Package 'RoBMA'

November 8, 2026

Title Robust Bayesian Meta-Analyses

Version 4.1.1

Maintainer František Bartoš <f.bartos96@gmail.com>

Description A framework for Bayesian meta-analysis, including model estimation, prior specification, model comparison, prediction, summaries, visualizations, and diagnostics. The package fits single and model-averaged meta-analytic, meta-regression, multilevel and multivariate, publication bias adjusted, and generalized linear mixed models. The model-averaged meta-analytic models combine competing models based on their predictive performance, weight inference by posterior model probabilities, and test model components using Bayes factors (e.g., effect vs. no effect; Bartoš et al., 2022, <doi:10.1002/jrsm.1594>; Maier, Bartoš & Wagenmakers, 2022, <doi:10.1037/met0000405>; Bartoš et al., 2025, <doi:10.1037/met0000737>). Users can specify flexible prior distributions for effect sizes, heterogeneity, publication bias (including selection models and PET-PEESE), and moderators.

URL https://fbartos.github.io/RoBMA/

BugReports https://github.com/FBartos/RoBMA/issues

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

SystemRequirements JAGS (>= 4.3.1, < 5.0.0) (https://mcmc-jags.sourceforge.io/)

NeedsCompilation yes

Depends R (>= 4.3.0)

Imports BayesTools (>= 0.4.0), bridgesampling, loo (>= 2.10.0), MASS, Matrix (>= 1.6-2), parallel, ps, runjags, rjags, statmod, stats, graphics, mvtnorm, scales, Rdpack, rlang, coda, ggplot2

Suggests metafor (>= 5.0-1), posterior, weightr, lme4, fixest, metaBMA, emmeans, metadat, testthat (>= 3.3.2), R6, withr, cli, vdiffr, knitr, rmarkdown, covr, pkgload, processx

LinkingTo Matrix (>= 1.6-2), mvtnorm

RdMacros Rdpack

VignetteBuilder knitr

Author František Bartoš [aut, cre] (ORCID:
<https://orcid.org/0000-0002-0018-5573>),
Maximilian Maier [aut] (ORCID: <https://orcid.org/0000-0002-9873-6096>),
Eric-Jan Wagenmakers [ths] (ORCID:
<https://orcid.org/0000-0003-1596-1034>),
Joris Goosen [ctb],
Matthew Denwood [cph] (Original copyright holder of some modified code
where indicated.),
Martyn Plummer [cph] (Original copyright holder of some modified code
where indicated.)

Repository CRAN

Date/Publication 2026-11-08 18:40:02 UTC

Contents

RoBMA-package 5
add_loo.brma 6
add_marglik.brma 8
add_waic.brma 11
AIC.brma 12
Anderson2010 13
Andrews2021 13
as.data.frame.brma 15
as.data.frame.brma_samples 16
as.data.frame.covratio.brma 17
as.data.frame.dfbetas.brma 18
as.data.frame.infl.brma 18
as.data.frame.marginal_means.brma 19
as.data.frame.summary_heterogeneity.brma 20
as.data.frame.summary_models.RoBMA 21
as.data.frame.vif.brma 22
as.data.frame.zplot_brma 23
as.matrix.brma_samples 24
as_draws.brma 24
as_draws.brma_samples 27
as_metafor_forest 28
as_zplot.brma 30
Bem2011 33
bf.brma 34
blup 35
blup.brma 36
BMA 38
BMA.glmm 43
BMA.mv 49

bPEESE . . . 55
bPEESE.mv . . . 60
bPET . . . 66
bPET.mv . . . 70
bridge_sampler.brma . . . 76
brma . . . 77
brma.glmm . . . 84
brma.mv . . . 90
bselmodel . . . 98
bselmodel.mv . . . 103
check_loo.brma . . . 111
coef.brma . . . 112
contr.BayesTools . . . 112
cooks.distance.brma . . . 113
covratio.brma . . . 115
density_diagnostics . . . 116
dfbetas.brma . . . 120
dffits.brma . . . 122
estimate_unit_information_sd . . . 123
fitted.brma . . . 124
forest.brma . . . 126
funnel . . . 129
hatvalues.brma . . . 134
Havrankova2025 . . . 135
hist.zplot_brma . . . 136
Hoogeveen2023 . . . 137
Hoppen2025 . . . 138
hypothesis . . . 139
influence.brma . . . 148
interpret . . . 149
Johnides2025 . . . 151
known_v_factor . . . 151
Kroupova2021 . . . 153
lines.zplot_brma . . . 154
logLik.brma . . . 156
logml.brma . . . 157
log_lik . . . 158
loo.brma . . . 159
loo_compare.brma . . . 160
loo_compare.loo . . . 161
loo_model_weights.brma . . . 162
loo_weights.brma . . . 164
Lui2015 . . . 165
ManyLabs16 . . . 165
marginal_means . . . 166
marginal_means.brma . . . 167
nobs.brma . . . 170
plot.brma . . . 171

plot.marginal_means.brma . . . 176
plot.zplot_brma . . . 178
plot_diagnostic . . . 180
plot_pet_peese . . . 182
plot_prior . . . 184
plot_weightfunction . . . 187
pooled_effect . . . 189
pooled_effect.brma . . . 189
pooled_heterogeneity . . . 191
pooled_heterogeneity.brma . . . 192
post_prob.brma . . . 194
Poulsen2006 . . . 195
predict.brma . . . 196
print.brma_samples . . . 201
print.marginal_means.brma . . . 202
print.RoBMA_data . . . 202
print.summary.brma_samples . . . 203
print.summary.marginal_means.brma . . . 204
print.summary.zplot_brma . . . 204
print.summary_heterogeneity.brma . . . 205
print.vif.brma . . . 205
print_prior . . . 206
prior . . . 207
prior_factor . . . 208
prior_informed . . . 209
prior_none . . . 210
prior_ordered . . . 211
prior_PEESE . . . 212
prior_PET . . . 213
prior_random . . . 214
prior_weightfunction . . . 221
qqnorm.brma . . . 223
radial.brma . . . 226
random_effect_formula_tags . . . 229
ranef . . . 232
ranef.brma . . . 233
regplot.brma . . . 235
remove_selection_cache . . . 240
residuals.brma . . . 242
RoBMA . . . 245
RoBMA.mv . . . 252
RoBMA_control . . . 260
RoBMA_options . . . 262
rstandard.brma . . . 265
rstudent.brma . . . 267
selection_cache_info . . . 268
selection_model . . . 270
selection_sampler_info . . . 272

selection_sensitivity_diagnostics . . . 273
set_selection_likelihood_control . . . 275
summary.brma . . . 278
summary.brma_samples . . . 281
summary.marginal_means.brma . . . 281
summary.zplot_brma . . . 282
summary_heterogeneity . . . 283
summary_heterogeneity.brma . . . 284
summary_models . . . 286
true_effects . . . 288
true_effects.brma . . . 288
update.brma . . . 290
vif . . . 292
vif.brma . . . 292
waic.brma . . . 295
Wang2025 . . . 296
Weingarten2018 . . . 297
zplot.brma . . . 298

Index **300**

`RoBMA-package` *RoBMA: Robust Bayesian Meta-Analysis*

Description

RoBMA provides Bayesian meta-analysis, meta-regression, multilevel meta-analysis, model averaging, and publication-bias adjustment. The main user-facing fitters are `RoBMA`, `RoBMA.mv`, `BMA`, `BMA.mv`, `brma`, `brma.mv`, `brma.glmm`, `bselmodel`, `bselmodel.mv`, `bPET`, `bPEESE`, `bPET.mv`, and `bPEESE.mv`.

User guide

See Bartoš et al. (2023), Maier et al. (2023), Bartoš et al. (2022), and Bartoš et al. (2026) for the RoBMA methodology. Use `vignette(package = "RoBMA")` to list installed vignettes.

Author(s)

Frantisek Bartos <f.bartos96@gmail.com>

References

Bartoš F, Maier M, Quintana DS, Wagenmakers E (2022). "Adjusting for publication bias in JASP and R — Selection models, PET-PEESE, and robust Bayesian meta-analysis." *Advances in Methods and Practices in Psychological Science*, **5**(3), 1–19. doi:10.1177/25152459221109259.

Bartoš F, Maier M, Wagenmakers E (2026). "Robust Bayesian multilevel meta-analysis: Adjusting for publication bias in the presence of dependent effect sizes." *Behavior Research Methods*.

doi:10.31234/osf.io/9tgp2_v2. Preprint available at https://doi.org/10.31234/osf.io/9tgp2_v2.

Bartoš F, Maier M, Wagenmakers E, Doucouliagos H, Stanley TD (2023). "Robust Bayesian meta-analysis: Model-averaging across complementary publication bias adjustment methods." *Research Synthesis Methods*, **14**(1), 99–116. doi:10.1002/jrsm.1594.

Maier M, Bartoš F, Wagenmakers E (2023). "Robust Bayesian Meta-Analysis: Addressing publication bias with model-averaging." *Psychological Methods*, **28**(1), 107–122. doi:10.1037/met0000405.

See Also

Useful links:

- https://fbartos.github.io/RoBMA/
- Report bugs at https://github.com/FBartos/RoBMA/issues

`add_loo.brma` *Add LOO-PSIS to brma Objects*

Description

Compute approximate leave-one-out cross-validation (LOO-CV) using Pareto smoothed importance sampling (PSIS) for brma model objects and store the result in the object.

Usage

```
## S3 method for class 'brma'
add_loo(object, unit = "estimate", r_eff = NULL, parallel = FALSE, ...)
```

Arguments

`object`	a brma model object.
`unit`	output/deletion unit. `"estimate"` computes one contribution per effect-size estimate. `"cluster"` computes one contribution per cluster and is available only for multilevel models fitted through the specialized clustered interface.
`r_eff`	optional vector of relative effective sample sizes. If not provided, it is computed from the log-likelihood values.
`parallel`	Logical. If `TRUE`, `loo::relative_eff()` and `loo::loo()` use `RoBMA.get_option("max_cores")`. Log-likelihood construction is unchanged. If `FALSE`, those computations use one core.
`...`	additional arguments (currently ignored).

Details

With `unit = "estimate"`, LOO-CV is computed with one contribution per effect-size estimate. For binomial and Poisson models, each pair of counts (ai/ci or x1i/x2i) that defines a single effect size estimate is treated as one contribution. Estimate-unit GLMM log-likelihoods integrate the nuisance base rate or log-rate over its prior: point priors and untruncated beta (base rate) and normal (log-rate) priors adaptively, other priors on a refined prior-CDF grid; a uniform(0, 1) base-rate prior is a Beta(1, 1) prior and is integrated adaptively. The grid is accepted only when its log-likelihood changes by less than `1e-6` between refinements. Any nuisance prior without an adaptive route (other than an untruncated beta base rate or an untruncated normal log-rate; truncated ones take the grid too) can fail to reach this, especially with few events or very large arms, and then stops with an error of class `RoBMA_glmm_grid_nonconvergence` (also of class `RoBMA_glmm_numerical_error`); gamma, lognormal, and inverse-gamma log-rate priors and lognormal base-rate priors are examples. A beta base-rate prior or a normal log-rate prior avoids it. When the likelihood is computed automatically after fitting (the `autocompute.loo` and `autocompute.waic` options), such a failure gives a warning and the fit is returned without LOO/WAIC; `add_loo()` then gives the same error.

With `unit = "cluster"`, LOO-CV is computed with one joint contribution per cluster. For normal models without selection, weighted or not, this uses the closed-form Gaussian integral over the held-out cluster effect, with the likelihood weights applied as powers of the row densities. Cluster-unit binomial and Poisson GLMM log-likelihoods are unavailable until certified nested adaptive quadrature is implemented; use `unit = "estimate"` for GLMMs. Cluster-unit contributions are also unavailable for `brma.mv()` models, whose random-formula deletion semantics still need a dedicated design; use `unit = "estimate"` there.

Selection-model deletion conditions the Gaussian law on retained outcomes and on the context declared by the fitted selection model. It preserves the original publication event and selection standard errors. Product weights on retained outcomes cancel; best selection continues to depend on the best p-value among both retained and deleted outcomes. This finite-vector deletion score differs from prediction for an entirely new publication event.

For Gaussian multilevel and known-`V` `brma.mv()` models, estimate-unit LOO integrates Gaussian local effects and uses $p(y_i \mid y_{-i}, \theta)$ within each fitted dependency block. Thus deleting one estimate retains the other estimates in its cluster or random- effect block; it does not represent prediction for a wholly new group. Gaussian `brma(..., cluster = ...)` models with non-unit likelihood `weights` also integrate the cluster effect. Their score integrates the entire powered row density against the cluster-effect distribution given the retained weighted observations, including the power-dependent normalizing constant. It is not the power of an unweighted marginal density. Earlier stored weighted estimate-unit LOO results must be recomputed with `add_loo()`; the fitted posterior does not need refitting for this post-fit calculation. Selection and GLMM models retain their supported routes. Numerically unresolved weighted Gaussian retained moments stop with `RoBMA_weighted_gaussian_unavailable`, a `RoBMA_numerical_unavailable` condition.

For `brma.mv()` models with known random-effect group covariance `R`, the known `R` matrix and fitted random-effect covariance enter the marginal ZGZ' covariance through BayesTools' compiled random-effect metadata. The same calculation applies to every supported random structure.

Estimate-unit deletion retains the fitted grouping and known-`V` dependency structure; it does not represent deletion of an entire new random group. Models with no local effects, sampled local effects, marginalized local effects, or a mixture of sampled and marginalized effects remain comparable when their data hash, deletion unit, retained context, and likelihood target agree. The sampled/marginalized labels are retained as provenance rather than comparison keys.

The PSIS object is essential for model comparison via loo_compare and is automatically saved in the loo result. RoBMA stores target metadata so comparisons can reject mismatched data, unit, or retained-context targets. Each finite log-likelihood column observed to be constant across posterior draws uses exact uniform importance ratios. Such columns bypass relative-ESS and generalized-Pareto fitting, including when other columns vary. Their Pareto-k diagnostic is recorded as the sentinel zero, meaning constant ratios with no tail instability rather than an estimated Pareto shape. Finite varying columns are centered by a column-specific constant before relative-ESS and PSIS evaluation. The offset is restored exactly afterward, preserving the LOO target while avoiding probability-scale underflow.

Important for model comparison: When comparing models via loo_compare, the selection is based on expected predictive performance for the documented deletion target, not in-sample fit. For dependent estimates, deleting one estimate may retain other observations from its fitted dependency block; this is not new-group prediction.

Value

The brma object with the LOO result stored in `object[["loo"]][[unit]]`.

References

(Vehtari et al. 2017) (Vehtari et al. 2024)

See Also

loo.brma, loo, loo_compare, pareto_k_ids

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- bPET(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")

  fit <- add_loo(fit)
  loo_fit <- loo(fit)
  print(loo_fit)
  plot(loo_fit)
}

## End(Not run)
```

---

add_marglik.brma *Add Marginal Likelihood to brma Objects*

Description

Compute the marginal likelihood of a brma model using bridge sampling and store the result in the object.

Usage

```
## S3 method for class 'brma'
add_marglik(
  object,
  parallel = NULL,
  cores = NULL,
  repetitions = 1L,
  method = c("normal", "warp3"),
  maxiter = 10000L,
  silent = TRUE,
  ...
)
```

Arguments

`object`	a brma model object.
`parallel`	whether bridge-density evaluations run in parallel. The default, `NULL`, inherits the setting used to fit `object`.
`cores`	number of bridge-sampling worker processes. The default, `NULL`, inherits the fitted core count and is capped by `RoBMA.get_option("max_cores")`. It is used only when `parallel = TRUE`.
`repetitions`	number of independent bridge-sampling repetitions.
`method`	bridge transformation; either `"normal"` or `"warp3"`.
`maxiter`	maximum number of bridge iterations per repetition.
`silent`	whether bridge-sampling progress is suppressed. The typed iteration-limit diagnostic remains visible even when `silent = TRUE`.
`...`	reserved for future use. Unused arguments produce a warning.

Details

The marginal likelihood is computed using the `bridgesampling` package via the `BayesTools::JAGS_bridgesampling` wrapper. The result is stored in the object and can be extracted using `bridge_sampler.brma`. Proposal draws use the seed stored with the fit. When that seed is non-NULL, the caller's random-number state is restored after the call; a NULL fit seed uses and advances the caller's random-number stream. A finite bridge estimate remains available when observed iteration counts exceed `maxiter`. The stored result's `success` records finite availability, not certified convergence. Its optional `iteration_limit_exceeded` is `TRUE` for observed excess, `FALSE` for complete counts without excess, and `NA` or absent when telemetry is unavailable. Counts may include upstream retries. Observed excess raises `BayesTools_marglik_iteration_limit`, inheriting from `BayesTools_marglik_diagnostic`, under either `silent` setting. Inspect `print(object[["marglik"]])` and its repetition diagnostics. `bridge_sampler(object)` returns the original upstream bridge object; it does not promise the stored wrapper's optional telemetry field.

Product-space model-averaging objects (`BMA.norm`, `BMA.glmm`, `BMA.mv`, `RoBMA`, and `RoBMA.mv`) do not expose a bridge-sampling marginal likelihood; use predictive comparison methods such as `loo.brma` instead. For a single model whose bridge parameters are all fixed by point priors,

the log marginal likelihood is evaluated exactly at those fixed values; no bridge-sampling repetitions are required. For `brma.mv()` known-`V` objects, bridge sampling evaluates the joint likelihood corresponding to the fitted known-`V` backend, not the conditional estimate-wise target used by LOO/WAIC diagnostics. Sampled Gaussian location random-effect blocks are integrated exactly during bridge evaluation as their draw-specific ZGZ' covariance. Their SD, allocation, and correlation parameters remain bridge coordinates; only the standardized latent effects are removed. This is a reparameterization of the same marginal-likelihood target, not a likelihood approximation. Fitted estimate-level marginalized blocks remain in the diagonal row variance. For Gaussian models fitted with the specialized `cluster` interface, the standardized cluster effects are likewise integrated exactly. The bridge retains total heterogeneity and its allocation while evaluating the implied diagonal-plus-cluster-rank-one covariance, including row-specific scale regression and likelihood weights. Selection models retain every context that the fitted model conditions on, including its normalized prior, and integrate only the Gaussian sources declared by that model. The likelihood uses the same full publication events and integration plan as fitting. A context integrated before selection normalization cannot be replaced by a retained-context parameterization. With `known_sampling_variance = "condition"`, the bridge retains the independent Gaussian augmentation coordinates and their normalized priors. The complete sampling error and random effects are reconstructed for each proposed bridge state before evaluating its selection normalizer.

A non-finite bridge repetition raises a `RoBMA_marglik_repetition_failure` error, inheriting from `RoBMA_marglik_error`. Its `parent` retains the complete upstream condition, including the failed `repetitions` table. Increase `maxiter` when a failed repetition reached its iteration limit; otherwise inspect the posterior samples and priors for non-finite densities. Fitting more posterior draws (for example with a larger `sample` or `update()`) may help in either case. These actions do not guarantee that the diagnostics will pass.

A checked posterior sample that cannot form a full-rank bridge proposal raises `RoBMA_marglik_rank_deficiency`, inheriting from `RoBMA_marglik_error`. Its unchanged upstream `parent` and exposed `reason`, `rank`, `n_coordinates`, `n_draws`, and `dependent_coordinates` describe the observed sample. For `reason = "insufficient_draws"`, fit more posterior draws with a larger `sample` or `update()`; dependence may persist. For `reason = "dependent_coordinates"`, inspect the parameterization and sampling; more draws alone need not resolve dependence. The message names the first reported pivot coordinates; they identify one dependent set, not uniquely faulty nodes.

The stored result records an identity of the data, weights, priors, and model it was computed for. `bridge_sampler()`, `logml()`, `bf()`, and `post_prob()` refuse a stored result that no longer matches the object, or that an earlier RoBMA version stored, with an error of class `RoBMA_marglik_stale` (inheriting from `RoBMA_marglik_error`); recompute it with `add_marglik()`. `update(slab = )` keeps it, and extending the fit with `update()` recomputes or drops it.

Value

The brma object with the marginal likelihood result stored in `object[["marglik"]]`.

See Also

`bridge_sampler.brma`, `logml.brma`, `bf.brma`, `post_prob.brma`

Examples

```
## Not run:
```

```
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- brma(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")

  fit <- add_marglik(fit)

  bridge <- bridge_sampler(fit)
  print(bridge)

  logml(fit)
}

## End(Not run)
```

`add_waic.brma` *Add WAIC to brma Objects*

Description

Compute the Widely Applicable Information Criterion (WAIC) for brma model objects and store the result in the object.

Usage

```
## S3 method for class 'brma'
add_waic(object, unit = "estimate", ...)
```

Arguments

`object`	a brma model object.
`unit`	output/deletion unit. See `add_loo`; the same accepted values and multilevel constraint apply.
`...`	additional arguments passed to `waic`.

Details

WAIC is an alternative to LOO-CV for estimating out-of-sample predictive accuracy. Like LOO, it evaluates expected predictive performance for new observations.

In most cases, LOO-PSIS (via `add_loo`) is preferred over WAIC because it provides better estimates and includes diagnostics (Pareto k values) that indicate when the approximation may be unreliable.

For Gaussian multilevel and known-V `brma.mv()` models, estimate-unit WAIC uses the same integrated conditional log-likelihood matrix as estimate-unit LOO, $p(y_i \mid y_{-i}, \theta)$. Estimate deletion retains the fitted grouping and dependency structure. Weighted Gaussian cluster models integrate the powered row likelihood, with its normalizing constant, over the retained-data cluster-effect distribution. Earlier stored weighted estimate-unit WAIC results require recomputation with `add_waic()`.

For `brma.mv()` models with known random-effect group covariance `R`, the known `R` matrix and fitted random-effect covariance enter through BayesTools' metadata-defined marginal ZGZ' covariance. The same four-field comparison policy applies across sampled, marginalized, and mixed local-effect representations.

Value

The brma object with the WAIC result stored in `object[["waic"]][[unit]]`.

See Also

`waic.brma`, `add_loo`, `waic`

`AIC.brma` *Unsupported Information Criteria for brma Objects*

Description

`brma` objects expose posterior pointwise log-likelihood draws (`logLik.brma`, `log_lik`), not a maximized scalar log-likelihood. AIC and BIC are therefore undefined and these methods always stop with an explicit error.

Usage

```
## S3 method for class 'brma'
AIC(object, ..., k = 2)

## S3 method for class 'brma'
BIC(object, ...)
```

Arguments

`object`	a brma model object.
`...`	ignored.
`k`	ignored; retained for the `stats::AIC()` method signature.

Value

These methods do not return a value.

See Also

`log_lik`, `logLik.brma`, `loo.brma`, `waic.brma`

`Anderson2010` *23 experimental studies from Anderson et al. (2010) that meet the best practice criteria*

Description

The data set contains correlation coefficients, sample sizes, and labels for 23 experimental studies focusing on the effect of violent video games on aggressive behavior.

Usage

```
Anderson2010
```

Format

A data.frame with 3 columns and 23 observations:

`r` Correlation coefficient.

`n` Sample size.

`name` Study label.

Source

https://github.com/Joe-Hilgard/Anderson-meta

References

Anderson CA, Shibuya A, Ihori N, Swing EL, Bushman BJ, Sakamoto A, Rothstein HR, Saleem M (2010). "Violent video game effects on aggression, empathy, and prosocial behavior in Eastern and Western countries: A meta-analytic review." *Psychological Bulletin*, **136**(2), 151–173. doi:10.1037/a0018251.

`Andrews2021` *39 study rows on household chaos and child executive functions from a meta-analysis by Andrews et al. (2021)*

Description

The data set contains raw correlation coefficients and sampling information using **metafor**-compatible column names (`ri`, `vi`, `sei`, `ni`, and `slab`), together with study-level moderators from the original data set. Three source rows have missing effect-size inputs and are omitted automatically by model-fitting functions that require complete outcomes. The original data set assessed the effect of household chaos on child executive functions (Andrews et al. 2021) and was used as an example in Bartoš et al. (2025).

Usage

```
Andrews2021
```

Format

A data.frame with 15 columns and 39 observations:

`study_id` Source row identifier.

`slab` Study label.

`ri` Raw correlation coefficient.

`vi` Sampling variance of the raw correlation coefficient.

`sei` Sampling standard error of the raw correlation coefficient.

`ni` Sample size.

`year` Publication year.

`percent_female` Percentage of female children in the sample.

`age` Mean age of the children in years.

`assessment_interval_months` Time between household-chaos and executive-function assessments in months.

`dissertation` Whether the study was a dissertation.

`percent_minority` Percentage of minority participants in the sample.

`percent_low_parental_education` Percentage of parents with high school, GED, or lower education.

`hc_dimension` Household-chaos dimension using the source coding.

`measure` Executive-function assessment type, direct or informant.

References

Andrews K, Atkinson L, Harris M, Gonzalez A (2021). "Examining the effects of household chaos on child executive functions: A meta-analysis." *Psychological Bulletin*, **147**(1), 16–32. doi:10.1037/bul0000311.

Bartoš F, Maier M, Stanley TD, Wagenmakers E (2025). "Robust Bayesian meta-regression: Model-averaged moderation analysis in the presence of publication bias." *Psychological Methods*. doi:10.1037/met0000737.

`as.data.frame.brma` *Convert brma Summaries to a Data Frame*

Description

Converts the non-empty tables displayed by a fitted `brma` object or its `summary.brma` result to one plain, long-form data frame. The leading `component` column identifies the summary section and `parameter` retains the displayed row label. Printed quantile labels are returned as syntactic `CI_` column names.

Usage

```
## S3 method for class 'brma'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  stringsAsFactors = FALSE,
  ...
)

## S3 method for class 'summary.brma'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  stringsAsFactors = FALSE,
  ...
)
```

Arguments

`x`	a fitted `brma` or `summary.brma` object.
`row.names`	`NULL` or a character vector giving the row names.
`optional`	logical; passed to the final data-frame coercion.
`stringsAsFactors`	accepted for compatibility with `data.frame()`.
`...`	for a fitted object, additional arguments passed to `summary.brma()`; otherwise unused.

Value

A plain `data.frame` containing all displayed summary rows. An object created with `only_data = TRUE` or `only_priors = TRUE` has no fitted model to export and stops with an error of class `RoBMA_unfitted_object` (also of class `RoBMA_unsupported_request`); `summary()` shows the data of an `only_data` object, and the summary of an `only_priors` object shows the priors in its `priors` element.

`as.data.frame.brma_samples` *Convert brma_samples Results to Data Frames*

Description

`as.data.frame()` and `data.frame()` of a `brma_samples` result return its posterior draws (one row per draw, one column per parameter), as for the underlying matrix. The posterior summary table displayed by `print()` is `as.data.frame(summary(x))`: a long data frame with component and parameter identifiers (multi-component paths use `/` as a separator), which is also the result for a multi-component `brma_samples_list`. Printed quantile labels such as `0.025` and `PI 0.025` are returned as the syntactic column names `CI_0.025` and `PI_0.025`, respectively.

Usage

```
## S3 method for class 'brma_samples'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'summary.brma_samples'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  stringsAsFactors = FALSE,
  ...
)

## S3 method for class 'brma_samples_list'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  format = c("long", "list"),
  stringsAsFactors = FALSE,
  ...
)
```

Arguments

`x`	a `brma_samples`, `summary.brma_samples`, or multi-component `brma_samples_list` result
`row.names`	`NULL` or a character vector giving the row names for the resulting data frame. Custom row names are unsupported when `format = "list"`.
`optional`	logical; passed to the final data-frame coercion method.
`...`	for a `brma_samples` result, additional arguments passed to the matrix coercion; for a `brma_samples_list`, additional arguments passed to `summary.brma_samples()`

`stringsAsFactors`
: logical; passed to the output data frame.

`format`
: for multi-component results, whether to return a single `"long"` data frame or a named, possibly nested `"list"` with a plain data frame at each leaf. Defaults to `"long"`.

Value

For an individual `brma_samples` object, a `data.frame` of the posterior draws. For a `summary.brma_samples` object, a plain long `data.frame` containing the displayed posterior summary statistics and leading `component` and `parameter` columns. For a `brma_samples_list`, a long data frame by default or, with `format = "list"`, a named, possibly nested list of such data frames.

---

as.data.frame.covratio.brma

Convert COVRATIO Results to a Data Frame

Description

Converts a `covratio.brma` vector to a component-aware long data frame.

Usage

```
## S3 method for class 'covratio.brma'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  stringsAsFactors = FALSE,
  ...
)
```

Arguments

`x`
: a `covratio.brma` object.

`row.names`
: `NULL` or a character vector giving the row names.

`optional`
: logical; passed to the final data-frame coercion.

`stringsAsFactors`
: accepted for compatibility with `data.frame()`.

`...`
: unused additional arguments.

Value

A plain `data.frame` with leading `component` and `parameter` columns and a numeric `value` column.

as.data.frame.dfbetas.brma

Convert DFBETAS Results to a Data Frame

Description

Converts a dfbetas.brma object to a component-aware long data frame while retaining its displayed coefficient columns.

Usage

```
## S3 method for class 'dfbetas.brma'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  stringsAsFactors = FALSE,
  ...
)
```

Arguments

x	a dfbetas.brma object.
row.names	NULL or a character vector giving the row names.
optional	logical; passed to the final data-frame coercion.
stringsAsFactors	accepted for compatibility with data.frame().
...	unused additional arguments.

Value

A plain data.frame with leading component and parameter columns.

as.data.frame.infl.brma

Convert Influence Results to a Data Frame

Description

Converts the influence and DFBETAS tables displayed by an infl.brma object to one component-aware long data frame.

Usage

```
## S3 method for class 'infl.brma'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  stringsAsFactors = FALSE,
  ...
)
```

Arguments

x	an infl.brma object.
row.names	NULL or a character vector giving the row names.
optional	logical; passed to the final data-frame coercion.
stringsAsFactors	accepted for compatibility with data.frame().
...	unused additional arguments.

Value

A plain data.frame with leading component and parameter columns.

---

as.data.frame.marginal_means.brma

Convert Marginal Means to a Data Frame

Description

Converts the summary displayed for a marginal_means.brma object to a plain long data frame with leading component and parameter columns. Printed quantile labels are returned as syntactic CI_ column names.

Usage

```
## S3 method for class 'marginal_means.brma'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  stringsAsFactors = FALSE,
  ...
)

## S3 method for class 'summary.marginal_means.brma'
as.data.frame(
```

```
  x,
  row.names = NULL,
  optional = FALSE,
  stringsAsFactors = FALSE,
  ...
)
```

Arguments

x	a marginal_means.brma or summary.marginal_means.brma object.
row.names	NULL or a character vector giving the row names.
optional	logical; passed to the final data-frame coercion.
stringsAsFactors	accepted for compatibility with data.frame().
...	additional arguments passed to summary().

Value

A plain long data.frame containing the displayed marginal-means summary and leading component and parameter columns.

---

as.data.frame.summary_heterogeneity.brma

Convert Heterogeneity Summaries to Data Frames

Description

Converts heterogeneity summaries to plain long data frames with leading component and parameter columns. A multi-component result can instead be returned as a named list with format = "list".

Usage

```
## S3 method for class 'summary_heterogeneity.brma'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  stringsAsFactors = FALSE,
  ...
)

## S3 method for class 'summary_heterogeneity.brma_list'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
```

```
  format = c("long", "list"),
  stringsAsFactors = FALSE,
  ...
)
```

Arguments

x	a summary_heterogeneity.brma or summary_heterogeneity.brma_list object.
row.names	NULL or a character vector giving the row names for the result. Custom row names are unsupported when format = "list".
optional	logical; passed to the final data-frame coercion.
stringsAsFactors	accepted for compatibility with data.frame().
...	unused additional arguments.
format	for multi-component results, whether to return one "long" data frame or a named "list" of data frames.

Value

A plain data.frame, or a named list of data frames when format = "list".

---

as.data.frame.summary_models.RoBMA

Convert Model-Weight Summaries to a Data Frame

Description

Converts every table displayed by a summary_models.RoBMA object to one component-aware long data frame.

Usage

```
## S3 method for class 'summary_models.RoBMA'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  stringsAsFactors = FALSE,
  ...
)
```

Arguments

x	a summary_models.RoBMA object.
row.names	NULL or a character vector giving the row names.
optional	logical; passed to the final data-frame coercion.
stringsAsFactors	accepted for compatibility with data.frame().
...	unused additional arguments.

Value

A plain data.frame with leading component and parameter columns.

---

as.data.frame.vif.brma

Convert VIF Results to a Data Frame

Description

Converts every table displayed by a vif.brma object to one component-aware long data frame.

Usage

```
## S3 method for class 'vif.brma'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  stringsAsFactors = FALSE,
  ...
)
```

Arguments

x	a vif.brma object.
row.names	NULL or a character vector giving the row names.
optional	logical; passed to the final data-frame coercion.
stringsAsFactors	accepted for compatibility with data.frame().
...	unused additional arguments.

Value

A plain data.frame with leading component and parameter columns.

`as.data.frame.zplot_brma` *Convert Z-Plot Summaries to a Data Frame*

Description

Converts the table displayed for a `zplot_brma` or `summary.zplot_brma` object to a component-aware long data frame. Printed quantile labels are returned as syntactic `CI_` column names.

Usage

```
## S3 method for class 'zplot_brma'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  stringsAsFactors = FALSE,
  ...
)

## S3 method for class 'summary.zplot_brma'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  stringsAsFactors = FALSE,
  ...
)
```

Arguments

`x`	a `zplot_brma` or `summary.zplot_brma` object.
`row.names`	`NULL` or a character vector giving the row names.
`optional`	logical; passed to the final data-frame coercion.
`stringsAsFactors`	accepted for compatibility with `data.frame()`.
`...`	additional arguments passed to `summary()` for a `zplot_brma` object and otherwise unused.

Value

A plain `data.frame` with leading `component` and `parameter` columns.

as.matrix.brma_samples *Convert brma_samples to Matrix*

Description

Converts a `brma_samples` object to a plain matrix, removing all brma_samples-specific attributes.

Usage

```
## S3 method for class 'brma_samples'
as.matrix(x, ...)
```

Arguments

`x`	a `brma_samples` object
`...`	additional arguments (ignored)

Value

A plain matrix of posterior samples.

as_draws.brma *Convert brma Objects to posterior Draws Formats*

Description

Provides an interface to the **posterior** package for `brma` objects. These functions convert the MCMC samples from a fitted `brma` model to various draws formats supported by the **posterior** package, enabling the use of posterior's rich set of diagnostics and summary functions.

Usage

```
as_draws(x, ...)

as_draws_array(x, ...)

as_draws_df(x, ...)

as_draws_list(x, ...)

as_draws_matrix(x, ...)

as_draws_rvars(x, ...)
```

```
## Default S3 method:
as_draws(x, ...)

## Default S3 method:
as_draws_array(x, ...)

## Default S3 method:
as_draws_df(x, ...)

## Default S3 method:
as_draws_list(x, ...)

## Default S3 method:
as_draws_matrix(x, ...)

## Default S3 method:
as_draws_rvars(x, ...)

## S3 method for class 'brma'
as_draws(x, include_auxiliary = FALSE, ...)

## S3 method for class 'brma'
as_draws_array(x, include_auxiliary = FALSE, ...)

## S3 method for class 'brma'
as_draws_df(x, include_auxiliary = FALSE, ...)

## S3 method for class 'brma'
as_draws_list(x, include_auxiliary = FALSE, ...)

## S3 method for class 'brma'
as_draws_matrix(x, include_auxiliary = FALSE, ...)

## S3 method for class 'brma'
as_draws_rvars(x, include_auxiliary = FALSE, ...)
```

Arguments

x	an object to convert. The brma methods expect a fitted brma object; default methods forward non-brma objects to the corresponding **posterior** conversion function.
...	additional arguments passed to the corresponding **posterior** function.
include_auxiliary	logical; whether to include the hidden intermediate parameters (see Details) in addition to the parameters shown by the default print() and summary() of the fit. Formula-random parameters retain their RoBMA semantic names in either case.

Details

By default, the draws hold exactly the parameters that the default `print()` and `summary()` of the fit show (the location, heterogeneity, and moderator coefficients, the multilevel allocation `rho`, and the publication-bias parameters), together with the model-inclusion indicators behind the reported inclusion probabilities. The hidden intermediate parameters are added by `include_auxiliary = TRUE`: the latent cluster and estimate effects (`gamma[k]` and the GLMM `theta[i]`), the selection-model latent variables (`eta`), the GLMM nuisance base rates and rates (`pi[i]` and `phi[i]`), the random-effect variances, component SDs derived from a variance allocation, and total variance of formula-random models (for example `tau2` or `tau2_total`), and the other backend variables listed below.

These functions are S3 generics. Their default methods forward to the corresponding **posterior** generics so attaching **RoBMA** preserves the usual **posterior** behavior for non-`brma` objects.

The following conversion functions are available:

- `as_draws`: converts to the default draws format
- `as_draws_array`: converts to a 3-D array (iteration x chain x variable)
- `as_draws_df`: converts to a data frame with columns for iterations, chains, and variables
- `as_draws_list`: converts to a list of lists
- `as_draws_matrix`: converts to a 2-D matrix (draw x variable)
- `as_draws_rvars`: converts to random variable objects

These methods require the **posterior** package to be installed. For `brma` methods, conversion is performed by first extracting the MCMC samples as a `mcmc.list` object and then using the corresponding **posterior** conversion function. By default, backend-private latent effects, known-`V` dependency factors, random-effect simulation and Cholesky factors, and prior-parameterization variables are omitted. Stable fitted coordinates such as model indicators remain available. Ordinary spike-and-slab backend slab and inclusion coordinates are omitted by default; `include_auxiliary = TRUE` retains them. Semantic random-effect indicators remain available as described below. Formula-random coordinates are replaced by their semantic RoBMA quantities using the same names as summaries, plots, and hypotheses, such as `tau_total`, `tau_common`, `rho(...)`, and `tau2_prop(...)`. A `rho(...)` column can contain `NA` draws: an original-scale correlation is undefined in draws where one of its standard deviations is zero, e.g. where an inclusion gate or a point prior sets it to zero. Summaries and hypothesis tests of the correlation use its defined draws; summaries computed from the exported draws need to handle the `NA` values. Random-effect SD switches are exposed as exact columns named after the standard deviation they act on, with an `_indicator` suffix, pairing with the semantic SD column the same way `mu_x_indicator` pairs with `mu_x`. A heterogeneity-component inclusion gate is 0/1, such as `study: tau_indicator` and `observation: tau_indicator`, and carries the inclusion state the summary reports under `Heterogeneity:`. An allocation-SD prior indicator, such as `tau_total_indicator`, records which mixture component allocated the total standard deviation and therefore takes one value per component, matching the order of that prior's components. Ordered predictor terms return their fitted totals and normalized sampled allocation shares by default, replacing their deterministic increment coordinates. Shares are normalized from the sampled Gamma primitives and remain defined when a total is zero or its inclusion event is off. Totals with point priors are exact constants for every fitted slice; fixed allocations do not create share columns. The labels come from the fitted BayesTools ordered specification, including reached factor levels and interaction slices. These quantities remain on the fitted predictor scale. Ordered-total mixture indicators and inclusion probabilities remain available,

while nested slab and pseudo-prior coordinates are omitted. With `include_auxiliary = TRUE`, raw ordered increments and eligible total auxiliaries are also retained. Internal Gamma coordinates are never exported. `as_draws_rvars()` preserves allocation labels as scalar random variables and keeps ordinary vector and multi-slice total indices. If an ordered sampled parameter conflicts with an existing fitted column, conversion stops with classes `RoBMA_ordered_draws_conflict` and `RoBMA_ordered_draws_unavailable`, with the message `Ordered sampled parameter '<name>' conflicts with existing fitted draws. Refit the model with the current RoBMA/BayesTools build.` Missing required ordered sources stop with `RoBMA_ordered_draws_unavailable` and `BayesTools_ordered_coord` with parent class `BayesTools_ordered_unavailable` and the message `Ordered source draws are unavailable. Refit the model with the current RoBMA/BayesTools build.` Entirely fixed numeric-point terms need only the existing chain row count to materialize their declared totals. Both RoBMA ordered refit errors also inherit `RoBMA_refit_required` and `BayesTools_refit_required`. Structural point-prior coordinates are returned as exact constant columns, with chain timing taken from the fitted draw geometry. Private backend anchors are never exposed. Use `include_auxiliary = TRUE` to skip RoBMA auxiliary filtering; added backend variables retain their coordinate names, while public random-effect parameters retain their RoBMA names. `brma_samples` objects have separate methods documented at `as_draws.brma_samples`.

Value

An object of the corresponding **posterior** draws class.

See Also

`draws`, `brma`, `as_draws.brma_samples`

---

as_draws.brma_samples *Convert brma_samples to posterior Draws Formats*

Description

Provides an interface to the **posterior** package for `brma_samples` objects. These functions convert the posterior samples to various draws formats supported by the **posterior** package.

Usage

```
## S3 method for class 'brma_samples'
as_draws(x, ...)

