
mean(g$theta) # about 1/3
```

```
# Joint spec: the graph law carries the per-graph normalizer Z(Gamma).
g = sample_graph_prior(
  p = 6, n_samples = 500,
  edge_prior = bernoulli_prior(0.3), spec = "joint",
  interaction_prior = normal_prior(scale = 0.5),
  precision_scale_prior = exponential_prior(rate = 2),
  seed = 11, verbose = FALSE
)
mean(g$edge_indicators) # shifted away from 0.3 by the Z(Gamma) tilt
```

`sample_sbm_prior` *Sample from the Stochastic-Block Edge-Prior Hyperprior*

Description

Ancestral draws from the MFM-SBM hyperprior of `sbm_prior()`: a partition of the nodes into blocks (shifted-Poisson number of components, Dirichlet-weighted allocation) and Beta-distributed within- and between-block edge-inclusion probabilities. These are the hyperparameters that `sample_graph_prior` and `bgm` integrate over when the edge prior is a Stochastic-Block prior.

Usage

```
sample_sbm_prior(p, n_samples, edge_prior = sbm_prior(), seed = 1L)
```

Arguments

`p`	Integer. Number of nodes ($p \geq 2$).
`n_samples`	Integer. Number of prior draws.
`edge_prior`	An `sbm_prior()` object. Default `sbm_prior()`.
`seed`	Integer. Seed for the draw; the caller's RNG state is restored on exit.

Value

A list with elements:

`allocations` Integer matrix (`n_samples` x `p`) of block labels.

`pair_probability` Numeric matrix (`n_samples` x `p(p-1)/2`) of implied pair-inclusion probabilities, columns in row-major upper-triangle order.

`num_blocks` Integer vector: number of occupied blocks per draw.

`pair_names` Character vector labeling the pair columns as `"i-j"`.

`p` The node count.

See Also

`sbm_prior`, `sample_graph_prior`, `bgm`

Examples

```
draws = sample_sbm_prior(p = 8, n_samples = 100, seed = 4)
table(draws$num_blocks)
range(draws$pair_probability)
```

---

sbm_prior *Stochastic Block Model Prior for Inclusion Indicators*

Description

Specifies a Stochastic Block Model (SBM) prior for inclusion indicators. Variables are assigned to latent clusters, with separate Beta priors on within-cluster and between-cluster inclusion probabilities.

Usage

```
sbm_prior(
  alpha = 1,
  beta = 1,
  alpha_between = 1,
  beta_between = 1,
  dirichlet_alpha = 1,
  lambda = 1
)
```

Arguments

alpha	Positive numeric. First shape parameter of the Beta distribution for within-cluster edges. Default: 1.
beta	Positive numeric. Second shape parameter of the Beta distribution for within-cluster edges. Default: 1.
alpha_between	Positive numeric. First shape parameter of the Beta distribution for between-cluster edges. Default: 1.
beta_between	Positive numeric. Second shape parameter of the Beta distribution for between-cluster edges. Default: 1.
dirichlet_alpha	Positive numeric. Concentration parameter of the Dirichlet prior on cluster assignments. Default: 1.
lambda	Positive numeric. Rate parameter of the shifted Poisson prior on the number of clusters B: $B - 1 \sim \text{Poisson}(\lambda)$, so $P(B = b) = \lambda^{b-1}e^{-\lambda}/(b-1)!$ for $b = 1, 2, \ldots$ and $E[B] = 1 + \lambda$. Default: 1.

Value

An object of class "bgms_indicator_prior" with family = "Stochastic-Block".

See Also

bernoulli_prior, beta_bernoulli_prior, bgm

Other prior-constructors: bernoulli_prior(), beta_bernoulli_prior(), beta_prime_prior(), cauchy_prior(), exponential_prior(), gamma_prior(), normal_prior()

Examples