## S3 method for class 'brma_samples'
as_draws_array(x, ...)

## S3 method for class 'brma_samples'
as_draws_df(x, ...)

## S3 method for class 'brma_samples'
as_draws_list(x, ...)
```

```
## S3 method for class 'brma_samples'
as_draws_matrix(x, ...)

## S3 method for class 'brma_samples'
as_draws_rvars(x, ...)
```

Arguments

x	a brma_samples object
...	additional arguments passed to the corresponding **posterior** function

Details

The conversion reconstructs the MCMC chain structure from the stored nchains and niter attributes. The samples are assumed to be ordered with chains concatenated (i.e., all iterations from chain 1, then all from chain 2, etc.). Conditional RoBMA samples are intentionally stored as one flattened chain because conditioning subsets posterior rows across chains. The generic as_draws() returns a draws_matrix, which retains the chain counts in its metadata; request as_draws_array() for the chain-by-iteration array.

Value

An object of the corresponding **posterior** draws class.

See Also

draws, as_draws.brma

---

as_metafor_forest *Prepare brma Data for metafor Forest Plots*

Description

as_metafor_forest extracts observed effect sizes, standard errors, study labels, and RoBMA posterior summary intervals in a structure that can be passed to **metafor**'s forest plotting functions.

Usage

```
as_metafor_forest(x, ...)

## S3 method for class 'brma'
as_metafor_forest(
  x,
  level = 95,
  addpred = FALSE,
  newdata = NULL,
```

```
  bias_adjusted = TRUE,
  conditional = FALSE,
  ...
)
```

Arguments

x	a fitted brma object.
...	additional named arguments stored in forest_args and later passed to forest.default. Shared graphical arguments are also stored in addpoly_args. This includes **metafor**'s auxiliary-column arguments ilab, ilab.lab, ilab.xpos, and ilab.pos.
level	credible/confidence level in percent, greater than 0 and less than 100. A value below 1 is read as a proportion (0.95 is 95 percent), so 1 means 1 percent. Defaults to 95.
addpred	logical; whether to compute a marginal posterior prediction interval for one new true effect. Defaults to FALSE.
newdata	optional one-row prediction design used when addpred = TRUE. When omitted, the interval is evaluated at the average expanded design using pooled_effect(). When supplied, each row passed to predict.brma() is one new true effect; with bias_adjusted = FALSE, it targets a selected new publication. The pooled reference declares no new-publication SE or selection event. The forest adapter accepts exactly one because it draws one summary polygon.
bias_adjusted	logical; whether pooled and prediction summaries should adjust for publication bias. Defaults to TRUE.
conditional	logical; whether to return conditional posterior summaries for RoBMA product-space objects. Defaults to FALSE.

Value

A list of class metafor_forest.brma with study-level data, pooled posterior summaries, and ready-to-use argument lists for forest.default and addpoly.default. The forest_args include enough vertical space for the default summary row. The returned object can also be plotted directly with forest.

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE) &&
    requireNamespace("metafor", quietly = TRUE)) {
  data(dat.bcg, package = "metadat")
  dat <- metafor::escalc(
    measure = "RR",
    ai      = tpos,
    bi      = tneg,
    ci      = cpos,
    di      = cneg,
    data    = dat.bcg
  )
```

```
  fit <- brma(yi = yi, vi = vi, data = dat, measure = "RR")
  count_columns <- as.matrix(dat[, c("tpos", "tneg", "cpos", "cneg")])
  forest_data <- as_metafor_forest(
    fit,
    addpred   = TRUE,
    slab      = paste(dat$author, dat$year),
    ilab      = count_columns,
    ilab.lab  = c("T+", "T-", "C+", "C-"),
    ilab.xpos = c(-7, -6, -5, -4),
    xlim      = c(-10, 4),
    atransf   = exp
  )
  metafor::forest(forest_data, predstyle = "bar")

  # Or draw the metafor calls directly:
  do.call(metafor::forest.default, forest_data$forest_args)
  do.call(metafor::addpoly.default, forest_data$addpoly_args)
}

## End(Not run)
```

`as_zplot.brma` *Transform brma Object to Zplot*

Description

Transforms an estimated brma model into a zplot object that can be summarized and plotted to assess replicability.

Usage

```
## S3 method for class 'brma'
as_zplot(
  object,
  significance_level = stats::qnorm(0.975),
  max_samples = 10000,
  conditioning_depth = "marginal",
  integration_control = set_selection_likelihood_control(),
  ...
)
```

Arguments

`object`	a normal-outcome `brma` object.
`significance_level`	positive z-value threshold for significance. Defaults to `qnorm(0.975)` (two-sided alpha = 0.05). The threshold must be finite and its two-sided alpha, `2`

* pnorm(significance_level, lower.tail = FALSE), strictly between 0 and 1. Zero (alpha = 1), a threshold so small that its alpha rounds to 1 (below about 1e-16), and one so large that its alpha underflows to 0 (above about 38.4) are refused because the Soric false discovery rate is undefined there.

max_samples
: maximum number of posterior samples for the EDR and missing-study summaries. Defaults to 10000. Use Inf to use all posterior samples. The density budget is supplied separately to plot() or lines() and defaults to 1000 for brma.mv models and 10000 for other models.

conditioning_depth
: predictive conditioning depth. Options are "marginal" (default), which integrates all newly realized random effects; "cluster", which retains fitted cluster effects and integrates within-cluster heterogeneity; and "estimate", which retains each fitted latent true effect, integrates its posterior conditional uncertainty, and adds new sampling error. Cluster conditioning is available only for multilevel models. Selection models currently support only "marginal" conditioning.

integration_control
: numerical integration settings created by set_selection_likelihood_control(). These control integration within each posterior draw, independently of max_samples. Factor blocks and supported full-covariance product-selection blocks use deterministic Gauss-Hermite quadrature where possible, with the existing adaptive randomized quasi-Monte Carlo fallback. Density-error checks use the peak within each posterior draw and also check the selection normalizer. Bounded CDF interpolation, Gaussian-mixture approximation, and omitted contributions are charged to the same numerical error budget.

...
: additional arguments (currently unused).

Details

Zplot is the pushforward into z-space of the design-conditional posterior predictive distribution. It retains the empirical mix of observed standard errors and moderators, evaluates one row-specific predictive distribution for each estimate, transforms each through $Z = Y/SE$, and averages the resulting densities with equal weight per estimate.

The default marginal target integrates every latent effect that would be newly realized in a replicated literature. For a specialized multilevel model fitted with cluster, the predictive variance before sampling error is the sum of between- and within-cluster heterogeneity. Cluster conditioning instead retains the fitted cluster effects and includes only within-cluster heterogeneity. Estimate conditioning uses the conditional true-effect means returned by predict(type = "blup") and additionally integrates the posterior conditional variance of those latent effects before adding new sampling error. It therefore represents a posterior predictive replication of the same uncertain latent effects rather than a BLUP-mean plug-in distribution. Thus the marginal zplot collapses the row-specific slices of the marginal funnel target onto the common significance scale. Unlike LOO diagnostics, it is a full-posterior predictive projection and does not leave observations or clusters out when estimating model parameters.

The Expected Discovery Rate (EDR) is the posterior average probability that a new outcome from the bias-adjusted reference distribution is statistically significant at the supplied threshold. It is therefore not an exact-repeat estimand unless conditioning_depth = "estimate" is requested.

The EDR uses the bias-adjusted reference distribution: selection and PET/PEESE offsets are removed. For multivariate selection models, models retaining estimate-level or sampling-error sources or integrating applicable contextual sources, and best-rule selection models, this reference is the normalized pre-selection Gaussian marginal, integrating all newly realized contexts. Its integral is one and EDR is its significance probability. Relative selection weights do not identify an absolute publication probability or missing count, so `Missing N` is unavailable for these targets.

The fitted selected curve preserves the original publication event and the fitted model's normalization order. It integrates the model's integrated sources inside each full-event normalizer and averages conditionally normalized projections over newly realized retained contexts. A best rule still selects the full original event when displaying one estimate. Supported full-covariance product blocks reuse numerator and normalizer calculations while refining their quadrature separately. Computation by posterior row and covariance block preserves the full publication event.

Numerical diagnostics distinguish raw normalizer-mass ratios from bounds on omitted contributions. A raw mass ratio does not certify that the compressed displayed curve integrates to one. An omitted-mass bound covers only the omitted components, not the total integrated error of the compressed curve. Retained component weights are not renormalized to hide omissions.

Independent univariate product-weight models and the released conditional cluster display retain their reference-bin-normalized inverse-weight extrapolation convention and corresponding `Missing N` summary. Product selection does not use publication groups; its covariance dependencies and source settings determine the reference distribution and diagnostic. Best-p-value selection preserves the resolved publication groups. A literal count interpretation additionally treats the reference bin as fully observed and requires a reporting/stopping design relating attempted to observed estimates. These assumptions are not supplied by the fitted relative-weight likelihood, which conditions on the reported vector and does not identify an absolute publication probability or finite missing pool. Valid above-one relative weights can give a negative diagnostic; such a value is not a literal missing count. Restricting a plot with `from` or `to` does not restrict the summary integrals.

Zplot diagnostics are available only for normal outcome models. GLMM objects are rejected because their raw likelihood is on a count scale while zplot diagnostics require observed effect-size z-statistics with standard errors. For correlated known-V `brma.mv()` models, zplot is a scalar display using $z_i = y_i/se_i$. The histogram does not display dependence, but the fitted selected marginals retain the full covariance. Numerical integration can be substantially more expensive for correlated selection than for ordinary Gaussian models. Density plotting uses its own `max_samples` argument, independently of the summaries stored here.

The resulting object retains all original brma properties while adding zplot results, enabling both standard meta-analytic summaries and zplot diagnostics on the same object.

Value

The input object with added class `"zplot_brma"` and a new `zplot` list component containing:

estimates a list with posterior samples for `EDR` and extrapolation `weights`, when a missing-count convention applies

data a list with `significance_level`, `conditioning_depth`, stored `integration_control`, observed `z`-statistics, `N_significant`, and `N_observed`

target the fitted reporting specification, reference distribution, source and publication identities, and missing-count applicability

See Also

summary.zplot_brma(), plot.zplot_brma(), hist.zplot_brma(), predict.brma(), funnel.brma(), regplot.brma()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- bPET(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")

  zfit <- as_zplot(fit)
  summary(zfit)
  plot(zfit)
}

## End(Not run)
```

Bem2011 — *9 experimental studies from Bem (2011) as described in Bem et al. (2011)*

Description

The data set contains Cohen's d effect sizes, standard errors, and labels for 9 experimental studies of precognition from the infamous Bem (2011) as analyzed in his later meta-analysis (Bem et al. 2011).

Usage

```
Bem2011
```

Format

A data.frame with 3 columns and 9 observations:

`d` Cohen's d effect size.

`se` Standard error of d.

`study` Study label.

References

Bem DJ (2011). "Feeling the future: Experimental evidence for anomalous retroactive influences on cognition and affect." *Journal of Personality and Social Psychology*, **100**(3), 407–425. doi:10.1037/a0021524.

Bem DJ, Utts J, Johnson WO (2011). "Must psychologists change the way they analyze their data?" *Journal of Personality and Social Psychology*, **101**(4), 716–719. doi:10.1037/a0024777.

`bf.brma` *Bayes Factor for brma Objects*

Description

Compute the Bayes factor comparing two brma models.

Usage

```
## S3 method for class 'brma'
bf(x1, x2, log = FALSE, ...)

## S3 method for class 'brma'
bayes_factor(x1, x2, log = FALSE, ...)
```

Arguments

`x1`	a brma model object (numerator).
`x2`	a brma model object (denominator).
`log`	logical; if `TRUE`, the log Bayes factor is returned. Default is `FALSE`.
`...`	additional arguments (currently not used).

Details

Computes the Bayes factor in favor of the model `x1` over the model `x2`. The marginal likelihoods must first be computed using `add_marglik`. Both models must be fitted to the same outcome target/data, including outcome type and, when present, likelihood weights. Cluster identifiers define model structure and are not part of the outcome-data identity. A known sampling covariance without correlated blocks (e.g., `V = diag(sei^2)` in `brma.mv`) is the outcome data of its standard errors, whereas a correlated `V` is a different target.

When either model contains repeated bridge estimates, the shorter sequence is recycled with a warning, following **bridgesampling**. Repetitions are compared in order. The median-based estimate is computed from the separate medians of the log marginal likelihoods before recycling, rather than the median of the repetition-wise Bayes factors. Standardized marginal-likelihood results must be finite. Legacy raw `"bridge_list"` results retain failed repetitions in the returned estimates, with upstream missing-value behavior when printed.

Value

For single estimates, a list of class `"bf_default"` with components:

- `bf`: (scalar) value of the Bayes factor in favor of `x1` over `x2`.
- `log`: Boolean indicating whether `bf` corresponds to the log Bayes factor.

For repeated estimates, a list of class `c("bf.brma", "bf_bridge_list")` with a vector `bf`, scalar `bf_median_based`, and `log`. It uses the upstream repeated-Bayes-factor print method. Both `as.data.frame()` and `data.frame()` return a long data frame with `component`, `parameter`, and full-precision numeric `value` columns, containing every repetition and the displayed median-based, range, and interquartile-range summaries.

See Also

`add_marglik`, `bridge_sampler.brma`, `post_prob.brma`, `logml.brma`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit1 <- brma(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")
  fit2 <- brma(
    yi           = yi,
    vi           = vi,
    data         = dat.lehmann2018,
    measure      = "SMD",
    prior_effect = FALSE
  )

  fit1 <- add_marglik(fit1)
  fit2 <- add_marglik(fit2)

  bf(fit1, fit2)
}

## End(Not run)
```

`blup` *Best Linear Unbiased Predictions (BLUPs)*

Description

Computes the estimated true effects (theta) from a fitted model. These correspond to Best Linear Unbiased Predictions (BLUPs) or empirical Bayes estimates.

Usage

```
blup(object, ...)
```

Arguments

`object`	a fitted model object
`...`	additional arguments passed to methods

Value

Method-specific return value, typically a summary table or posterior samples of BLUP or empirical-Bayes true-effect summaries.

See Also

predict.brma()

`blup.brma` *Best Linear Unbiased Predictions for brma Objects*

Description

Computes the estimated true effects (theta) for a fitted brma object. These correspond to Best Linear Unbiased Predictions (BLUPs) or empirical Bayes estimates.

Usage

```
## S3 method for class 'brma'
blup(
  object,
  bias_adjusted = FALSE,
  output_measure = NULL,
  transform = NULL,
  probs = c(0.025, 0.975),
  ...
)
```

Arguments

`object`	a fitted brma object
`bias_adjusted`	whether to adjust for publication bias. Defaults to `FALSE`, which returns estimates including publication bias effects (i.e., what we expect the true effects to be given the biased observations). Set to `TRUE` to obtain bias-corrected estimates.
`output_measure`	effect-size measure for location/effect predictions. Defaults to the fitted measure. Supported conversions are among `"SMD"`, `"COR"`, `"ZCOR"`, and `"OR"`; `"RR"`, `"HR"`, `"IRR"`, `"RD"`, and `"GEN"` can only be returned on their fitted measure. Use `transform = "EXP"` for ratio-scale output from log-scale measures.
`transform`	optional display transformation. Currently `"EXP"` exponentiates log-scale measures `"OR"`, `"RR"`, `"HR"`, and `"IRR"`.
`probs`	quantiles of the posterior distribution to be displayed. Defaults to `c(.025, .975)` for 95% credible intervals.
`...`	additional arguments passed to `predict.brma`; wrapper arguments such as `newdata`, `type`, `quiet`, `output_measure`, and `transform` are fixed by this method (supplying one is R's error for an argument matched twice, as for `predict()`), and other unknown arguments are unused and trigger a warning.

Details

This function is a convenience wrapper around `predict.brma(...,type = "blup", newdata = NULL)`. Unlike `predict(..., type = "estimate", conditioning_depth = "estimate")`, it returns conditional location/BLUP means rather than adding conditional latent-effect uncertainty.

For unweighted two-level normal models, true effects are computed using empirical Bayes shrinkage:

$$\theta_i = \lambda_i \cdot y_i + (1 - \lambda_i) \cdot \mu_i$$

where $\lambda_i = \tau^2/(\tau^2 + se_i^2)$. With likelihood weights, se_i^2 is replaced by the weighted sampling variance se_i^2/w_i.

For GLMM models (binomial, Poisson), the estimate-level random effects are extracted directly from the posterior samples.

For multilevel (3-level) normal models, cluster-level effects are estimated jointly within cluster blocks and estimate-level effects are then shrunk conditional on those cluster effects.

For known-V models, the conditional means require applying the inverse of the marginal covariance. A covariance that is too ill-conditioned for that (a reciprocal condition number of about `2e-10` or less in any posterior draw used) is unavailable: the call stops with an error of class `RoBMA_known_v_precision_unavailable`, whose parent class is `RoBMA_numerical_unavailable`, instead of returning a rounded number. The condition carries the draw positions (`rows`) and their `reciprocal_condition`.

Value

A `brma_samples` object containing posterior draws of BLUP or empirical-Bayes true-effect summaries with one column per estimate. For existing normal data, these are conditional BLUP means, not simulated latent-effect draws. When printed, displays a summary table. Use `summary()`, or `as.data.frame(summary(x))`, to obtain the summary table; `as.data.frame(x)` returns the posterior draws. The samples can be converted to **posterior** draws formats using `as_draws()`.

See Also

`predict.brma()`, `pooled_effect()`, `pooled_heterogeneity()`, `true_effects()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- brma(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )

  blup(fit)
}
```

```
## End(Not run)
```

BMA *Bayesian Model-Averaged Meta-Analysis*

Description

Fits Bayesian model-averaged meta-analytic models without publication-bias adjustment. `BMA()` is an alias for `BMA.norm()`.

Usage

```
BMA(
  yi,
  vi,
  sei,
  weights,
  ni,
  mods,
  scale,
  cluster,
  data,
  slab,
  subset,
  measure,
  prior_effect,
  prior_heterogeneity,
  prior_mods,
  prior_scale,
  prior_heterogeneity_allocation,
  prior_effect_null,
  prior_heterogeneity_null,
  prior_mods_null,
  prior_scale_null,
  prior_heterogeneity_allocation_null,
  standardize_continuous_predictors = TRUE,
  set_contrast_factor_predictors = "meandif",
  prior_unit_information_sd,
  rescale_priors = 1,
  prior_informed_field,
  prior_informed_subfield,
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  chains = 3,
```

```
  thin = 1,
  parallel = FALSE,
  autofit = FALSE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  seed = NULL,
  silent = TRUE,
  ...
)
```

Arguments

yi	a vector of effect sizes, or a formula with the effect size on the left-hand side and location moderators on the right-hand side (for example `yi ~ x1 + x2`). If a formula is supplied, `mods` must not be specified.
vi	a vector of sampling variances. Either `vi` or `sei` must be supplied for normal models.
sei	a vector of standard errors. Either `vi` or `sei` must be supplied for normal models.
weights	an optional vector of finite positive likelihood weights. For normal/effect-size models, each weight powers the estimate likelihood. For constructors with GLMM raw-count input, each weight powers the paired two-arm likelihood for one study. An integer weight w repeats that conditional likelihood contribution w times, with the declared latent-effect and grouping structure held fixed. Positive fractional likelihood powers are also supported.
ni	an optional vector of sample sizes. Used for `measure = "GEN"` or when estimating the unit information standard deviation.
mods	an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data` and supports explicit columns and standard formula operators, including `^` as R's crossing operator with a literal whole exponent of at least 2: `(x + z)^2` expands to `x + z + x:z`, and `x^2` is not a quadratic term (`x + x^2` is `x`). Precompute transformed predictors, such as a squared predictor, as columns in `data`; inline calls such as `I()`, `poly()`, spline constructors, and user functions are not supported.
scale	an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data` and follows the same explicit-column formula grammar as `mods`. Random-formula models also accept a named list targeting top-level random components or concrete random-effect blocks; see `brma.mv()`.
cluster	an optional vector of cluster identifiers for multilevel meta-analysis.
data	an optional data frame containing the variables.
slab	an optional vector of study labels.
subset	an optional logical or numeric vector specifying a subset of data to be used. Subsetting precedes missing-row removal; retained-row outcome and predictor checks follow these steps. Type, shape, and existing extraction checks can still reject malformed inputs before subsetting. Row selection (by `subset` or missing-row removal) drops factor levels left without rows from `mods`, `scale`,

and random-effect grouping factors. The index variable of a structured random-effect term (`cs()`, `hcs()`, `ar1()`, `har()` with one index variable) keeps its declared levels, which define the covariance structure; the variables of a multivariable `cs(a + b | group)` index follow the ordinary rule, because that index is built from their observed combinations. If an index variable that keeps its levels is also used in `mods` or `scale` and loses one of its levels to the selection, the fit stops with an error of class `RoBMA_structured_index_level_removed` (also `RoBMA_data_error`); remove the level from the data with `droplevels()` or use a separate column for the moderator.

`measure` a character string specifying the effect size measure. Normal/effect-size constructors require an explicit value and support `"SMD"`, `"ZCOR"`, `"RR"`, `"OR"`, `"HR"`, `"RD"`, `"IRR"`, and `"GEN"`. Use `"GEN"` only for general effect sizes without a known unit information standard deviation. GLMM raw-count constructors support only `"OR"` and `"IRR"` and default to `"OR"`.

`prior_effect` prior distribution(s) for the alternative effect component(s).

`prior_heterogeneity`
prior distribution(s) for the alternative heterogeneity component(s).

`prior_mods` prior distribution(s) for alternative moderator components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `mods` is specified. For a formula without an intercept (`~ 0 + x`), an `intercept` entry other than a point prior at zero stops with an error of class `RoBMA_prior_intercept_conflict` (also of class `RoBMA_prior_error`). A named alternative `intercept` must be a single simple or point prior; it is the complete intercept prior and replaces the global effect mixture. Specify it in one place: null-list intercepts and explicit global effect declarations left without a consumer are refused. Ordinary term entries replace only their alternative component.

`prior_scale` prior distribution(s) for alternative scale-regression components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `scale` is specified. A named alternative `intercept` must be a single positive simple or point prior; it is the complete SD prior used by `log(intercept)`. Global heterogeneity declarations remain meaningful when they supply another random root or its inclusion weights; declarations replaced without a consumer are refused. Null-list intercept declarations are unavailable.

`prior_heterogeneity_allocation`
prior distribution(s) for the alternative cluster-level heterogeneity allocation component(s). This argument can be specified only when `cluster` is specified.

`prior_effect_null`
prior distribution(s) for the null effect component(s).

`prior_heterogeneity_null`
prior distribution(s) for the null heterogeneity component(s).

`prior_mods_null`
prior distribution(s) for null moderator components. A named `intercept` entry, including `NULL` or `FALSE`, is unavailable: put the complete intercept prior in `prior_mods`, or use global prior arguments without a formula-list intercept. The same rule applies to `prior_scale_null`. A single prior applies to all

terms; a named list can specify term-specific components. This argument can be specified only when mods is specified. The null and the alternative prior of a factor term need the same contrast; a conflict stops with an error of class RoBMA_prior_contrast_conflict (also of class RoBMA_prior_error) that names the term and the remedies. The same rule applies to prior_scale_null.

prior_scale_null
: prior distribution(s) for null scale-regression components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when scale is specified.

prior_heterogeneity_allocation_null
: prior distribution(s) for the null cluster-level heterogeneity allocation component(s). This argument can be specified only when cluster is specified.

standardize_continuous_predictors
: logical. Whether to standardize continuous predictors. Defaults to TRUE.

set_contrast_factor_predictors
: character. How to set contrast for factor predictors. Defaults are constructor-specific and shown in each function usage. Single-model constructors use "treatment"; model-averaging constructors use "meandif". Accepted values are "treatment", "meandif", "orthonormal", and "independent". Independent coding creates one coefficient per factor level. Random-coefficient blocks reuse the resolved fixed-factor contrast unless random_block(contrasts = ...) overrides it. A warning is issued when default or inherited independent contrasts make a fixed or random design overspecified. With "meandif" or "orthonormal" contrasts, a formula term that interacts a factor with a predictor whose own term is missing, such as g:x in ~ g + g:x or ~ g / x, stops the fit: the term codes g by level indicators, with one coefficient per level, on which mean-difference and orthonormal priors are not defined. Specify an independent prior for that term, e.g., prior_mods = list("g:x" = prior_factor("normal", list(0, 1), contrast = "independent")) (or prior_scale for a scale formula), include the missing term (~ g * x), or use set_contrast_factor_predictors = "treatment".

prior_unit_information_sd
: numeric. The unit information standard deviation (σ_{unit}). Cannot be used together with prior_informed_field.

rescale_priors
: numeric. A scaling factor for supported prior distributions. Point and none priors are unchanged; supplied prior_random() scale architectures and publication-bias priors are not rescaled. The default PEESE prior separately uses its documented UISD adjustment, independently of rescale_priors. Defaults to 1.

prior_informed_field
: character. The field of the informed prior distributions. Omit to use the standard default prior specification; explicit NULL is invalid.

prior_informed_subfield
: character. The subfield of the informed prior distributions. Can only be specified together with prior_informed_field. Omit to use the field-specific default, such as "Cochrane" for prior_informed_field = "medicine"; explicit NULL is invalid.

sample
: numeric. Number of MCMC samples to save. Defaults to 5000.

burnin	numeric. Number of burn-in iterations. Defaults to 2000.
adapt	numeric. Number of adaptation iterations. Defaults to 500.
chains	numeric. Number of MCMC chains. Defaults to 3.
thin	numeric. Thinning interval. Defaults to 1.
parallel	logical. Whether to run MCMC chains in parallel. Defaults to FALSE.
autofit	logical. Whether to automatically extend the MCMC chains if convergence is not met. Defaults to FALSE.
autofit_control	list of autofit control settings. See set_autofit_control() for details.
convergence_checks	list of convergence check settings. See set_convergence_checks() for details.
seed	numeric. Random seed for reproducibility. Defaults to NULL.
silent	logical. Whether to suppress output. Constructors with no explicit default use RoBMA.get_option("silent") when silent is omitted. Model-averaging wrappers default to TRUE unless explicitly changed.
...	additional advanced arguments. Fitting functions reject unused arguments; currently recognized internal arguments include only_data, only_priors, is_JASP, and is_JASP_prefix.

Details

BMA.norm (and its alias BMA) provides a simplified interface for Bayesian model-averaged meta-analysis when publication bias adjustment is not needed. It uses the same product-space mixture-prior machinery as RoBMA() but omits selection models and PET-PEESE bias adjustment.

For publication bias adjusted meta-analysis, use RoBMA directly.

Value

A fitted object of class c("BMA.norm", "RoBMA", "brma"). The object contains checked data, checked mixture priors, the JAGS fit, cached summary, and cached coefficients.

See Also

RoBMA() brma() summary.brma() plot.brma()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")

  fit <- BMA(
    yi      = yi,
    vi      = vi,
    mods    = ~ Preregistered,
    data    = dat.lehmann2018,
    measure = "SMD",
```

```
    seed    = 1,
    silent  = TRUE
  )

  summary(fit)
}

## End(Not run)
```

BMA.glmm *Bayesian Model-Averaged Generalized Meta-Analysis*

Description

Function for fitting Bayesian model-averaged meta-analytic models directly to binary or count data using a generalized linear mixed model (GLMM) framework. Unlike RoBMA, this function does not adjust for publication bias, as weight function and regression-based bias adjustment methods are not available for GLMM outcomes.

Usage

```
BMA.glmm(
  ai,
  bi,
  ci,
  di,
  n1i,
  n2i,
  x1i,
  x2i,
  t1i,
  t2i,
  weights,
  mods,
  scale,
  cluster,
  data,
  slab,
  subset,
  measure = "OR",
  prior_effect,
  prior_heterogeneity,
  prior_mods,
  prior_scale,
  prior_heterogeneity_allocation,
  prior_baserate,
  prior_lograte,
```

```
  prior_effect_null,
  prior_heterogeneity_null,
  prior_mods_null,
  prior_scale_null,
  prior_heterogeneity_allocation_null,
  standardize_continuous_predictors = TRUE,
  set_contrast_factor_predictors = "meandif",
  prior_unit_information_sd,
  rescale_priors = 1,
  prior_informed_field,
  prior_informed_subfield,
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  chains = 3,
  thin = 1,
  parallel = FALSE,
  autofit = FALSE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  seed = NULL,
  silent = TRUE,
  ...
)
```

Arguments

ai	a vector of the number of events in the treatment or experimental group for binomial GLMM models.
bi	a vector of the number of non-events in the treatment or experimental group for binomial GLMM models.
ci	a vector of the number of events in the control group for binomial GLMM models.
di	a vector of the number of non-events in the control group for binomial GLMM models.
n1i	a vector of the sample size in the treatment or experimental group. If omitted for binomial GLMMs, it is computed as `ai + bi`.
n2i	a vector of the sample size in the control group. If omitted for binomial GLMMs, it is computed as `ci + di`.
x1i	a vector of the number of events in the treatment/experimental group (for Poisson data).
x2i	a vector of the number of events in the control group (for Poisson data).
t1i	a vector of the person-time in the treatment/experimental group.
t2i	a vector of the person-time in the control group.
weights	an optional vector of finite positive likelihood weights. For normal/effect-size models, each weight powers the estimate likelihood. For constructors with

GLMM raw-count input, each weight powers the paired two-arm likelihood for one study. An integer weight w repeats that conditional likelihood contribution w times, with the declared latent-effect and grouping structure held fixed. Positive fractional likelihood powers are also supported.

`mods` an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data` and supports explicit columns and standard formula operators, including `^` as R's crossing operator with a literal whole exponent of at least 2: `(x + z)^2` expands to `x + z + x:z`, and `x^2` is not a quadratic term (`x + x^2` is `x`). Precompute transformed predictors, such as a squared predictor, as columns in `data`; inline calls such as `I()`, `poly()`, spline constructors, and user functions are not supported.

`scale` an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data` and follows the same explicit-column formula grammar as `mods`. Random-formula models also accept a named list targeting top-level random components or concrete random-effect blocks; see `brma.mv()`.

`cluster` an optional vector of cluster identifiers for multilevel meta-analysis.

`data` an optional data frame containing the variables.

`slab` an optional vector of study labels.

`subset` an optional logical or numeric vector specifying a subset of data to be used. Subsetting precedes missing-row removal; retained-row outcome and predictor checks follow these steps. Type, shape, and existing extraction checks can still reject malformed inputs before subsetting. Row selection (by `subset` or missing-row removal) drops factor levels left without rows from `mods`, `scale`, and random-effect grouping factors. The index variable of a structured random-effect term (`cs()`, `hcs()`, `ar1()`, `har()` with one index variable) keeps its declared levels, which define the covariance structure; the variables of a multivariable `cs(a + b | group)` index follow the ordinary rule, because that index is built from their observed combinations. If an index variable that keeps its levels is also used in `mods` or `scale` and loses one of its levels to the selection, the fit stops with an error of class `RoBMA_structured_index_level_removed` (also `RoBMA_data_error`); remove the level from the data with `droplevels()` or use a separate column for the moderator.

`measure` a character string specifying the effect size measure. Normal/effect-size constructors require an explicit value and support `"SMD"`, `"ZCOR"`, `"RR"`, `"OR"`, `"HR"`, `"RD"`, `"IRR"`, and `"GEN"`. Use `"GEN"` only for general effect sizes without a known unit information standard deviation. GLMM raw-count constructors support only `"OR"` and `"IRR"` and default to `"OR"`.

`prior_effect` prior distribution(s) for the alternative effect component(s).

`prior_heterogeneity`
prior distribution(s) for the alternative heterogeneity component(s).

`prior_mods` prior distribution(s) for alternative moderator components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `mods` is specified. For a formula without an intercept (`~ 0 + x`), an `intercept` entry other than a point prior at zero stops with an error of class `RoBMA_prior_intercept_conflict` (also of class

RoBMA_prior_error). A named alternative intercept must be a single simple or point prior; it is the complete intercept prior and replaces the global effect mixture. Specify it in one place: null-list intercepts and explicit global effect declarations left without a consumer are refused. Ordinary term entries replace only their alternative component.

prior_scale prior distribution(s) for alternative scale-regression components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when scale is specified. A named alternative intercept must be a single positive simple or point prior; it is the complete SD prior used by log(intercept). Global heterogeneity declarations remain meaningful when they supply another random root or its inclusion weights; declarations replaced without a consumer are refused. Null-list intercept declarations are unavailable.

prior_heterogeneity_allocation
prior distribution(s) for the alternative cluster-level heterogeneity allocation component(s). This argument can be specified only when cluster is specified.

prior_baserate prior distribution for the estimate-specific midpoint base-rate probability in binomial GLMM models (measure = "OR"). If omitted or NULL, defaults to independent Beta(1, 1) priors. Continuous priors are restricted to [0, 1] (see prior_specification) and point priors must lie strictly inside (0, 1). The estimate-unit log-likelihood of the model (log_lik(), LOO, WAIC) integrates the base rate over any of these priors: point priors and untruncated beta priors adaptively (a uniform(0, 1) prior is a Beta(1, 1) prior), other families on a refined prior-CDF grid. The grid is accepted only when its log-likelihood changes by less than 1e-6 between refinements; any prior without an adaptive route (other than an untruncated beta prior; a truncated beta prior takes the grid too) can fail to reach this, especially with few events or very large arms, and then stops with an error of class RoBMA_glmm_grid_nonconvergence, as a lognormal base-rate prior does on cells with few events. A beta prior avoids it. The automatic computation of LOO/WAIC after fitting warns instead and returns the fit without them.

prior_lograte prior distribution for the estimate-specific midpoint log-rate in Poisson GLMM models (measure = "IRR"). If omitted or NULL, a normal prior centered at the pooled crude log-rate is used independently for each estimate. Its standard deviation defaults to 1 and can be changed via RoBMA.options(default_lograte.sd = ...). Rescaling all person-times shifts the prior mean by the corresponding log conversion while leaving its standard deviation unchanged. The estimate-unit log-likelihood of the model (log_lik(), LOO, WAIC) integrates the log-rate over the prior: point priors and untruncated normal priors adaptively, other families on a refined prior-CDF grid. The grid is accepted only when its log-likelihood changes by less than 1e-6 between refinements; any prior without an adaptive route (other than an untruncated normal prior; a truncated normal prior takes the grid too) can fail to reach this, especially with few events or very large arms, and then stops with an error of class RoBMA_glmm_grid_nonconvergence, as gamma, lognormal, and inverse-gamma log-rate priors do on cells with few events. A normal prior avoids it. The automatic computation of LOO/WAIC after fitting warns instead and returns the fit without them.

prior_effect_null
: prior distribution(s) for the null effect component(s).

prior_heterogeneity_null
: prior distribution(s) for the null heterogeneity component(s).

prior_mods_null
: prior distribution(s) for null moderator components. A named `intercept` entry, including `NULL` or `FALSE`, is unavailable: put the complete intercept prior in `prior_mods`, or use global prior arguments without a formula-list intercept. The same rule applies to `prior_scale_null`. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `mods` is specified. The null and the alternative prior of a factor term need the same contrast; a conflict stops with an error of class `RoBMA_prior_contrast_conflict` (also of class `RoBMA_prior_error`) that names the term and the remedies. The same rule applies to `prior_scale_null`.

prior_scale_null
: prior distribution(s) for null scale-regression components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `scale` is specified.

prior_heterogeneity_allocation_null
: prior distribution(s) for the null cluster-level heterogeneity allocation component(s). This argument can be specified only when `cluster` is specified.

standardize_continuous_predictors
: logical. Whether to standardize continuous predictors. Defaults to `TRUE`.

set_contrast_factor_predictors
: character. How to set contrast for factor predictors. Defaults are constructor-specific and shown in each function usage. Single-model constructors use `"treatment"`; model-averaging constructors use `"meandif"`. Accepted values are `"treatment"`, `"meandif"`, `"orthonormal"`, and `"independent"`. Independent coding creates one coefficient per factor level. Random-coefficient blocks reuse the resolved fixed-factor contrast unless `random_block(contrasts = ...)` overrides it. A warning is issued when default or inherited independent contrasts make a fixed or random design overspecified. With `"meandif"` or `"orthonormal"` contrasts, a formula term that interacts a factor with a predictor whose own term is missing, such as `g:x` in `~ g + g:x` or `~ g / x`, stops the fit: the term codes `g` by level indicators, with one coefficient per level, on which mean-difference and orthonormal priors are not defined. Specify an independent prior for that term, e.g., `prior_mods = list("g:x" = prior_factor("normal", list(0, 1), contrast = "independent"))` (or `prior_scale` for a scale formula), include the missing term (`~ g * x`), or use `set_contrast_factor_predictors = "treatment"`.

prior_unit_information_sd
: numeric. The unit information standard deviation (σ_{unit}). Cannot be used together with `prior_informed_field`.

rescale_priors
: numeric. A scaling factor for supported prior distributions. Point and none priors are unchanged; supplied `prior_random()` scale architectures and publication-bias priors are not rescaled. The default PEESE prior separately uses its documented UISD adjustment, independently of `rescale_priors`. Defaults to 1.

prior_informed_field
: character. The field of the informed prior distributions. Omit to use the standard default prior specification; explicit NULL is invalid.

prior_informed_subfield
: character. The subfield of the informed prior distributions. Can only be specified together with prior_informed_field. Omit to use the field-specific default, such as "Cochrane" for prior_informed_field = "medicine"; explicit NULL is invalid.

sample
: numeric. Number of MCMC samples to save. Defaults to 5000.

burnin
: numeric. Number of burn-in iterations. Defaults to 2000.

adapt
: numeric. Number of adaptation iterations. Defaults to 500.

chains
: numeric. Number of MCMC chains. Defaults to 3.

thin
: numeric. Thinning interval. Defaults to 1.

parallel
: logical. Whether to run MCMC chains in parallel. Defaults to FALSE.

autofit
: logical. Whether to automatically extend the MCMC chains if convergence is not met. Defaults to FALSE.

autofit_control
: list of autofit control settings. See set_autofit_control() for details.

convergence_checks
: list of convergence check settings. See set_convergence_checks() for details.

seed
: numeric. Random seed for reproducibility. Defaults to NULL.

silent
: logical. Whether to suppress output. Constructors with no explicit default use RoBMA.get_option("silent") when silent is omitted. Model-averaging wrappers default to TRUE unless explicitly changed.