```
sbm_prior()
sbm_prior(alpha = 2, beta = 1, alpha_between = 1, beta_between = 5)
```

---

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.

Other prediction: `predict.bgmCompare()`, `predict.bgms()`, `simulate.bgms()`, `simulate_mrf()`

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. Supports ordinal, Blume-Capel, continuous (GGM), and mixed MRF models.

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.

Discrete columns are returned on the original category scale of the training data (the values supplied to `bgm()`), so the output can be passed straight to `predict()`. For mixed MRF models, discrete columns contain non-negative integers and continuous columns contain real-valued observations, ordered as in the original data.

See Also

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

Other prediction: predict.bgmCompare(), predict.bgms(), simulate.bgmCompare(), simulate_mrf()

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 a Markov Random Field*

Description

simulate_mrf() generates observations from a Markov Random Field using user-specified parameters. For ordinal and Blume-Capel variables, observations are generated via Gibbs sampling. For continuous variables (Gaussian graphical model), observations are drawn directly from the multivariate normal distribution implied by the precision matrix.

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 observations to be generated.

`num_variables` The number of variables in the 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. Only used for ordinal and Blume-Capel variables; ignored when `variable_type = "continuous"`.

`pairwise` A symmetric `num_variables` by `num_variables` matrix. For ordinal and Blume-Capel variables, this contains the pairwise interaction parameters; only the off-diagonal elements are used. For continuous variables, this is the precision matrix Ω (including diagonal) and must be positive definite.

`main` For ordinal and Blume-Capel variables: 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. For continuous variables: a numeric vector of length `num_variables` containing the means μ for each variable. Defaults to zeros if not supplied (missing(main)).

`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"`, `"blume-capel"`, and `"continuous"`. Binary variables are automatically treated as `"ordinal"`. Ordinal and Blume-Capel variables can be mixed freely, but continuous variables cannot be mixed with ordinal or Blume-Capel variables. When `variable_type = "continuous"`, the function simulates from a Gaussian graphical model. 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 (ordinal/Blume-Capel variables only). The function provides the last state of the Gibbs sampler as output. Ignored for continuous variables. 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

Ordinal / Blume-Capel variables: The Gibbs sampler is initiated with random values from the response options, after which it proceeds by simulating states for each variable from its full conditional distribution given the other variable states.

Continuous variables (GGM): Observations are drawn from $N(\mu, \Omega^{-1})$ where Ω is the precision matrix specified via `pairwise` and μ is the means vector specified via `main`. No Gibbs sampling is needed; `iter` is ignored.

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:

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

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 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 observations. For ordinal/Blume-Capel variables, entries are non-negative integers. For continuous variables, entries are real-valued.

See Also

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

Other prediction: `predict.bgmCompare()`, `predict.bgms()`, `simulate.bgmCompare()`, `simulate.bgms()`

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
)

# Generate responses from a Gaussian graphical model (GGM) with 4 variables.
num_variables = 4

# Precision matrix (symmetric, positive definite)
Omega = diag(c(1, 1.2, 0.8, 1.5))
Omega[2, 1] = Omega[1, 2] = 0.3
Omega[3, 1] = Omega[1, 3] = 0.3
Omega[4, 2] = Omega[2, 4] = -0.2

x = simulate_mrf(
  num_states = 500,
  num_variables = num_variables,
  pairwise = Omega,
  variable_type = "continuous"
)
```

---

summarize_zratio_gauge

Summarize the Hierarchical Prior Trust Gauge

Description

Reports the per-chain trust gauge for the hierarchical graph prior. Under that prior the sampler decides each edge with a fast approximation; in a set of assessment sweeps after sampling the gauge redoes a subset of each chain's own edge decisions with the exact calculation and records two statistics per chain:

`flip_rate` The fraction of add/remove decisions that would come out differently under the exact calculation. A chain is flagged on this channel when `flip_rate` exceeds the tolerance by more than the exact reference's own Monte-Carlo noise. This channel detects error that changed decisions the chain actually made; it is insensitive to a small coherent error at chains whose decisions are far from their accept/reject boundaries.

`harm_pred` The projected distortion of the mean posterior inclusion probability under the measured approximation error, a first-order (linear-response) quantity targeted at coherent error: `harm_pred = |mean(m_e s_e)| * A`, where `s_e` is the signed log-ratio error of audited edge `e` against the exact reference, `m_e = p_e (1 - p_e)` is that edge's inclusion sensitivity, and `A = 1 / (1 - g)` is the inclusion-probability feedback amplification with linearized gain `g = E m / (theta (1 - theta) (a + b + E))`, `m = mean(p_e (1 - p_e))` over all edges, under a Beta-Bernoulli edge prior (for a fixed inclusion probability `g = 0`, so `A = 1`). When the per-pair audit stream is unavailable the unweighted form `|se_mean| * m * A` is used. A chain is flagged on this channel when `harm_pred` exceeds the tolerance and the weighted error is resolved above twice its standard error (per-edge cluster-robust spread plus the reference Monte-Carlo noise). This channel detects coherent error whose equilibrium effect exceeds the tolerance even when no individual decision visibly flips; it targets the mean-inclusion shift and does not bound edge-specific distortions that cancel in the mean. It is computed for Bernoulli and Beta-Bernoulli edge priors; under other priors it is `NA`.

The gauge reads the `zratio$gauge` block that the sampler attaches under the hierarchical prior; it does not re-scan the stored draws.

Usage

```
summarize_zratio_gauge(
  chains,
  threshold = 0.01,
  verbose = TRUE,
  harm_inputs = NULL,
  harm_threshold = 0.02
)
```

Arguments

`chains`	List of per-chain sampler outputs, each carrying a `zratio$gauge` block (`flip_rate`, `noise_floor`, `se_mean`, `se_sd`, `se_mcse`, `n_ent`, `n_ref`, `n_capped`).
`threshold`	Numeric flag threshold on `flip_rate` (default `0.01`).
`verbose`	Logical: message flagged chains (default `TRUE`).
`harm_inputs`	Optional list enabling the `harm_pred` channel: `pip` (a list with one numeric vector of posterior edge-inclusion probabilities per chain, over the audited block, which is every edge on a Gaussian graphical model and the continuous-continuous

block on a mixed model) and, for a Beta-Bernoulli edge prior, its shape parameters a and b (NULL for a fixed inclusion probability). An optional pool_pip, of the same shape as pip, gives the edges the shared inclusion parameter is drawn from when that pool is wider than the audited block, as on a mixed fit; omit it when the two coincide. When NULL (default) the harm_pred columns are NA and only the flip_rate channel flags.

harm_threshold Numeric flag threshold on harm_pred, in inclusion-probability units (default 0.02). The default sits above the spread this projection shows across healthy reference fits and below its value on a known-biased reference kernel, near the geometric midpoint of the two, so a flag marks a projected distortion outside anything a healthy fit produces.

Details

The gauge audits only the edge moves whose mediating block is non-trivial (two or more variables). On a sparse posterior no block reaches that size, the normalizer ratio is exact, and the gauge reports nothing because there is nothing to audit: silence there is exactness, not blindness. The cost follows the same rule – it is zero where the approximation is exact and grows with the size of the blocks the chain actually visits.

The audit is a sample. Each sweep references a capped number of edge moves, so on a dense large graph a few tens of moves stand in for tens of thousands of non-trivial ones (n_ref against n_ent, with the remainder counted in n_capped). That sample resolves coherent error – error with a consistent sign across edges, which is what harm_pred projects onto the inclusion-probability scale and what shifts a recovered network. It does not resolve rare edge-specific failures: flip_rate is the flip rate among audited decisions, not a per-edge guarantee over all of them.

Value

An invisible named list:

per_chain Data frame, one row per chain: flip_rate, its flag, the signed mean and spread of the log-ratio error (se_mean, se_sd), the reference-noise component se_mcse and combined standard error se_se of se_mean, the reference noise_floor, the pair counts (n_ent non-trivial seen, n_ref referenced, n_capped cap hits), the mediating-block sizes the audit covered (block_lo, block_hi), and the harm channel (amplification, kappa = the predicted mean-inclusion shift per nat of coherent error, harm_pred, harm_flag).

threshold The flag tolerance on flip_rate.

harm_threshold The flag tolerance on harm_pred.

flagged Logical: any chain flagged on either channel.

When the additive correction collapses

Below a gamma_prior() diagonal shape of 0.5 the per-edge correction is the additive kernel rather than the fitted surface. That kernel sums per-channel moment constants over the mediating block's node and edge counts and passes the pair to a saddle map, which returns a ratio of 1 – log-ratio exactly 0 – whenever the first moment is not positive. On a common-neighbour block the edge constant is a difference and is negative in most cells, while the edge count grows as the square of the block size against a linear node count, so the sum crosses zero at a size the cell and the density

fix. Past that size the ratio is *discarded*, not approximated: those edge decisions are made with no correction at all.