...
: additional advanced arguments. Fitting functions reject unused arguments; currently recognized internal arguments include only_data, only_priors, is_JASP, and is_JASP_prefix.

Details

BMA.glmm combines the data input style of brma.glmm with the mixture prior specification of RoBMA for Bayesian model-averaging.

Model for odds ratios (measure = "OR") uses a binomial-normal model as described in Jackson et al. (2018).

Model for incidence rate ratios (measure = "IRR") uses a Poisson-normal model as described in Bagos and Nikolopoulos (2009).

When weights are supplied, they are treated as likelihood weights on the paired two-arm study contribution.

For factor moderators, BMA.glmm() defaults to set_contrast_factor_predictors = "meandif", matching the other model-averaging constructors. This differs from brma.glmm(), which defaults to "treatment" as a single-model constructor.

Value

A fitted object of class c("BMA.glmm", "RoBMA", "brma.glmm", "brma"). The object contains checked data, checked mixture priors, the JAGS fit, cached summary, and cached coefficients.

See Also

brma.glmm() RoBMA() summary.brma() plot.brma()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.bcg, package = "metadat")

  fit <- BMA.glmm(
    ai      = tpos,
    bi      = tneg,
    ci      = cpos,
    di      = cneg,
    data    = dat.bcg,
    measure = "OR",
    seed    = 1,
    silent  = TRUE
  )

  summary(fit)
}

## End(Not run)
```

BMA.mv *Bayesian Model-Averaged Multivariate Meta-Analysis*

Description

Fits Bayesian model-averaged multivariate and multilevel meta-analytic models with known sampling covariance and formula random effects. Random structures supplied through `random` are model-averaged with independent component gates.

Usage

```
BMA.mv(
  yi,
  V,
  ni,
  mods,
  scale,
  random,
  R = NULL,
  Rscale = "cor",
  data,
  slab,
```

```
  subset,
  measure,
  prior_effect,
  prior_heterogeneity,
  prior_mods,
  prior_scale,
  prior_effect_null,
  prior_heterogeneity_null,
  prior_mods_null,
  prior_scale_null,
  standardize_continuous_predictors = TRUE,
  set_contrast_factor_predictors = "meandif",
  prior_unit_information_sd,
  rescale_priors = 1,
  prior_informed_field,
  prior_informed_subfield,
  known_v_parameterization = "auto",
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  chains = 3,
  thin = 1,
  parallel = FALSE,
  autofit = FALSE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  seed = NULL,
  silent = TRUE,
  ...,
  vi = NULL,
  sei = NULL
)
```

Arguments

yi	a vector of effect sizes, or a formula with the effect size on the left-hand side and location moderators on the right-hand side (for example `yi ~ x1 + x2`). If a formula is supplied, `mods` must not be specified.
V	a known working variance-covariance matrix, a list of block variance-covariance matrices, or an exact diagonal-plus-factor declaration created by `known_v_factor()`, used by `brma.mv()`. Ordinary covariance matrices from `metafor::vcalc()` can be supplied directly. Product selection requires no publication groups. For best-p-value selection, declare groups separately with `selection_model(weight_rule = "best", group = ...)`. Matrix symmetry uses base R's numerical tolerance. Accepted matrices are stored symmetrically by averaging off-diagonal pairs; the supplied object and its diagonal are unchanged. Dependency blocks that are block constant on nested partitions are represented exactly as a diagonal-plus-factor structure without a `known_v_factor()` declaration. Dimensions and row

alignment are checked on the complete input; when rows are dropped (subset or missing values), the numerical covariance checks apply to the retained rows only, so acceptance says nothing about the entries of the dropped rows.

ni — an optional vector of sample sizes. Used for measure = "GEN" or when estimating the unit information standard deviation.

mods — an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in data and supports explicit columns and standard formula operators, including ^ as R's crossing operator with a literal whole exponent of at least 2: (x + z)^2 expands to x + z + x:z, and x^2 is not a quadratic term (x + x^2 is x). Precompute transformed predictors, such as a squared predictor, as columns in data; inline calls such as I(), poly(), spline constructors, and user functions are not supported.

scale — an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in data and follows the same explicit-column formula grammar as mods. Random-formula models also accept a named list targeting top-level random components or concrete random-effect blocks; see brma.mv().

random — Optional BayesTools random-effect formula or list of formulas. Omission or NULL means a fixed-effect model with no implicit heterogeneity term. Known sampling covariance V remains unchanged. Declare random before supplying heterogeneity priors or scale formulas.

R — optional known random-effect group covariance/correlation matrix, or a named list of matrices. Matrix row and column names identify grouping levels. This is distinct from the known sampling covariance V.

Rscale — scaling applied to each supplied R matrix before fitting. Supported values are "cor", "none", "cor0", "cov0", with metafor-compatible aliases TRUE, FALSE, 0, 1, 2, and 3. An unnamed scalar is recycled across all supplied matrices. A vector or list can specify one scaling value per named R entry, with exactly matching names. An unnamed one-element list is supported only for a single R matrix.

data — an optional data frame containing the variables.

slab — an optional vector of study labels.

subset — an optional logical or numeric vector specifying a subset of data to be used. Subsetting precedes missing-row removal; retained-row outcome and predictor checks follow these steps. Type, shape, and existing extraction checks can still reject malformed inputs before subsetting. Row selection (by subset or missing-row removal) drops factor levels left without rows from mods, scale, and random-effect grouping factors. The index variable of a structured random-effect term (cs(), hcs(), ar1(), har() with one index variable) keeps its declared levels, which define the covariance structure; the variables of a multi-variable cs(a + b | group) index follow the ordinary rule, because that index is built from their observed combinations. If an index variable that keeps its levels is also used in mods or scale and loses one of its levels to the selection, the fit stops with an error of class RoBMA_structured_index_level_removed (also RoBMA_data_error); remove the level from the data with droplevels() or use a separate column for the moderator.

`measure` a character string specifying the effect size measure. Normal/effect-size constructors require an explicit value and support `"SMD"`, `"ZCOR"`, `"RR"`, `"OR"`, `"HR"`, `"RD"`, `"IRR"`, and `"GEN"`. Use `"GEN"` only for general effect sizes without a known unit information standard deviation. GLMM raw-count constructors support only `"OR"` and `"IRR"` and default to `"OR"`.

`prior_effect` prior distribution(s) for the alternative effect component(s).

`prior_heterogeneity`
prior distribution(s) for the alternative heterogeneity component(s).

`prior_mods` prior distribution(s) for alternative moderator components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `mods` is specified. For a formula without an intercept (`~ 0 + x`), an `intercept` entry other than a point prior at zero stops with an error of class `RoBMA_prior_intercept_conflict` (also of class `RoBMA_prior_error`). A named alternative `intercept` must be a single simple or point prior; it is the complete intercept prior and replaces the global effect mixture. Specify it in one place: null-list intercepts and explicit global effect declarations left without a consumer are refused. Ordinary term entries replace only their alternative component.

`prior_scale` prior distribution(s) for alternative scale-regression components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `scale` is specified. A named alternative `intercept` must be a single positive simple or point prior; it is the complete SD prior used by `log(intercept)`. Global heterogeneity declarations remain meaningful when they supply another random root or its inclusion weights; declarations replaced without a consumer are refused. Null-list intercept declarations are unavailable.

`prior_effect_null`
prior distribution(s) for the null effect component(s).

`prior_heterogeneity_null`
prior distribution(s) for the null heterogeneity component(s).

`prior_mods_null`
prior distribution(s) for null moderator components. A named `intercept` entry, including `NULL` or `FALSE`, is unavailable: put the complete intercept prior in `prior_mods`, or use global prior arguments without a formula-list intercept. The same rule applies to `prior_scale_null`. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `mods` is specified. The null and the alternative prior of a factor term need the same contrast; a conflict stops with an error of class `RoBMA_prior_contrast_conflict` (also of class `RoBMA_prior_error`) that names the term and the remedies. The same rule applies to `prior_scale_null`.

`prior_scale_null`
prior distribution(s) for null scale-regression components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `scale` is specified.

`standardize_continuous_predictors`
logical. Whether to standardize continuous predictors. Defaults to `TRUE`.

`set_contrast_factor_predictors`
: character. How to set contrast for factor predictors. Defaults are constructor-specific and shown in each function usage. Single-model constructors use `"treatment"`; model-averaging constructors use `"meandif"`. Accepted values are `"treatment"`, `"meandif"`, `"orthonormal"`, and `"independent"`. Independent coding creates one coefficient per factor level. Random-coefficient blocks reuse the resolved fixed-factor contrast unless `random_block(contrasts = ...)` overrides it. A warning is issued when default or inherited independent contrasts make a fixed or random design overspecified. With `"meandif"` or `"orthonormal"` contrasts, a formula term that interacts a factor with a predictor whose own term is missing, such as `g:x` in `~ g + g:x` or `~ g / x`, stops the fit: the term codes `g` by level indicators, with one coefficient per level, on which mean-difference and orthonormal priors are not defined. Specify an independent prior for that term, e.g., `prior_mods = list("g:x" = prior_factor("normal", list(0, 1), contrast = "independent"))` (or `prior_scale` for a scale formula), include the missing term (`~ g * x`), or use `set_contrast_factor_predictors = "treatment"`.

`prior_unit_information_sd`
: numeric. The unit information standard deviation (σ_{unit}). Cannot be used together with `prior_informed_field`.

`rescale_priors`
: numeric. A scaling factor for supported prior distributions. Point and none priors are unchanged; supplied `prior_random()` scale architectures and publication-bias priors are not rescaled. The default PEESE prior separately uses its documented UISD adjustment, independently of `rescale_priors`. Defaults to 1.

`prior_informed_field`
: character. The field of the informed prior distributions. Omit to use the standard default prior specification; explicit `NULL` is invalid.

`prior_informed_subfield`
: character. The subfield of the informed prior distributions. Can only be specified together with `prior_informed_field`. Omit to use the field-specific default, such as `"Cochrane"` for `prior_informed_field = "medicine"`; explicit `NULL` is invalid.

`known_v_parameterization`
: known-V backend used by `brma.mv()`. `"auto"` chooses an exact backend when feasible; `"latent"` uses a latent `D + BB'` decomposition; `"whitened"` uses an eigen-rotated normal likelihood; `"block_mvn"` uses an exact native block multivariate-normal likelihood.

`sample`
: numeric. Number of MCMC samples to save. Defaults to `5000`.

`burnin`
: numeric. Number of burn-in iterations. Defaults to `2000`.

`adapt`
: numeric. Number of adaptation iterations. Defaults to `500`.

`chains`
: numeric. Number of MCMC chains. Defaults to `3`.

`thin`
: numeric. Thinning interval. Defaults to `1`.

`parallel`
: logical. Whether to run MCMC chains in parallel. Defaults to `FALSE`.

`autofit`
: logical. Whether to automatically extend the MCMC chains if convergence is not met. Defaults to `FALSE`.

`autofit_control`
: list of autofit control settings. See `set_autofit_control()` for details.

convergence_checks
: list of convergence check settings. See set_convergence_checks() for details.

seed
: numeric. Random seed for reproducibility. Defaults to NULL.

silent
: logical. Whether to suppress output. Constructors with no explicit default use RoBMA.get_option("silent") when silent is omitted. Model-averaging wrappers default to TRUE unless explicitly changed.

...
: additional advanced arguments. Fitting functions reject unused arguments; currently recognized internal arguments include only_data, only_priors, is_JASP, and is_JASP_prefix.

vi
: a vector of sampling variances. Either vi or sei must be supplied for normal models.

sei
: a vector of standard errors. Either vi or sei must be supplied for normal models.

Details

Omitted or NULL random specifies no heterogeneity and no random-effect inclusion mixture. Heterogeneity priors and scale formulas require an explicit random structure; the known sampling covariance remains unchanged.

BMA.mv() combines the product-space model-averaging workflow of BMA() with the likelihood, formula, random-effect, and known-covariance machinery of brma.mv(). For multiple top-level random components, a Dirichlet prior allocates the slab total variance and an independent Bernoulli gate multiplies each allocated contribution. The allocation is not renormalized when a gate is off. Thus component j contributes

$$I_j w_j \tau^2,$$

where I_j is its inclusion indicator, w_j is its slab allocation, and τ is the slab total SD. With one component, no artificial Dirichlet weight is created and the contribution is $I_1 \tau^2$.

By default, every top-level component has an independent 0.5 inclusion probability. prior_heterogeneity defines the shared positive slab and prior_heterogeneity_null defines exclusion at exactly zero. Their prior weights determine the common gate probability; multiple alternative priors retain their relative weights within the slab. Setting prior_heterogeneity_null = NULL or FALSE fixes all component gates on, while prior_heterogeneity = NULL or FALSE fixes them off. Nested random terms are gated as one top-level component before their internal Dirichlet split. Component-specific or partial scale models instead have independent SD roots, with allocation within each component; unscaled components keep their own base SD prior. See random_effect_prior_specification.

A partial BayesTools::prior_random() supplied through prior_heterogeneity may override contrasts, correlations, monitoring, and parameterization. BMA.mv() owns the gated scale architecture, so a custom prior_random() must not supply an SD, SD source, term-specific SD, covariance-owned SD, or variance allocation.

Random-effect inclusion probabilities are reported separately from fixed effect and scale-regression inclusion. Averaged random-component SDs retain the excluded zero branch; conditional random summaries condition each component on its own gate. tau_total and tau2_total are the realized gated aggregate, including zero when every component is excluded. tau2_prop(j) is the realized share $I_j w_j / \sum_k I_k w_k$, conditional on positive total heterogeneity. Excluded components therefore have zero share, and the all-off branch is omitted only from variance-proportion summaries. The positive slab scale and raw Dirichlet weights remain internal coordinates.

Product-space marginal likelihood and bridge-sampling methods are not available. Predictive comparison through `loo.brma()` and `waic.brma()` remains available. If `V` is singular, every allowed product-space branch must contain structural variance sufficient to regularize its null space; an allowed all-components-off branch is therefore rejected.

The function does not add publication-bias models. Use `RoBMA.mv()` to add model-averaged selection, PET, and PEESE publication-bias adjustments.

Value

A fitted object of class `c("BMA.mv", "RoBMA", "brma.mv", "brma.norm", "brma")`.

See Also

`BMA()`, `RoBMA.mv()`, `brma.mv()`, `summary.brma()`, `summary_models()`

Examples

```
## Not run:
data("dat.assink2016", package = "metadat")
V <- metafor::vcalc(
  vi,
  cluster = study,
  type = deltype,
  obs = esid,
  rho = c(0.7, 0.5),
  data = dat.assink2016
)
fit <- BMA.mv(
  yi = yi,
  V = V,
  mods = ~ deltype,
  random = ~ 1 | study / esid,
  data = dat.assink2016,
  measure = "SMD",
  seed = 1,
  silent = TRUE
)
summary(fit)
summary_models(fit)

## End(Not run)
```

---

bPEESE *Bayesian Precision-Effect Estimate with Standard Errors (PEESE) Model*

Description

Function for fitting random-effects, meta-regression, multilevel, and location-scale meta-analytic PEESE models.

Usage

```
bPEESE(
  yi,
  vi,
  sei,
  weights,
  ni,
  mods,
  scale,
  cluster,
  data,
  slab,
  subset,
  measure,
  prior_effect,
  prior_heterogeneity,
  prior_mods,
  prior_scale,
  prior_heterogeneity_allocation,
  prior_bias,
  standardize_continuous_predictors = TRUE,
  set_contrast_factor_predictors = "treatment",
  prior_unit_information_sd,
  rescale_priors = 1,
  prior_informed_field,
  prior_informed_subfield,
  effect_direction = "detect",
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  chains = 3,
  thin = 1,
  parallel = FALSE,
  autofit = FALSE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  seed = NULL,
  silent,
  ...
)
```

Arguments

yi	a vector of effect sizes, or a formula with the effect size on the left-hand side and location moderators on the right-hand side (for example `yi ~ x1 + x2`). If a formula is supplied, `mods` must not be specified.
vi	a vector of sampling variances. Either `vi` or `sei` must be supplied for normal models.
sei	a vector of standard errors. Either `vi` or `sei` must be supplied for normal models.
weights	an optional vector of finite positive likelihood weights. For normal/effect-size models, each weight powers the estimate likelihood. For constructors with GLMM raw-count input, each weight powers the paired two-arm likelihood for one study. An integer weight w repeats that conditional likelihood contribution w times, with the declared latent-effect and grouping structure held fixed. Positive fractional likelihood powers are also supported.
ni	an optional vector of sample sizes. Used for `measure = "GEN"` or when estimating the unit information standard deviation.
mods	an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data` and supports explicit columns and standard formula operators, including `^` as R's crossing operator with a literal whole exponent of at least 2: `(x + z)^2` expands to `x + z + x:z`, and `x^2` is not a quadratic term (`x + x^2` is `x`). Precompute transformed predictors, such as a squared predictor, as columns in `data`; inline calls such as `I()`, `poly()`, spline constructors, and user functions are not supported.
scale	an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data` and follows the same explicit-column formula grammar as `mods`. Random-formula models also accept a named list targeting top-level random components or concrete random-effect blocks; see `brma.mv()`.
cluster	an optional vector of cluster identifiers for multilevel meta-analysis.
data	an optional data frame containing the variables.
slab	an optional vector of study labels.
subset	an optional logical or numeric vector specifying a subset of data to be used. Subsetting precedes missing-row removal; retained-row outcome and predictor checks follow these steps. Type, shape, and existing extraction checks can still reject malformed inputs before subsetting. Row selection (by `subset` or missing-row removal) drops factor levels left without rows from `mods`, `scale`, and random-effect grouping factors. The index variable of a structured random-effect term (`cs()`, `hcs()`, `ar1()`, `har()` with one index variable) keeps its declared levels, which define the covariance structure; the variables of a multivariable `cs(a + b | group)` index follow the ordinary rule, because that index is built from their observed combinations. If an index variable that keeps its levels is also used in `mods` or `scale` and loses one of its levels to the selection, the fit stops with an error of class `RoBMA_structured_index_level_removed` (also `RoBMA_data_error`); remove the level from the data with `droplevels()` or use a separate column for the moderator.

measure — a character string specifying the effect size measure. Normal/effect-size constructors require an explicit value and support "SMD", "ZCOR", "RR", "OR", "HR", "RD", "IRR", and "GEN". Use "GEN" only for general effect sizes without a known unit information standard deviation. GLMM raw-count constructors support only "OR" and "IRR" and default to "OR".

prior_effect — prior distribution for the effect size (μ) parameter (the intercept). If omitted and consumed, a default prior is constructed. In single-model functions, explicit NULL or FALSE sets a spike at zero.

prior_heterogeneity — prior distribution for the heterogeneity (τ) parameter. If omitted and consumed, a default prior is constructed. In single-model functions, explicit NULL or FALSE sets a spike at zero. For brma.mv random-effect formulas, this can also be a partial or complete prior_random object; see random_effect_prior_specification.

prior_mods — prior distribution for the moderators (β) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or NULL, default priors are used. This argument can be specified only when mods is specified. A formula without an intercept (~ 0 + x) fixes the location intercept at zero, so an intercept entry of the list is accepted only as a point prior at zero; any other intercept prior stops with an error of class RoBMA_prior_intercept_conflict (also of class RoBMA_prior_error). A named intercept entry supplies the complete intercept prior. Explicit competing global effect declarations are refused when this would discard them. Named NULL or FALSE intercept entries are invalid; specify the complete prior in one place.

prior_scale — prior distribution for the scale (δ) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or NULL, default priors are used. This argument can be specified only when scale is specified. A named intercept entry supplies the complete positive SD prior used by log(intercept). Global heterogeneity declarations remain meaningful when they supply another random root or its model-inclusion weights; declarations fully replaced without a consumer raise RoBMA_prior_intercept_conflict.

prior_heterogeneity_allocation — prior distribution for the fraction of heterogeneity allocated to the cluster-level component in multilevel models (ρ, a variance fraction rather than a random-coefficient correlation). If omitted or NULL, defaults to Beta(1, 1). This argument can be specified only when cluster is specified.

prior_bias — PEESE regression-coefficient prior created by prior_PEESE(). If omitted or NULL, the default is a Cauchy prior centered at 0, truncated to positive values, with scale from RoBMA.get_option("default_bias_PEESE.scale") after UISD adjustment.

standardize_continuous_predictors — logical. Whether to standardize continuous predictors. Defaults to TRUE.

set_contrast_factor_predictors — character. How to set contrast for factor predictors. Defaults are constructor-specific and shown in each function usage. Single-model constructors use "treatment"; model-averaging constructors use "meandif". Accepted values are "treatment", "meandif", "orthonormal", and "independent". Independent coding creates one coefficient per factor level. Random-coefficient blocks reuse the resolved

fixed-factor contrast unless `random_block(contrasts = ...)` overrides it. A warning is issued when default or inherited independent contrasts make a fixed or random design overspecified. With `"meandif"` or `"orthonormal"` contrasts, a formula term that interacts a factor with a predictor whose own term is missing, such as `g:x` in `~ g + g:x` or `~ g / x`, stops the fit: the term codes `g` by level indicators, with one coefficient per level, on which mean-difference and orthonormal priors are not defined. Specify an independent prior for that term, e.g., `prior_mods = list("g:x" = prior_factor("normal", list(0, 1), contrast = "independent"))` (or `prior_scale` for a scale formula), include the missing term (`~ g * x`), or use `set_contrast_factor_predictors = "treatment"`.

`prior_unit_information_sd`
: numeric. The unit information standard deviation (σ_{unit}). Cannot be used together with `prior_informed_field`.

`rescale_priors`
: numeric. A scaling factor for supported prior distributions. Point and none priors are unchanged; supplied `prior_random()` scale architectures and publication-bias priors are not rescaled. The default PEESE prior separately uses its documented UISD adjustment, independently of `rescale_priors`. Defaults to 1.

`prior_informed_field`
: character. The field of the informed prior distributions. Omit to use the standard default prior specification; explicit `NULL` is invalid.

`prior_informed_subfield`
: character. The subfield of the informed prior distributions. Can only be specified together with `prior_informed_field`. Omit to use the field-specific default, such as `"Cochrane"` for `prior_informed_field = "medicine"`; explicit `NULL` is invalid.

`effect_direction`
: direction used by publication-bias adjustments. `"positive"` assumes statistically significant positive estimates are more likely to be selected; `"negative"` mirrors the selection direction; `"detect"` infers the direction from the fitted data.

`sample`
: numeric. Number of MCMC samples to save. Defaults to `5000`.

`burnin`
: numeric. Number of burn-in iterations. Defaults to `2000`.

`adapt`
: numeric. Number of adaptation iterations. Defaults to `500`.

`chains`
: numeric. Number of MCMC chains. Defaults to `3`.

`thin`
: numeric. Thinning interval. Defaults to `1`.

`parallel`
: logical. Whether to run MCMC chains in parallel. Defaults to `FALSE`.

`autofit`
: logical. Whether to automatically extend the MCMC chains if convergence is not met. Defaults to `FALSE`.

`autofit_control`
: list of autofit control settings. See `set_autofit_control()` for details.

`convergence_checks`
: list of convergence check settings. See `set_convergence_checks()` for details.

`seed`
: numeric. Random seed for reproducibility. Defaults to `NULL`.

`silent`
: logical. Whether to suppress output. Constructors with no explicit default use `RoBMA.get_option("silent")` when `silent` is omitted. Model-averaging wrappers default to `TRUE` unless explicitly changed.

...	additional advanced arguments. Fitting functions reject unused arguments; currently recognized internal arguments include `only_data`, `only_priors`, `is_JASP`, and `is_JASP_prefix`.

Value

A fitted object of class `c("bPEESE", "brma")` containing a single PEESE publication-bias model fit.

See Also

`bPEESE.mv()`, publication_bias_prior_specification, `RoBMA()`, `bPET()`, `bselmodel()`, `summary.brma()`, `funnel.brma()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")

  fit <- bPEESE(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )

  summary(fit)
  funnel(fit)
}

## End(Not run)
```

`bPEESE.mv` *Bayesian Multivariate and Multilevel PEESE Model*

Description

Fits Bayesian precision-effect estimate with standard errors (PEESE) models with known sampling covariance and BayesTools random-effect formulas.

Usage

```
bPEESE.mv(
  yi,
  V,
```

```
  ni,
  mods,
  scale,
  random,
  R = NULL,
  Rscale = "cor",
  data,
  slab,
  subset,
  measure,
  prior_effect,
  prior_heterogeneity,
  prior_mods,
  prior_scale,
  prior_bias,
  standardize_continuous_predictors = TRUE,
  set_contrast_factor_predictors = "treatment",
  prior_unit_information_sd,
  rescale_priors = 1,
  prior_informed_field,
  prior_informed_subfield,
  effect_direction = "detect",
  known_v_parameterization = "auto",
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  chains = 3,
  thin = 1,
  parallel = FALSE,
  autofit = FALSE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  seed = NULL,
  silent,
  ...,
  vi = NULL,
  sei = NULL
)
```

Arguments

yi	a vector of effect sizes, or a formula with the effect size on the left-hand side and location moderators on the right-hand side (for example `yi ~ x1 + x2`). If a formula is supplied, `mods` must not be specified.
V	a known working variance-covariance matrix, a list of block variance-covariance matrices, or an exact diagonal-plus-factor declaration created by `known_v_factor()`, used by `brma.mv()`. Ordinary covariance matrices from `metafor::vcalc()` can be supplied directly. Product selection requires no publication groups. For

best-p-value selection, declare groups separately with `selection_model(weight_rule = "best", group = ...)`. Matrix symmetry uses base R's numerical tolerance. Accepted matrices are stored symmetrically by averaging off-diagonal pairs; the supplied object and its diagonal are unchanged. Dependency blocks that are block constant on nested partitions are represented exactly as a diagonal-plus-factor structure without a `known_v_factor()` declaration. Dimensions and row alignment are checked on the complete input; when rows are dropped (`subset` or missing values), the numerical covariance checks apply to the retained rows only, so acceptance says nothing about the entries of the dropped rows.

`ni` an optional vector of sample sizes. Used for `measure = "GEN"` or when estimating the unit information standard deviation.

`mods` an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data` and supports explicit columns and standard formula operators, including `^` as R's crossing operator with a literal whole exponent of at least 2: `(x + z)^2` expands to `x + z + x:z`, and `x^2` is not a quadratic term (`x + x^2` is `x`). Precompute transformed predictors, such as a squared predictor, as columns in `data`; inline calls such as `I()`, `poly()`, spline constructors, and user functions are not supported.

`scale` an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data` and follows the same explicit-column formula grammar as `mods`. Random-formula models also accept a named list targeting top-level random components or concrete random-effect blocks; see `brma.mv()`.

`random` Optional BayesTools random-effect formula or list of formulas. Omission or `NULL` means a fixed-effect model with no implicit heterogeneity term. Known sampling covariance `V` remains unchanged. Declare `random` before supplying heterogeneity priors or scale formulas.

`R` optional known random-effect group covariance/correlation matrix, or a named list of matrices. Matrix row and column names identify grouping levels. This is distinct from the known sampling covariance `V`.

`Rscale` scaling applied to each supplied `R` matrix before fitting. Supported values are `"cor"`, `"none"`, `"cor0"`, `"cov0"`, with metafor-compatible aliases `TRUE`, `FALSE`, `0`, `1`, `2`, and `3`. An unnamed scalar is recycled across all supplied matrices. A vector or list can specify one scaling value per named `R` entry, with exactly matching names. An unnamed one-element list is supported only for a single `R` matrix.

`data` an optional data frame containing the variables.

`slab` an optional vector of study labels.

`subset` an optional logical or numeric vector specifying a subset of data to be used. Subsetting precedes missing-row removal; retained-row outcome and predictor checks follow these steps. Type, shape, and existing extraction checks can still reject malformed inputs before subsetting. Row selection (by `subset` or missing-row removal) drops factor levels left without rows from `mods`, `scale`, and random-effect grouping factors. The index variable of a structured random-effect term (`cs()`, `hcs()`, `ar1()`, `har()` with one index variable) keeps its declared levels, which define the covariance structure; the variables of a multivariable `cs(a + b | group)` index follow the ordinary rule, because that index is

built from their observed combinations. If an index variable that keeps its levels is also used in `mods` or `scale` and loses one of its levels to the selection, the fit stops with an error of class `RoBMA_structured_index_level_removed` (also `RoBMA_data_error`); remove the level from the data with `droplevels()` or use a separate column for the moderator.

`measure` a character string specifying the effect size measure. Normal/effect-size constructors require an explicit value and support `"SMD"`, `"ZCOR"`, `"RR"`, `"OR"`, `"HR"`, `"RD"`, `"IRR"`, and `"GEN"`. Use `"GEN"` only for general effect sizes without a known unit information standard deviation. GLMM raw-count constructors support only `"OR"` and `"IRR"` and default to `"OR"`.

`prior_effect` prior distribution for the effect size (μ) parameter (the intercept). If omitted and consumed, a default prior is constructed. In single-model functions, explicit `NULL` or `FALSE` sets a spike at zero.

`prior_heterogeneity`
prior distribution for the heterogeneity (τ) parameter. If omitted and consumed, a default prior is constructed. In single-model functions, explicit `NULL` or `FALSE` sets a spike at zero. For `brma.mv` random-effect formulas, this can also be a partial or complete `prior_random` object; see `random_effect_prior_specification`.

`prior_mods` prior distribution for the moderators (β) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or `NULL`, default priors are used. This argument can be specified only when `mods` is specified. A formula without an intercept (`~ 0 + x`) fixes the location intercept at zero, so an `intercept` entry of the list is accepted only as a point prior at zero; any other `intercept` prior stops with an error of class `RoBMA_prior_intercept_conflict` (also of class `RoBMA_prior_error`). A named `intercept` entry supplies the complete intercept prior. Explicit competing global effect declarations are refused when this would discard them. Named `NULL` or `FALSE` intercept entries are invalid; specify the complete prior in one place.

`prior_scale` prior distribution for the scale (δ) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or `NULL`, default priors are used. This argument can be specified only when `scale` is specified. A named `intercept` entry supplies the complete positive SD prior used by `log(intercept)`. Global heterogeneity declarations remain meaningful when they supply another random root or its model-inclusion weights; declarations fully replaced without a consumer raise `RoBMA_prior_intercept_conflict`.

`prior_bias` PEESE regression-coefficient prior created by `prior_PEESE()`. If omitted or `NULL`, the default is a Cauchy prior centered at 0, truncated to positive values, with scale from `RoBMA.get_option("default_bias_PEESE.scale")` after UISD adjustment.

`standardize_continuous_predictors`
logical. Whether to standardize continuous predictors. Defaults to `TRUE`.

`set_contrast_factor_predictors`
character. How to set contrast for factor predictors. Defaults are constructor-specific and shown in each function usage. Single-model constructors use `"treatment"`; model-averaging constructors use `"meandif"`. Accepted values are `"treatment"`, `"meandif"`, `"orthonormal"`, and `"independent"`. Independent coding creates

one coefficient per factor level. Random-coefficient blocks reuse the resolved fixed-factor contrast unless `random_block(contrasts = ...)` overrides it. A warning is issued when default or inherited independent contrasts make a fixed or random design overspecified. With `"meandif"` or `"orthonormal"` contrasts, a formula term that interacts a factor with a predictor whose own term is missing, such as `g:x` in `~ g + g:x` or `~ g / x`, stops the fit: the term codes `g` by level indicators, with one coefficient per level, on which mean-difference and orthonormal priors are not defined. Specify an independent prior for that term, e.g., `prior_mods = list("g:x" = prior_factor("normal", list(0, 1), contrast = "independent"))` (or `prior_scale` for a scale formula), include the missing term (`~ g * x`), or use `set_contrast_factor_predictors = "treatment"`.

`prior_unit_information_sd`
: numeric. The unit information standard deviation (σ_{unit}). Cannot be used together with `prior_informed_field`.

`rescale_priors`
: numeric. A scaling factor for supported prior distributions. Point and none priors are unchanged; supplied `prior_random()` scale architectures and publication-bias priors are not rescaled. The default PEESE prior separately uses its documented UISD adjustment, independently of `rescale_priors`. Defaults to 1.

`prior_informed_field`
: character. The field of the informed prior distributions. Omit to use the standard default prior specification; explicit `NULL` is invalid.

`prior_informed_subfield`
: character. The subfield of the informed prior distributions. Can only be specified together with `prior_informed_field`. Omit to use the field-specific default, such as `"Cochrane"` for `prior_informed_field = "medicine"`; explicit `NULL` is invalid.

`effect_direction`
: direction used by publication-bias adjustments. `"positive"` assumes statistically significant positive estimates are more likely to be selected; `"negative"` mirrors the selection direction; `"detect"` infers the direction from the fitted data.

`known_v_parameterization`
: known-V backend used by `brma.mv()`. `"auto"` chooses an exact backend when feasible; `"latent"` uses a latent `D + BB'` decomposition; `"whitened"` uses an eigen-rotated normal likelihood; `"block_mvn"` uses an exact native block multivariate-normal likelihood.

`sample`
: numeric. Number of MCMC samples to save. Defaults to `5000`.

`burnin`
: numeric. Number of burn-in iterations. Defaults to `2000`.

`adapt`
: numeric. Number of adaptation iterations. Defaults to `500`.

`chains`
: numeric. Number of MCMC chains. Defaults to `3`.

`thin`
: numeric. Thinning interval. Defaults to `1`.

`parallel`
: logical. Whether to run MCMC chains in parallel. Defaults to `FALSE`.

`autofit`
: logical. Whether to automatically extend the MCMC chains if convergence is not met. Defaults to `FALSE`.

`autofit_control`
: list of autofit control settings. See `set_autofit_control()` for details.

convergence_checks
: list of convergence check settings. See set_convergence_checks() for details.

seed
: numeric. Random seed for reproducibility. Defaults to NULL.

silent
: logical. Whether to suppress output. Constructors with no explicit default use RoBMA.get_option("silent") when silent is omitted. Model-averaging wrappers default to TRUE unless explicitly changed.

...
: additional advanced arguments. Fitting functions reject unused arguments; currently recognized internal arguments include only_data, only_priors, is_JASP, and is_JASP_prefix.

vi
: a vector of sampling variances. Either vi or sei must be supplied for normal models.

sei
: a vector of standard errors. Either vi or sei must be supplied for normal models.

Details

Omitted or NULL random specifies a fixed-effect model with no implicit heterogeneity. Heterogeneity priors and scale formulas require an explicit random structure; the known sampling covariance remains unchanged.

bPEESE.mv() combines the PEESE publication-bias regression of bPEESE() with the known-covariance and random-formula model of brma.mv(). The PEESE predictor is the marginal sampling variance diag(V); known off-diagonal sampling covariance remains part of the joint likelihood.

The supported known-V backends, random-effect marginalization rules, predictive targets, and diagnostic semantics are the same as for brma.mv(). In particular, random replaces the specialized cluster argument used by bPEESE().

Value

A fitted object of class c("bPEESE.mv", "bPEESE", "brma.mv", "brma.norm", "brma").

See Also

bPEESE(), bPET.mv(), brma.mv(), summary.brma()

Examples

```
## Not run:
V <- matrix(c(0.04, 0.01, 0.01, 0.09), 2, 2)
fit <- bPEESE.mv(
  yi      = c(0.10, 0.20),
  V       = V,
  measure = "GEN",
  seed    = 1,
  silent  = TRUE
)
summary(fit)

## End(Not run)
```

bPET *Bayesian Precision-Effect Test (PET) Model*

Description

Function for fitting random-effects, meta-regression, multilevel, and location-scale meta-analytic PET models.

Usage

```
bPET(
  yi,
  vi,
  sei,
  weights,
  ni,
  mods,
  scale,
  cluster,
  data,
  slab,
  subset,
  measure,
  prior_effect,
  prior_heterogeneity,
  prior_mods,
  prior_scale,
  prior_heterogeneity_allocation,
  prior_bias,
  standardize_continuous_predictors = TRUE,
  set_contrast_factor_predictors = "treatment",
  prior_unit_information_sd,
  rescale_priors = 1,
  prior_informed_field,
  prior_informed_subfield,
  effect_direction = "detect",
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  chains = 3,
  thin = 1,
  parallel = FALSE,
  autofit = FALSE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  seed = NULL,
  silent,
```

```
  ...
)
```

Arguments

yi	a vector of effect sizes, or a formula with the effect size on the left-hand side and location moderators on the right-hand side (for example `yi ~ x1 + x2`). If a formula is supplied, `mods` must not be specified.
vi	a vector of sampling variances. Either `vi` or `sei` must be supplied for normal models.
sei	a vector of standard errors. Either `vi` or `sei` must be supplied for normal models.
weights	an optional vector of finite positive likelihood weights. For normal/effect-size models, each weight powers the estimate likelihood. For constructors with GLMM raw-count input, each weight powers the paired two-arm likelihood for one study. An integer weight w repeats that conditional likelihood contribution w times, with the declared latent-effect and grouping structure held fixed. Positive fractional likelihood powers are also supported.
ni	an optional vector of sample sizes. Used for `measure = "GEN"` or when estimating the unit information standard deviation.
mods	an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data` and supports explicit columns and standard formula operators, including `^` as R's crossing operator with a literal whole exponent of at least 2: `(x + z)^2` expands to `x + z + x:z`, and `x^2` is not a quadratic term (`x + x^2` is `x`). Precompute transformed predictors, such as a squared predictor, as columns in `data`; inline calls such as `I()`, `poly()`, spline constructors, and user functions are not supported.
scale	an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data` and follows the same explicit-column formula grammar as `mods`. Random-formula models also accept a named list targeting top-level random components or concrete random-effect blocks; see `brma.mv()`.
cluster	an optional vector of cluster identifiers for multilevel meta-analysis.
data	an optional data frame containing the variables.
slab	an optional vector of study labels.
subset	an optional logical or numeric vector specifying a subset of data to be used. Subsetting precedes missing-row removal; retained-row outcome and predictor checks follow these steps. Type, shape, and existing extraction checks can still reject malformed inputs before subsetting. Row selection (by `subset` or missing-row removal) drops factor levels left without rows from `mods`, `scale`, and random-effect grouping factors. The index variable of a structured random-effect term (`cs()`, `hcs()`, `ar1()`, `har()` with one index variable) keeps its declared levels, which define the covariance structure; the variables of a multivariable `cs(a + b | group)` index follow the ordinary rule, because that index is built from their observed combinations. If an index variable that keeps its levels is also used in `mods` or `scale` and loses one of its levels to the selection, the fit stops with an error of class `RoBMA_structured_index_level_removed` (also

RoBMA_data_error); remove the level from the data with droplevels() or use a separate column for the moderator.

measure
: a character string specifying the effect size measure. Normal/effect-size constructors require an explicit value and support "SMD", "ZCOR", "RR", "OR", "HR", "RD", "IRR", and "GEN". Use "GEN" only for general effect sizes without a known unit information standard deviation. GLMM raw-count constructors support only "OR" and "IRR" and default to "OR".

prior_effect
: prior distribution for the effect size (μ) parameter (the intercept). If omitted and consumed, a default prior is constructed. In single-model functions, explicit NULL or FALSE sets a spike at zero.

prior_heterogeneity
: prior distribution for the heterogeneity (τ) parameter. If omitted and consumed, a default prior is constructed. In single-model functions, explicit NULL or FALSE sets a spike at zero. For brma.mv random-effect formulas, this can also be a partial or complete prior_random object; see random_effect_prior_specification.

prior_mods
: prior distribution for the moderators (β) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or NULL, default priors are used. This argument can be specified only when mods is specified. A formula without an intercept (~ 0 + x) fixes the location intercept at zero, so an intercept entry of the list is accepted only as a point prior at zero; any other intercept prior stops with an error of class RoBMA_prior_intercept_conflict (also of class RoBMA_prior_error). A named intercept entry supplies the complete intercept prior. Explicit competing global effect declarations are refused when this would discard them. Named NULL or FALSE intercept entries are invalid; specify the complete prior in one place.

prior_scale
: prior distribution for the scale (δ) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or NULL, default priors are used. This argument can be specified only when scale is specified. A named intercept entry supplies the complete positive SD prior used by log(intercept). Global heterogeneity declarations remain meaningful when they supply another random root or its model-inclusion weights; declarations fully replaced without a consumer raise RoBMA_prior_intercept_conflict.

prior_heterogeneity_allocation
: prior distribution for the fraction of heterogeneity allocated to the cluster-level component in multilevel models (ρ, a variance fraction rather than a random-coefficient correlation). If omitted or NULL, defaults to Beta(1, 1). This argument can be specified only when cluster is specified.

prior_bias
: PET regression-coefficient prior created by prior_PET(). If omitted or NULL, the default is a Cauchy prior centered at 0, truncated to positive values, with scale from RoBMA.get_option("default_bias_PET.scale").

standardize_continuous_predictors
: logical. Whether to standardize continuous predictors. Defaults to TRUE.

set_contrast_factor_predictors
: character. How to set contrast for factor predictors. Defaults are constructor-specific and shown in each function usage. Single-model constructors use "treatment"; model-averaging constructors use "meandif". Accepted values are "treatment",

"meandif", "orthonormal", and "independent". Independent coding creates one coefficient per factor level. Random-coefficient blocks reuse the resolved fixed-factor contrast unless random_block(contrasts = ...) overrides it. A warning is issued when default or inherited independent contrasts make a fixed or random design overspecified. With "meandif" or "orthonormal" contrasts, a formula term that interacts a factor with a predictor whose own term is missing, such as g:x in ~ g + g:x or ~ g / x, stops the fit: the term codes g by level indicators, with one coefficient per level, on which mean-difference and orthonormal priors are not defined. Specify an independent prior for that term, e.g., prior_mods = list("g:x" = prior_factor("normal", list(0, 1), contrast = "independent")) (or prior_scale for a scale formula), include the missing term (~ g * x), or use set_contrast_factor_predictors = "treatment".

prior_unit_information_sd
: numeric. The unit information standard deviation (σ_{unit}). Cannot be used together with prior_informed_field.

rescale_priors
: numeric. A scaling factor for supported prior distributions. Point and none priors are unchanged; supplied prior_random() scale architectures and publication-bias priors are not rescaled. The default PEESE prior separately uses its documented UISD adjustment, independently of rescale_priors. Defaults to 1.

prior_informed_field
: character. The field of the informed prior distributions. Omit to use the standard default prior specification; explicit NULL is invalid.

prior_informed_subfield
: character. The subfield of the informed prior distributions. Can only be specified together with prior_informed_field. Omit to use the field-specific default, such as "Cochrane" for prior_informed_field = "medicine"; explicit NULL is invalid.

effect_direction
: direction used by publication-bias adjustments. "positive" assumes statistically significant positive estimates are more likely to be selected; "negative" mirrors the selection direction; "detect" infers the direction from the fitted data.

sample
: numeric. Number of MCMC samples to save. Defaults to 5000.

burnin
: numeric. Number of burn-in iterations. Defaults to 2000.

adapt
: numeric. Number of adaptation iterations. Defaults to 500.

chains
: numeric. Number of MCMC chains. Defaults to 3.

thin
: numeric. Thinning interval. Defaults to 1.

parallel
: logical. Whether to run MCMC chains in parallel. Defaults to FALSE.

autofit
: logical. Whether to automatically extend the MCMC chains if convergence is not met. Defaults to FALSE.

autofit_control
: list of autofit control settings. See set_autofit_control() for details.

convergence_checks
: list of convergence check settings. See set_convergence_checks() for details.

seed
: numeric. Random seed for reproducibility. Defaults to NULL.

silent	logical. Whether to suppress output. Constructors with no explicit default use RoBMA.get_option("silent") when silent is omitted. Model-averaging wrappers default to TRUE unless explicitly changed.
...	additional advanced arguments. Fitting functions reject unused arguments; currently recognized internal arguments include only_data, only_priors, is_JASP, and is_JASP_prefix.

Value

A fitted object of class c("bPET", "brma") containing a single PET publication-bias model fit.

See Also

bPET.mv(), publication_bias_prior_specification, RoBMA(), bPEESE(), bselmodel(), summary.brma(), funnel.brma()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")

  fit <- bPET(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )

  summary(fit)
  funnel(fit)
}

## End(Not run)
```

---

bPET.mv *Bayesian Multivariate and Multilevel PET Model*

Description

Fits Bayesian precision-effect test (PET) models with known sampling covariance and BayesTools random-effect formulas.

Usage

```
bPET.mv(
  yi,
  V,
  ni,
  mods,
  scale,
  random,
  R = NULL,
  Rscale = "cor",
  data,
  slab,
  subset,
  measure,
  prior_effect,
  prior_heterogeneity,
  prior_mods,
  prior_scale,
  prior_bias,
  standardize_continuous_predictors = TRUE,
  set_contrast_factor_predictors = "treatment",
  prior_unit_information_sd,
  rescale_priors = 1,
  prior_informed_field,
  prior_informed_subfield,
  effect_direction = "detect",
  known_v_parameterization = "auto",
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  chains = 3,
  thin = 1,
  parallel = FALSE,
  autofit = FALSE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  seed = NULL,
  silent,
  ...,
  vi = NULL,
  sei = NULL
)
```

Arguments

yi	a vector of effect sizes, or a formula with the effect size on the left-hand side and location moderators on the right-hand side (for example `yi ~ x1 + x2`). If a formula is supplied, `mods` must not be specified.

V — a known working variance-covariance matrix, a list of block variance-covariance matrices, or an exact diagonal-plus-factor declaration created by known_v_factor(), used by brma.mv(). Ordinary covariance matrices from metafor::vcalc() can be supplied directly. Product selection requires no publication groups. For best-p-value selection, declare groups separately with selection_model(weight_rule = "best", group = ...). Matrix symmetry uses base R's numerical tolerance. Accepted matrices are stored symmetrically by averaging off-diagonal pairs; the supplied object and its diagonal are unchanged. Dependency blocks that are block constant on nested partitions are represented exactly as a diagonal-plus-factor structure without a known_v_factor() declaration. Dimensions and row alignment are checked on the complete input; when rows are dropped (subset or missing values), the numerical covariance checks apply to the retained rows only, so acceptance says nothing about the entries of the dropped rows.

ni — an optional vector of sample sizes. Used for measure = "GEN" or when estimating the unit information standard deviation.

mods — an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in data and supports explicit columns and standard formula operators, including ^ as R's crossing operator with a literal whole exponent of at least 2: (x + z)^2 expands to x + z + x:z, and x^2 is not a quadratic term (x + x^2 is x). Precompute transformed predictors, such as a squared predictor, as columns in data; inline calls such as I(), poly(), spline constructors, and user functions are not supported.

scale — an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in data and follows the same explicit-column formula grammar as mods. Random-formula models also accept a named list targeting top-level random components or concrete random-effect blocks; see brma.mv().

random — Optional BayesTools random-effect formula or list of formulas. Omission or NULL means a fixed-effect model with no implicit heterogeneity term. Known sampling covariance V remains unchanged. Declare random before supplying heterogeneity priors or scale formulas.

R — optional known random-effect group covariance/correlation matrix, or a named list of matrices. Matrix row and column names identify grouping levels. This is distinct from the known sampling covariance V.

Rscale — scaling applied to each supplied R matrix before fitting. Supported values are "cor", "none", "cor0", "cov0", with metafor-compatible aliases TRUE, FALSE, 0, 1, 2, and 3. An unnamed scalar is recycled across all supplied matrices. A vector or list can specify one scaling value per named R entry, with exactly matching names. An unnamed one-element list is supported only for a single R matrix.

data — an optional data frame containing the variables.

slab — an optional vector of study labels.

subset — an optional logical or numeric vector specifying a subset of data to be used. Subsetting precedes missing-row removal; retained-row outcome and predictor checks follow these steps. Type, shape, and existing extraction checks can still reject malformed inputs before subsetting. Row selection (by subset or

missing-row removal) drops factor levels left without rows from `mods`, `scale`, and random-effect grouping factors. The index variable of a structured random-effect term (`cs()`, `hcs()`, `ar1()`, `har()` with one index variable) keeps its declared levels, which define the covariance structure; the variables of a multivariable `cs(a + b | group)` index follow the ordinary rule, because that index is built from their observed combinations. If an index variable that keeps its levels is also used in `mods` or `scale` and loses one of its levels to the selection, the fit stops with an error of class `RoBMA_structured_index_level_removed` (also `RoBMA_data_error`); remove the level from the data with `droplevels()` or use a separate column for the moderator.

`measure` a character string specifying the effect size measure. Normal/effect-size constructors require an explicit value and support `"SMD"`, `"ZCOR"`, `"RR"`, `"OR"`, `"HR"`, `"RD"`, `"IRR"`, and `"GEN"`. Use `"GEN"` only for general effect sizes without a known unit information standard deviation. GLMM raw-count constructors support only `"OR"` and `"IRR"` and default to `"OR"`.

`prior_effect` prior distribution for the effect size (μ) parameter (the intercept). If omitted and consumed, a default prior is constructed. In single-model functions, explicit `NULL` or `FALSE` sets a spike at zero.

`prior_heterogeneity`
prior distribution for the heterogeneity (τ) parameter. If omitted and consumed, a default prior is constructed. In single-model functions, explicit `NULL` or `FALSE` sets a spike at zero. For `brma.mv` random-effect formulas, this can also be a partial or complete `prior_random` object; see `random_effect_prior_specification`.

`prior_mods` prior distribution for the moderators (β) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or `NULL`, default priors are used. This argument can be specified only when `mods` is specified. A formula without an intercept (`~ 0 + x`) fixes the location intercept at zero, so an `intercept` entry of the list is accepted only as a point prior at zero; any other `intercept` prior stops with an error of class `RoBMA_prior_intercept_conflict` (also of class `RoBMA_prior_error`). A named `intercept` entry supplies the complete intercept prior. Explicit competing global effect declarations are refused when this would discard them. Named `NULL` or `FALSE` intercept entries are invalid; specify the complete prior in one place.

`prior_scale` prior distribution for the scale (δ) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or `NULL`, default priors are used. This argument can be specified only when `scale` is specified. A named `intercept` entry supplies the complete positive SD prior used by `log(intercept)`. Global heterogeneity declarations remain meaningful when they supply another random root or its model-inclusion weights; declarations fully replaced without a consumer raise `RoBMA_prior_intercept_conflict`.

`prior_bias` PET regression-coefficient prior created by `prior_PET()`. If omitted or `NULL`, the default is a Cauchy prior centered at 0, truncated to positive values, with scale from `RoBMA.get_option("default_bias_PET.scale")`.

`standardize_continuous_predictors`
logical. Whether to standardize continuous predictors. Defaults to `TRUE`.

`set_contrast_factor_predictors`
character. How to set contrast for factor predictors. Defaults are constructor-

specific and shown in each function usage. Single-model constructors use "treatment"; model-averaging constructors use "meandif". Accepted values are "treatment", "meandif", "orthonormal", and "independent". Independent coding creates one coefficient per factor level. Random-coefficient blocks reuse the resolved fixed-factor contrast unless random_block(contrasts = ...) overrides it. A warning is issued when default or inherited independent contrasts make a fixed or random design overspecified. With "meandif" or "orthonormal" contrasts, a formula term that interacts a factor with a predictor whose own term is missing, such as g:x in ~ g + g:x or ~ g / x, stops the fit: the term codes g by level indicators, with one coefficient per level, on which mean-difference and orthonormal priors are not defined. Specify an independent prior for that term, e.g., prior_mods = list("g:x" = prior_factor("normal", list(0, 1), contrast = "independent")) (or prior_scale for a scale formula), include the missing term (~ g * x), or use set_contrast_factor_predictors = "treatment".

prior_unit_information_sd
: numeric. The unit information standard deviation (σ_{unit}). Cannot be used together with prior_informed_field.

rescale_priors
: numeric. A scaling factor for supported prior distributions. Point and none priors are unchanged; supplied prior_random() scale architectures and publication-bias priors are not rescaled. The default PEESE prior separately uses its documented UISD adjustment, independently of rescale_priors. Defaults to 1.

prior_informed_field
: character. The field of the informed prior distributions. Omit to use the standard default prior specification; explicit NULL is invalid.

prior_informed_subfield
: character. The subfield of the informed prior distributions. Can only be specified together with prior_informed_field. Omit to use the field-specific default, such as "Cochrane" for prior_informed_field = "medicine"; explicit NULL is invalid.

effect_direction
: direction used by publication-bias adjustments. "positive" assumes statistically significant positive estimates are more likely to be selected; "negative" mirrors the selection direction; "detect" infers the direction from the fitted data.

known_v_parameterization
: known-V backend used by brma.mv(). "auto" chooses an exact backend when feasible; "latent" uses a latent D + BB' decomposition; "whitened" uses an eigen-rotated normal likelihood; "block_mvn" uses an exact native block multivariate-normal likelihood.

sample
: numeric. Number of MCMC samples to save. Defaults to 5000.

burnin
: numeric. Number of burn-in iterations. Defaults to 2000.

adapt
: numeric. Number of adaptation iterations. Defaults to 500.

chains
: numeric. Number of MCMC chains. Defaults to 3.

thin
: numeric. Thinning interval. Defaults to 1.

parallel
: logical. Whether to run MCMC chains in parallel. Defaults to FALSE.

autofit	logical. Whether to automatically extend the MCMC chains if convergence is not met. Defaults to FALSE.
autofit_control	list of autofit control settings. See set_autofit_control() for details.
convergence_checks	list of convergence check settings. See set_convergence_checks() for details.
seed	numeric. Random seed for reproducibility. Defaults to NULL.
silent	logical. Whether to suppress output. Constructors with no explicit default use RoBMA.get_option("silent") when silent is omitted. Model-averaging wrappers default to TRUE unless explicitly changed.
...	additional advanced arguments. Fitting functions reject unused arguments; currently recognized internal arguments include only_data, only_priors, is_JASP, and is_JASP_prefix.
vi	a vector of sampling variances. Either vi or sei must be supplied for normal models.
sei	a vector of standard errors. Either vi or sei must be supplied for normal models.

Details

Omitted or NULL random specifies a fixed-effect model with no implicit heterogeneity. Heterogeneity priors and scale formulas require an explicit random structure; the known sampling covariance remains unchanged.

bPET.mv() combines the PET publication-bias regression of bPET() with the known-covariance and random-formula model of brma.mv(). The PET predictor is the marginal sampling standard error sqrt(diag(V)); known off-diagonal sampling covariance remains part of the joint likelihood.

The supported known-V backends, random-effect marginalization rules, predictive targets, and diagnostic semantics are the same as for brma.mv(). In particular, random replaces the specialized cluster argument used by bPET().

Value

A fitted object of class c("bPET.mv", "bPET", "brma.mv", "brma.norm", "brma").

See Also

bPET(), bPEESE.mv(), brma.mv(), summary.brma()

Examples

```
## Not run:
V <- matrix(c(0.04, 0.01, 0.01, 0.09), 2, 2)
fit <- bPET.mv(
  yi      = c(0.10, 0.20),
  V       = V,
  measure = "GEN",
  seed    = 1,
  silent  = TRUE
)
```

```
summary(fit)

## End(Not run)
```

`bridge_sampler.brma` *Bridge Sampling for brma Objects*

Description

Extract the marginal likelihood bridge sampling object from a brma model. The marginal likelihood must first be computed using `add_marglik`.

Usage

```
## S3 method for class 'brma'
bridge_sampler(samples, ...)
```

Arguments

`samples`	a brma model object.
`...`	additional arguments (currently not used).

Details

This function extracts the raw **bridgesampling** object underlying the marginal-likelihood result stored by `add_marglik`. If the marginal likelihood has not been computed, an error is thrown. Product-space model-averaging objects (`BMA.norm`, `BMA.glmm`, `BMA.mv`, `RoBMA`, and `RoBMA.mv`) do not expose bridge-sampling marginal likelihoods.

Fully fixed models can have a zero-dimensional marginal likelihood evaluated exactly. Such fits have no bridge-sampling object, so this method raises a `RoBMA_exact_marglik_no_bridge` error; use `logml` to retrieve their exact log marginal likelihood. For Bayesian model comparison, pass fitted `brma` objects directly to `bf` or `post_prob`.

When a stored marginal-likelihood result has no raw bridge object, extraction raises `RoBMA_bridge_unavailable`. Its `reason` is `"exact_zero_dimensional"` for an exact result, or `"missing_upstream"` when a stored result lacks its raw bridge object. For the latter, recompute the result with `add_marglik`.

Value

An object of class `"bridge"`, or `"bridge_list"` for repeated bridge estimates, as returned by `bridge_sampler`.

See Also

`add_marglik`, `bridge_sampler`, `logml.brma`, `bf.brma`, `post_prob.brma`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- brma(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")

  fit <- add_marglik(fit)

  bridge <- bridge_sampler(fit)
  print(bridge)
}

## End(Not run)
```

---

brma *Bayesian Meta-Analysis*

Description

Function for fitting normal-likelihood/effect-size random-effects, meta-regression, multilevel, and location-scale meta-analytic models. `brma.norm()` is an alias for `brma()`; raw-count GLMM models use `brma.glmm()`.

Usage

```
brma(
  yi,
  vi,
  sei,
  weights,
  ni,
  mods,
  scale,
  cluster,
  data,
  slab,
  subset,
  measure,
  prior_effect,
  prior_heterogeneity,
  prior_mods,
  prior_scale,
  prior_heterogeneity_allocation,
  standardize_continuous_predictors = TRUE,
  set_contrast_factor_predictors = "treatment",
  prior_unit_information_sd,
```

```
  rescale_priors = 1,
  prior_informed_field,
  prior_informed_subfield,
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  chains = 3,
  thin = 1,
  parallel = FALSE,
  autofit = FALSE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  seed = NULL,
  silent,
  ...
)
```

Arguments

yi	a vector of effect sizes, or a formula with the effect size on the left-hand side and location moderators on the right-hand side (for example `yi ~ x1 + x2`). If a formula is supplied, `mods` must not be specified.
vi	a vector of sampling variances. Either `vi` or `sei` must be supplied for normal models.
sei	a vector of standard errors. Either `vi` or `sei` must be supplied for normal models.
weights	an optional vector of finite positive likelihood weights. For normal/effect-size models, each weight powers the estimate likelihood. For constructors with GLMM raw-count input, each weight powers the paired two-arm likelihood for one study. An integer weight w repeats that conditional likelihood contribution w times, with the declared latent-effect and grouping structure held fixed. Positive fractional likelihood powers are also supported.
ni	an optional vector of sample sizes. Used for `measure = "GEN"` or when estimating the unit information standard deviation.
mods	an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data` and supports explicit columns and standard formula operators, including `^` as R's crossing operator with a literal whole exponent of at least 2: `(x + z)^2` expands to `x + z + x:z`, and `x^2` is not a quadratic term (`x + x^2` is `x`). Precompute transformed predictors, such as a squared predictor, as columns in `data`; inline calls such as `I()`, `poly()`, spline constructors, and user functions are not supported.
scale	an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data` and follows the same explicit-column formula grammar as `mods`. Random-formula models also accept a named list targeting top-level random components or concrete random-effect blocks; see `brma.mv()`.
cluster	an optional vector of cluster identifiers for multilevel meta-analysis.

`data` an optional data frame containing the variables.

`slab` an optional vector of study labels.

`subset` an optional logical or numeric vector specifying a subset of data to be used. Subsetting precedes missing-row removal; retained-row outcome and predictor checks follow these steps. Type, shape, and existing extraction checks can still reject malformed inputs before subsetting. Row selection (by `subset` or missing-row removal) drops factor levels left without rows from `mods`, `scale`, and random-effect grouping factors. The index variable of a structured random-effect term (`cs()`, `hcs()`, `ar1()`, `har()` with one index variable) keeps its declared levels, which define the covariance structure; the variables of a multivariable `cs(a + b | group)` index follow the ordinary rule, because that index is built from their observed combinations. If an index variable that keeps its levels is also used in `mods` or `scale` and loses one of its levels to the selection, the fit stops with an error of class `RoBMA_structured_index_level_removed` (also `RoBMA_data_error`); remove the level from the data with `droplevels()` or use a separate column for the moderator.

`measure` a character string specifying the effect size measure. Normal/effect-size constructors require an explicit value and support `"SMD"`, `"ZCOR"`, `"RR"`, `"OR"`, `"HR"`, `"RD"`, `"IRR"`, and `"GEN"`. Use `"GEN"` only for general effect sizes without a known unit information standard deviation. GLMM raw-count constructors support only `"OR"` and `"IRR"` and default to `"OR"`.

`prior_effect` prior distribution for the effect size (μ) parameter (the intercept). If omitted and consumed, a default prior is constructed. In single-model functions, explicit `NULL` or `FALSE` sets a spike at zero.

`prior_heterogeneity`
prior distribution for the heterogeneity (τ) parameter. If omitted and consumed, a default prior is constructed. In single-model functions, explicit `NULL` or `FALSE` sets a spike at zero. For `brma.mv` random-effect formulas, this can also be a partial or complete `prior_random` object; see `random_effect_prior_specification`.

`prior_mods` prior distribution for the moderators (β) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or `NULL`, default priors are used. This argument can be specified only when `mods` is specified. A formula without an intercept (`~ 0 + x`) fixes the location intercept at zero, so an `intercept` entry of the list is accepted only as a point prior at zero; any other `intercept` prior stops with an error of class `RoBMA_prior_intercept_conflict` (also of class `RoBMA_prior_error`). A named `intercept` entry supplies the complete intercept prior. Explicit competing global effect declarations are refused when this would discard them. Named `NULL` or `FALSE` intercept entries are invalid; specify the complete prior in one place.

`prior_scale` prior distribution for the scale (δ) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or `NULL`, default priors are used. This argument can be specified only when `scale` is specified. A named `intercept` entry supplies the complete positive SD prior used by `log(intercept)`. Global heterogeneity declarations remain meaningful when they supply another random root or its model-inclusion weights; declarations fully replaced without a consumer raise `RoBMA_prior_intercept_conflict`.

`prior_heterogeneity_allocation`
: prior distribution for the fraction of heterogeneity allocated to the cluster-level component in multilevel models (ρ, a variance fraction rather than a random-coefficient correlation). If omitted or `NULL`, defaults to `Beta(1, 1)`. This argument can be specified only when `cluster` is specified.

`standardize_continuous_predictors`
: logical. Whether to standardize continuous predictors. Defaults to `TRUE`.

`set_contrast_factor_predictors`
: character. How to set contrast for factor predictors. Defaults are constructor-specific and shown in each function usage. Single-model constructors use `"treatment"`; model-averaging constructors use `"meandif"`. Accepted values are `"treatment"`, `"meandif"`, `"orthonormal"`, and `"independent"`. Independent coding creates one coefficient per factor level. Random-coefficient blocks reuse the resolved fixed-factor contrast unless `random_block(contrasts = ...)` overrides it. A warning is issued when default or inherited independent contrasts make a fixed or random design overspecified. With `"meandif"` or `"orthonormal"` contrasts, a formula term that interacts a factor with a predictor whose own term is missing, such as `g:x` in `~ g + g:x` or `~ g / x`, stops the fit: the term codes `g` by level indicators, with one coefficient per level, on which mean-difference and orthonormal priors are not defined. Specify an independent prior for that term, e.g., `prior_mods = list("g:x" = prior_factor("normal", list(0, 1), contrast = "independent"))` (or `prior_scale` for a scale formula), include the missing term (`~ g * x`), or use `set_contrast_factor_predictors = "treatment"`.

`prior_unit_information_sd`
: numeric. The unit information standard deviation (σ_{unit}). Cannot be used together with `prior_informed_field`.

`rescale_priors`
: numeric. A scaling factor for supported prior distributions. Point and none priors are unchanged; supplied `prior_random()` scale architectures and publication-bias priors are not rescaled. The default PEESE prior separately uses its documented UISD adjustment, independently of `rescale_priors`. Defaults to 1.

`prior_informed_field`
: character. The field of the informed prior distributions. Omit to use the standard default prior specification; explicit `NULL` is invalid.

`prior_informed_subfield`
: character. The subfield of the informed prior distributions. Can only be specified together with `prior_informed_field`. Omit to use the field-specific default, such as `"Cochrane"` for `prior_informed_field = "medicine"`; explicit `NULL` is invalid.

`sample`
: numeric. Number of MCMC samples to save. Defaults to `5000`.

`burnin`
: numeric. Number of burn-in iterations. Defaults to `2000`.

`adapt`
: numeric. Number of adaptation iterations. Defaults to `500`.

`chains`
: numeric. Number of MCMC chains. Defaults to `3`.

`thin`
: numeric. Thinning interval. Defaults to `1`.

`parallel`
: logical. Whether to run MCMC chains in parallel. Defaults to `FALSE`.

`autofit`
: logical. Whether to automatically extend the MCMC chains if convergence is not met. Defaults to `FALSE`.

`autofit_control`
: list of autofit control settings. See `set_autofit_control()` for details.

`convergence_checks`
: list of convergence check settings. See `set_convergence_checks()` for details.

`seed`
: numeric. Random seed for reproducibility. Defaults to `NULL`.

`silent`
: logical. Whether to suppress output. Constructors with no explicit default use `RoBMA.get_option("silent")` when `silent` is omitted. Model-averaging wrappers default to `TRUE` unless explicitly changed.

`...`
: additional advanced arguments. Fitting functions reject unused arguments; currently recognized internal arguments include `only_data`, `only_priors`, `is_JASP`, and `is_JASP_prefix`.

Details

Prior distributions:

Prior distributions must be specified for all model parameters. This typically includes the pooled effect μ and between-study heterogeneity τ. In the case of meta-regression, the pooled effect μ corresponds to the intercept, and additional prior distributions for the regression coefficients are required. In the case of a location-scale model, the between-study heterogeneity corresponds to the intercept of the scale regression, and additional prior distributions for the scale regression coefficients are required.

There are several ways to specify the prior distributions:

1. via a standardized effect size `measure` with known unit information standard deviation,
2. by estimating unit information standard deviation using sample sizes `ni`,
3. by manually setting `prior_unit_information_sd`,
4. by specifying informed empirical prior distributions via `prior_informed_field` and `prior_informed_subfield`,
5. or via fully custom specification using the `prior_effect`, `prior_heterogeneity`, `prior_mods`, `prior_scale`, and `prior_heterogeneity_allocation` arguments.

In all cases, the prior behavior can be further modified by the `rescale_priors`, `standardize_continuous_predictors`, and `set_contrast_factor_predictors` arguments.

(1) Specifying prior distributions via standardized effect size measures with known:
unit information standard deviation This is the easiest way to specify prior distributions. The width of prior distributions is based on a fraction of the known unit information standard deviation (`UISD`) (Röver et al. 2021). The default prior distributions for the parameters are set as follows:

effect size:	Normal(0, $\frac{1}{2}$ `UISD`)
heterogeneity:	Normal+(0, $\frac{1}{4}$ `UISD`)
effect moderation:	Normal(0, $\frac{1}{4}$ `UISD`)
heterogeneity moderation:	Normal(0, $\frac{1}{2}$)

The heterogeneity moderation parameters are multiplicative, as such they are independent of `UISD`.
The default fraction of the `UISD` can be changed via `RoBMA.options()` using one of the following arguments: `"default_UISD.effect"`, `"default_UISD.heterogeneity"`, `"default_UISD.mods"`, `"default_UISD.scale"`.

The known UISD for standardized effect size measures (measure) are set as follows:

"SMD":	$\sqrt{2}$
"ZCOR":	1
"RR":	$\sqrt{4}$
"OR":	$\sqrt{4}$
"HR":	$\sqrt{4}$
"IRR":	$\sqrt{4}$

See Chapter 2.4 in Spiegelhalter et al. (2004) and Chapter 1 in Grieve (2022).

(2) Estimating unit information standard deviation using sample sizes:
When effect sizes are on a non-standardized scale (measure = "GEN") or use a standardized effect size without known UISD, the UISD can be estimated from sample sizes (ni) and standard errors (sei) following Equation 6 in Röver et al. (2021). The estimated UISD is then used to scale the default prior distributions as described in section (1).
Note that the known UISD for standardized effect size measures (section (1)) is used if available, even when ni is provided.

(3) Manually setting unit information standard deviation:
Alternatively, the UISD can be specified directly via the prior_unit_information_sd argument. This is useful when the appropriate scale for prior distributions is known a priori or when multiple analyses are to be performed on subsets of the same data (re-estimating UISD on different subsets of the data can lead to slightly different prior distributions for different subsets; see estimate_unit_information_sd()). The specified prior_unit_information_sd is then used to scale the default prior distributions as described in section (1).
Note that the manually specified prior_unit_information_sd takes precedence over the estimated UISD from ni (section (2)) and the known UISD from measure (section (1)). It cannot be combined with prior_informed_field.

(4) Specifying informed empirical prior distributions:
Informed prior distributions can be specified via the prior_informed_field and prior_informed_subfield arguments. Currently, only prior_informed_field = "medicine" with subfields defined in BayesTools::prior_informed_medicine_names is supported, which uses empirically derived prior distributions from medical meta-analyses as described in Bartoš et al. (2021) and Bartoš et al. (2023).
When prior_informed_field = "medicine", the default prior_informed_subfield is "Cochrane" (i.e., using the whole CDSR database as a reference). The informed prior distributions are available for the following effect size measures:

"SMD":	standardized mean difference
"OR":	log odds ratio
"RR":	log risk ratio
"RD":	risk difference
"HR":	log hazard ratio

Note that informed prior distributions are only available for the effect size (μ) and heterogeneity (τ) parameters. For effect moderation (prior_mods), the informed effect prior is scaled by a factor of RoBMA.get_option("default_informed_priors.mods"). For heterogeneity moderation (prior_scale), a normal prior with standard deviation specified by RoBMA.get_option("default_informed_prio
is used.

(5) Fully custom prior distributions:
Prior distributions can be fully customized by directly specifying the `prior_effect`, `prior_heterogeneity`, `prior_mods`, `prior_scale`, and `prior_heterogeneity_allocation` arguments. These should be prior distribution objects created via `BayesTools::prior()` or related functions (e.g., `prior_factor()`).

Rescaling prior distributions:
The `rescale_priors` argument allows rescaling supported prior distributions by a multiplicative factor. For example, `rescale_priors = 2` doubles the standard deviations/scales of normal, Cauchy, t, and inverse-gamma prior distributions, making them more diffuse. Point and none priors are unchanged.

Handling of continuous and factor predictors:
When `standardize_continuous_predictors = TRUE`, continuous moderator and scale predictors are internally standardized before fitting. Reported summaries are transformed back to the original predictor scale by default; use `standardized_coefficients = TRUE` in `summary()` or related methods to inspect coefficients on the standardized scale.
Factor predictors use the contrast specified by `set_contrast_factor_predictors`. The default `"treatment"` follows standard treatment coding. Model-averaging functions default to `"meandif"` so inclusion priors for factor levels are centered on deviations from the grand mean.

Value

A fitted object of class `c("brma.norm", "brma")`. The object contains checked `data`, checked `priors`, the JAGS `fit`, cached `summary`, and cached `coefficients`. If the corresponding package options are enabled, it can also contain cached LOO, WAIC, or marginal likelihood results. The advanced internal `only_data = TRUE` and `only_priors = TRUE` paths return partially constructed objects.

References

Bartoš F, Gronau QF, Timmers B, Otte WM, Ly A, Wagenmakers E (2021). "Bayesian model-averaged meta-analysis in medicine." *Statistics in Medicine*, **40**(30), 6743–6761. doi:10.1002/sim.9170.

Bartoš F, Otte WM, Gronau QF, Timmers B, Ly A, Wagenmakers E (2023). "Empirical prior distributions for Bayesian meta-analyses of binary and time-to-event outcomes." doi:10.48550/arXiv.2306.11468. Preprint available at https://doi.org/10.48550/arXiv.2306.11468.

Grieve AP (2022). *Hybrid frequentist/Bayesian power and Bayesian power in planning clinical trials*. Chapman and Hall/CRC.

Röver C, Bender R, Dias S, Schmid CH, Schmidli H, Sturtz S, Weber S, Friede T (2021). "On weakly informative prior distributions for the heterogeneity parameter in Bayesian random-effects meta-analysis." *Research Synthesis Methods*, **12**(4), 448–474. doi:10.1002/jrsm.1475.

Spiegelhalter DJ, Abrams KR, Myles JP (2004). *Bayesian approaches to clinical trials and health-care evaluation*. John Wiley and Sons. doi:10.1002/0470092602.