MEASURED over 4160 cells spanning both interaction slabs, both block families, ten diagonal shapes, two standardized rates, thirteen block sizes and four densities: the moment-sign condition accounts for the collapse in every one of them, and a closed-form expression for the size at which it fires matches an exact size-by-size scan in 40 of 40 cells. Bipartite bridge blocks are unaffected, since only the positive bridge constant contributes there.

That boundary is not a single number. It moves with the cell and with the determinant-tilt exponent `delta`, and `bgm()` resolves `delta` from the model dimension by default ($0.5 \log p$), which is also what bounds the largest possible mediating block at $p - 2$. Both therefore move together. MEASURED at the default `delta` across p from 6 to 120: the boundary runs from about 12 to 32 common-neighbour variables, and the collapse is *out of reach* at small p (the largest possible block is below the boundary) and reachable from roughly 20 to 30 variables upward. A fit below that is not exposed to this at all.

The per-chain `counters` vector records `n_collapsed` and `max_collapse_size`, and a fit on which it happened says so in a note. This is a documented limitation of the fallback kernel and not a property of the surface, which serves every shape from 0.5 up.

Fits that route around the correction entirely

Above the Gamma diagonal shape range the correction is scored on, the mediating correction is switched off and every edge is served the isolated-edge normalizer ratio, which is exact for an edge with no mediating structure. The per-chain `counters` vector records this as `n_isolated`, and a fit that took the route says so in a note reporting the measured bound on what it leaves out. The gauge still runs there and still measures that residual directly, so a flag on such a fit is as meaningful as on any other.

See Also

`sample_ggm_prior`

Other diagnostics: `calibration_check()`, `plot.bgms_calibration()`, `plot.bgms_prior_sensitivity()`, `print.bgms_calibration()`, `print.bgms_prior_sensitivity()`, `prior_sensitivity_check()`

Examples

```
draws = sample_ggm_prior(
  p = 8, n_samples = 100, n_warmup = 200,
  interaction_prior = normal_prior(scale = 0.5),
  precision_scale_prior = gamma_prior(shape = 1, rate = 2),
  spec = "hierarchical", verbose = FALSE
)
draws$zratio_diagnostics$per_chain
```

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.

Details

In the printed "Group differences (main effects)" block, a threshold difference that rests on the prior rather than on the data is marked with a leading *, and the block gains the footnote *a group lacks observations in this category or in the reference category; the estimate reflects the prior, not the data*.

bgmCompare() keeps the union of the categories the groups observe, so a group can contribute no observations at all to a retained category. Two things follow. If the empty category is the row's own, that group's threshold for it has nothing behind it. If the empty category is the reference category, the group has no data fixing the level of its threshold vector at all, and *every* threshold of that variable is affected, not just one — so the whole variable is marked. Either way the reported difference is large and very uncertain without being evidence of a group difference.

The per-group counts behind the mark are in extract_arguments(fit)$category_support, one matrix of category-by-group observation counts per variable (NULL for Blume-Capel variables, which are exempt from the union recode and are never marked). Fits made before that field existed print unmarked.

Value

An object of class summary.bgmCompare with posterior summaries.

See Also

bgmCompare(), print.bgmCompare(), coef.bgmCompare()

Other posterior-methods: coef.bgmCompare(), coef.bgms(), plot.bgmCompare(), plot.bgms(), plot_edge_posterior(), print.bgmCompare(), print.bgms(), print.bgms_verdicts(), summary.bgms(), verdicts()

Examples

```
# See ?bgmCompare for a full example
```

---

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.

See Also

bgm(), print.bgms(), coef.bgms()

Other posterior-methods: coef.bgmCompare(), coef.bgms(), plot.bgmCompare(), plot.bgms(), plot_edge_posterior(), print.bgmCompare(), print.bgms(), print.bgms_verdicts(), summary.bgmCompare(), verdicts()

Examples

```
fit = bgm(x = Wenchuan[, 1:3])
summary(fit)
```

summary.bgms_centrality

Summarize Posterior Centrality

Description

Posterior mean, credible interval, and probability of being the most central node, per variable.