See Also

`RoBMA()`, `BMA()`, `brma.glmm()`, `summary.brma()`, `plot.brma()`, `predict.brma()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE) &&
    requireNamespace("metafor", quietly = TRUE)) {
  data(dat.bcg, package = "metadat")
  dat <- metafor::escalc(
    measure = "RR",
    ai      = tpos,
    bi      = tneg,
    ci      = cpos,
    di      = cneg,
    data    = dat.bcg
  )

  fit <- brma(
    yi      = yi,
    vi      = vi,
    mods    = ~ ablat + year,
    data    = dat,
    measure = "RR",
    seed    = 1,
    silent  = TRUE
  )

  summary(fit)
  predict(fit, type = "terms")
}

## End(Not run)
```

`brma.glmm` *Bayesian Generalized Meta-Analysis*

Description

Function for fitting random-effects, meta-regression, multilevel, and location-scale meta-analytic models directly to either binary or count data.

Usage

```
brma.glmm(
  ai,
  bi,
  ci,
  di,
  n1i,
  n2i,
  x1i,
```

```
  x2i,
  t1i,
  t2i,
  weights,
  mods,
  scale,
  cluster,
  data,
  slab,
  subset,
  measure = "OR",
  prior_effect,
  prior_heterogeneity,
  prior_mods,
  prior_scale,
  prior_heterogeneity_allocation,
  prior_baserate,
  prior_lograte,
  standardize_continuous_predictors = TRUE,
  set_contrast_factor_predictors = "treatment",
  prior_unit_information_sd,
  rescale_priors = 1,
  prior_informed_field,
  prior_informed_subfield,
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  chains = 3,
  thin = 1,
  parallel = FALSE,
  autofit = FALSE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  seed = NULL,
  silent,
  ...
)
```

Arguments

ai	a vector of the number of events in the treatment or experimental group for binomial GLMM models.
bi	a vector of the number of non-events in the treatment or experimental group for binomial GLMM models.
ci	a vector of the number of events in the control group for binomial GLMM models.
di	a vector of the number of non-events in the control group for binomial GLMM models.

`n1i` a vector of the sample size in the treatment or experimental group. If omitted for binomial GLMMs, it is computed as `ai + bi`.

`n2i` a vector of the sample size in the control group. If omitted for binomial GLMMs, it is computed as `ci + di`.

`x1i` a vector of the number of events in the treatment/experimental group (for Poisson data).

`x2i` a vector of the number of events in the control group (for Poisson data).

`t1i` a vector of the person-time in the treatment/experimental group.

`t2i` a vector of the person-time in the control group.

`weights` an optional vector of finite positive likelihood weights. For normal/effect-size models, each weight powers the estimate likelihood. For constructors with GLMM raw-count input, each weight powers the paired two-arm likelihood for one study. An integer weight w repeats that conditional likelihood contribution w times, with the declared latent-effect and grouping structure held fixed. Positive fractional likelihood powers are also supported.

`mods` an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data` and supports explicit columns and standard formula operators, including `^` as R's crossing operator with a literal whole exponent of at least 2: `(x + z)^2` expands to `x + z + x:z`, and `x^2` is not a quadratic term (`x + x^2` is `x`). Precompute transformed predictors, such as a squared predictor, as columns in `data`; inline calls such as `I()`, `poly()`, spline constructors, and user functions are not supported.

`scale` an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data` and follows the same explicit-column formula grammar as `mods`. Random-formula models also accept a named list targeting top-level random components or concrete random-effect blocks; see `brma.mv()`.

`cluster` an optional vector of cluster identifiers for multilevel meta-analysis.

`data` an optional data frame containing the variables.

`slab` an optional vector of study labels.

`subset` an optional logical or numeric vector specifying a subset of data to be used. Subsetting precedes missing-row removal; retained-row outcome and predictor checks follow these steps. Type, shape, and existing extraction checks can still reject malformed inputs before subsetting. Row selection (by `subset` or missing-row removal) drops factor levels left without rows from `mods`, `scale`, and random-effect grouping factors. The index variable of a structured random-effect term (`cs()`, `hcs()`, `ar1()`, `har()` with one index variable) keeps its declared levels, which define the covariance structure; the variables of a multivariable `cs(a + b | group)` index follow the ordinary rule, because that index is built from their observed combinations. If an index variable that keeps its levels is also used in `mods` or `scale` and loses one of its levels to the selection, the fit stops with an error of class `RoBMA_structured_index_level_removed` (also `RoBMA_data_error`); remove the level from the data with `droplevels()` or use a separate column for the moderator.

measure — a character string specifying the effect size measure. Normal/effect-size constructors require an explicit value and support "SMD", "ZCOR", "RR", "OR", "HR", "RD", "IRR", and "GEN". Use "GEN" only for general effect sizes without a known unit information standard deviation. GLMM raw-count constructors support only "OR" and "IRR" and default to "OR".

prior_effect — prior distribution for the effect size (μ) parameter (the intercept). If omitted and consumed, a default prior is constructed. In single-model functions, explicit NULL or FALSE sets a spike at zero.

prior_heterogeneity — prior distribution for the heterogeneity (τ) parameter. If omitted and consumed, a default prior is constructed. In single-model functions, explicit NULL or FALSE sets a spike at zero. For brma.mv random-effect formulas, this can also be a partial or complete prior_random object; see random_effect_prior_specification.

prior_mods — prior distribution for the moderators (β) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or NULL, default priors are used. This argument can be specified only when mods is specified. A formula without an intercept (~ 0 + x) fixes the location intercept at zero, so an intercept entry of the list is accepted only as a point prior at zero; any other intercept prior stops with an error of class RoBMA_prior_intercept_conflict (also of class RoBMA_prior_error). A named intercept entry supplies the complete intercept prior. Explicit competing global effect declarations are refused when this would discard them. Named NULL or FALSE intercept entries are invalid; specify the complete prior in one place.

prior_scale — prior distribution for the scale (δ) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or NULL, default priors are used. This argument can be specified only when scale is specified. A named intercept entry supplies the complete positive SD prior used by log(intercept). Global heterogeneity declarations remain meaningful when they supply another random root or its model-inclusion weights; declarations fully replaced without a consumer raise RoBMA_prior_intercept_conflict.

prior_heterogeneity_allocation — prior distribution for the fraction of heterogeneity allocated to the cluster-level component in multilevel models (ρ, a variance fraction rather than a random-coefficient correlation). If omitted or NULL, defaults to Beta(1, 1). This argument can be specified only when cluster is specified.

prior_baserate — prior distribution for the estimate-specific midpoint base-rate probability in binomial GLMM models (measure = "OR"). If omitted or NULL, defaults to independent Beta(1, 1) priors. Continuous priors are restricted to [0, 1] (see prior_specification) and point priors must lie strictly inside (0, 1). The estimate-unit log-likelihood of the model (log_lik(), LOO, WAIC) integrates the base rate over any of these priors: point priors and untruncated beta priors adaptively (a uniform(0, 1) prior is a Beta(1, 1) prior), other families on a refined prior-CDF grid. The grid is accepted only when its log-likelihood changes by less than 1e-6 between refinements; any prior without an adaptive route (other than an untruncated beta prior; a truncated beta prior takes the grid too) can fail to reach this, especially with few events or very large arms, and then stops with an error of class RoBMA_glmm_grid_nonconvergence, as a lognormal base-rate

prior does on cells with few events. A beta prior avoids it. The automatic computation of LOO/WAIC after fitting warns instead and returns the fit without them.

`prior_lograte` prior distribution for the estimate-specific midpoint log-rate in Poisson GLMM models (`measure = "IRR"`). If omitted or `NULL`, a normal prior centered at the pooled crude log-rate is used independently for each estimate. Its standard deviation defaults to `1` and can be changed via `RoBMA.options(default_lograte.sd = ...)`. Rescaling all person-times shifts the prior mean by the corresponding log conversion while leaving its standard deviation unchanged. The estimate-unit log-likelihood of the model (`log_lik()`, LOO, WAIC) integrates the log-rate over the prior: point priors and untruncated normal priors adaptively, other families on a refined prior-CDF grid. The grid is accepted only when its log-likelihood changes by less than `1e-6` between refinements; any prior without an adaptive route (other than an untruncated normal prior; a truncated normal prior takes the grid too) can fail to reach this, especially with few events or very large arms, and then stops with an error of class `RoBMA_glmm_grid_nonconvergence`, as gamma, lognormal, and inverse-gamma log-rate priors do on cells with few events. A normal prior avoids it. The automatic computation of LOO/WAIC after fitting warns instead and returns the fit without them.

`standardize_continuous_predictors`
logical. Whether to standardize continuous predictors. Defaults to `TRUE`.

`set_contrast_factor_predictors`
character. How to set contrast for factor predictors. Defaults are constructor-specific and shown in each function usage. Single-model constructors use `"treatment"`; model-averaging constructors use `"meandif"`. Accepted values are `"treatment"`, `"meandif"`, `"orthonormal"`, and `"independent"`. Independent coding creates one coefficient per factor level. Random-coefficient blocks reuse the resolved fixed-factor contrast unless `random_block(contrasts = ...)` overrides it. A warning is issued when default or inherited independent contrasts make a fixed or random design overspecified. With `"meandif"` or `"orthonormal"` contrasts, a formula term that interacts a factor with a predictor whose own term is missing, such as `g:x` in `~ g + g:x` or `~ g / x`, stops the fit: the term codes `g` by level indicators, with one coefficient per level, on which mean-difference and orthonormal priors are not defined. Specify an independent prior for that term, e.g., `prior_mods = list("g:x" = prior_factor("normal", list(0, 1), contrast = "independent"))` (or `prior_scale` for a scale formula), include the missing term (`~ g * x`), or use `set_contrast_factor_predictors = "treatment"`.

`prior_unit_information_sd`
numeric. The unit information standard deviation (σ_{unit}). Cannot be used together with `prior_informed_field`.

`rescale_priors` numeric. A scaling factor for supported prior distributions. Point and none priors are unchanged; supplied `prior_random()` scale architectures and publication-bias priors are not rescaled. The default PEESE prior separately uses its documented UISD adjustment, independently of `rescale_priors`. Defaults to 1.

`prior_informed_field`
character. The field of the informed prior distributions. Omit to use the standard default prior specification; explicit `NULL` is invalid.

`prior_informed_subfield`
: character. The subfield of the informed prior distributions. Can only be specified together with `prior_informed_field`. Omit to use the field-specific default, such as `"Cochrane"` for `prior_informed_field = "medicine"`; explicit `NULL` is invalid.

`sample`
: numeric. Number of MCMC samples to save. Defaults to `5000`.

`burnin`
: numeric. Number of burn-in iterations. Defaults to `2000`.

`adapt`
: numeric. Number of adaptation iterations. Defaults to `500`.

`chains`
: numeric. Number of MCMC chains. Defaults to `3`.

`thin`
: numeric. Thinning interval. Defaults to `1`.

`parallel`
: logical. Whether to run MCMC chains in parallel. Defaults to `FALSE`.

`autofit`
: logical. Whether to automatically extend the MCMC chains if convergence is not met. Defaults to `FALSE`.

`autofit_control`
: list of autofit control settings. See `set_autofit_control()` for details.

`convergence_checks`
: list of convergence check settings. See `set_convergence_checks()` for details.

`seed`
: numeric. Random seed for reproducibility. Defaults to `NULL`.

`silent`
: logical. Whether to suppress output. Constructors with no explicit default use `RoBMA.get_option("silent")` when `silent` is omitted. Model-averaging wrappers default to `TRUE` unless explicitly changed.

`...`
: additional advanced arguments. Fitting functions reject unused arguments; currently recognized internal arguments include `only_data`, `only_priors`, `is_JASP`, and `is_JASP_prefix`.

Details

Model for odds ratios (`measure = "OR"`) corresponds to Model 4 described in Jackson et al. (2018). `logit(pi[i])` is the study-specific midpoint of the two arm logits. `prior_baserate` defines the estimate-specific prior distribution on `pi[i]`.

Model for incidence rate ratios (`measure = "IRR"`) corresponds to Bagos and Nikolopoulos (2009). `phi[i]` is the study-specific midpoint of the two arm log incidence rates. `prior_lograte` defines the estimate-specific prior distribution on `phi[i]`. If unspecified, a unit-information prior is based on the data and used independently for each estimate.

When `weights` are supplied, they are treated as likelihood weights on the paired two-arm study contribution.

Value

A fitted object of class `c("brma.glmm", "brma")`. The object contains checked `data`, checked `priors`, the JAGS `fit`, cached `summary`, and cached `coefficients`. If the corresponding package options are enabled, it can also contain cached LOO, WAIC, or marginal likelihood results.

See Also

`brma()`, `BMA.glmm()`, `summary.brma()`, `predict.brma()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.bcg, package = "metadat")

  fit <- brma.glmm(
    ai      = tpos,
    bi      = tneg,
    ci      = cpos,
    di      = cneg,
    mods    = ~ alloc,
    data    = dat.bcg,
    measure = "OR",
    seed    = 1,
    silent  = TRUE
  )

  summary(fit)
}

## End(Not run)
```

`brma.mv` *Bayesian Meta-Analysis With Known Sampling Covariance*

Description

Fits normal-likelihood Bayesian meta-analytic models with a known working variance-covariance matrix `V`. The interface supports location moderators, BayesTools random-effect formulas using random-effect formula structure tags, and `brma.mv()`-specific known-`V` backends.

Usage

```
brma.mv(
  yi,
  V,
  ni,
  mods,
  scale,
  random,
  R = NULL,
  Rscale = "cor",
  data,
  slab,
  subset,
  measure,
  prior_effect,
```

```
  prior_heterogeneity,
  prior_mods,
  prior_scale,
  standardize_continuous_predictors = TRUE,
  set_contrast_factor_predictors = "treatment",
  prior_unit_information_sd,
  rescale_priors = 1,
  prior_informed_field,
  prior_informed_subfield,
  known_v_parameterization = "auto",
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  chains = 3,
  thin = 1,
  parallel = FALSE,
  autofit = FALSE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  seed = NULL,
  silent,
  ...,
  vi = NULL,
  sei = NULL
)
```

Arguments

yi	a vector of effect sizes, or a formula with the effect size on the left-hand side and location moderators on the right-hand side (for example `yi ~ x1 + x2`). If a formula is supplied, `mods` must not be specified.
V	a known working variance-covariance matrix, a list of block variance-covariance matrices, or an exact diagonal-plus-factor declaration created by `known_v_factor()`, used by `brma.mv()`. Ordinary covariance matrices from `metafor::vcalc()` can be supplied directly. Product selection requires no publication groups. For best-p-value selection, declare groups separately with `selection_model(weight_rule = "best", group = ...)`. Matrix symmetry uses base R's numerical tolerance. Accepted matrices are stored symmetrically by averaging off-diagonal pairs; the supplied object and its diagonal are unchanged. Dependency blocks that are block constant on nested partitions are represented exactly as a diagonal-plus-factor structure without a `known_v_factor()` declaration. Dimensions and row alignment are checked on the complete input; when rows are dropped (`subset` or missing values), the numerical covariance checks apply to the retained rows only, so acceptance says nothing about the entries of the dropped rows.
ni	an optional vector of sample sizes. Used for `measure = "GEN"` or when estimating the unit information standard deviation.
mods	an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data` and supports explicit columns

and standard formula operators, including `^` as R's crossing operator with a literal whole exponent of at least 2: `(x + z)^2` expands to `x + z + x:z`, and `x^2` is not a quadratic term (`x + x^2` is `x`). Precompute transformed predictors, such as a squared predictor, as columns in `data`; inline calls such as `I()`, `poly()`, spline constructors, and user functions are not supported.

`scale` an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data` and follows the same explicit-column formula grammar as `mods`. Random-formula models also accept a named list targeting top-level random components or concrete random-effect blocks; see `brma.mv()`.

`random` Optional BayesTools random-effect formula or list of formulas. Omission or `NULL` means a fixed-effect model with no implicit heterogeneity term. Known sampling covariance `V` remains unchanged. Declare `random` before supplying heterogeneity priors or scale formulas.

`R` optional known random-effect group covariance/correlation matrix, or a named list of matrices. Matrix row and column names identify grouping levels. This is distinct from the known sampling covariance `V`.

`Rscale` scaling applied to each supplied `R` matrix before fitting. Supported values are `"cor"`, `"none"`, `"cor0"`, `"cov0"`, with metafor-compatible aliases `TRUE`, `FALSE`, `0`, `1`, `2`, and `3`. An unnamed scalar is recycled across all supplied matrices. A vector or list can specify one scaling value per named `R` entry, with exactly matching names. An unnamed one-element list is supported only for a single `R` matrix.

`data` an optional data frame containing the variables.

`slab` an optional vector of study labels.

`subset` an optional logical or numeric vector specifying a subset of data to be used. Subsetting precedes missing-row removal; retained-row outcome and predictor checks follow these steps. Type, shape, and existing extraction checks can still reject malformed inputs before subsetting. Row selection (by `subset` or missing-row removal) drops factor levels left without rows from `mods`, `scale`, and random-effect grouping factors. The index variable of a structured random-effect term (`cs()`, `hcs()`, `ar1()`, `har()` with one index variable) keeps its declared levels, which define the covariance structure; the variables of a multivariable `cs(a + b | group)` index follow the ordinary rule, because that index is built from their observed combinations. If an index variable that keeps its levels is also used in `mods` or `scale` and loses one of its levels to the selection, the fit stops with an error of class `RoBMA_structured_index_level_removed` (also `RoBMA_data_error`); remove the level from the data with `droplevels()` or use a separate column for the moderator.

`measure` a character string specifying the effect size measure. Normal/effect-size constructors require an explicit value and support `"SMD"`, `"ZCOR"`, `"RR"`, `"OR"`, `"HR"`, `"RD"`, `"IRR"`, and `"GEN"`. Use `"GEN"` only for general effect sizes without a known unit information standard deviation. GLMM raw-count constructors support only `"OR"` and `"IRR"` and default to `"OR"`.

`prior_effect` prior distribution for the effect size (μ) parameter (the intercept). If omitted and consumed, a default prior is constructed. In single-model functions, explicit `NULL` or `FALSE` sets a spike at zero.

prior_heterogeneity
: prior distribution for the heterogeneity (τ) parameter. If omitted and consumed, a default prior is constructed. In single-model functions, explicit NULL or FALSE sets a spike at zero. For brma.mv random-effect formulas, this can also be a partial or complete prior_random object; see random_effect_prior_specification.

prior_mods
: prior distribution for the moderators (β) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or NULL, default priors are used. This argument can be specified only when mods is specified. A formula without an intercept (~ 0 + x) fixes the location intercept at zero, so an intercept entry of the list is accepted only as a point prior at zero; any other intercept prior stops with an error of class RoBMA_prior_intercept_conflict (also of class RoBMA_prior_error). A named intercept entry supplies the complete intercept prior. Explicit competing global effect declarations are refused when this would discard them. Named NULL or FALSE intercept entries are invalid; specify the complete prior in one place.

prior_scale
: prior distribution for the scale (δ) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or NULL, default priors are used. This argument can be specified only when scale is specified. A named intercept entry supplies the complete positive SD prior used by log(intercept). Global heterogeneity declarations remain meaningful when they supply another random root or its model-inclusion weights; declarations fully replaced without a consumer raise RoBMA_prior_intercept_conflict.

standardize_continuous_predictors
: logical. Whether to standardize continuous predictors. Defaults to TRUE.

set_contrast_factor_predictors
: character. How to set contrast for factor predictors. Defaults are constructor-specific and shown in each function usage. Single-model constructors use "treatment"; model-averaging constructors use "meandif". Accepted values are "treatment", "meandif", "orthonormal", and "independent". Independent coding creates one coefficient per factor level. Random-coefficient blocks reuse the resolved fixed-factor contrast unless random_block(contrasts = ...) overrides it. A warning is issued when default or inherited independent contrasts make a fixed or random design overspecified. With "meandif" or "orthonormal" contrasts, a formula term that interacts a factor with a predictor whose own term is missing, such as g:x in ~ g + g:x or ~ g / x, stops the fit: the term codes g by level indicators, with one coefficient per level, on which mean-difference and orthonormal priors are not defined. Specify an independent prior for that term, e.g., prior_mods = list("g:x" = prior_factor("normal", list(0, 1), contrast = "independent")) (or prior_scale for a scale formula), include the missing term (~ g * x), or use set_contrast_factor_predictors = "treatment".

prior_unit_information_sd
: numeric. The unit information standard deviation (σ_{unit}). Cannot be used together with prior_informed_field.

rescale_priors
: numeric. A scaling factor for supported prior distributions. Point and none priors are unchanged; supplied prior_random() scale architectures and publication-bias priors are not rescaled. The default PEESE prior separately uses its documented UISD adjustment, independently of rescale_priors. Defaults to 1.

prior_informed_field
: character. The field of the informed prior distributions. Omit to use the standard default prior specification; explicit NULL is invalid.

prior_informed_subfield
: character. The subfield of the informed prior distributions. Can only be specified together with prior_informed_field. Omit to use the field-specific default, such as "Cochrane" for prior_informed_field = "medicine"; explicit NULL is invalid.

known_v_parameterization
: known-V backend used by brma.mv(). "auto" chooses an exact backend when feasible; "latent" uses a latent D + BB' decomposition; "whitened" uses an eigen-rotated normal likelihood; "block_mvn" uses an exact native block multivariate-normal likelihood.

sample
: numeric. Number of MCMC samples to save. Defaults to 5000.

burnin
: numeric. Number of burn-in iterations. Defaults to 2000.

adapt
: numeric. Number of adaptation iterations. Defaults to 500.

chains
: numeric. Number of MCMC chains. Defaults to 3.

thin
: numeric. Thinning interval. Defaults to 1.

parallel
: logical. Whether to run MCMC chains in parallel. Defaults to FALSE.

autofit
: logical. Whether to automatically extend the MCMC chains if convergence is not met. Defaults to FALSE.

autofit_control
: list of autofit control settings. See set_autofit_control() for details.

convergence_checks
: list of convergence check settings. See set_convergence_checks() for details.

seed
: numeric. Random seed for reproducibility. Defaults to NULL.

silent
: logical. Whether to suppress output. Constructors with no explicit default use RoBMA.get_option("silent") when silent is omitted. Model-averaging wrappers default to TRUE unless explicitly changed.

...
: additional advanced arguments. Fitting functions reject unused arguments; currently recognized internal arguments include only_data, only_priors, is_JASP, and is_JASP_prefix.

vi
: a vector of sampling variances. Either vi or sei must be supplied for normal models.

sei
: a vector of standard errors. Either vi or sei must be supplied for normal models.

Details

brma.mv() represents known sampling dependence through a latent D + BB' decomposition of V (known_v_parameterization = "latent"), through an eigen-rotated likelihood (known_v_parameterization = "whitened"), or through an exact native block multivariate-normal likelihood (known_v_parameterization = "block_mvn"). The default known_v_parameterization = "auto" chooses the whitened backend whenever the extra diagonal variance is scalar and no scale regression is present, otherwise chooses block-MVN when the largest dense covariance block is at or below the configured feasibility threshold, and falls back to the latent backend for large blocks. The automatic block-MVN

threshold defaults to 128 rows per covariance block and can be changed with `options(RoBMA.known_v_block_mvn_max_blo = value)`, where `value` is a positive integer or `Inf`. Correlated `V` with scale regression cannot use `"whitened"`; request `"block_mvn"` explicitly if an automatic latent decomposition is unavailable for a nearly singular block. Covariance inputs retain their symmetric representation. Numerical positive-semidefinite checks use standardized correlations so small marginal variances are not mistaken for null directions. At a numerically singular boundary, the computational factor uses the existing eigensolver roundoff tolerance; sampling, whitening, and conditional calculations share its support. This numerical rule is separate from the selection model's condition/integrate source choices. Exact rank-one dependency blocks are represented by one latent sampling factor, including when another backend was requested, so correlations of one and any separate positive diagonal variance retain their mathematical structure. Singular positive-semidefinite `V` matrices are supported only when integrated conditional variance is structurally positive on every retained null direction. This can come from strictly positive heterogeneity or a supported marginalized random-effect variance. The check is performed after formulas and priors are compiled. Sampled random effects change the conditional mean and therefore do not regularize the likelihood covariance. For the block-MVN backend, a fixed point regularizer must also be large enough to make each block numerically positive definite at double precision; ineffective fixed values fail during preflight rather than inside JAGS. The latent known-`V` backend is unavailable for other singular `V` matrices; use `"auto"`, `"whitened"`, or `"block_mvn"`. Unsupported configurations fail before fitting rather than invoking a degenerate-normal likelihood. As a convenience for diagonal known-`V` inputs, `brma.mv()` also accepts named `vi` or `sei` arguments when `V` is omitted; these are converted to `diag(vi)` or `diag(sei^2)`. Do not supply `V` together with `vi` or `sei`.

Omitted or NULL `random` declares no random-effects structure. No implicit `tau` or heterogeneity mixture is added. Do not supply `prior_heterogeneity` or `prior_heterogeneity_null` without `random`, even as NULL or a zero prior; omit these irrelevant arguments instead. Scale formulas also require an explicit random structure. This rule applies to all .mv constructors; specialized univariate models retain their usual implicit heterogeneity and clustered `tau`/`rho`/`I2` behavior.

The `random` argument uses BayesTools random-effect formula structure tags. Plain `(expr | group)` syntax creates an unstructured random-coefficient block, and `expr || group` creates a diagonal block. Therefore `~ 1 | study`, `~ 1 + x | study`, and nested forms such as `~ 1 | study/effect` are supported. The explicit `us()` / `un()`, `diag()`, and `id()` tags use the same coefficient-formula semantics. The structured `cs()` / `hcs()`, `ar1()` / `ar()` / `har()`, and `car()` tags instead own their index basis; see the linked tag documentation before specifying factor slopes or index variables. BayesTools stores fully explicit canonical random-effect names of the form `(formula) owner: quantity(arguments)`. RoBMA uses their simplified display aliases through RoBMA's meta-analytic naming map consistently in summaries, plotting, density estimation, hypotheses, and posterior draws. A sole random intercept drops its redundant argument, so a bare block is displayed and accepted as `tau`, while an explicitly named block is `study: tau`; the unique shorthand `tau` is also accepted. A non-intercept coefficient remains explicit, for example `study: tau(x)`. Bare formulas and unnamed one-entry lists omit a redundant owner; explicitly named one-entry lists retain it, and missing names in longer lists become `component 1`, `component 2`, and so on. Allocation names include `tau_total`, `tau_common`, and `tau2_prop(study)`. BayesTools retains its general `sd`/`var`/`cor` vocabulary internally; none of these public RoBMA names expose backend coordinates. The optional `R` argument supplies known covariance or correlation matrices across random-effect grouping levels, following `metafor::rma.mv()` naming. `R` is separate from the known sampling covariance `V`: `V` describes sampling-error dependence between estimates, while `R` describes the latent group-axis covariance of random intercepts. Current `R` support is limited to random-intercept blocks. `Rscale` controls how each supplied `R` matrix is scaled before fitting and accepts `"cor"`, `"none"`, `"cor0"`, `"cov0"`,

plus metafor-compatible aliases TRUE/1 for "cor", FALSE/0 for "none", 2 for "cor0", and 3 for "cov0". In non-selection Gaussian models with random, shared sampled random-effect blocks enter the conditional mean through BayesTools. A supported one-to-one random-intercept block is automatically integrated into the likelihood variance. Other terms remain sampled with their full covariance. With a single top-level random component, scale models the row-wise total random-effect SD consumed by that component. A named list of scale formulas can instead target top-level random components or concrete random-effect blocks. Block names and grouping labels are accepted, and the terminal grouping variable is an alias for an unambiguous nested block; for example, scale = list(esid = ~ x) targets the esid block in random = ~ 1 | study / esid. Untargeted blocks retain their ordinary SD priors. Component names can be ordinary text; RoBMA normalizes them internally for JAGS. A single scale formula is rejected for multiple top-level random components because the target is ambiguous. In model- averaged models, block-specific scales inside one grouped component require the blocks to be supplied as separately named random entries so their inclusion gates remain well-defined. When scale is a named list, prior_scale must either be omitted or use the same named-list shape with one prior list per scale component, using the same user-facing names. Post-fit scale predictions for random-formula models keep these component scales separate rather than collapsing them into one total τ. In non-selection Gaussian models, a supported random-intercept block that maps one-to-one to estimates is automatically compiled as a marginalized variance component instead of sampled latent coefficients. This preserves estimate-level heterogeneity semantics while keeping shared higher-level random effects conditional. For known R, this automatic marginalization is available only when BayesTools validates a one-column random-intercept block whose Z K Z' row-space contribution is diagonal and one-to-one with the fitted rows; RoBMA then adds the BayesTools-prepared row multiplier to the known-V extra variance. Other known-R blocks remain sampled or are rejected by BayesTools validation. Slopes and other unsupported variance forms remain sampled. Multiple declared terms with one grouping level per estimate are rejected, regardless of their coefficient basis or known group covariance.

The implementation intentionally omits the weights and cluster arguments from metafor::rma.mv(). Use random for multilevel structures. Estimate-unit log_lik(), add_loo(), add_waic(), residuals(), and rstudent() are available for known-V models. Bridge-sampling marginal likelihoods via add_marglik() are available for fitted single-model known-V objects. rstandard() is available for known-V models at supported estimate and marginal conditioning depths.

Value

A fitted object of class c("brma.mv", "brma.norm", "brma"). The advanced internal only_data = TRUE path returns checked data before prior validation and random-effect marginalization decisions; use only_priors = TRUE when inspecting the compiled random-effect structure.

Diagnostic target semantics

The main brma.mv() post-fit methods intentionally use different targets:

- log_lik(unit = "estimate"), add_loo()/loo(), add_waic()/waic(), and rstudent() use estimate-unit log-score targets. When known V induces dependence, these contributions are Schur-complement conditionals $p(y_i \mid y_{-i}, \theta)$ within dependency blocks. They are existing-estimate diagnostics, not independent new-estimate prediction targets. Every local Gaussian random effect is integrated, so the target is the same whether the fitting parameterization sampled or marginalized the effect. When known random-effect covariance R is supplied, the

known R matrix and the fitted random-effect covariance enter the marginal ZGZ' covariance of the Schur conditional through BayesTools' compiled random-effect metadata.

- dfbetas() and covratio() use estimate-unit PSIS weights. With correlated known V, their interpretation is parameter influence under conditional estimate deletion, not independent-study deletion.
- add_marglik()/bridge_sampler() uses the full fitted likelihood corresponding to the selected known-V backend. It does not use the estimate-wise conditional target from LOO/WAIC. During bridge evaluation, sampled Gaussian location random effects are integrated exactly as ZGZ' while retaining every covariance parameter and prior; fitted diagonal marginalized blocks remain in the row variance. Selection likelihoods instead preserve the normalization boundary declared by selection_model(): retained contextual sources remain outside the normalizer and integrated sources enter the conditional covariance.
- hatvalues(), marginal rstandard(), and vif() use a marginal GLS covariance target based on V + ZGZ', where formula random effects are marginalized through the BayesTools covariance metadata.
- dffits() and cooks.distance() use fixed-location fitted-value influence targets. Cluster/block LOO is deferred for brma.mv() because its deletion target is a joint dependency block, not an independent row.

Without random, the model estimates no heterogeneity. Explicit random-formula models report component-specific standard deviation, correlation and allocation summaries. A single homogeneous random component uses the RoBMA names tau and tau2. summary_heterogeneity(component = ...) describes component heterogeneity over the observed design, whereas pooled_heterogeneity(component = ...) evaluates absolute component heterogeneity at the average expanded random design. Both report the fitted heterogeneity scale without fixed known-R group covariance multipliers; row-specific scale prediction and covariance consumers retain the full known R. General known-V models do not report scalar I2/H2.

See Also

random_effect_formula_tags, random_effect_prior_specification, brma(), BMA.mv(), RoBMA.mv(), bPET.mv(), bPEESE.mv(), bselmodel.mv(), summary.brma(), predict.brma()

Examples

```
## Not run:
V <- matrix(c(0.04, 0.01, 0.01, 0.09), 2, 2)
fit <- brma.mv(
  yi                        = c(0.10, 0.20),
  V                         = V,
  measure                   = "GEN",
  prior_unit_information_sd = 1,
  seed                      = 1,
  silent                    = TRUE
)
summary(fit)

## End(Not run)
```

`bselmodel` *Bayesian Selection Model*

Description

Function for fitting random-effects, meta-regression, multilevel, and location-scale meta-analytic selection models.

Usage

```
bselmodel(
  yi,
  vi,
  sei,
  weights,
  ni,
  mods,
  scale,
  cluster,
  data,
  slab,
  subset,
  measure,
  prior_effect,
  prior_heterogeneity,
  prior_mods,
  prior_scale,
  prior_heterogeneity_allocation,
  prior_bias,
  standardize_continuous_predictors = TRUE,
  set_contrast_factor_predictors = "treatment",
  prior_unit_information_sd,
  rescale_priors = 1,
  prior_informed_field,
  prior_informed_subfield,
  effect_direction = "detect",
  steps,
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  chains = 3,
  thin = 1,
  parallel = FALSE,
  autofit = FALSE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  seed = NULL,
```

```
  silent,
  selection = BayesTools::selection_model(),
  selection_control = set_selection_likelihood_control(),
  ...
)
```

Arguments

yi	a vector of effect sizes, or a formula with the effect size on the left-hand side and location moderators on the right-hand side (for example `yi ~ x1 + x2`). If a formula is supplied, `mods` must not be specified.
vi	a vector of sampling variances. Either `vi` or `sei` must be supplied for normal models.
sei	a vector of standard errors. Either `vi` or `sei` must be supplied for normal models.
weights	an optional vector of finite positive likelihood weights. For normal/effect-size models, each weight powers the estimate likelihood. For constructors with GLMM raw-count input, each weight powers the paired two-arm likelihood for one study. An integer weight w repeats that conditional likelihood contribution w times, with the declared latent-effect and grouping structure held fixed. Positive fractional likelihood powers are also supported.
ni	an optional vector of sample sizes. Used for `measure = "GEN"` or when estimating the unit information standard deviation.
mods	an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data` and supports explicit columns and standard formula operators, including `^` as R's crossing operator with a literal whole exponent of at least 2: `(x + z)^2` expands to `x + z + x:z`, and `x^2` is not a quadratic term (`x + x^2` is `x`). Precompute transformed predictors, such as a squared predictor, as columns in `data`; inline calls such as `I()`, `poly()`, spline constructors, and user functions are not supported.
scale	an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data` and follows the same explicit-column formula grammar as `mods`. Random-formula models also accept a named list targeting top-level random components or concrete random-effect blocks; see `brma.mv()`.
cluster	an optional vector of cluster identifiers for multilevel meta-analysis.
data	an optional data frame containing the variables.
slab	an optional vector of study labels.
subset	an optional logical or numeric vector specifying a subset of data to be used. Subsetting precedes missing-row removal; retained-row outcome and predictor checks follow these steps. Type, shape, and existing extraction checks can still reject malformed inputs before subsetting. Row selection (by `subset` or missing-row removal) drops factor levels left without rows from `mods`, `scale`, and random-effect grouping factors. The index variable of a structured random-effect term (`cs()`, `hcs()`, `ar1()`, `har()` with one index variable) keeps its declared levels, which define the covariance structure; the variables of a multivariable `cs(a + b | group)` index follow the ordinary rule, because that index is

built from their observed combinations. If an index variable that keeps its levels is also used in mods or scale and loses one of its levels to the selection, the fit stops with an error of class RoBMA_structured_index_level_removed (also RoBMA_data_error); remove the level from the data with droplevels() or use a separate column for the moderator.

measure — a character string specifying the effect size measure. Normal/effect-size constructors require an explicit value and support "SMD", "ZCOR", "RR", "OR", "HR", "RD", "IRR", and "GEN". Use "GEN" only for general effect sizes without a known unit information standard deviation. GLMM raw-count constructors support only "OR" and "IRR" and default to "OR".

prior_effect — prior distribution for the effect size (μ) parameter (the intercept). If omitted and consumed, a default prior is constructed. In single-model functions, explicit NULL or FALSE sets a spike at zero.

prior_heterogeneity — prior distribution for the heterogeneity (τ) parameter. If omitted and consumed, a default prior is constructed. In single-model functions, explicit NULL or FALSE sets a spike at zero. For brma.mv random-effect formulas, this can also be a partial or complete prior_random object; see random_effect_prior_specification.

prior_mods — prior distribution for the moderators (β) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or NULL, default priors are used. This argument can be specified only when mods is specified. A formula without an intercept (~ 0 + x) fixes the location intercept at zero, so an intercept entry of the list is accepted only as a point prior at zero; any other intercept prior stops with an error of class RoBMA_prior_intercept_conflict (also of class RoBMA_prior_error). A named intercept entry supplies the complete intercept prior. Explicit competing global effect declarations are refused when this would discard them. Named NULL or FALSE intercept entries are invalid; specify the complete prior in one place.

prior_scale — prior distribution for the scale (δ) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or NULL, default priors are used. This argument can be specified only when scale is specified. A named intercept entry supplies the complete positive SD prior used by log(intercept). Global heterogeneity declarations remain meaningful when they supply another random root or its model-inclusion weights; declarations fully replaced without a consumer raise RoBMA_prior_intercept_conflict.

prior_heterogeneity_allocation — prior distribution for the fraction of heterogeneity allocated to the cluster-level component in multilevel models (ρ, a variance fraction rather than a random-coefficient correlation). If omitted or NULL, defaults to Beta(1, 1). This argument can be specified only when cluster is specified.

prior_bias — selection-model bias prior, usually created by prior_weightfunction(). If omitted or NULL, a default one-sided weightfunction prior is constructed from steps.

standardize_continuous_predictors — logical. Whether to standardize continuous predictors. Defaults to TRUE.

set_contrast_factor_predictors
: character. How to set contrast for factor predictors. Defaults are constructor-specific and shown in each function usage. Single-model constructors use "treatment"; model-averaging constructors use "meandif". Accepted values are "treatment", "meandif", "orthonormal", and "independent". Independent coding creates one coefficient per factor level. Random-coefficient blocks reuse the resolved fixed-factor contrast unless random_block(contrasts = ...) overrides it. A warning is issued when default or inherited independent contrasts make a fixed or random design overspecified. With "meandif" or "orthonormal" contrasts, a formula term that interacts a factor with a predictor whose own term is missing, such as g:x in ~ g + g:x or ~ g / x, stops the fit: the term codes g by level indicators, with one coefficient per level, on which mean-difference and orthonormal priors are not defined. Specify an independent prior for that term, e.g., prior_mods = list("g:x" = prior_factor("normal", list(0, 1), contrast = "independent")) (or prior_scale for a scale formula), include the missing term (~ g * x), or use set_contrast_factor_predictors = "treatment".

prior_unit_information_sd
: numeric. The unit information standard deviation (σ_{unit}). Cannot be used together with prior_informed_field.

rescale_priors
: numeric. A scaling factor for supported prior distributions. Point and none priors are unchanged; supplied prior_random() scale architectures and publication-bias priors are not rescaled. The default PEESE prior separately uses its documented UISD adjustment, independently of rescale_priors. Defaults to 1.

prior_informed_field
: character. The field of the informed prior distributions. Omit to use the standard default prior specification; explicit NULL is invalid.

prior_informed_subfield
: character. The subfield of the informed prior distributions. Can only be specified together with prior_informed_field. Omit to use the field-specific default, such as "Cochrane" for prior_informed_field = "medicine"; explicit NULL is invalid.

effect_direction
: direction used by publication-bias adjustments. "positive" assumes statistically significant positive estimates are more likely to be selected; "negative" mirrors the selection direction; "detect" infers the direction from the fitted data.

steps
: numeric vector of one-sided p-value cut points for the default selection model. If prior_bias is supplied, the prior carries its own side, steps, and weights. If omitted, the default is 0.025, yielding intervals [0, .025] and (.025, 1].

sample
: numeric. Number of MCMC samples to save. Defaults to 5000.

burnin
: numeric. Number of burn-in iterations. Defaults to 2000.

adapt
: numeric. Number of adaptation iterations. Defaults to 500.

chains
: numeric. Number of MCMC chains. Defaults to 3.

thin
: numeric. Thinning interval. Defaults to 1.

parallel
: logical. Whether to run MCMC chains in parallel. Defaults to FALSE.

`autofit` logical. Whether to automatically extend the MCMC chains if convergence is not met. Defaults to `FALSE`.

`autofit_control` list of autofit control settings. See `set_autofit_control()` for details.

`convergence_checks` list of convergence check settings. See `set_convergence_checks()` for details.

`seed` numeric. Random seed for reproducibility. Defaults to `NULL`.

`silent` logical. Whether to suppress output. Constructors with no explicit default use `RoBMA.get_option("silent")` when `silent` is omitted. Model-averaging wrappers default to `TRUE` unless explicitly changed.

`selection` specification created by `selection_model()` for automatically constructed weightfunction priors. The default integrates estimate-level random effects and the complete sampling error, and conditions on other random effects. Each source can instead be conditioned upon or integrated through the corresponding selection-model setting. The default `weight_rule = "product"` needs no publication groups and ignores `group`. Publication grouping is active only for `weight_rule = "best"`. Explicit `prior_bias` objects retain their own selection specification; this argument does not overwrite it. Conditioning on sampling variation retains the entire sampling-error realization, including in ordinary univariate models. This sampling setting requires omitted or unit observation `weights`.

`selection_control` numerical integration settings created by `set_selection_likelihood_control()`. These affect numerical evaluation, without changing the conditioning model or the vector weighting rule.

`...` additional advanced arguments. Fitting functions reject unused arguments; currently recognized internal arguments include `only_data`, `only_priors`, `is_JASP`, and `is_JASP_prefix`.

Details

`bselmodel()` is a normal/effect-size selection-model constructor. Custom `prior_bias` can be a weightfunction prior or a supported BayesTools selection-kernel prior; p-hacking kernels are not supported in active RoBMA. By default, the model conditions on contextual cluster effects and uses the product of estimate weights. For independent rows conditional on these effects, this gives the usual selected-normal likelihood with within-cluster heterogeneity integrated. Use `selection = selection_model(...)` to choose `weight_rule = "best"`, which uses the weight at the smallest p-value in each publication group. For `"best"`, `group` identifies a data column; when `group = NULL`, the specialized `cluster` argument supplies publication groups, or an unclustered model uses one group per estimate. Product models use neither input to define selection groups.

All conditioning choices use the same Gaussian source model. Conditioned sources remain latent; their population distributions stay outside selection normalization. Integrated sources may be reweighted by selection. With all sources conditioned and positive weights, the weights cancel and the observed law is the ordinary Gaussian model. See `bselmodel.mv()` for the complete source and covariance contract.

Product normalizers factorize only for conditionally independent rows. Dependent events use supported covariance-factor quadrature or fixed randomized quasi-Monte Carlo integration with

explicit error diagnostics. Non-unit observation `weights` require `known_sampling_variance = "integrate"` and independent product factors. Unit weights are equivalent to omitting `weights`.

Value

A fitted object of class `c("bselmodel", "brma")` containing a single Bayesian selection model fit.