Usage

```
## S3 method for class 'bgms_centrality'
summary(object, probs = c(0.025, 0.975), ...)
```

Arguments

object	An object of class bgms_centrality, from extract_centrality().
probs	Numeric of length two; the credible-interval quantiles. Default c(0.025, 0.975).
...	Ignored.

Details

A difference has a point mass at exactly zero, from the draws in which every one of the node's difference indicators is excluded and the two groups share the network. p_positive and its mirror therefore need not sum to one, and a node can have a positive posterior mean with p_positive well below 0.5: the remaining mass is on no difference at all, which is the model averaging reporting itself.

Value

A data frame with one row per variable, ordered by decreasing posterior mean, with columns node, mean, lower, upper, and p_most_central, the posterior probability that the node has the largest centrality of all nodes.

For a difference between two groups the last column is p_positive instead: the posterior probability that the node's centrality is higher in the first group than in the second. Which node is *most* central is not the question a difference answers.

See Also

extract_centrality()

Other extractors: extract_arguments(), extract_category_thresholds(), extract_centrality(), extract_ess(), extract_group_params(), extract_inclusion_bf(), extract_indicator_priors(), extract_indicators(), extract_log_odds(), extract_main_effects(), extract_pairwise_interactions(), extract_partial_correlations(), extract_posterior_inclusion_probabilities(), extract_precision(), extract_prior_inclusion_probabilities(), extract_rhat(), extract_sbm(), plot.bgms_centrality()

Examples

```
fit = bgm(Wenchuan[, 1:5], display_progress = "none")
summary(extract_centrality(fit))
```

`verdicts` *Edge Verdicts and Their Monte Carlo Fragility*

Description

Reads each edge (or difference) indicator's inclusion Bayes factor as a three-way verdict – evidence of presence, undecided, evidence of absence – and flags the verdicts that a rerun of the sampler could change.

Usage

```
verdicts(bgms_object, evidence_threshold = 10, ...)
```

Arguments

`bgms_object`	A fitted model object of class `bgms` (from `bgm()` with `edge_selection = TRUE`) or `bgmCompare` (from `bgmCompare()` with `difference_selection = TRUE`).
`evidence_threshold`	Numeric > 1; the inclusion Bayes factor above which an edge is called present. Its reciprocal is the threshold below which an edge is called absent. Default `10`.
`...`	Passed to methods.

Details

Monte Carlo verdict errors are a boundary phenomenon. In a known-truth calibration study of 37,010 graded edge-fits across ordinal, binary, and Gaussian graphical models, every one of the 66 verdict errors sat within 0.58 of a threshold on the log Bayes factor scale, and no edge further out was ever misclassified. The fragility flag turns that into a per-edge statement: an edge is fragile when a verdict boundary lies within two standard errors of the estimated evidence, which is the regime where the reported verdict rests on Monte Carlo noise.

Two standard errors are computed because neither catches every error alone. The two-state standard error models the binary indicator chain as a first-order two-state Markov chain with Jeffreys-smoothed transition rates, which keeps it defined when the indicator never flips; on its own it caught 74% of the study's verdict errors. The Rao-Blackwellized standard error is the Monte Carlo standard error of the one-step inclusion draws, carried to the logit scale by the delta method; on its own it caught 94%. Flagging when either places a boundary within two standard errors caught all 66, at the cost of also flagging 3.0% of correct verdicts. The union transfers across model types, so the flag applies to Gaussian graphical models as it does to ordinal and binary ones.

A fragile verdict is not a wrong verdict; it is a verdict the run is too short to settle. The remedy is more sampling iterations.

Every arm of that study fitted a single network, so the operating point applies to the edge indicators of bgm(). On the difference indicators of bgmCompare() the flag still marks verdicts sitting near a boundary, but no study has measured what share of difference-verdict errors it catches or how many correct verdicts it rejects; the print method says so, and the returned object carries a flag_validated attribute.

Value

A data frame of class bgms_verdicts, one row per indicator, in the order of the fit's raw indicator draws, with columns:

parameter Indicator name, as in summary(fit)$indicator.

pip Rao-Blackwellized posterior inclusion probability.

bf, log_bf Inclusion Bayes factor and its natural logarithm, from extract_inclusion_bf(), which returns the same natural-log scale under log = TRUE. log_bf stays finite where bf saturates at 0 or Inf.

verdict Factor with levels presence, undecided, absence, and NA for indicators that were never updated (main-effect differences under main_difference_selection = FALSE).

se_two_state, se_rb Standard errors of the logit inclusion probability, from the Jeffreys-smoothed two-state model of the indicator chain and from the Rao-Blackwellized draws. se_rb is NA where the Rao-Blackwellized draws are constant to double precision.

distance_two_state, distance_rb Distance from log_bf to the nearer verdict boundary, in units of each standard error.

fragile TRUE when either distance is below 2.

The evidence threshold is attached as the evidence_threshold attribute, and whether the fragility flag's operating point covers this kind of indicator as flag_validated. A bgmCompare fit additionally carries difference_fit, and, under main_difference_selection = FALSE, unselected_main, marking the main-effect rows the print leaves out of its counts because no indicator exists for them.

See Also

extract_inclusion_bf() for the Bayes factors, extract_posterior_inclusion_probabilities() for the inclusion probabilities, prior_sensitivity_check() for whether a verdict depends on the prior rather than on the run length

Other posterior-methods: coef.bgmCompare(), coef.bgms(), plot.bgmCompare(), plot.bgms(), plot_edge_posterior(), print.bgmCompare(), print.bgms(), print.bgms_verdicts(), summary.bgmCompare(), summary.bgms()

Examples

```
fit = bgm(Wenchuan[, 1:5], display_progress = "none")
v = verdicts(fit)
v[v$fragile, ]

# A stricter reading of the evidence moves the boundaries, and with them
# which edges sit close enough to one to be fragile.
verdicts(fit, evidence_threshold = 30)
```

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.

`$.bgmCompare` *Access elements of a bgmCompare object*

Description

Provides `$` access to S7 properties. Lazy `posterior_summary_*` properties trigger computation on first access via S7 property getters. Also supports legacy S3 list-based fit objects.

Usage

```
## S3 method for class 'bgmCompare'
x$name

## S3 method for class 'bgmCompare'
x[[name, ...]]
```

Arguments

`x`	A `bgmCompare` object.
`name`	Name of the element to access.
`...`	Ignored.

Value

The requested element.

`$.bgms` *Access elements of a bgms object*

Description

Provides `$` access to S7 properties. Lazy `posterior_summary_*` properties trigger computation on first access via S7 property getters. Also supports legacy S3 list-based fit objects.

Usage

```
## S3 method for class 'bgms'
x$name

## S3 method for class 'bgms'
x[[name, ...]]
```

Arguments

`x`	A `bgms` object.
`name`	Name of the element to access.
`...`	Ignored.

Value

The requested element.

Index

∗ **datasets**
ADHD, 3
Boredom, 22
Wenchuan, 92
∗ **diagnostics**
calibration_check, 23
plot.bgms_calibration, 53
plot.bgms_prior_sensitivity, 55
print.bgms_calibration, 63
print.bgms_prior_sensitivity, 64
prior_sensitivity_check, 65
summarize_zratio_gauge, 83
∗ **extractors**
extract_arguments, 29
extract_category_thresholds, 29
extract_centrality, 30
extract_ess, 32
extract_group_params, 33
extract_inclusion_bf, 34
extract_indicator_priors, 36
extract_indicators, 35
extract_log_odds, 37
extract_main_effects, 38
extract_pairwise_interactions, 39
extract_partial_correlations, 40
extract_posterior_inclusion_probabilities, 41
extract_precision, 42
extract_prior_inclusion_probabilities, 43
extract_rhat, 45
extract_sbm, 46
plot.bgms_centrality, 54
summary.bgms_centrality, 89
∗ **model-fitting**
bgm, 7
bgmCompare, 16
∗ **posterior-methods**
coef.bgmCompare, 26
coef.bgms, 27
plot.bgmCompare, 48
plot.bgms, 51
plot_edge_posterior, 56
print.bgmCompare, 62
print.bgms, 62
print.bgms_verdicts, 65
summary.bgmCompare, 87
summary.bgms, 88
verdicts, 90
∗ **prediction**
predict.bgmCompare, 58
predict.bgms, 59
simulate.bgmCompare, 77
simulate.bgms, 78
simulate_mrf, 80
∗ **prior-constructors**
bernoulli_prior, 5
beta_bernoulli_prior, 5
beta_prime_prior, 6
cauchy_prior, 25
exponential_prior, 28
gamma_prior, 46
normal_prior, 47
sbm_prior, 76
[[.bgmCompare *($.bgmCompare)*, 93
[[.bgms *($.bgms)*, 94
$.bgmCompare, 93
$.bgms, 94