See Also

publication_bias_prior_specification, `RoBMA()`, `bPET()`, `bPEESE()`, `summary.brma()`, `funnel.brma()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")

  fit <- bselmodel(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    steps   = 0.025,
    seed    = 1,
    silent  = TRUE
  )

  summary(fit)
  funnel(fit)
}

## End(Not run)
```

`bselmodel.mv` *Bayesian Multivariate and Multilevel Selection Model*

Description

Fits estimate-level Bayesian selection models with known sampling covariance and BayesTools random-effect formulas.

Usage

```
bselmodel.mv(
  yi,
  V,
  ni,
  mods,
```

```
  scale,
  random,
  R = NULL,
  Rscale = "cor",
  data,
  slab,
  subset,
  measure,
  prior_effect,
  prior_heterogeneity,
  prior_mods,
  prior_scale,
  prior_bias,
  standardize_continuous_predictors = TRUE,
  set_contrast_factor_predictors = "treatment",
  prior_unit_information_sd,
  rescale_priors = 1,
  prior_informed_field,
  prior_informed_subfield,
  effect_direction = "detect",
  steps,
  selection = BayesTools::selection_model(),
  selection_control = set_selection_likelihood_control(),
  known_v_parameterization = "auto",
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  chains = 3,
  thin = 1,
  parallel = FALSE,
  autofit = FALSE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  seed = NULL,
  silent,
  ...,
  vi = NULL,
  sei = NULL
)
```

Arguments

yi	a vector of effect sizes, or a formula with the effect size on the left-hand side and location moderators on the right-hand side (for example `yi ~ x1 + x2`). If a formula is supplied, `mods` must not be specified.
V	a known working variance-covariance matrix, a list of block variance-covariance matrices, or an exact diagonal-plus-factor declaration created by `known_v_factor()`, used by `brma.mv()`. Ordinary covariance matrices from `metafor::vcalc()`

can be supplied directly. Product selection requires no publication groups. For best-p-value selection, declare groups separately with `selection_model(weight_rule = "best", group = ...)`. Matrix symmetry uses base R's numerical tolerance. Accepted matrices are stored symmetrically by averaging off-diagonal pairs; the supplied object and its diagonal are unchanged. Dependency blocks that are block constant on nested partitions are represented exactly as a diagonal-plus-factor structure without a `known_v_factor()` declaration. Dimensions and row alignment are checked on the complete input; when rows are dropped (`subset` or missing values), the numerical covariance checks apply to the retained rows only, so acceptance says nothing about the entries of the dropped rows.

`ni` an optional vector of sample sizes. Used for `measure = "GEN"` or when estimating the unit information standard deviation.

`mods` an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data` and supports explicit columns and standard formula operators, including `^` as R's crossing operator with a literal whole exponent of at least 2: `(x + z)^2` expands to `x + z + x:z`, and `x^2` is not a quadratic term (`x + x^2` is `x`). Precompute transformed predictors, such as a squared predictor, as columns in `data`; inline calls such as `I()`, `poly()`, spline constructors, and user functions are not supported.

`scale` an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data` and follows the same explicit-column formula grammar as `mods`. Random-formula models also accept a named list targeting top-level random components or concrete random-effect blocks; see `brma.mv()`.

`random` Optional BayesTools random-effect formula or list of formulas. Omission or `NULL` means a fixed-effect model with no implicit heterogeneity term. Known sampling covariance `V` remains unchanged. Declare `random` before supplying heterogeneity priors or scale formulas.

`R` optional known random-effect group covariance/correlation matrix, or a named list of matrices. Matrix row and column names identify grouping levels. This is distinct from the known sampling covariance `V`.

`Rscale` scaling applied to each supplied `R` matrix before fitting. Supported values are `"cor"`, `"none"`, `"cor0"`, `"cov0"`, with metafor-compatible aliases `TRUE`, `FALSE`, `0`, `1`, `2`, and `3`. An unnamed scalar is recycled across all supplied matrices. A vector or list can specify one scaling value per named `R` entry, with exactly matching names. An unnamed one-element list is supported only for a single `R` matrix.

`data` an optional data frame containing the variables.

`slab` an optional vector of study labels.

`subset` an optional logical or numeric vector specifying a subset of data to be used. Subsetting precedes missing-row removal; retained-row outcome and predictor checks follow these steps. Type, shape, and existing extraction checks can still reject malformed inputs before subsetting. Row selection (by `subset` or missing-row removal) drops factor levels left without rows from `mods`, `scale`, and random-effect grouping factors. The index variable of a structured random-effect term (`cs()`, `hcs()`, `ar1()`, `har()` with one index variable) keeps its de-

clared levels, which define the covariance structure; the variables of a multivariable cs(a + b | group) index follow the ordinary rule, because that index is built from their observed combinations. If an index variable that keeps its levels is also used in mods or scale and loses one of its levels to the selection, the fit stops with an error of class RoBMA_structured_index_level_removed (also RoBMA_data_error); remove the level from the data with droplevels() or use a separate column for the moderator.

measure — a character string specifying the effect size measure. Normal/effect-size constructors require an explicit value and support "SMD", "ZCOR", "RR", "OR", "HR", "RD", "IRR", and "GEN". Use "GEN" only for general effect sizes without a known unit information standard deviation. GLMM raw-count constructors support only "OR" and "IRR" and default to "OR".

prior_effect — prior distribution for the effect size (μ) parameter (the intercept). If omitted and consumed, a default prior is constructed. In single-model functions, explicit NULL or FALSE sets a spike at zero.

prior_heterogeneity — prior distribution for the heterogeneity (τ) parameter. If omitted and consumed, a default prior is constructed. In single-model functions, explicit NULL or FALSE sets a spike at zero. For brma.mv random-effect formulas, this can also be a partial or complete prior_random object; see random_effect_prior_specification.

prior_mods — prior distribution for the moderators (β) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or NULL, default priors are used. This argument can be specified only when mods is specified. A formula without an intercept (~ 0 + x) fixes the location intercept at zero, so an intercept entry of the list is accepted only as a point prior at zero; any other intercept prior stops with an error of class RoBMA_prior_intercept_conflict (also of class RoBMA_prior_error). A named intercept entry supplies the complete intercept prior. Explicit competing global effect declarations are refused when this would discard them. Named NULL or FALSE intercept entries are invalid; specify the complete prior in one place.

prior_scale — prior distribution for the scale (δ) parameters. A single prior applies to all terms; a named list can specify term-specific priors. If omitted or NULL, default priors are used. This argument can be specified only when scale is specified. A named intercept entry supplies the complete positive SD prior used by log(intercept). Global heterogeneity declarations remain meaningful when they supply another random root or its model-inclusion weights; declarations fully replaced without a consumer raise RoBMA_prior_intercept_conflict.

prior_bias — selection-model bias prior, usually created by prior_weightfunction(). If omitted or NULL, a default one-sided weightfunction prior is constructed from steps.

standardize_continuous_predictors — logical. Whether to standardize continuous predictors. Defaults to TRUE.

set_contrast_factor_predictors — character. How to set contrast for factor predictors. Defaults are constructor-specific and shown in each function usage. Single-model constructors use "treatment"; model-averaging constructors use "meandif". Accepted values are "treatment", "meandif", "orthonormal", and "independent". Independent coding creates

one coefficient per factor level. Random-coefficient blocks reuse the resolved fixed-factor contrast unless `random_block(contrasts = ...)` overrides it. A warning is issued when default or inherited independent contrasts make a fixed or random design overspecified. With `"meandif"` or `"orthonormal"` contrasts, a formula term that interacts a factor with a predictor whose own term is missing, such as `g:x` in `~ g + g:x` or `~ g / x`, stops the fit: the term codes `g` by level indicators, with one coefficient per level, on which mean-difference and orthonormal priors are not defined. Specify an independent prior for that term, e.g., `prior_mods = list("g:x" = prior_factor("normal", list(0, 1), contrast = "independent"))` (or `prior_scale` for a scale formula), include the missing term (`~ g * x`), or use `set_contrast_factor_predictors = "treatment"`.

`prior_unit_information_sd`
: numeric. The unit information standard deviation (σ_{unit}). Cannot be used together with `prior_informed_field`.

`rescale_priors`
: numeric. A scaling factor for supported prior distributions. Point and none priors are unchanged; supplied `prior_random()` scale architectures and publication-bias priors are not rescaled. The default PEESE prior separately uses its documented UISD adjustment, independently of `rescale_priors`. Defaults to 1.

`prior_informed_field`
: character. The field of the informed prior distributions. Omit to use the standard default prior specification; explicit `NULL` is invalid.

`prior_informed_subfield`
: character. The subfield of the informed prior distributions. Can only be specified together with `prior_informed_field`. Omit to use the field-specific default, such as `"Cochrane"` for `prior_informed_field = "medicine"`; explicit `NULL` is invalid.

`effect_direction`
: direction used by publication-bias adjustments. `"positive"` assumes statistically significant positive estimates are more likely to be selected; `"negative"` mirrors the selection direction; `"detect"` infers the direction from the fitted data.

`steps`
: numeric vector of one-sided p-value cut points for the default selection model. If `prior_bias` is supplied, the prior carries its own side, steps, and weights. If omitted, the default is `0.025`, yielding intervals `[0, .025]` and `(.025, 1]`.

`selection`
: specification created by `selection_model()` for automatically constructed weight-function priors. The default integrates estimate-level random effects and the complete sampling error, and conditions on other random effects. Each source can instead be conditioned upon or integrated through the corresponding selection-model setting. The default `weight_rule = "product"` needs no publication groups and ignores `group`. Publication grouping is active only for `weight_rule = "best"`. Explicit `prior_bias` objects retain their own selection specification; this argument does not overwrite it. Conditioning on sampling variation retains the entire sampling-error realization, including in ordinary univariate models. This sampling setting requires omitted or unit observation `weights`.

`selection_control`
: numerical integration settings created by `set_selection_likelihood_control()`. These affect numerical evaluation, without changing the conditioning model or the vector weighting rule.

known_v_parameterization
: known-V backend used by brma.mv(). "auto" chooses an exact backend when feasible; "latent" uses a latent D + BB' decomposition; "whitened" uses an eigen-rotated normal likelihood; "block_mvn" uses an exact native block multivariate-normal likelihood.

sample
: numeric. Number of MCMC samples to save. Defaults to 5000.

burnin
: numeric. Number of burn-in iterations. Defaults to 2000.

adapt
: numeric. Number of adaptation iterations. Defaults to 500.

chains
: numeric. Number of MCMC chains. Defaults to 3.

thin
: numeric. Thinning interval. Defaults to 1.

parallel
: logical. Whether to run MCMC chains in parallel. Defaults to FALSE.

autofit
: logical. Whether to automatically extend the MCMC chains if convergence is not met. Defaults to FALSE.

autofit_control
: list of autofit control settings. See set_autofit_control() for details.

convergence_checks
: list of convergence check settings. See set_convergence_checks() for details.

seed
: numeric. Random seed for reproducibility. Defaults to NULL.

silent
: logical. Whether to suppress output. Constructors with no explicit default use RoBMA.get_option("silent") when silent is omitted. Model-averaging wrappers default to TRUE unless explicitly changed.

...
: additional advanced arguments. Fitting functions reject unused arguments; currently recognized internal arguments include only_data, only_priors, is_JASP, and is_JASP_prefix.

vi
: a vector of sampling variances. Either vi or sei must be supplied for normal models.

sei
: a vector of standard errors. Either vi or sei must be supplied for normal models.

Details

Omitted or NULL random specifies a fixed-effect selection model with no implicit heterogeneity. Both random-effect selection axes are then inapplicable; the sampling-error selection axis still applies. Heterogeneity priors and scale formulas require an explicit random structure.

bselmodel.mv() combines the selection-model interface of bselmodel() with the known-covariance and random-formula model of brma.mv(). Each weightfunction carries a selection_model() specifying whether estimate-level random effects, other random effects, and the complete sampling error are conditioned upon or integrated before selection normalization. The default integrates estimate-level random effects and sampling error, conditions on other random effects, and uses the product of estimate weights. The default product rule needs no publication groups, so adding a three-level structure requires only random = ~ 1 | study / esid. weight_rule = "best" instead uses the weight at the smallest actual p-value in the publication group.

With one cutoff and all three sources integrated, "best" corresponds to the relaxed report-all rule of van Aert et al. (in press): a favorable result can select the publication vector, after which all measured outcomes are reported. The other conditioning cells and multiple-cutoff best rules are extensions of that specification.

Publication grouping is active only for `weight_rule = "best"`. For grouped multivariate models, specify it with `selection = selection_model(weight_rule = "best", group = study)`, or in the selection model of an explicit weightfunction prior. Independent models without a random structure can use singleton groups automatically. Covariance matrices, including standard `metafor::vcalc()` results, and random-effect grouping factors do not establish publication identities. With `"product"`, `group` is ignored: it is neither bound to a data column nor checked as a publication partition. Integration follows the dependencies in `V` and the integrated random effects, independently of publication grouping.

Sampling error is $e \sim N(0, V)$. With positive sampling standard errors it can equivalently be written as $e = Sz$, with $z \sim N(0, R)$ and $V = SRS$. Here S contains the original sampling standard errors and R preserves their complete correlation structure. Conditioning retains the entire realized vector e during selection normalization. No residual sampling variation remains inside that normalizer. Integration instead contributes the full V to candidate variation. These choices apply equally to univariate, diagonal, and correlated sampling covariance, including `vi` and `sei` inputs.

The covariance `V` remains authoritative. A diagonal-plus-factor input from `known_v_factor()` can support more efficient calculations, but equivalent representations define the same selection model with matching source settings and, for `"best"`, publication groups. Selection thresholds always use the original `sqrt(diag(V))`. The same faster routes are reached from an ordinary matrix whose dependency blocks are block constant on nested partitions, such as compound symmetry or nested `metafor::vcalc()` type/obs structures: that structure is recovered from the supplied entries and reproduces them to working precision. It changes how the same integral is evaluated, not which sampling sources are conditioned upon.

The likelihood and post-fit methods use the same resolved conditioning model. `known_v_parameterization` and numerical integration settings cannot change it. The `estimate_random_effects` setting controls estimate-level true-effect variation. Shared covariance-factor and dense numerical paths use explicit integration diagnostics.

The source settings define the reporting model. An optional comparison with integrating retained contexts can be requested with `selection_sensitivity_diagnostics()` or stored with `add_selection_sensitivity_di`. Fitting does not compute this separate comparison automatically.

`estimate_random_effects`, `other_random_effects`, and `known_sampling_variance` each accept `"condition"` or `"integrate"`. Both `"product"` and `"best"` support the resulting choices for supported Gaussian sources, subject to the dependency restrictions below.

Retained contexts remain unknown and are estimated. Their population mixing laws stay outside the selection normalizer; integrating a context instead allows selection to reweight its mixing law. An absent context makes its two settings coincide. BayesTools identifies an estimate-level random term by a distinct grouping level for every retained estimate. At most one such term is allowed. Every other declared term is an other-level source, including terms with independent coefficient supports inside repeated groups. These roles do not split coefficient families or discard known group covariance; each choice applies to the complete declared random term.

With every outcome-generating source conditioned upon, candidate outcomes are deterministic within a retained context. If weights are positive almost surely, they cancel under conditional normalization: the observed law is the ordinary Gaussian model and carries no information about the selection weights. A zero acceptance probability on a positive-probability set of retained contexts makes this selection process invalid. It must not be handled by silently discarding those contexts. Ordinary population-level publication selection instead integrates all outcome-generating sources.

Product normalization retains all supported Gaussian dependencies. Best weights require integrated

dependencies to remain within each publication group; an unsupported source is identified before fitting. Conditioned sources may connect groups. Integrated candidate covariance can be singular while the observed law remains nondegenerate after averaging over retained sources. No residual noise is added to change this boundary. Ensembles require a common conditioning cell across active selection branches and a common publication partition across their `"best"` branches only. Product branches impose no grouping requirements; an ensemble without `"best"` accepts the default `group = NULL`. Bins, weight priors, and supported weight rules may differ across branches. Like the other `brma.mv()`-family constructors, `bselmodel.mv()` does not accept observation `weights`.

Joint likelihoods, single-model bridge evidence, conditional-density estimation, selected prediction, latent summaries, and z-plots use this resolved model. Estimate-unit LOO, WAIC, and LOO-PIT integrate deleted outcomes within the original publication event. Their conditional row scores must not be summed and interpreted as a joint group likelihood. Cluster or block deletion remains unavailable for arbitrary `brma.mv()` random formulas. Product-space ensembles do not provide bridge marginal likelihoods. With conditioned sampling, estimate deletion integrates the deleted sampling error conditional on the retained sampling errors of the remaining estimates.

Marginal selected predictions draw new retained contexts from their original mixing laws before drawing selected outcomes and reconstructing latent true effects. At estimate depth, latent predictions draw fitted true effects; `blup()` and `fitted.brma()` instead return conditional means. Response prediction at estimate depth describes a new reporting event around fitted truth, with new sampling error. Prediction's `conditioning_depth` is a separate choice from the selection-model source settings. Cluster depth is available only for the specialized `cluster` interface. Known-covariance latent and response predictions with `newdata` require `V_new` and, for `"best"`, the resolved publication-group reference; cross-covariance with fitted outcomes and non-marginal `newdata` are unavailable. See `predict.brma()` for prediction targets and labels.

When the complete sampling error is conditioned upon, total fitted truth equals the observed estimate minus its fitted sampling error. Conditional means and estimate-depth latent draws then coincide within each posterior draw; uncertainty remains across posterior draws. Blockwise `ranef()` retains its conditional-mean target for each random-effect block.

Partial-vector calculations retain the full selection event. Densities need a nondegenerate observed Gaussian law, except for supported one-coordinate projections of explicitly declared rank-one sources. Multicoordinate singular Lebesgue densities and general singular conditional latent posteriors are unavailable; zero integrated true-effect covariance gives deterministic latent effects. No general public multivariate CDF is provided. LOO-PIT and z-plots use their respective deletion and marginal projection targets.

Value

A fitted object of class `c("bselmodel.mv", "bselmodel", "brma.mv", "brma.norm", "brma")`.

References

van Aert RCM, Riley R, Jackson D (in press). "Correcting for publication bias in multivariate and multilevel meta-analysis: A multivariate step function selection model approach." *Psychological Methods*. doi:10.31234/osf.io/rk4mn_v1.

See Also

`bselmodel()`, `brma.mv()`, `set_selection_likelihood_control()`, `summary.brma()`

Examples

```
## Not run:
dat <- data.frame(yi = c(0.10, 0.20))
V <- matrix(c(0.04, 0.01, 0.01, 0.09), 2, 2)
fit <- bselmodel.mv(
  yi      = yi,
  V       = V,
  data    = dat,
  measure = "GEN",
  seed    = 1,
  silent  = TRUE
)
summary(fit)

## End(Not run)
```

`check_loo.brma` *Check LOO Diagnostics for brma Objects*

Description

Check Pareto k diagnostics for a brma model object and warn if any values are high.

Usage

```
## S3 method for class 'brma'
check_loo(object, unit = "estimate", ...)
```

Arguments

`object`	a brma model object.
`unit`	output/deletion unit. See `add_loo`.
`...`	currently unused.

Details

LOO must first be computed with `object <- add_loo(object, unit = unit)`. With S posterior draws in the stored LOO result, the method warns when any Pareto k diagnostic is greater than $\min(1 - 1/\log_{10}(S), 0.7)$. The warning gives that cutoff and the affected target indices.

Value

NULL (throws warning if diagnostics are unreliable).

coef.brma *Extract Model Coefficients for brma Objects*

Description

Extract model coefficients (posterior means) from a fitted brma object.

Usage

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

Arguments

object	a fitted brma object
...	additional arguments (currently ignored)

Value

A named numeric vector of posterior mean coefficients.

See Also

summary.brma()

contr.BayesTools *BayesTools Contrast Matrices*

Description

BayesTools provides several contrast matrix functions for Bayesian factor analysis. These functions create different types of contrast matrices that can be used with factor variables in Bayesian models.

Usage

```
contr.orthonormal(n, contrasts = TRUE)

contr.meandif(n, contrasts = TRUE)

contr.independent(n, contrasts = TRUE)
```

Arguments

n	a vector of levels for a factor, or the number of levels
contrasts	logical indicating whether contrasts should be computed

Details

The package includes the following contrast functions:

`contr.orthonormal` Return a matrix of orthonormal contrasts: a normalised Helmert basis, whose columns have unit length and sum to zero and which is identical on every platform. Any orthonormal basis of the centred space implies the same exchangeable prior on the factor level effects under a spherical prior on the coefficients, which corresponds to the description by Rouder et al. (2012). Returns a matrix with n rows and k columns, with k = n - 1 if `contrasts` = `TRUE` and k = n if `contrasts` = `FALSE`.

`contr.meandif` Return a matrix of mean difference contrasts. This is an adjustment to the `contr.orthonormal` that ascertains that the prior distributions on difference between the grand mean and factor level are identical independent of the number of factor levels (which does not hold for the orthonormal contrast). Furthermore, the contrast is re-scaled so the specified prior distribution exactly corresponds to the prior distribution on difference between each factor level and the grand mean – this is approximately twice the scale of `contr.orthonormal`. Returns a matrix with n rows and k columns, with k = n - 1 if `contrasts` = `TRUE` and k = n if `contrasts` = `FALSE`.

`contr.independent` Return a matrix of independent contrasts – a level for each term. Returns a matrix with n rows and k columns, with k = n if `contrasts` = `TRUE` and k = n if `contrasts` = `FALSE`.

References

Rouder JN, Morey RD, Speckman PL, Province JM (2012). "Default Bayes factors for ANOVA designs." *Journal of Mathematical Psychology*, **56**(5), 356–374. doi:10.1016/j.jmp.2012.08.001.

Examples

```
# Orthonormal contrasts
contr.orthonormal(c(1, 2))
contr.orthonormal(c(1, 2, 3))

# Mean difference contrasts
contr.meandif(c(1, 2))
contr.meandif(c(1, 2, 3))

# Independent contrasts
contr.independent(c(1, 2))
contr.independent(c(1, 2, 3))
```

---

cooks.distance.brma *Cook's Distance for brma Objects*

Description

Computes Cook's distance for a fitted brma object. Cook's distance measures the aggregate influence of each observation on the model coefficients.

Usage

```
## S3 method for class 'brma'
cooks.distance(model, ...)
```

Arguments

`model`	a fitted normal-outcome `brma` object without a weightfunction component.
`...`	additional arguments (currently ignored).

Details

Cook's distance is computed as a PSIS leave-one-out deletion diagnostic. For each observation i, normalized PSIS weights estimate the fitted values under the leave-one-out posterior. The distance is the squared posterior Mahalanobis distance between the full-data and leave-one-out fitted-value vectors:

$$D_i = \Delta_i' V_\mu^+ \Delta_i$$

where $\Delta_i = \hat{\mu} - \hat{\mu}_{(-i)}$, V_μ^+ is the generalized inverse of the full-posterior fitted-value covariance. This uses the unscaled meta-analytic convention of **metafor** and Viechtbauer and Cheung (2010), without division by the fixed-effect model rank. Values can differ from **metafor** because RoBMA uses posterior moments and PSIS deletion rather than classical refits. For `brma.mv()` known-`V` models, Cook's distance uses estimate-unit PSIS weights. With correlated known-`V`, deletion is conditional estimate deletion and the reported fitted-value target is the fixed-location mean $\mu = X\beta$; sampled or marginalized random effects are not included in the reported fitted value. Selection `brma.mv()` influence is currently unavailable. The PSIS fixed-location moments do not require the inverse marginal covariance used by `hatvalues()`.

Value

A numeric vector of Cook's distance values, one for each observation.

References

(Viechtbauer and Cheung 2010)

See Also

`influence.brma`, `dffits.brma`, `hatvalues.brma`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- bPET(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")
  fit <- add_loo(fit)

  cooks.distance(fit)
}
```

```
## End(Not run)
```

`covratio.brma` *COVRATIO for brma Objects*

Description

Computes COVRATIO for a fitted brma object. COVRATIO measures the change in the determinant of the covariance matrix of the estimates when observation i is removed.

Usage

```
## S3 method for class 'brma'
covratio(model, type = "mods", component = NULL, parameter = NULL, ...)
```

Arguments

`model`	a fitted brma object.
`type`	type of parameters to be summarized. Defaults to `"mods"` (for the effect size and meta-regression coefficients). Use `"scale"` for heterogeneity and scale-regression coefficients.
`component`	optional parameter namespace. Use `"random"` with one explicitly selected semantic random-effect quantity.
`parameter`	semantic random-effect quantity used when `component = "random"`.
`...`	additional arguments. The internal `.weights` argument can supply precomputed PSIS weights for callers that already extracted them.

Details

COVRATIO is computed using importance sampling weights to approximate the leave-one-out covariance matrices without refitting the model. Estimate-unit LOO must first be computed with `model <- add_loo(model, unit = "estimate")`, unless internal weights are supplied.

$$COVRATIO_i = \frac{\det(Cov(\beta)_{-i})}{\det(Cov(\beta))}$$

Values > 1 indicate that the observation improves precision (decreases variance), while values < 1 indicate that the observation decreases precision (increases variance). For `brma.mv()` objects, COVRATIO uses estimate-unit PSIS weights. With correlated known-`V` data, this is influence under conditional estimate deletion, not independent-study deletion. A requested random-effect correlation must be defined in every posterior draw. If either SD is zero in some draws, the complete influence request is unavailable; explicitly select an everywhere-defined quantity with `component = "random"` and `parameter` instead. Parameters with zero posterior variance are excluded from the covariance determinant and reported in a printed note when available. Ordered terms retain their fitted totals and omit one share per metadata-declared Dirichlet simplex. Aliases of a shared allocation source contribute its independent coordinates only once. If no estimable parameters remain, or if a full or LOO covariance determinant is zero or non-finite after exclusion, COVRATIO is undefined and values are reported as `NaN`.

Value

A named numeric vector of COVRATIO values, one for each observation.

See Also

influence.brma, dffits.brma, cooks.distance.brma

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- bPET(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")
  fit <- add_loo(fit)

  covratio(fit)
}

## End(Not run)
```

---

density_diagnostics *Extract qCMDE/IWMDE Density Diagnostics*

Description

Extracts the compact reliability diagnostics attached to a qCMDE/IWMDE hypothesis result. The diagnostics describe the posterior ordinate computation. They do not include uncertainty in the prior ordinate or, for IWMDE, uncertainty from estimating its conditional weight function.

Usage

```
density_diagnostics(object, ...)
```

Arguments

object	an object returned by hypothesis() with density_method = "qCMDE" or density_method = "IWMDE", or an RoBMA_density_ordinate_error caught from a failed or rejected point-ordinate computation.
...	unused.

Details

Point ordinates evaluate one fixed posterior-row sample whose size is chosen by samples before any ordinate contributions are computed. Precision, effective-sample-size, and contribution-concentration diagnostics never determine whether that finite estimate is returned. An unmet target_relative_mcse produces a warning; increase samples or set it to Inf to use every eligible row. The table records the requested and achieved sample sizes, all_rows_used, whether the precision target was met,

and whether the ordinate passed the independent numerical-stability gates. Finite row budgets use a reproducible simple random sample without replacement. For a purely continuous ordinate, `relative_mcse` and `ess` describe the selected continuous posterior-row sequence. For a mixed point/continuous ordinate evaluated on every active continuous row, they use the full conditioned-chain sequence, with zero contributions for inactive rows, and therefore include active-mass uncertainty and its covariance with continuous contributions. With a subset of active rows, `active_branch_relative_mcse` describes the selected active-row contribution sequence and `active_mass_relative_mcse` comes from the complete binary indicator chain. `relative_mcse` is their conservative worst-correlation delta upper bound on the unconditional density scale. It is suitable as a descriptive precision diagnostic but does not certify the unobserved full-mixture covariance, so `bf_grade_met` remains false. `sampling_relative_mcse` separately reports the finite-population simple-random-sampling component conditional on the fitted rows (`sampling_uncertainty_type = "finite_population_srswor"`). The sampling design is exact, but the standard error is an estimate: the usual without-replacement formula applied to the sample variance of the selected contributions, and zero when every eligible row is used. It is not added to the selected-row sequence MCSE, which already describes the actual selected estimator sequence.

These fields concern the continuous density ordinate. At a location that is also a posterior atom, the atom probability remains a separate discrete mass and is never added to the density ordinate.

`ordinate` and `log_ordinate` describe the posterior density at the requested point, not the model's marginal likelihood. `log_ordinate` is retained only when actually computed; it is not reconstructed for older results. `numerical_status` distinguishes `finite`, `underflow`, `overflow`, `arithmetic_failure`, `invalid_density`, `zero_without_log_evidence`, `nonfinite`, and `not_computed`. Underflow/overflow requires finite log evidence and confirmation that exponentiating that log loses range. A zero without this evidence is not classified as a mathematical zero. A small density alone does not establish unreliability.

Failed requested points retain separate diagnostic records with a `failure_reason`, including failures occurring before a final ordinate was obtained. `status = "unavailable"` distinguishes these failures from a finite positive ordinate rejected by accuracy diagnostics. Ordinary-scale precision diagnostics lost to underflow/overflow are unavailable, not zero error or proof of convergence. Retaining a log estimate does not authorize a Bayes factor: the existing finite-positive ordinate requirement is unchanged. Legacy tables remain inspectable with unknown new fields set to `NA`. A hypothesis result without attached diagnostics, such as one whose point ordinates were estimated with `density_method = "KDE"` or one without a point hypothesis, stops with the class `RoBMA_density_diagnostics_unavailable`. Density ordinate, diagnostic-unavailable, plot, and construction (`RoBMA_density_construction_error`) conditions share the `RoBMA_density_error` parent while retaining their specific condition classes.

The reliability policy warns when relative MCSE is at least 5 percent, ESS is below 100, the largest contribution share is at least 20 percent, or fewer than 100 finite contributions remain. These same-sample diagnostics do not reject a finite fixed-design estimate. Their warning thresholds are returned as columns rather than being implicit. For a descriptive mixed density curve using the conservative partial-row bound, a finite ESS below the usual curve rejection threshold is retained as a warning when the quantitative mixture-scale MCSE checks pass.

qCMDE's full method diagnostics distinguish `pilot_normalization_integral` from `final_normalization_integral`; IWMDE uses `support_grid_normalization_integral`. This compact accessor reports the relative difference of the final integral from the active mass as `normalization_relative_error`. For qCMDE it is a bookkeeping check, not a discretization error: rows normalized on the lattice are normalized by the same trapezoid rule, on the same grid, that integrates them, so they sum

to the active mass up to roundoff by construction, and rows normalized in closed form contribute their analytic mass within the range. The discretization and truncation errors of the lattice normalizers are reported by `ordinate_relative_change`, `lattice_resolution_relative_error`, and `truncation_ordinate_bound`. The active policy check is identified by `stability_metric`: qCMDE checks its posterior ordinate error estimate (`ordinate_error_estimate`), whereas IWMDE checks normalization mass. The qCMDE estimate adds three parts. `ordinate_relative_change` is the ordinate's movement between the selected normalization grid and the coarser nested grid over the same range. It estimates the discretization error of the lattice normalizers; it is an estimate, not a bound, because a row narrower than the lattice step can escape both grids alike. Rows whose lattice normalization is under-resolved also contribute an ordinate-share weighted resolution estimate (`lattice_resolution_relative_error`), using the kernel width or nested-grid log curvature at each row's maximum. This component and their sum are estimates, not rigorous bounds: a sharp prior core can be narrower than its likelihood kernel. Rows whose Gaussian likelihood kernel meets a normal prior are normalized by the closed-form mass of their support when their evaluated values are those of that closed form: their normalizer is the same on every grid, so they add no discretization change, lattice-resolution component or truncation. `truncation_ordinate_bound` is $t/(1-t)$ for the largest fraction t of any row's conditional mass outside the normalization range (`normalization_truncation`): a row normalizer missing that fraction overstates the row's density by at most this relative amount. `normalization_truncation_status` states how t is known: `"exact"` for rows whose Gaussian likelihood kernel meets a normal prior, `"bound"` for a Gaussian kernel with another proper prior, and `"estimate"` (from the exponential envelope of the tails at the range ends) otherwise; it reports the weakest status among the rows. The range covers every row's central `normalization_prob` interval, so t is at most `1 - normalization_prob` (for estimated rows, as estimated once their extension reaches its target) and `truncation_ordinate_bound` is at most `(1 - normalization_prob) / normalization_prob`, about 0.1 percent at the default `0.999`. The extension of an estimated row stops at its limit (four initial range widths per side and 40 passes), at a value the joint density cannot evaluate, or for a tail that does not decay; the reported `normalization_truncation` and `truncation_ordinate_bound` can then exceed these values. Setting `normalization_prob` closer to 1 widens the range and lowers the truncation; increasing `normalization_points` refines the grid. The table also reports the stability and adaptive-quadrature warning/rejection thresholds.

Ordered effects use normalized conditional-source contributions for both requested methods. Regular exact offsets can use a positive total/share block. The complete numerical ratio error must be at most 0.025 for every evaluated row, separately from the MCMC precision diagnostics. The columns `internal_method`, `working_distribution`, and `normalization_basis` identify this computation while `estimator` retains the requested method. `ordered_error_status` preserves its estimated error status. Block numerator and denominator errors, maximum quadrature order/panel count, and actual native normalizer node count are reported in the `block_*` columns. Exact offsets classified as nonfinite or unsupported are excluded only from density curves; explicit point requests retain their structural or precision refusal. The ordered per-row limit of 0.025 is separate from the generic stability warning/rejection thresholds reported in this table. Source-normalization refusals have class `RoBMA_ordered_normalization_error` and retain `iwmde_ordered_normalization_error` as a compatibility ancestor. Their condition's `proof` field contains the numerical integration evidence. When only some ordered evaluations lack certified contribution variation, their MCMC error and ESS stay unavailable while other evaluations retain their metrics. A printed table footnote reports the `partial` uncertainty status and its reason; the original selected-row scope stays in its column. Such curves fail the MCMC availability gate, and an affected point cannot meet its precision target. If every evaluation is affected, the full unavailable scope is retained. A finite log MCSE

whose absolute value underflows remains a separate representation limitation.

Value

A data frame of class `RoBMA_density_diagnostics`, with one row per computed or failed requested point ordinate. Columns identify the estimator, parameter, requested and evaluated values, schema/source provenance, row counts, active mass, relative MCSE, ESS, largest contribution share, finite terms, normalization, truncation, and quadrature checks, the qCMDE pilot-gate columns (the nested normalization grids have no refinement sequence to stop, so `pilot_gate_stopped` is `FALSE` and `pilot_bulk_ess` is `NA`), fixed-sampling state, policy thresholds, weight fallbacks, status, and warnings.

The exact columns, in order, are `schema_version`, `algorithm_version`, `source_fingerprint`, `estimator`, `internal_method`, `working_distribution`, `normalization_basis`, `ordered_error_status`, `block_numerator_relative_error`, `block_denominator_relative_error`, `block_max_order`, `block_max_panels`, `block_native_nodes`, `density_method`, `parameter`, `level`, `requested_value`, `evaluation_value`, `requested_samples`, `achieved_row_budget`, `eligible_rows`, `evaluated_rows`, `finite_terms`, `active_mass`, `relative_mcse`, `active_branch_relative_mcse`, `active_mass_relative_mcse`, `mixture_mcse_type`, `sampling_relative_mcse`, `sampling_fraction`, `mcmc_uncertainty_scope`, `sampling_uncertainty_type`, `ess`, `max_weight_share`, `normalization_relative_error`, `stability_metric`, `stability_relative_error`, `ordinate_relative_change`, `lattice_resolution_relative_error`, `normalization_truncation`, `normalization_truncation_status`, `truncation_ordinate_bound`, `quadrature_relative_change`, `pilot_gate_stopped`, `pilot_bulk_ess`, `target_relative_mcse`, `stability_warning_threshold`, `stability_rejection_threshold`, `quadrature_warning_threshold`, `quadrature_rejection_threshold`, `warning_relative_mcse`, `warning_min_finite_terms`, `warning_min_ess`, `warning_max_weight_share`, `all_rows_used`, `target_met`, `precision_target_met`, `sampling_target_met`, `bf_grade_met`, `n_weight_fallbacks`, `weight_fallback_reasons`, `status`, `ordinate`, `log_ordinate`, `numerical_status`, `failure_reason`, and `warnings`.

See Also

`hypothesis()`

Examples

```
## Not run:
result <- hypothesis(
  fit,
  hypothesis = "mu != 0 vs mu = 0"
)
density_diagnostics(result)

failed <- tryCatch(
  hypothesis(
    fit,
    hypothesis     = "mu != 0 vs mu = 0",
    density_method = "IWMDE"
  ),
  RoBMA_density_ordinate_error = identity
)
density_diagnostics(failed)
```

```
## End(Not run)
```

`dfbetas.brma` *DFBETAS for brma Objects*

Description

Computes DFBETAS (Difference in BETAS, standardized) for a fitted brma object. DFBETAS measures the influence of each observation on the estimated model coefficients. Positive values indicate that deleting the observation yields a smaller estimate, negative values indicate that deleting the observation yields a larger estimate.

Usage

```
## S3 method for class 'brma'
dfbetas(
  model,
  type = "mods",
  standardized_coefficients = FALSE,
  transform_factors = TRUE,
  return_loo_estimates = FALSE,
  component = NULL,
  parameter = NULL,
  ...
)
```

Arguments

`model`	a fitted brma object.
`type`	type of parameters to be summarized. Defaults to `"mods"` (for the effect size and meta-regression coefficients). The other options are `"scale"` (for the heterogeneity and scale-regression coefficients) and `"bias"` (for omega, PET, and PEESE publication-bias parameters).
`standardized_coefficients`	whether to show standardized meta-regression and scale-regression coefficients. Defaults to `FALSE`. When set to `TRUE`, standardized coefficients are returned for the intercept and continuous predictors of the location (`type = "mods"`) and scale (`type = "scale"`) formulas. These coefficients correspond to the standardized scale on which prior distributions are specified by default (i.e., `standardize_continuous_predictc = TRUE`). Otherwise, the coefficients are returned on the original scale of the predictors.
`transform_factors`	whether to transform factors to their original names. Defaults to `TRUE`.

`return_loo_estimates`
: whether to return the leave-one-out coefficient estimates used to compute DFBETAS instead of standardized DFBETAS values. Defaults to `FALSE`.

`component`
: optional parameter namespace. Use `"random"` for semantic random-effect SD, correlation, and allocation quantities.

`parameter`
: optional semantic random-effect quantity. When omitted with `component = "random"`, all available random quantities are returned.

`...`
: additional arguments (currently ignored).

Details

This function computes DFBETAS values using the Leave-One-Out (LOO) approximation based on Pareto Smoothed Importance Sampling (PSIS) weights. Ideally, DFBETAS is defined as:

$$DFBETAS_{ij} = \frac{\hat{\beta}_j - \hat{\beta}_{j(-i)}}{SE(\hat{\beta}_{j(-i)})}$$

where $\hat{\beta}_j$ is the estimate of the j-th coefficient using the full data, $\hat{\beta}_{j(-i)}$ is the estimate when observation i is omitted, and $SE(\hat{\beta}_{j(-i)})$ is the standard error of the coefficient when observation i is omitted.

In the Bayesian context using LOO approximation:

- $\hat{\beta}_{j(-i)}$ is estimated as the importance sampling weighted mean of the posterior samples, using PSIS weights w_{is}.
- $SE(\hat{\beta}_{j(-i)})$ is estimated as the importance sampling weighted standard deviation of the posterior samples.

This approximation allows computing influence statistics without refitting the model K times, making it computationally efficient. For `brma.mv()` objects, DFBETAS use estimate-unit PSIS weights. With correlated known-`V` data, this is influence under conditional estimate deletion, not independent-study deletion. A requested random-effect correlation must be defined in every posterior draw. If either SD is zero in some draws, the complete influence request is unavailable; explicitly select an everywhere-defined quantity with `component = "random"` and `parameter` instead. For `type = "bias"`, fixed identification parameters (e.g., the reference $\omega = 1$ interval) can have zero LOO posterior standard deviation. These parameters are retained in the output, but their DFBETAS values are reported as `NaN` because the standardized diagnostic is undefined.

Note: This function requires that LOO-CV has been computed for the model using `add_loo`.

Value

If `return_loo_estimates = FALSE`, a data frame with K rows (observations) and P columns (parameters), containing DFBETAS values. If `return_loo_estimates = TRUE`, returns the corresponding leave-one-out coefficient estimates. Row names correspond to study labels (if available) or indices.

See Also

`add_loo`, `loo_weights.brma`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- bPET(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")
  fit <- add_loo(fit)

  inf <- dfbetas(fit)
  plot(inf[, 1], type = "h")
}

## End(Not run)
```

`dffits.brma` *DFFITS for brma Objects*

Description

Computes DFFITS (Difference in FITS, standardized) for a fitted brma object. DFFITS measures how much the fitted value for observation i changes if observation i is removed, standardized by the estimated standard error of the fit.

Usage

```
## S3 method for class 'brma'
dffits(model, ...)
```

Arguments

`model`	a fitted normal-outcome `brma` object without a weightfunction component.
`...`	additional arguments (currently ignored).

Details

DFFITS values are computed as a PSIS leave-one-out deletion diagnostic. For each observation i, the leave-one-out posterior mean fitted value at that observation is estimated with normalized PSIS weights and compared to the full-posterior fitted value:

$$DFFITS_i = \frac{\hat{\mu}_i - \hat{\mu}_{i(-i)}}{SD_{(-i)}(\mu_i)}$$

This targets deletion influence on fitted values directly. It does not use LOO-PIT residuals, which are predictive outlier diagnostics rather than fitted-value deletion diagnostics. For `brma.mv()` known-`V` models, DFFITS uses estimate-unit PSIS weights. With correlated known-`V`, deletion is conditional estimate deletion and the reported fitted-value target is the fixed-location mean $\mu = X\beta$; sampled or marginalized random effects are not included in the reported fitted value.

Estimate-unit LOO must first be computed with `model <- add_loo(model, unit = "estimate")`. If the leave-one-out posterior SD of a fitted value is zero, the corresponding DFFITS value is returned as `NA`, without a separate explanatory note. The diagnostic is available for normal outcomes without a weightfunction component; selection `brma.mv()` influence is currently unavailable. Fixed-location PSIS moments do not require the inverse marginal covariance used by `hatvalues()`.

Value

A named numeric vector of DFFITS values, one for each observation.

See Also

`influence.brma`, `cooks.distance.brma`, `hatvalues.brma`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- bPET(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")
  fit <- add_loo(fit)

  dffits(fit)
}

## End(Not run)
```

---

estimate_unit_information_sd

Estimate Unit Information Standard Deviation

Description

Estimates the unit information standard deviation (UISD) from sample sizes and standard errors. The UISD is used to scale weakly informative prior distributions for meta-analytic parameters.

Usage

```
estimate_unit_information_sd(sei, ni)
```

Arguments

`sei`	a non-empty complete numeric vector of finite strictly positive standard errors for each study.
`ni`	a complete numeric vector of finite strictly positive sample sizes for each study, with the same length as `sei`.

Details

The unit information standard deviation is computed following Equation 6 in Röver et al. (2021):

$$\text{UISD} = \sqrt{\frac{\sum n_i}{\sum \text{se}_i^{-2}}}$$

where n_i is the sample size and se_i is the standard error for each study.

This function is useful when you want to compute the UISD once and pass it to multiple analyses via the `prior_unit_information_sd` argument in `brma()` or related functions. This ensures consistent prior scaling across analyses performed on different subsets of the same data.

Value

Returns a single numeric value representing the estimated unit information standard deviation.

References

Röver C, Bender R, Dias S, Schmid CH, Schmidli H, Sturtz S, Weber S, Friede T (2021). "On weakly informative prior distributions for the heterogeneity parameter in Bayesian random-effects meta-analysis." *Research Synthesis Methods*, **12**(4), 448–474. doi:10.1002/jrsm.1475.

Examples

```
# Example with simulated data
sei <- c(0.2, 0.3, 0.25, 0.15)
ni  <- c(50, 30, 40, 80)
estimate_unit_information_sd(sei = sei, ni = ni)
```

`fitted.brma` *Fitted Values for brma Objects*

Description

Extract in-sample fitted values from a fitted `brma` object.

Usage

```
## S3 method for class 'brma'
fitted(
  object,
  unit = "estimate",
  conditioning_depth = "marginal",
  component = "location",
  bias_adjusted = FALSE,
  output_measure = NULL,
  transform = NULL,
```

```
  conditional = FALSE,
  ...
)
```

Arguments

object	a fitted brma object.
unit	output unit. Only "estimate" is implemented currently.
conditioning_depth	conditioning depth for location fitted values. "marginal" uses fixed effects only, "cluster" conditions on cluster-level random effects, and "estimate" conditions on the full estimate-level fitted value.
component	fitted component to return. "location" (alias "mods") returns location fitted values, "scale" returns fitted heterogeneity τ_i, and "all" returns both as a named list.
bias_adjusted	whether location fitted values should adjust for publication bias. Defaults to FALSE.
output_measure	effect-size measure for location/effect predictions. Defaults to the fitted measure. Supported conversions are among "SMD", "COR", "ZCOR", and "OR"; "RR", "HR", "IRR", "RD", and "GEN" can only be returned on their fitted measure. Use transform = "EXP" for ratio-scale output from log-scale measures.
transform	optional display transformation. Currently "EXP" exponentiates log-scale measures "OR", "RR", "HR", and "IRR".
conditional	whether to return fitted values from conditional posterior predictions for RoBMA product-space objects.
...	additional arguments. Currently only quiet is honored.

Details

This method is a compact adapter around predict.brma. It summarizes posterior prediction draws by their column means and returns a base numeric vector, matching the usual fitted contract. Use predict() directly when posterior draws or intervals are needed.

The default conditioning_depth = "marginal" corresponds to predict(object, type = "terms") and matches the usual fitted-value convention for meta-regression. For normal models, conditioning_depth = "estimate" corresponds to BLUP means for the observed estimates.

For component = "all", conditioning_depth, output_measure, and transform apply only to the location component. The scale component always returns fitted τ_i values.

Value

A numeric vector of fitted values, or a named list with location and scale components when component = "all".

See Also

predict.brma(), residuals.brma(), blup.brma()

forest.brma *Forest Plot for brma Objects Using metafor*

Description

`metafor::forest()` creates a forest plot for a fitted brma object by adapting RoBMA results to **metafor**'s public forest plotting API. Study rows are drawn by `forest.default` and the RoBMA pooled posterior summary is added with `addpoly.default`.

Usage

```
## S3 method for class 'brma'
forest(
  x,
  addfit = TRUE,
  addpred = FALSE,
  newdata = NULL,
  predstyle = "line",
  mlab = NULL,
  row = -1,
  level = 95,
  bias_adjusted = TRUE,
  conditional = FALSE,
  as_data = FALSE,
  predlim,
  border,
  constarea,
  ...
)

## S3 method for class 'metafor_forest.brma'
forest(
  x,
  addfit = TRUE,
  predstyle = "line",
  mlab = NULL,
  row = -1,
  predlim,
  border,
  constarea,
  ...
)
```

Arguments

x	a fitted brma object or an object returned by `as_metafor_forest()`.

addfit	logical; whether to add the RoBMA pooled-effect summary polygon. Defaults to TRUE.
addpred	logical; whether to add a posterior prediction interval for one new true effect. Defaults to FALSE.
newdata	optional one-row prediction design used when addpred = TRUE. See as_metafor_forest().
predstyle	character; prediction interval style passed to addpoly.default. Defaults to "line".
mlab	label for the summary row. Defaults to "Pooled Effect" for a pooled summary and "Predicted Effect" when newdata requests a moderator-specific prediction.
row	numeric row position for the pooled-effect summary polygon. Defaults to -1.
level	credible/confidence level in percent, greater than 0 and less than 100. A value below 1 is read as a proportion (0.95 is 95 percent), so 1 means 1 percent. Defaults to 95.
bias_adjusted	logical; whether pooled and prediction summaries should adjust for publication bias. Defaults to TRUE.
conditional	logical; whether to return conditional posterior summaries for RoBMA product-space objects. Defaults to FALSE.
as_data	logical; if TRUE, returns the prepared data instead of creating a plot.
predlim	argument passed to addpoly.default for prediction interval rendering.
border, constarea	arguments passed to addpoly.default for summary polygon rendering.
...	additional arguments passed to forest.default. This includes auxiliary-column arguments such as ilab, ilab.lab, ilab.xpos, and ilab.pos. Shared graphical arguments such as annotate, digits, transf, atransf, efac, col, lty, fonts, and cex are also forwarded to addpoly.default.

Details

This method deliberately uses metafor's public vector-based forest plotting API instead of constructing a fake metafor model object. The observed study rows are shown as usual Wald intervals based on the supplied effect sizes and standard errors. The model summary row uses the posterior mean and credible interval from pooled_effect. When addpred = TRUE without newdata, the displayed center, credible interval, and marginal prediction interval use the average-design target returned by pooled_effect. With explicit newdata, all three refer to that one prediction row and the prediction interval uses predict.brma with type = "estimate" and conditioning_depth = "marginal". For **metafor**'s predstyle = "shade" and predstyle = "dist", the adapter supplies the predictive density. Without newdata it is the posterior predictive density, a normal mixture over the posterior draws, smoothed by a normal kernel for display (a kernel density estimate of predictive draws without their simulation noise); the smoothing keeps point masses (draws without heterogeneity) visible. With newdata it is a kernel density estimate of the predictive draws from predict.brma. The displayed prediction-interval endpoints are the quantiles reported by pooled_effect or predict.brma.

Additional study-level columns can be added with metafor's ilab interface. When supplied to as_metafor_forest() or to metafor::forest(..., as_data = TRUE), those arguments are stored

in the adapter object's `forest_args`; presentation options such as `addfit`, `row`, and `predstyle` are retained by `forest(..., as_data = TRUE)` for later rendering. Explicit plot-time arguments override stored defaults. When supplied to `metafor::forest()` at plot time, they are forwarded directly.

Use `as_metafor_forest()` to prepare the adapter object without requiring a **metafor** installation.

Value

Invisibly returns the object returned by `forest.default`. If `as_data = TRUE`, returns a `metafor_forest.brma` object.

See Also

`as_metafor_forest()`, `pooled_effect()`, `predict.brma()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE) &&
    requireNamespace("metafor", quietly = TRUE)) {
  data(dat.bcg, package = "metadat")
  dat <- metafor::escalc(
    measure = "RR",
    ai      = tpos,
    bi      = tneg,
    ci      = cpos,
    di      = cneg,
    data    = dat.bcg
  )

  fit <- brma(yi = yi, vi = vi, data = dat, measure = "RR")
  metafor::forest(fit)
  metafor::forest(fit, addpred = TRUE, atransf = exp)

  count_columns <- as.matrix(dat[, c("tpos", "tneg", "cpos", "cneg")])
  metafor::forest(
    fit,
    addpred   = TRUE,
    slab      = paste(dat$author, dat$year),
    ilab      = count_columns,
    ilab.lab  = c("T+", "T-", "C+", "C-"),
    ilab.xpos = c(-8.5, -7.5, -6.5, -5.5),
    xlim      = c(-10, 4),
    atransf   = exp
  )
}

## End(Not run)
```

funnel *Frequentist-Style and Bayesian Funnel Plots for brma Objects*

Description

`funnel.brma` creates fast plug-in funnel contours. Location and bias parameters are fixed at their within-model posterior means; heterogeneity uses the root mean square SD, $\sqrt{E(\tau^2)}$. For model-averaged fits, complete joint models retain their posterior model probabilities and their plug-in sampling CDFs are averaged before the contours are inverted. This makes the ordinary normal-model construction directly comparable to a conventional plug-in funnel such as `metafor::funnel(..., addtau2 = TRUE)`.

`bfunnel.brma` instead averages the sampling CDF over posterior draws before inversion. It therefore incorporates continuous parameter uncertainty as well as model uncertainty.

Usage

```
## S3 method for class 'brma'
funnel(
  x,
  residual,
  type = "LOO-PIT",
  unit = "estimate",
  conditioning_depth = "marginal",
  sampling_heterogeneity = TRUE,
  sampling_bias = TRUE,
  max_samples = 10000,
  plot_type = "base",
  ...
)

bfunnel(x, ...)

## Default S3 method:
bfunnel(x, ...)

## S3 method for class 'brma'
bfunnel(
  x,
  sampling_heterogeneity = TRUE,
  sampling_bias = TRUE,
  max_samples = 10000,
  plot_type = "base",
  ...
)
```

Arguments

x	a fitted brma object
residual	whether to use residual mode. Defaults to not specified, which means the function automatically determines the mode: • not specified: Displays outcomes only for intercept-only models with one common marginal heterogeneity distribution. Otherwise it automatically uses LOO-PIT residual mode. • FALSE: explicitly requests outcome mode. This is available only for intercept-only models with common marginal heterogeneity. • TRUE: explicitly requests residual mode, displaying residuals on the x-axis and using type to determine how those residuals are computed.
type	the type of residuals to use when in residual mode. Options are: • "LOO-PIT" (alias: "rstudent"; default): Leave-one-out probability integral transform residuals returned by rstudent.brma. • "rstandard": Internally standardized residuals using rstandard.brma. Only available for normal outcome models. • "outcome": Raw outcome residuals from residuals.brma with type = "outcome". Its funnel bands are descriptive sampling-error-only reference bands. Only used when funnel is in residual mode.
unit	output unit for residual mode. Only "estimate" is currently implemented.
conditioning_depth	conditioning depth for non-LOO residual mode. Options are "marginal", "cluster", and "estimate". Omit this argument for type = "LOO-PIT" (the default) or its "rstudent" alias: the estimate-deletion context determines the LOO predictive target.
sampling_heterogeneity	whether heterogeneity should be incorporated into the sampling distribution funnel. Defaults to TRUE. Only used in outcome mode and ignored in residual mode.
sampling_bias	whether publication bias should be incorporated into the sampling distribution funnel. Defaults to TRUE. Only used when residual = FALSE or when automatic mode selects outcome mode. Ignored in residual mode. When TRUE and the model includes selection models (weightfunction), uses selected-normal quantiles where the fitted source and publication structure admit a scalar calculation. Unsupported joint-selection configurations fail explicitly; use sampling_bias = FALSE for bias-adjusted contours. Exact-zero integrated selected cells require an existing positive-weight cancellation certificate; otherwise the selected contour is unavailable. sampling_bias = FALSE remains an explicitly unselected reference. Ordinary and constant-positive step-selection rows are evaluated as exact Gaussian components before conditioned-source quadrature. Remaining selected rows are unavailable when the required odd quadrature order exceeds the existing 401-node cap or is nonfinite; the call then stops with an error of class RoBMA_funnel_quadrature_rejected, whose parent class is RoBMA_numerical_unavailable, carrying the standard deviation ratio, the required_order and the maximum order. max_samples controls posterior rows

and does not increase this quadrature budget. When TRUE and the model includes PET/PEESE, incorporates the expected skew from these regression adjustments.

`max_samples` maximum number of posterior draws used by `bfunnel()`. Defaults to `10000`; use `Inf` for all draws. `funnel()` uses all draws to estimate conditional plug-in values and posterior model probabilities, then evaluates only one CDF per joint model.

`plot_type` whether to use a base plot `"base"` or ggplot2 `"ggplot"` for plotting. Defaults to `"base"`.

`...` additional graphical arguments to customize the plot appearance:

xlim, ylim numeric vectors of length 2 specifying axis limits

xlab, ylab character strings for axis labels

main character string for plot title (default: no title)

pch point symbol (default: 21, filled circle). Use standard R pch values.

col point border color (default: "black")

bg point fill/background color (default: "#A6A6A6")

cex point size multiplier for base graphics (default: 1)

size point size for ggplot2 (default: 2)

las axis-label style for base graphics (default: 1)

back background region color (default: "grey"). Set to NA to suppress.

shade funnel region fill color (default: "white"). Set to NA to suppress.

lty line type for funnel edges and center line (default: "dotted")

col.line color for funnel edge lines (default: "black")

refline numeric override for the reference line. By default, residual mode uses 0, while outcome mode uses the center of the fitted sampling distribution, which may be curved when PET/PEESE bias adjustment is incorporated.

col.refline color of vertical reference line (default: "black")

as_data if TRUE, returns plot data instead of creating a plot

Details

The ordinary `funnel()` has outcome and residual modes. If `residual` is not specified, outcomes are used only when the fitted model is intercept-only and its marginal heterogeneity is identical across studies. The latter check uses fitted row-marginal random-effect variances, including complete ZGZ' contributions for `brma.mv()` random- formula models. All other models use residual mode.

Outcome mode (intercept-only models with common heterogeneity): Displays observed effect sizes on the x-axis and standard errors on the y-axis. The reference line follows the center of the fitted sampling distribution. When `sampling_bias = FALSE`, this center is the plug-in mixture median (the pooled effect in a single model); when PET/PEESE bias adjustment is incorporated, the center line can vary with the standard error. The funnel region represents the central 95\ region of the plug-in sampling distribution, optionally incorporating heterogeneity and publication bias. Location and bias parameters use posterior means, and heterogeneity uses the square root of the posterior mean variance, separately within every complete joint model. Integrated and conditioned variance components are averaged on the variance scale. The resulting model-specific CDFs are averaged with posterior model probabilities and only then inverted; neither parameters nor contour endpoints are averaged across models. For normal models, these contours use the marginal response target:

heterogeneity represents newly realized latent effects and fitted study effects are not retained. For correlated known-V `brma.mv()` models, outcome-mode funnels use the diagonal of `V` for Gaussian marginal contours. The contour describes a new, independent estimate at each hypothetical standard error, so the fitted rows' correlations do not enter it: under `weight_rule = "product"` such an estimate forms its own dependency block and its selected law is the scalar selected normal at that standard error, mixed over the population law of any conditioned source. Nonconstant joint selection under a best weight rule instead weighs an estimate against the others of its publication; it requires a full publication design at each hypothetical standard error and remains unavailable in the scalar contour calculation. Conditioning every applicable selection source cancels positive weights and therefore retains the Gaussian contours. The observed estimates' own selected marginals inside a correlated block differ from this new-study contour; use `zplot()` to display them at the fitted publication designs. Residual mode follows the fitted estimate-unit residual target.

`bfunnel()` is available only for intercept-only normal models with common marginal heterogeneity. It has no residual mode: explicitly supplied `residual`, `type`, `unit`, or `conditioning_depth` arguments are refused. It averages the same sampling CDF over posterior draws and then inverts the averaged CDF. Thus it incorporates posterior uncertainty in the effect, heterogeneity, PET/PEESE coefficients, and selection parameters in addition to model uncertainty. For models without a common outcome-scale target, use the default LOO-PIT residual funnel from `funnel()`.

Residual mode (models without a common outcome-scale target): Displays residuals on the x-axis and standard errors on the y-axis. The funnel region is bounded by the 0.025 and 0.975 normal quantiles times the displayed standard error. It is a descriptive reference band, not a posterior-predictive coverage interval. With `type = "LOO-PIT"`, the plotted x-coordinate is the PIT-normalized `z` value returned by `rstudent.brma` multiplied by its LOO predictive standard deviation. Consequently, each plotted `x / se` is exactly its LOO-PIT `z`. This reduces to the ordinary raw deleted residual when the LOO predictive distribution is normal and extends the same funnel geometry to non-normal continuous predictive mixtures. With `type = "rstandard"`, the plotted values are internally standardized residual companions from `rstandard.brma`. With `type = "outcome"`, these are raw outcome residuals and the band uses sampling error only; it does not incorporate fitted heterogeneity.

The `type` argument controls how residuals are computed in residual mode. See `residuals.brma` for details on each type. The `sampling_heterogeneity` and `sampling_bias` arguments are ignored in residual mode.

For GLMM models, outcome-mode observed effect sizes are computed from the raw frequency data using formulas equivalent to `metafor::escalc`. Their contours use the corresponding continuity-corrected effect-size estimates and approximate standard errors in a normal reference calculation. The default x-axis label and a warning identify this as a descriptive approximation; it is not a coverage interval from the fitted binomial or Poisson likelihood and does not use the prior-nuisance marginal GLMM response target from `predict()`. Exact discrete-likelihood coverage cannot be indexed by a scalar standard error alone because it also depends on arm totals or exposures and nuisance rates. LOO-PIT residual mode of GLMMs uses the descriptive normal approximation on the effect-size scale of `residuals.brma` (the fitted predictive distribution is discrete); the default x-axis label identifies it. The sampling-error-only bands of the residual funnel are not exact diagnostics for the fitted count likelihood.

Outcome-mode plot data carry the canonical `RoBMA_target` attribute. Its `reference_law` identifies the independently published new-estimate product-rule law; `estimand` separately records plug-in versus posterior-predictive averaging. Declared source roles retain their provenance.

Value

If `as_data = TRUE`, `funnel.brma` and `bfunnel.brma` return a list with the data used for plotting, including the plotted points, funnel polygons, plotting limits, labels, reference line, and outcome-funnel estimand. GLMM outcome-mode data of `funnel.brma` also contain an `approximation` record identifying the descriptive normal effect-size approximation and stating that it is not fitted discrete- likelihood coverage (`bfunnel.brma` is available only for normal outcome models). Otherwise, it returns `NULL` invisibly if `plot_type = "base"` or a ggplot object if `plot_type = "ggplot"`.

See Also

`residuals.brma()`, `rstandard.brma()`, `rstudent.brma()`, `predict.brma()`, `zplot.brma()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE) &&
    requireNamespace("metafor", quietly = TRUE)) {
  data(dat.bcg, package = "metadat")
  dat <- metafor::escalc(
    measure = "RR",
    ai      = tpos,
    bi      = tneg,
    ci      = cpos,
    di      = cneg,
    data    = dat.bcg
  )

  fit <- brma(yi = yi, vi = vi, data = dat, measure = "RR")
  funnel(fit)
  bfunnel(fit)
  funnel(fit, pch = 19, col = "blue", bg = "lightblue")

  fit_reg <- brma(
    yi      = yi,
    vi      = vi,
    mods    = ~ ablat + year,
    data    = dat,
    measure = "RR"
  )
  fit_reg <- add_loo(fit_reg)
  funnel(fit_reg)
  funnel(fit_reg, type = "outcome")

  funnel_data <- funnel(fit, as_data = TRUE)
  funnel(fit, plot_type = "ggplot")
}

## End(Not run)
```

`hatvalues.brma` *Hat Values for brma Objects*

Description

Computes hat values (leverages) from a fitted brma object. Returns posterior mean leverages, one for each observation.

Usage

```
## S3 method for class 'brma'
hatvalues(model, max_samples = Inf, ...)
```

Arguments

`model`	a fitted brma object
`max_samples`	maximum posterior draws used for known-`V` `brma.mv()` marginal covariance computations. Defaults to `Inf`. Finite values deterministically thin draws across posterior row order.
`...`	additional arguments (currently ignored)

Details

Hat values (leverages) measure the influence of each observation on the fitted values. In a Bayesian meta-analysis, the random effects variance τ^2 is uncertain, so the hat matrix depends on the posterior samples of τ^2.

This function computes the diagonal elements of the hat matrix:

$$h_i = (X(X^T W X)^{-1} X^T W)_{ii}$$

where W is the fitted precision matrix. Without additional likelihood weights it is the inverse marginal covariance; likelihood weights modify estimation precision without redefining the original outcome covariance used to standardize residuals.

The hat matrix is computed for each posterior draw and then averaged over draws, matching the vector output shape used by `metafor`. The members of a `RoBMA()` ensemble can differ in their PET and PEESE terms; each draw then uses the design of the member it belongs to (the location columns plus the `PET` column of PET members or the `PEESE` column of PEESE members), and the reported leverage is the posterior mean over all draws. For known-`V` `brma.mv()` random-formula models, leverages use the marginal GLS covariance $V + ZGZ'$.

This method is available only for normal outcome models without weightfunction selection.

For known-`V` models, the leverages require applying the inverse of the marginal covariance. A covariance that is too ill-conditioned for that (a reciprocal condition number of about `2e-10` or less in any posterior draw used) is unavailable: the call stops with an error of class `RoBMA_known_v_precision_unavailable`, whose parent class is `RoBMA_numerical_unavailable`, instead of returning a rounded number. The condition carries the draw positions (`rows`) and their `reciprocal_condition`.

Value

A numeric vector of length K, one posterior mean leverage per observation.

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE) &&
    requireNamespace("metafor", quietly = TRUE)) {
  data(dat.bcg, package = "metadat")
  dat <- metafor::escalc(
    measure = "RR",
    ai      = tpos,
    bi      = tneg,
    ci      = cpos,
    di      = cneg,
    data    = dat.bcg
  )

  fit <- brma(yi = yi, vi = vi, data = dat, measure = "RR")
  hatvalues(fit)
}

## End(Not run)
```

Havrankova2025 *1159 effect sizes from a meta-analysis of beauty and professional success by Havránková et al. (2025)*

Description

The data set contains effect sizes (percent increase in earnings), standard errors, study identifiers, sample sizes, and the type of customer contact (no, some, or direct). The meta-analysis examined the relationship between perceived beauty and professional success (Havránková et al. 2025).

Usage

```
Havrankova2025
```

Format

A data.frame with 5 columns and 1159 observations:

`y` Effect size: percent increase in earnings.

`se` Standard error of y.

`facing_customer` Type of customer contact.

`study_id` Study identifier.

`N` Sample size.

References

Havránková Z, Havránek T, Bortnikova K, Bartoš F (2025). "Meta-analysis of field studies on beauty and professional success." Preprint available at https://meta-analysis.cz/beauty/beauty.pdf.

`hist.zplot_brma` *Histogram of Z-Statistics*

Description

Plots a histogram of observed z-values from the meta-analysis.

Usage

```
## S3 method for class 'zplot_brma'
hist(
  x,
  plot_type = "base",
  from = -6,
  to = 6,
  by = 0.5,
  length.out = NULL,
  add = FALSE,
  plot_thresholds = TRUE,
  dots_thresholds = NULL,
  dots_hist = NULL,
  dots_all = NULL,
  ...
)
```

Arguments

`x`	a zplot_brma object.
`plot_type`	graphics system: `"base"` or `"ggplot"`. Defaults to `"base"`.
`from, to`	z-value range for plotting. Defaults to `-6` and `6`.
`by`	bin width. Defaults to 0.5.
`length.out`	number of bins (alternative to `by`).
`add`	whether to add to an existing base plot. Defaults to `FALSE`. For ggplot, combine the histogram and density layers with `+`; `add = TRUE` is unsupported.
`plot_thresholds`	whether to show significance threshold lines. Defaults to `TRUE`.
`dots_thresholds`	graphical parameters for threshold lines (list).
`dots_hist`	graphical parameters for histogram (list).
`dots_all`	graphical parameters passed to all components (list).
`...`	additional graphical parameters, such as `xlim` and `ylim`. Use `as_data = TRUE` to return histogram data without drawing.

Details

Z-statistics are computed as effect size divided by standard error (`yi / sei`). Histogram bins are adjusted to align with significance thresholds when selection model priors are present. The bins cover `from` to `to`; when by does not divide that range, the last bin ends at `to`. `xlim` and `ylim` set the displayed axis ranges without changing the bins. Standalone ggplot histograms scale automatically to added layers unless `ylim` is supplied.

Value

`NULL` invisibly for base graphics, a ggplot2 object, or a data frame with columns `x`, `density`, and `breaks` (bin widths) when `as_data = TRUE`.

See Also

`plot.zplot_brma()`, `lines.zplot_brma()`

Hoogeveen2023 *106 analyst-team effect-size estimates of the relation between religiosity and well-being from Hoogeveen et al. (2023)*

Description

The data set contains standardized effect-size estimates, standard errors, 95% confidence interval bounds, sample sizes, effect-size types, sequential analyst-team identifiers, and team-level characteristics describing psychological dependent-variable coding and self-reported methodological knowledge. The Many-Analysts Religion Project examined whether religious people self-report greater well-being (Hoogeveen et al. 2023).

Usage

```
Hoogeveen2023
```

Format

A data.frame with 9 columns and 106 observations:

`yi` Standardized effect-size estimate.

`sei` Standard error of the effect-size estimate.

`lci` Lower bound of the reported 95% confidence interval.

`uci` Upper bound of the reported 95% confidence interval.

`ni` Analysis sample size.

`type` Reported effect-size type: standardized regression coefficient beta or Cohen's d.

`team` Sequential analyst-team identifier in this example.

`team_psych` Whether the effect estimate used the psychological dependent-variable coding.

`team_knowledge` Team's self-reported methodological-knowledge rating, from 1 (no knowledge) to 5 (expert).

Source

https://osf.io/gkxqy/

References

Hoogeveen S, Sarafoglou A, Aczel B, Aditya Y, Alayan AJ, others (2023). "A many-analysts approach to the relation between religiosity and well-being." *Religion, Brain & Behavior*, **13**(3), 237–283. doi:10.1080/2153599X.2022.2070255.

`Hoppen2025` *37 studies from a meta-analysis of social comparison as a behavior change technique by Hoppen et al. (2025)*

Description

The data set contains Cohen's d effect sizes, variances, and study characteristics including outcome type, feedback level, social comparison type, number of sessions, sample type, sample size, and country. The meta-analysis examined social comparison as a behavior change technique across the behavioral sciences (Hoppen et al. 2025).

Usage

```
Hoppen2025
```

Format

A data.frame with 9 columns and 37 observations:

`d` Cohen's d effect size.

`v` Sampling variance of d.

`outcome` Outcome type.

`feedback_level` Feedback level.

`social_comparison_type` Social-comparison type.

`sessions` Number of sessions.

`sample_type` Sample type.

`sample_size` Sample size.

`country` Country.

References

Hoppen TH, Cuno RM, Nelson J, Lemmel F, Schlechter P, Morina N (2025). "Meta-analysis of randomized controlled trials examining social comparison as a behaviour change technique across the behavioural sciences." *Nature Human Behaviour*, **9**(8), 1595–1612. doi:10.1038/s41562025-022092.

hypothesis *Hypothesis Bayes Factors*

Description

Computes Bayes factors for scalar hypotheses written as expressions, for example `"mu = 0"`, `"year > 0"`, or `"mu_alloc[alternate] > mu_alloc[random]"`.

Usage

```
hypothesis(object, ...)

## Default S3 method:
hypothesis(object, hypothesis, ...)

## S3 method for class 'brma'
hypothesis(
  object,
  hypothesis,
  component = "auto",
  standardized_coefficients = FALSE,
  conditional = FALSE,
  logBF = FALSE,
  BF01 = FALSE,
  seed = NULL,
  density_method = "qCMDE",
  density_control = NULL,
  n_samples = 10000,
  columns = "default",
  ...
)

## S3 method for class 'marginal_means.brma'
hypothesis(
  object,
  hypothesis,
  parameter = NULL,
  logBF = FALSE,
  BF01 = FALSE,
  seed = NULL,
  density_method = NULL,
  density_control = NULL,
  columns = "default",
  ...
)

bf_hypothesis(object, ...)
```

```
BF_hypothesis(object, ...)
```

Arguments

object	a fitted brma, BMA, RoBMA, or a posterior object accepted by BayesTools::hypothesis_BF().
...	additional arguments passed to methods. For fitted brma objects, unused arguments trigger a warning.
hypothesis	character vector with scalar hypothesis statements. For marginal_means.brma objects, constants are specified on the fitted linear-predictor scale, even when the object stores a display transformation. Separate statements may reference different model parameters, also the marginal means of different parameters of marginal_means.brma objects (unless parameter is set); each statement itself must resolve to one scalar model target.
component	parameter component. Defaults to "auto", which infers the component when possible. Use "mods" (alias "location"), "scale", or "random" to disambiguate terms used in multiple model components. A statement whose references still name several model parameters stops with an error of class BayesTools_hypothesis_ambiguo with the parent class BayesTools_hypothesis_statement: the statement or component needs to change, the test itself is not unavailable. An ambiguous reference that BayesTools refuses with the class BayesTools_parameter_ambiguous (for example the contrast coefficient "g{1}" of a factor term in the location and the scale formula) has these classes too, before its BayesTools classes. A statement whose parameters belong to another component than component, including a name that is known only outside component next to references of component (with or without factor levels, e.g. "log_tau_g[a] > mu_g[a]" with component = "mods"), stops with the classes BayesTools_hypothesis_statement and BayesTools_hypothesis_component_mismatch (plot.brma() stops with RoBMA_component_mismatch for its own component mismatches). The exception is the alias of a factor term of both the location and the scale formula (for example g for mods = ~ g, scale = ~ g) next to a name that is unknown within component, such as a name of the other component or a display label (for example "g > log_tau_intercept" with component = "mods"): the statement is resolved without component, where the alias is ambiguous, and stops with BayesTools_hypothesis_statement followed by the classes of the BayesTools refusal of its first unresolved reference, the ambiguity of the alias (BayesTools_parameter_ambiguous, without BayesTools_hypothesis_ambiguous, as component is set already) or the unknown name (BayesTools_parameter_not_found, for example in "log_tau_intercept < g" or, with a level of the alias, in "g[a] > log_tau_intercept"). The display labels of the summary tables (for example "(mu) intercept", "exp(intercept)", or "(mu) g[a]") name the parameters they label, with and without component. The random component supports point, interval, and directional hypotheses for semantic standard deviation, variance, correlation, and allocation quantities. Point-null hypotheses require a direct parameter reference, a factor level, or a linear combination of the levels of one factor term (for example "g[a] = g[b]" or "2 * g[a] = 0.1"). Point hypotheses on a random-effect variance are evaluated through

its standard deviation, so that both give the same Bayes factor at positive variance values. At zero, the square map is singular: the variance is tested on its own scale and follows its own prior-ordinate eligibility rules. A statement comparing such a point with a region of the variance (for example `"tau2 = 0.09 vs tau2 > 0.09"`) evaluates the point through the standard deviation and the region on the variance draws. Certified `exp(affine)` fitted-scale hypotheses are available with KDE only for atom-free, unconditional scalar targets. Values at a prior point mass (for example, gated components at 0) have no Savage-Dickey Bayes factor; the Component Inclusion table compares the exclusion and inclusion of gated components. Point nulls at an exact support boundary (for example, a variance proportion at 0) use the one-sided prior ordinate when BayesTools classifies it as exact, finite, and positive. Publication-bias parameters are not supported (class `BayesTools_hypothesis_target`), including the quantities of the publication-bias prior such as the weight-function coordinates `"omega[0,0.025]"` and `"bias_indicator"`. Other quantities of the fitted model that are no tested model parameter, such as inclusion indicators (`"mu_x_indicator"`), the component inclusions of variance allocations (`"inclusion(study)"`), or latent cluster effects, are refused with the class `BayesTools_hypothesis_tar` as well; `hypothesis_quantities()` lists the tested quantities.

`standardized_coefficients`
: whether moderator and scale coefficients are tested on the standardized predictor scale. Defaults to `FALSE`.

`conditional`
: whether to use the conditional posterior for product-space model-averaging objects. Defaults to `FALSE`. When this argument is omitted and the selected parameter has both null and alternative components, a warning notes that the full ensemble is used. Pass `FALSE` explicitly to retain that test without the warning, or `TRUE` to test only models where the parameter is active. Point hypotheses on linear combinations of factor levels of model-averaged objects require `conditional = TRUE`: the levels share the null component of the averaged prior.

`logBF`
: whether to display the Bayes factor on the log scale.

`BF01`
: whether to display the inverse Bayes factor.

`seed`
: optional seed used by BayesTools for sampled prior quantities.

`density_method`
: posterior density method. Defaults to `"qCMDE"` for fitted `brma` objects. `"KDE"` uses the standard BayesTools kernel density estimate. `"normal"` uses the BayesTools normal approximation for point-null hypotheses on fitted `brma` objects. `"qCMDE"` and `"IWMDE"` attach RoBMA likelihood-aware posterior ordinates for point-null hypotheses and propagate their `BF_error` estimates printed as `error%(BF)`. `BF_error` is a Monte Carlo diagnostic for the computed posterior ordinate, not a complete standard error. For a subsampled mixed posterior it is a worst-correlation delta upper bound combining the selected active-row and complete indicator-chain components. It excludes uncertainty in the prior ordinate and, for IWMDE, uncertainty from estimating the conditional weight function. At a location that is also a posterior atom, qCMDE/IWMDE reports the continuous ordinate; the atom probability is a separate discrete mass. Use `density_diagnostics()` to inspect the attached computation and reliability diagnostics. For `marginal_means.brma` objects, `"normal"` is not supported. Their default `NULL` reuses the density method stored by `marginal_means()`; explicitly request `"KDE"` to override it. `"qCMDE"`

and "IWMDE" compute missing point-null ordinates from the stored source model. Matching is case-insensitive. qCMDE/IWMDE ordinates use the exact fitted-to-original structural coefficient map and induced prior density stored by BayesTools. qCMDE/IWMDE supports exact linear maps: coefficients, factor levels of every contrast, and linear combinations of levels. For nonlinear joint maps, such as an exponentiated intercept that also depends on varying slopes, KDE point and directional hypotheses are available only when structural metadata certifies an atom-free, unconditional scalar target and the point is inside the open transformed support. Alternatively, use standardized_coefficients = TRUE.

The statements are planned and evaluated by the hypothesis engine of BayesTools (BayesTools::hypothesis_BF_model()); RoBMA describes its models to it (BayesTools::hypothesis_provider()) and provides the density methods qCMDE and IWMDE (BayesTools::hypothesis_density_backend()). Every statement is planned before it is evaluated: the plan records the tested target (its weights on the fitted coefficients, its BayesTools prior density and declared atoms), the classification of each point value by BayesTools::prior_ordinate_status(), and whether each density method can evaluate it; hypothesis_quantities() renders the same plans. A point hypothesis whose prior ordinate is not exact, finite, and regular stops with the BayesTools condition class (for example BayesTools_point_mass_at_null, BayesTools_infinite_ordinate, or BayesTools_inexact_ordinate). Other refusals of a test that cannot be computed have the class BayesTools_hypothesis_unavailable and one of BayesTools_hypothesis_fixe (the quantity is fixed by the fitted model), BayesTools_hypothesis_target (no supported target or prior density), or BayesTools_hypothesis_method (the density method is unavailable for the target); linear combinations that BayesTools cannot certify keep the class BayesTools_linear_target_unavailable. A method refusal because the fitted model does not support qCMDE/IWMDE also has the classes of the refusals of these requests by plot.brma() and marginal_means(): RoBMA_density_method_unavailable and one class naming the cause, RoBMA_density_method_rando (brma.mv() random-formula models without known V), RoBMA_density_method_scale_components (component-specific scale formulas), or RoBMA_density_method_glmm (IWMDE for binomial and Poisson GLMMs). qCMDE/IWMDE refusals for a cause for which plot.brma() refuses these methods have its classes too: RoBMA_density_method_conditional_ (conditional random-effect statements), RoBMA_density_method_random_target (random-effect quantities without a supported scalar random-component coordinate), and RoBMA_density_method_original_scale (original-scale coefficients whose fitted map is nonlinear, such as exp(affine) targets). As in plot.brma(), allocated component SDs in a predictor-scaled block require standardized_coefficient = TRUE or density_method = 'KDE'; the block-wide restriction also covers an unchanged centered intercept. As in plot.brma(), the capability of the fitted model is checked before these causes and before other qCMDE/IWMDE refusals of the tested quantity: a request to which both apply stops with the capability refusal. A qCMDE/IWMDE ordinate that cannot be constructed stops with RoBMA_density_ordinate_error; when a fitted chain has no eligible row of the target, the error also has the class RoBMA_density_chain_coverage and its message names density_method = "KDE" and longer chains as remedies. The refusal for fewer than 20 continuous active samples of the target takes precedence over this diagnosis: such a target is refused for that reason alone, without

the class RoBMA_density_chain_coverage, even when a fitted chain has none of its rows.