ADHD, 3

bernoulli_prior, 5, *6*, *9*, *18*, *70*, *73*, *74*, *77*
bernoulli_prior(), *6*, *25*, *28*, *47*, *48*, *77*
beta_bernoulli_prior, *5*, 5, *9*, *18*, *70*, *73*, *74*, *77*
beta_bernoulli_prior(), *5*, *6*, *25*, *28*, *47*, *48*, *77*
beta_prime_prior, 6, *8*, *9*, *18*, *25*, *48*, *70*
beta_prime_prior(), *5*, *6*, *25*, *28*, *47*, *48*, *77*

bgm, *5, 6*, 7, *18*, *20*, *25*, *28*, *47, 48*, *65, 66*, *68–75*, *77*
bgm(), *21*, *23*, *27*, *29, 30*, *32–46*, *50, 51*, *56*, *63*, *88*, *90, 91*
bgmCompare, *5*, 16, *65, 66*
bgmCompare(), *15*, *23, 24*, *26*, *29–36*, *38, 39*, *41, 42*, *45, 46*, *48, 49*, *62*, *87*, *90, 91*
Boredom, 22

calibration_check, 23
calibration_check(), *53*, *55*, *63, 64*, *68*, *86*
cauchy_prior, *6*, *8, 9*, *18*, 25, *48*, *70*, *72, 73*
cauchy_prior(), *5, 6*, *28*, *47, 48*, *77*
coef.bgmCompare, 26
coef.bgmCompare(), *27*, *50*, *52*, *57*, *62, 63*, *65*, *87, 88*, *91*
coef.bgms, 27
coef.bgms(), *26*, *43*, *50*, *52*, *57*, *62, 63*, *65*, *87, 88*, *91*

exponential_prior, *9*, 28, *47*, *70*, *72, 73*
exponential_prior(), *5, 6*, *25*, *47, 48*, *77*
extract_arguments, 29
extract_arguments(), *30, 31*, *33*, *35–40*, *42–46*, *54*, *89*
extract_category_thresholds, 29
extract_category_thresholds(), *29*, *31*, *33*, *35–40*, *42–46*, *54*, *89*
extract_centrality, 30
extract_centrality(), *29, 30*, *33*, *35–40*, *42–46*, *48–52*, *54*, *89*
extract_ess, 32
extract_ess(), *29–31*, *33*, *35–40*, *42–46*, *54*, *89*
extract_group_params, 33
extract_group_params(), *29–31*, *33*, *35–40*, *42–46*, *49, 50*, *54*, *89*
extract_inclusion_bf, 34
extract_inclusion_bf(), *29–31*, *33*, *36–46*, *54*, *56, 57*, *89*, *91*
extract_indicator_priors, 36
extract_indicator_priors(), *29–31*, *33*, *35–40*, *42–46*, *54*, *89*
extract_indicators, 35
extract_indicators(), *29–31*, *33*, *35–40*, *42–46*, *54*, *89*
extract_log_odds, 37
extract_log_odds(), *14*, *29–31*, *33*, *35, 36*, *38–40*, *42–46*, *54*, *89*
extract_main_effects, 38
extract_main_effects(), *29–31*, *33*, *35–37*, *39, 40*, *42–46*, *54*, *89*
extract_pairwise_interactions, 39
extract_pairwise_interactions(), *29–31*, *33*, *35–38*, *40*, *42–46*, *54*, *57*, *89*
extract_partial_correlations, 40
extract_partial_correlations(), *14*, *29–31*, *33*, *35–39*, *42–46*, *54*, *89*
extract_posterior_inclusion_probabilities, 41, *68*
extract_posterior_inclusion_probabilities(), *29–33*, *35–40*, *43–46*, *54*, *89*, *91*
extract_precision, 42
extract_precision(), *14*, *27*, *29–31*, *33*, *35–40*, *42*, *44–46*, *54*, *89*
extract_prior_inclusion_probabilities, *10*, 43
extract_prior_inclusion_probabilities(), *29–31*, *33–40*, *42, 43*, *45, 46*, *54*, *89*
extract_rhat, 45
extract_rhat(), *29–31*, *33*, *35–40*, *42–44*, *46*, *54*, *89*
extract_sbm, 46
extract_sbm(), *29–31*, *33*, *35–40*, *42–45*, *54*, *89*

gamma_prior, *9*, *28*, 46, *70*, *72, 73*
gamma_prior(), *5, 6*, *25*, *28*, *48*, *77*

normal_prior, *6*, *8, 9*, *18*, *25*, 47, *70*, *72, 73*
normal_prior(), *5, 6*, *25*, *28*, *47*, *77*

plot(), *31*
plot.bgmCompare, 48
plot.bgmCompare(), *26, 27*, *52*, *57*, *62, 63*, *65*, *87, 88*, *91*
plot.bgms, 51
plot.bgms(), *26, 27*, *49, 50*, *57*, *62, 63*, *65*, *87, 88*, *91*
plot.bgms_calibration, 53
plot.bgms_calibration(), *25*, *55*, *64*, *68*, *86*
plot.bgms_centrality, 54
plot.bgms_centrality(), *29–31*, *33*, *35–40*, *42–46*, *49*, *51*, *89*
plot.bgms_prior_sensitivity, 55
plot.bgms_prior_sensitivity(), *25*, *53*, *64*, *68*, *86*

plot_edge_posterior, 56
plot_edge_posterior(), *26, 27*, *50*, *52*, *62, 63*, *65*, *87, 88*, *91*
predict.bgmCompare, 58, *78*
predict.bgmCompare(), *61*, *78*, *80*, *82*
predict.bgms, *59*, 59, *80*
predict.bgms(), *25*, *59*, *78*, *80*, *82*
print.bgmCompare, 62
print.bgmCompare(), *26, 27*, *50*, *52*, *57*, *63*, *65*, *87, 88*, *91*
print.bgms, 62
print.bgms(), *26, 27*, *50*, *52*, *57*, *62*, *65*, *87, 88*, *91*
print.bgms_calibration, 63
print.bgms_calibration(), *25*, *53*, *55*, *64*, *68*, *86*
print.bgms_prior_sensitivity, 64
print.bgms_prior_sensitivity(), *25*, *53*, *55*, *64*, *68*, *86*
print.bgms_verdicts, 65
print.bgms_verdicts(), *26, 27*, *50*, *52*, *57*, *62, 63*, *87, 88*, *91*
prior_sensitivity_check, *55*, *64*, 65
prior_sensitivity_check(), *25*, *50*, *53*, *55*, *64*, *86*, *91*

qgraph::qgraph(), *49–51*

sample_ggm_prior, 69, *73, 74*, *86*
sample_graph_prior, 72, *75*
sample_sbm_prior, *74*, 75
sbm_prior, *5, 6*, *9*, *18*, *70*, *73–75*, 76
sbm_prior(), *5, 6*, *25*, *28*, *47, 48*
simulate(), *24*
simulate.bgmCompare, *59*, 77
simulate.bgmCompare(), *59*, *61*, *80*, *82*
simulate.bgms, *61*, *78*, 78, *82*
simulate.bgms(), *25*, *59*, *61*, *78*, *82*
simulate_mrf, *80*, 80
simulate_mrf(), *59*, *61*, *78*, *80*
summarize_zratio_gauge, *11*, *71, 72*, 83
summarize_zratio_gauge(), *25*, *53*, *55*, *64*, *68*
summary(), *31*
summary.bgmCompare, 87
summary.bgmCompare(), *21*, *26, 27*, *29*, *50*, *52*, *57*, *62, 63*, *65*, *88*, *91*
summary.bgms, 88
summary.bgms(), *26, 27*, *29*, *50*, *52*, *57*, *62, 63*, *65*, *87*, *91*
summary.bgms_centrality, 89
summary.bgms_centrality(), *29–31*, *33*, *35–40*, *42–46*, *54*

verdicts, 90
verdicts(), *26, 27*, *31*, *49–52*, *57*, *62, 63*, *65*, *87, 88*

Wenchuan, 92