Statements that need to be restated, or an argument that needs to change, stop with the class BayesTools_hypothesis_statement without BayesTools_hypothesis_unavailable: point expressions that are no direct reference (for example "2 * mu = 0"), nonlinear expressions of factor levels, several point values of a random-effect variance in one statement, one quantity named by several names in one statement (for example "mu > 0 & intercept < 1" or "g[a] > mu_g[a]"; use one name), model-averaged marginal-means statements that mix point and region events or span several marginal means (levels or scalar marginal means), ambiguous references (BayesTools_hypothesis_ambiguous, see component), references to parameters of another component (BayesTools_hypothesis_component_mismatch), marginal-means statements that refer to another parameter than parameter (BayesTools_hypothesis_pa factor contrast coefficients such as "g{1}" (state the hypothesis on factor levels; a selector of a coefficient that is a level, such as "g{1}" of a treatment factor, also has the classes of the BayesTools refusal BayesTools_selector_unavailable, which names the level), references to unknown names or factor levels, and statements that reference no parameter (followed by the classes of the BayesTools refusal of such a statement, BayesTools_hypothesis_no_parameters and BayesTools_parameter_reso on fitted objects and on marginal means). An unknown name or level also has the classes BayesTools_parameter_not_found and BayesTools_parameter_resolution_error, with which BayesTools refuses it (on fitted objects and in linear combinations of marginal-means levels, the BayesTools condition itself with its fields alias and available). Every other refusal of a statement's references by BayesTools (class BayesTools_parameter_resolution_error) stops hypothesis() too, on fitted objects and on marginal means, as the BayesTools condition with its classes and fields after BayesTools_hypothesis_statement, also when BayesTools raises it only when it evaluates the statement (for example the whole factor term next to one of its levels, "g[a] > mu_g", as an unknown quantity). Missing or unsupported fitted metadata of the tested target (a parameter catalog without RoBMA parameters, the coefficient transform, the fitted coordinates, or the linear weights of a factor level; a fit of an older RoBMA/BayesTools build) stop with the class RoBMA_refit_required: refit the model. Its parent BayesTools_refit_required is the class of every error of RoBMA and BayesTools that asks for a refit; BayesTools' own refit errors stop hypothesis() unconverted.

density_control

named list of qCMDE/IWMDE tuning settings. Supported entries are n_points (default 100), samples (the fixed posterior-row sample size, default 500 for qCMDE and 1000 for IWMDE; use Inf for the eligible-row census), target_relative_mcse (default 0.05), normalization_points (default NULL, resolved to max(50, n_points)), normalization_prob (default 0.999; lower than 1 for qCMDE), and display_grid (default "adaptive"). integration_control (default NULL) retains the fitted selection-integration settings. Supply a control created by set_selection_likelihood_control() to change those settings for this post-fit calculation, for example list(integration_control = set_selection_likelihood_control(max = 32768)). This entry is available for Gaussian selection models with a fitted integration plan. The fitted object and posterior draws are unchanged. The max-

imum point budget controls factor QMC fallback; analytic and deterministic quadrature rules remain unchanged. Point ordinates use one state-independent simple random sample selected before ordinate contributions are evaluated. Multiple direct scalar point ordinates for one target share the same conditional-normalization pass while retaining separate diagnostics. A finite estimate is returned independently of its sample diagnostics. If the fixed sample does not meet the precision target, a warning recommends increasing `samples` or using the census; if the census does not meet the target, obtain more posterior draws. `display_grid` is immaterial for point-only requests. qCMDE normalizes each posterior row's conditional density over a range covering at least its central `normalization_prob` mass, on `normalization_points` nodes and the nested grid of those nodes and their midpoints. The mass of a row left outside the range, its truncation t, is therefore at most `1 - normalization_prob`: zero for rows whose Gaussian likelihood kernel meets a normal prior and whose evaluated values are those of the resulting truncated normal (their normalizer is its closed-form mass, so no lattice decides it), exactly computed for such rows that keep the lattice normalizer, bounded for a Gaussian kernel with another proper scalar prior, and estimated from the tails (each tail extended until its estimate is at most `(1 - normalization_prob) / 2`) otherwise. The tail extension has a limit, ends at values the joint density cannot evaluate, and cannot help a tail that does not decay, so the estimate for such a row can exceed that target and the bound below with it. A row normalizer missing the fraction t overstates that row's density by the factor $1/(1-t)$, so the ordinate is overstated by at most $t/(1-t)$ for the largest row truncation, which is at most `(1 - normalization_prob) / normalization_prob`; at the default `0.999` that is about 0.1 percent, and it is attained when all rows share one conditional law. `density_diagnostics()` report the largest row truncation (`normalization_truncation`), how it is known (`normalization_truncation_status`), and this bound (`truncation_ordinate_bound`), together with the change between the two nested grids (`ordinate_relative_change`), which estimates the discretization error of the lattice normalizers, and the contribution-share weighted lattice-resolution estimate (`lattice_resolution_relative_error`). The resolution estimate uses the Gaussian kernel width at the nested-grid maximum, or local log curvature when no kernel is available. Neither the resolution component nor the combined error estimate is a rigorous bound; a sharp prior core can be narrower than its likelihood kernel. Exact nonlattice normalizers have no lattice-resolution component. Set `normalization_prob` closer to 1 to reduce the truncation, and increase `normalization_points` to reduce the discretization. IWMDE checks its normalization mass on `normalization_points` points over the central `normalization_prob` quantile range of the target draws, widened by 10 percent on each side. A point ordinate warns at relative MCSE at least 5 percent, ESS below 100, maximum contribution share at least 20 percent, or fewer than 100 finite contributions. These same-sample diagnostics do not suppress a finite fixed-design estimate. The qCMDE ordinate error estimate (nested-grid change, lattice resolution and the tail-truncation bound) warns above 2.5 percent and is rejected above 5 percent; IWMDE normalization error warns above 5 percent and is rejected above 10 percent. For retained-location conditional normalizers, adaptive-quadrature sensitivity warns above 2.5 percent and is rejected above 5 percent.

`n_samples` number of prior samples/grid points used by BayesTools for deterministic marginal prior densities.

`columns` output columns passed to `BayesTools::hypothesis_BF()`. `"default"` returns the compact BayesTools table with `Alternative`, `Null`, `BF`, and `error%(BF)` columns. `"all"` also returns `prior`, `posterior`, and `method` columns. The `prior` and `posterior` columns are diagnostics rather than probabilities for all hypothesis types.

`parameter` optional marginal-means parameter/term used to disambiguate the hypothesis expression. Without it, each statement is tested on the marginal means of the parameter it names, so that separate statements may refer to different parameters (for example `c("g[a] > 0", "h[b] > 0")`); the rows are those of the statements tested one at a time, in the order of the statements. One statement must refer to the marginal means of one parameter: a statement that names several (for example `"g[a] > h[b]"`), or an alias that names several parameters, stops with an error of class `BayesTools_hypothesis_ambiguous`, with the parent class `BayesTools_hypothesis_statement`; `parameter` selects the parameter of an ambiguous alias. With `parameter`, every statement is tested on that parameter, and a statement that refers to the marginal means of another parameter stops with an error of class `BayesTools_hypothesis_parameter_mismatch`, with the parent class `BayesTools_hypothesis_statement`; for a statement that names several parameters (`"g[a] > h[b]"` with `parameter = "g"`) its message is the one given without `parameter`. A statement that references no marginal-means parameter stops with the class `BayesTools_hypothesis_statement`; for a statement without parameter symbols (for example `"1 > 0"`, also with `parameter`) also with `BayesTools_hypothesis_no_parameters` and `BayesTools_parameter_resolution_error`, and for an unknown name, or an unknown level of a factor term, also with `BayesTools_parameter_not_found` and `BayesTools_parameter_resolution_error`, as on fitted objects.

Details

For a continuous focal parameter θ, nuisance parameters ξ, and an unnormalized joint posterior kernel $q(\theta, \xi)$, qCMDE estimates the posterior ordinate as

$$\widehat{p}_Q(\theta^* \mid y) = \frac{1}{S} \sum_{i=1}^{S} \frac{q(\theta^*, \xi_i)}{\int_\Theta q(t, \xi_i)\, dt}.$$

Each row is therefore normalized over the focal-parameter support before the conditional densities are averaged. IWMDE instead uses

$$\widehat{p}_I(\theta^* \mid y) = \frac{1}{S} \sum_{i=1}^{S} w(\theta_i \mid \xi_i) \frac{q(\theta^*, \xi_i)}{q(\theta_i, \xi_i)},$$

where $w(\theta \mid \xi)$ is a normalized, fitted conditional weight. For half-bounded targets, IWMDE fits the distance from the declared boundary and normalizes a Normal weight over its central 95 percent positive-tail interval, using the actual computed endpoints. A draw outside that compact interval contributes zero while retaining its original row and chain. The same compact family is used when conditional fitting is unavailable. Compact focal support does not certify a finite joint second

moment or MCMC long-run variance; reported uncertainty is conditional on the fitted proposal and does not include uncertainty from learning it on the same draws. Spike-mixture and product-space results additionally retain the posterior mass of the continuous active branch. These estimators build on conditional marginal density estimation and importance-weighted marginal density estimation (Gelfand et al. 1992; Chen 1994).

qCMDE is the preferred likelihood-aware method when it is supported and its additional numerical normalization cost is acceptable. IWMDE can be faster, but is more sensitive to the adequacy and concentration of its fitted conditional weights. The normal approximation is useful as a rough check for near-normal interior ordinates; it is not a validation reference for skewed tails, bounded parameters, or one-sided support boundaries. Selected weightfunction models near zero cumulative ordered effects can take more than 10 minutes with qCMDE or IWMDE. qCMDE is the default for fitted `brma` objects. Use `"KDE"`, `"normal"`, or `"IWMDE"` explicitly to select another route. For binomial and Poisson GLMMs, the nuisance state includes the sampled estimate-level heterogeneity effects and baserates or log-rates. Averaging the normalized row-conditional focal densities over the posterior draws of this nuisance state targets the marginal posterior density by an exact identity; the estimate approximates that expectation with the posterior draws and numerical normalization, subject to the reported diagnostics. IWMDE is unavailable for GLMM density estimation because its high-dimensional conditional weights did not meet bridge-sampling certification. GLMM density curves and point-null hypotheses require qCMDE. qCMDE and IWMDE evaluate one scale formula: they are unavailable for `brma.mv()` models with component-specific scale formulas (a named `scale` list, e.g. one scale formula per random component), whose point-null hypotheses use `density_method = "KDE"`.

For an unrestricted alternative against $\theta = \theta_0$, the Savage-Dickey identity (Dickey 1971; Wagenmakers et al. 2010) gives $BF_{10} = p(\theta_0 \mid H_1)/p(\theta_0 \mid y, H_1)$ only when the null model's nuisance-parameter prior is the conditional prior induced by the unrestricted model at θ_0. Otherwise, fit the constrained model separately and compare marginal likelihoods, for example by bridge sampling.

Marginal-means hypotheses are specified on the fitted linear-predictor scale. Display transformations stored by `marginal_means()` do not transform hypothesis constants. Single-model hypotheses and model-averaged region hypotheses use averaged marginal draws. Model-averaged point-null hypotheses use alternative-conditioned draws to avoid null atoms in averaged marginals. For model-averaged objects, hypotheses that mix point and region events, and point hypotheses on more than one marginal mean (several levels of a factor, or levels and scalar marginal means of one parameter or of several), are not supported. Separate region statements may refer to different parameters, while each statement refers to the marginal means of one parameter. Single-model hypotheses use one common averaged posterior and may combine these events. Point hypotheses require the exact prior ordinate that BayesTools attaches to the marginal mean, as for fitted objects.

Value

A `BayesTools_hypothesis_BF` BayesTools table.

References

Chen M (1994). "Importance-weighted marginal Bayesian posterior density estimation." *Journal of the American Statistical Association*, **89**(427), 818–824. doi:10.1080/01621459.1994.10476815.

Dickey JM (1971). "The weighted likelihood ratio, linear hypotheses on normal location parameters." *The Annals of Mathematical Statistics*, 204–223. doi:10.1214/aoms/1177693507.

Gelfand AE, Smith AF, Lee T (1992). "Bayesian analysis of constrained parameter and truncated data problems using Gibbs sampling." *Journal of the American Statistical Association*, **87**(418), 523–532. doi:10.1080/01621459.1992.10475235.

Wagenmakers E, Lodewyckx T, Kuriyal H, Grasman R (2010). "Bayesian hypothesis testing for psychologists: A tutorial on the Savage–Dickey method." *Cognitive Psychology*, **60**(3), 158–189. doi:10.1016/j.cogpsych.2009.12.001.

See Also

density_diagnostics(), marginal_means(), plot.brma(), bridge_sampler.brma()

Examples

```
## Not run:
fit <- brma(
  yi      = c(0.12, 0.20, 0.05, 0.33, 0.18),
  sei     = c(0.10, 0.12, 0.09, 0.15, 0.11),
  measure = "GEN"
)

# Average-effect hypotheses
hypothesis(fit, "mu != 0 vs mu = 0")
hypothesis(fit, "mu > 0 vs mu = 0")
hypothesis(fit, "mu > 0 vs mu < 0")

# Heterogeneity hypothesis
hypothesis(fit, "tau = 0 vs tau > 0", component = "scale")

dat <- data.frame(
  yi    = c(0.12, 0.20, 0.05, 0.33, 0.18, 0.41),
  sei   = c(0.10, 0.12, 0.09, 0.15, 0.11, 0.14),
  group = factor(c("A", "A", "A", "B", "B", "B"))
)
fit_mod <- brma(
  yi      = yi,
  sei     = sei,
  mods    = ~ group,
  data    = dat,
  measure = "GEN"
)

# Marginal-mean hypothesis for a factor level
emm <- marginal_means(fit_mod, density_method = "qCMDE")
hypothesis(emm, "group[B] > 0 vs group[B] = 0")

## End(Not run)
```

`influence.brma` *Measure Influence for brma Objects*

Description

Computes DFFITS, Cook's distance, COVRATIO, and other influence diagnostics for a fitted brma object.

Usage

```
## S3 method for class 'brma'
influence(model, ...)
```

Arguments

`model`	a fitted brma object.
`...`	additional arguments (currently ignored).

Details

The `hat` column uses the design of each draw's member for ensembles whose members differ in their PET and PEESE terms, as described in `hatvalues.brma`.

Selection `brma.mv()` models are currently unavailable through this aggregate diagnostic. Standalone DFFITS and Cook's distance are available for normal outcomes without a weightfunction component.

For known-`V` models, the hat values require applying the inverse of the marginal covariance. DFFITS and Cook's distance use PSIS fixed-location moments and do not apply this inverse. A covariance that is too ill-conditioned for that (a reciprocal condition number of about `2e-10` or less in any posterior draw used) is unavailable: the call stops with an error of class `RoBMA_known_v_precision_unavailable`, whose parent class is `RoBMA_numerical_unavailable`, instead of returning a rounded number. The condition carries the draw positions (`rows`) and their `reciprocal_condition`.

Value

An object of class `"infl.brma"`, which corresponds to the structure of `metafor::influence` objects. It is a list containing:

`inf`	A data frame with columns: `rstudent`, `dffits`, `cook.d`, `cov.r`, `tau.del`, and `hat`, where available. The `tau.del` column is the PSIS leave-one-out posterior mean of the root mean square of total row-level heterogeneity over the remaining scale-model rows after each deletion, the root mean square over the observed design that `summary_heterogeneity` reports. It differs from the arithmetic mean of the observation-specific tau that 4.0.0 used, which is never larger. It is not a pooled log-scale design prediction.
`dfbs`	A data frame with DFBETAS values for the model coefficients.

Undefined COVRATIO and coefficient DFBETAS values are reported as NaN with an explanatory note. DFFITS returns NA when its leave-one-out posterior fitted-value SD is zero, without a separate note. For binomial and Poisson GLMMs the rstudent component is the descriptive normal approximation of rstudent.brma (and the DFFITS, Cook's distance, and hat columns are not available). For brma.mv() objects, influence returns only components with implemented estimate-unit targets. DFFITS and Cook's distance report influence on fixed-location fitted values $\mu = X\beta$; tau.del is currently deferred. With correlated known-V data, the returned PSIS influence components use conditional estimate deletion rather than independent-study deletion.

See Also

dffits.brma, cooks.distance.brma, covratio.brma, dfbetas.brma, hatvalues.brma

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- brma(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )
  fit <- add_loo(fit)

  inf <- influence(fit)
  print(inf)
}

## End(Not run)
```

---

interpret *Interpret brma Results*

Description

Creates a concise textual interpretation of fitted RoBMA brma objects.

Usage

```
interpret(object, ...)

## Default S3 method:
interpret(object, ...)
```

```
## S3 method for class 'brma'
interpret(
  object,
  output_measure = NULL,
  transform = NULL,
  conditional = FALSE,
  scope = "core",
  probs = c(0.025, 0.975),
  central = NULL,
  priors = FALSE,
  digits = 3,
  ...
)

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

Arguments

object	a fitted model object.
...	additional arguments passed to methods. The brma method reserves ...; unused arguments error, except deprecated output_scale, which is accepted with a warning.
output_measure	optional effect-size measure used for the pooled effect only. Supported conversion targets include "SMD", "COR", "ZCOR", and "OR" when the input scale allows conversion.
transform	optional display transformation. "EXP" exponentiates log-scale "OR", "RR", "HR", and "IRR" pooled effects; aliases for no transform are accepted.
conditional	whether to summarize conditional estimates for RoBMA product-space objects. Defaults to FALSE.
scope	character vector specifying sections to include. Use "core" for the default concise interpretation, "all" for all sections, or any of "components", "estimates", "moderators", "scale", and "bias".
probs	two quantiles used for credible intervals. Defaults to c(.025, .975).
central	whether estimates are described by posterior mean or median. Defaults to NULL, which uses the posterior mean, except for transform = "EXP" pooled effects where the posterior mode is used.
priors	whether to print prior distributions after the interpretation. Defaults to FALSE.
digits	number of digits after the decimal point.
x	an interpret.brma object.

Value

A character vector with class "interpret.brma". The normalized BayesTools interpretation records are stored in the "records" attribute; the brma method also stores "scope", "conditional", and optionally "priors".

Johnides2025 — *412 effect sizes from a meta-analysis of secondary benefits of family-based treatments by Johnides et al. (2025)*

Description

The data set contains Cohen's d effect sizes, standard errors, and study labels from a meta-analysis investigating the extent to which family-based treatments for children with mental health, physical health, and developmental disorders provide secondary benefits to the children's siblings and caregivers (Johnides et al. 2025).

Usage

```
Johnides2025
```

Format

A data.frame with 3 columns and 412 observations:

`study` Study label.

`d` Cohen's d effect size.

`se` Standard error of d.

References

Johnides BD, Borduin CM, Sheerin KM, Kuppens S (2025). "Secondary benefits of family member participation in treatments for childhood disorders: A multilevel meta-analytic review." *Psychological Bulletin*, **151**(1), 1–32. doi:10.1037/bul0000462.

known_v_factor — *Declare a diagonal-plus-factor sampling covariance*

Description

Constructs an exact known sampling covariance representation

$$V = \mathrm{diag}(d) + UU^{\mathsf{T}}.$$

The supplied diagonal and loading matrix define `V`; no numerical rank is inferred from a materialized covariance matrix. This representation permits likelihood implementations to use the declared factor dimension when that route is supported. Gaussian and selection models retain the same target as a conventional dense `V` input with the same source settings and, for best-p-value selection, publication groups.

With `selection_model(known_sampling_variance = "condition")`, the declared covariance defines the complete retained sampling-error vector, including both diagonal and factor variation.

With known_sampling_variance = "integrate" (the default), the full sampling error is integrated before selection normalization. The diagonal-plus-factor split is a numerical representation; it does not define which sampling sources are conditioned upon.

Ordinary covariance matrices, including results from metafor::vcalc(), can also be supplied directly. Covariance inputs do not specify publication groups. Only weight_rule = "best" uses publication groups; declare them separately with selection_model(weight_rule = "best", group = ...). The default product rule requires no publication grouping.

A declaration is not required to reach the factor routes. Dependency blocks whose correlation is constant on nested partitions of their rows, which is what compound symmetry and nested metafor::vcalc() type/obs structures produce, are recognized from an ordinary matrix and represented exactly as $\mathrm{diag}(d) + UU^{\mathsf{T}}$. Other blocks that are exactly a diagonal plus a few loading columns, such as covariance from overlapping samples, shared control arms, or partially shared raters, are recovered by a rank search that stops at the first successful candidate, the kernel rank cap, or the Ledermann bound. It finds the smallest rank reached by that search, without certifying the minimum possible rank. The recovered representation must reproduce the supplied entries to working precision; blocks that do not, such as autoregressive phi structures, keep the general route with the supplied entries. V remains the stored covariance either way. Use known_v_factor() to declare a factor that cannot be recovered from the matrix, or to name its sources.

Matrix inputs use base R's numerical symmetry tolerance. Accepted matrices are stored symmetrically by averaging corresponding off-diagonal entries, leaving the supplied R object and its diagonal unchanged. Positive semidefiniteness is judged from the standardized eigenvalues, with values inside the eigensolver's roundoff envelope treated as zero. Exactly singular designs are therefore accepted and factorized to their true rank, including metafor::vcalc() output with rho = 1, whose diagonal and off-diagonal reach working precision along different rounding paths. A singular covariance still requires model structure covering its null space; see brma.mv(). The stored covariance uses this symmetric representation and retains input type and row mapping; it does not include a separate copy of unequal input triangles.

Usage

```
known_v_factor(diagonal, loading)
```

Arguments

diagonal	finite non-negative numeric vector d.
loading	finite numeric matrix U with one row per element of diagonal. A zero-column matrix is allowed and represents a diagonal covariance.

Value

A RoBMA_known_v_factor object accepted as the V or V_new argument of multivariate-model interfaces.

`Kroupova2021` *881 estimates from 69 studies of a relationship between employment and educational outcomes collected by Kroupova et al. (2024)*

Description

The data set contains partial correlation coefficients, standard errors, study labels, sample sizes, type of the educational outcome, intensity of the employment, gender of the student population, study location, study design, whether the study controlled for endogeneity, and whether the study controlled for motivation. The original data set including additional variables and the publication are available from the project page. (Note that some standard errors and employment intensities are missing.)

Usage

```
Kroupova2021
```

Format

A data.frame with 11 columns and 881 observations:

`r` Partial correlation coefficient.

`se` Standard error of r.

`study` Study label.

`sample_size` Sample size.

`education_outcome` Type of educational outcome.

`employment_intensity` Employment intensity.

`students_gender` Gender composition of the student sample.

`location` Study location.

`design` Study design.

`endogenity_control` Whether endogeneity was controlled.

`motivation_control` Whether motivation was controlled.

Source

https://meta-analysis.cz/students/

References

Kroupova K, Havranek T, Irsova Z (2024). "Student employment and education: A meta-analysis." *Economics of Education Review*, **100**, 102539. doi:10.1016/j.econedurev.2024.102539.

`lines.zplot_brma` *Add Zplot Density Lines*

Description

Adds model-implied density lines to an existing zplot.

Usage

```
## S3 method for class 'zplot_brma'
lines(
  x,
  plot_type = "base",
  probs = c(0.025, 0.975),
  max_samples = if (inherits(x, "brma.mv")) 1000 else 10000,
  plot_ci = TRUE,
  extrapolate = FALSE,
  from = -6,
  to = 6,
  by = 0.05,
  length.out = NULL,
  col = "black",
  as_data = FALSE,
  integration_control = x[["zplot"]][["data"]][["integration_control"]],
  parallel = FALSE,
  cores = min(4, RoBMA.get_option("max_cores")),
  ...
)
```

Arguments

`x`	a zplot_brma object.
`plot_type`	graphics system: `"base"` or `"ggplot"`. Defaults to `"base"`.
`probs`	quantiles for credible intervals. Defaults to `c(.025, .975)`. The bands are pointwise Harrell-Davis quantiles of the posterior density draws (see Details).
`max_samples`	maximum posterior samples for density. Defaults to 1000 for `brma.mv` models and 10000 for other models. Use `Inf` to use all posterior samples. This plotting budget does not change the EDR and missing-study summaries stored by `as_zplot()`.
`plot_ci`	whether to show credible interval bands. Defaults to `TRUE`.
`extrapolate`	whether to remove bias adjustments. Defaults to `FALSE`.
`from, to`	z-value range for density. Defaults to `-6` and `6`.
`by`	step size for density points. Defaults to 0.05.
`length.out`	number of density points (alternative to `by`).

col — line color. Defaults to "black".

as_data — whether to return data instead of plotting. Defaults to FALSE.

integration_control — numerical integration settings created by set_selection_likelihood_control(). Defaults to the settings stored by as_zplot(); an explicit value overrides them for these lines. Legacy objects without stored settings require an explicit control, or recreation with as_zplot(), when their selected density needs integration.

parallel — whether to distribute multivariate selection-model density calculations across local workers. Defaults to FALSE. Only posterior draws are divided; the full publication event and integration settings are retained. Scalar models, Gaussian-reference-only curves, and EDR summaries remain serial.

cores — number of local workers when parallel = TRUE. Defaults to the smaller of 4 and RoBMA.get_option("max_cores"). One core or fewer posterior draws than workers uses serial computation. Parallel workers require matching installed package/native builds and share the configured selection-cache budget. Worker count is reported when parallel work starts.

... — additional graphical parameters.

Details

The density uses the conditioning_depth stored by as_zplot(). Recreate the zplot object to change the predictive target.

The line is the posterior mean of the density at each z value, and its credible band (plot_ci = TRUE, or y_lCI and y_uCI of the returned data) is formed by pointwise Harrell-Davis quantiles of the posterior density draws at the probabilities probs, computed by BayesTools::harrell_davis_quantile(). They estimate the same quantiles as the empirical quantiles of the draws but smooth the kinks that the empirical quantiles show along z. The bands remain pointwise, and, because every draw contributes, they can differ visibly from the empirical quantiles when few draws are used (max_samples below a few hundred). Near the edge of a point mass in the draws (e.g., draws of models that fix the plotted quantity in a model-averaged posterior) the band blends the point mass and the remaining draws, and extreme probabilities (e.g., probs = c(.0005, .9995)) are not estimated more accurately than by the empirical quantiles; see BayesTools::harrell_davis_quantile() for these caveats.

When extrapolate = FALSE, the density includes all bias adjustments (PET/PEESE regression, selection weights) representing the fitted model. When extrapolate = TRUE, bias adjustments are removed to show the hypothetical distribution without publication bias. Multivariate models, retained estimate-level or sampling-error sources, applicable integrated contexts, and best-rule selection models use a normalized pre-selection Gaussian reference. Released univariate product targets retain inverse-weight scaling and their extrapolated curve need not integrate to one.

Value

NULL invisibly for base graphics, ggplot2 layers for ggplot, or a data frame with columns x, y, y_lCI, y_uCI if as_data = TRUE.

See Also

plot.zplot_brma(), hist.zplot_brma()

---

logLik.brma *Extract Log-Likelihood Matrix from brma Object*

Description

Extract the pointwise log-likelihood matrix from a brma model object. This is an S x K or S x G matrix where S is the number of posterior samples, K is the number of estimates, and G is the number of clusters. This method implements the S3 'logLik' generic for brma objects and returns the matrix of pointwise log-likelihoods (one column per observation, one row per sample). It returns the draws of log_lik, which documents the targets of the columns; use log_lik() directly where the generic is not required.

Usage

```
## S3 method for class 'brma'
logLik(object, unit = "estimate", ...)
```

Arguments

object	a brma model object.
unit	output unit. See add_loo.
...	currently unused.

Details

The log-likelihood is computed for each observation at each posterior sample. For binomial and Poisson models, each observation consists of a pair of counts (ai/ci or x1i/x2i) that together define a single effect size estimate. The matrix holds posterior draws, not a scalar maximized log-likelihood, so AIC() and BIC() are not defined for brma objects (see AIC.brma).

Value

An S x K or S x G matrix of log-likelihood values.

See Also

log_lik, loo.brma

logml.brma *Log Marginal Likelihood for brma Objects*

Description

Extract the log marginal likelihood from a brma model. The marginal likelihood must first be computed using add_marglik.

Usage

```
## S3 method for class 'brma'
logml(x, ...)
```

Arguments

x	a brma model object.
...	additional arguments (currently not used).

Details

This function extracts the log marginal likelihood from the bridge sampling object that was previously computed and stored using add_marglik. Product-space model-averaging objects (BMA.norm, BMA.glmm, BMA.mv, RoBMA, and RoBMA.mv) do not expose bridge-sampling marginal likelihoods.

For repeated standardized results, this method returns the stored scalar median of the included log marginal likelihoods. Legacy raw bridge lists use the upstream scalar median method. Use bridge_sampler to extract the raw bridge repetitions, or bf and post_prob to compare models across repetitions.

Value

A scalar numeric value representing the log marginal likelihood.

See Also

add_marglik, bridge_sampler.brma, bf.brma, post_prob.brma

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- brma(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")

  fit <- add_marglik(fit)

  logml(fit)
}
```

```
## End(Not run)
```

`log_lik` *Extract Pointwise Log-Likelihood Draws*

Description

Extract posterior pointwise log-likelihood draws from a fitted `brma` object. The result is an $S \times K$ or $S \times G$ matrix, where S is the number of posterior draws, K is the number of estimates, and G is the number of clusters.

Usage

```
log_lik(object, ...)

## S3 method for class 'brma'
log_lik(object, unit = "estimate", ...)
```

Arguments

`object`	a brma model object.
`...`	currently unused.
`unit`	output unit. See `add_loo`.

Details

This function returns posterior log-likelihood draws, not a scalar maximized log-likelihood. The `logLik.brma` method returns the same matrix for the `stats::logLik()` generic; `AIC()` and `BIC()` are not defined for it.

The log-likelihood is computed for each target at each posterior sample. For binomial and Poisson models, each observation consists of a pair of counts (ai/ci or x1i/x2i) that together define a single effect size estimate. Estimate-unit columns condition on the retained observations in each declared dependency block. For Gaussian targets after integrating their declared local Gaussian sources, including correlated known-V `brma.mv()` and specialized multilevel normal models, these are Schur conditional scores $p(y_i \mid y_{-i}, \theta)$; their row sum generally is not the full joint likelihood. Likelihood powers and their normalizing constants are retained inside the local-effect integral for weighted Gaussian cluster models. The observation CDF used by residuals remains unpowered. Numerically unresolved retained moments raise `RoBMA_weighted_gaussian_unavailable`. Likelihood powers and retained-sampling context follow the declared target. Independent observations reduce to ordinary pointwise contributions. Cluster-unit columns delete each complete cluster jointly. Sampling-conditioned selection models retain the declared remaining sampling errors and random context for estimate deletion, including ordinary component branches and constant selection weights. Cluster deletion keeps its distinct joint target. Earlier affected estimate scores must be recomputed. Estimate-unit GLMM draws integrate the nuisance base rate or log-rate over its prior: point priors and untruncated beta (base rate) and normal (log-rate) priors adaptively, other priors on a refined

prior-CDF grid; a uniform(0, 1) base-rate prior is a Beta(1, 1) prior and is integrated adaptively. The adaptive Gauss-Hermite rules raise the quadrature order until successive orders agree within the tolerance; this agreement is an empirical refinement check, not an error bound. The grid is accepted only when its log-likelihood changes by less than `1e-6` between refinements. Any nuisance prior without an adaptive route (other than an untruncated beta base rate or an untruncated normal log-rate; truncated ones take the grid too) can fail to reach this, especially with few events or very large arms, and then stops with the error class `RoBMA_glmm_grid_nonconvergence` described below; gamma, lognormal, and inverse-gamma log-rate priors and lognormal base-rate priors are examples. A beta base-rate prior or a normal log-rate prior avoids it. The automatic computation after fitting (the `autocompute.loo` and `autocompute.waic` options) warns instead and returns the fit without LOO/WAIC. Numerical quadrature failures share class `RoBMA_glmm_numerical_error`. AGHQ failures have class `RoBMA_glmm_aghq_nonconvergence`; prior-CDF refinement failures retain `RoBMA_glmm_grid_nonconvergence`. Their fields identify the outcome family, observation, and available sample/refinement diagnostics. A beta baserate prior whose shapes are too extreme for the prior-CDF grid to be represented in double precision is unavailable and raises `RoBMA_glmm_prior_unavailable`, also of class `RoBMA_glmm_numerical_error`. Cluster-unit GLMM draws are unavailable until certified nested adaptive quadrature is implemented.

Value

An $S \times K$ or $S \times G$ matrix of pointwise log-likelihood draws with `RoBMA_target` metadata.

See Also

`loo.brma`, `add_waic`

`loo.brma` *LOO-PSIS for brma Objects*

Description

Extract the LOO-PSIS object from a brma model object. The LOO must first be computed using `add_loo`.

Usage

```
## S3 method for class 'brma'
loo(x, unit = "estimate", ...)
```

Arguments

`x`	a brma model object.
`unit`	output/deletion unit. See `add_loo`.
`...`	additional arguments (currently unused).

Details

This function extracts the LOO object that was previously computed and stored using `object <- add_loo(object, unit = unit)`. If LOO has not been computed for the requested unit, an error is thrown.

This is the RoBMA S3 generic and `brma` method. Use `loo` directly for raw log-likelihood arrays or matrices.

Value

An object of class `c("psis_loo", "loo")` as returned by `loo`.

See Also

`add_loo`, `loo`, `loo_compare`, `pareto_k_ids`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- bPET(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")
  fit <- add_loo(fit)

  loo_fit <- loo(fit)
  print(loo_fit)
}

## End(Not run)
```

---

loo_compare.brma *Compare brma Models Using LOO*

Description

Compare multiple brma models using LOO-PSIS cross-validation. This is a convenience wrapper around `loo_compare`.

Usage

```
## S3 method for class 'brma'
loo_compare(x, ..., unit = "estimate")
```

Arguments

`x`	a brma model object (the first model to compare).
`...`	additional brma model objects or `loo` objects to compare.
`unit`	output/deletion unit used when extracting LOO from brma objects.

Details

This function compares models based on their expected out-of-sample predictive performance (ELPD).

Important for model comparison: When comparing models via `loo_compare`, the selection is based on expected predictive performance for the documented deletion target, not in-sample fit. For dependent estimates, deleting one estimate may retain other observations from its fitted dependency block; this is not new-group prediction. RoBMA rejects comparisons with different outcome targets/data, `unit`, or retained context. A known sampling covariance without correlated blocks (e.g., `V = diag(sei^2)` in `brma.mv`) is the outcome data of its standard errors, whereas a correlated `V` is a different target. The same compatibility key is applied separately to LOO and WAIC; the two criteria cannot be mixed in one comparison table. Sampled versus marginalized local-effect provenance does not by itself block comparison.

Value

The comparison returned by `loo_compare` for the stored results: a data frame of class `"compare.loo"` with one row per model, ordered by expected predictive performance, which prints as loo prints it. The models are named `model1`, `model2`, ... unless the call names them.

See Also

`loo.brma`, `loo_compare`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")

  fit_bias <- RoBMA(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")
  fit_nobias <- BMA(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")

  fit_bias <- add_loo(fit_bias)
  fit_nobias <- add_loo(fit_nobias)

  loo_compare(fit_bias, fit_nobias)
  loo_compare(loo(fit_bias), loo(fit_nobias))
}

## End(Not run)
```

`loo_compare.loo` *Compare loo Objects Using LOO*

Description

Method for comparing RoBMA-targeted `loo` or `waic` objects directly.

Usage

```
## S3 method for class 'loo'
loo_compare(x, ..., unit = "estimate")
```

Arguments

x	a RoBMA-targeted loo or waic object (the first model to compare).
...	additional RoBMA-targeted loo/waic or brma objects to compare.
unit	output/deletion unit used when extracting LOO from brma objects.

Value

The comparison returned by loo_compare for the stored results: a data frame of class "compare.loo" with one row per model, ordered by expected predictive performance, which prints as loo prints it. The models are named model1, model2, ... unless the call names them.

See Also

loo.brma, loo_compare

---

loo_model_weights.brma

Predictive Model Weights for brma Objects

Description

Compute stacking or pseudo-BMA weights from LOO results already stored in multiple brma objects.

Usage

```
## S3 method for class 'brma'
loo_model_weights(
  x,
  ...,
  unit = "estimate",
  method = c("stacking", "pseudobma"),
  optim_method = "BFGS",
  optim_control = list(),
  BB = TRUE,
  BB_n = 1000,
  alpha = 1,
  r_eff_list = NULL,
  cores = getOption("mc.cores", 1)
)
```

Arguments

x	a brma model object (the first model to weight).
...	additional brma model objects or RoBMA-targeted psis_loo objects.
unit	output/deletion unit used when extracting stored LOO results from brma objects.
method	weighting method; "stacking" or "pseudobma".
optim_method, optim_control	optimization method and controls for stacking. See loo_model_weights.
BB, BB_n, alpha	Bayesian-bootstrap controls for pseudo-BMA weighting. See loo_model_weights.
r_eff_list	optional relative effective sample sizes. Retained for the upstream interface and ignored when stored psis_loo objects are supplied.
cores	number of cores passed to loo::loo_model_weights().

Details

Each brma object must first be updated with object <- add_loo(object, unit = unit). All models must use the same outcome data, deletion unit, and likelihood target. RoBMA validates these targets before delegating to loo::loo_model_weights().

Model names are derived from the supplied model expressions. See loo_model_weights for details about stacking, pseudo-BMA, and pseudo-BMA+.

Value

A named numeric vector containing one predictive weight per model.

See Also

add_loo, loo.brma, loo_compare.brma, loo_model_weights

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")

  fit_bias <- RoBMA(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")
  fit_nobias <- BMA(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")

  fit_bias   <- add_loo(fit_bias)
  fit_nobias <- add_loo(fit_nobias)

  loo_model_weights(fit_bias, fit_nobias)
}

## End(Not run)
```

`loo_weights.brma` *Extract Normalized PSIS Weights from brma Object*

Description

Extract the normalized Pareto smoothed importance sampling (PSIS) weights from a brma model object.

Usage

```
## S3 method for class 'brma'
loo_weights(object, unit = "estimate", ...)
```

Arguments

`object`	a brma model object.
`unit`	output/deletion unit. See `add_loo`.
`...`	currently unused.

Details

LOO must first be computed with `object <- add_loo(object, unit = unit)`. This method extracts the stored PSIS object and does not compute LOO. It does not compute weights across models. For predictive model weights, use `loo_model_weights(model_1, model_2)`; for a model comparison table, use `loo_compare(model_1, model_2)`.

When the score integrates local random effects, these weights reweight global-parameter draws. They must not be applied to full-data sampled local effects as though those were draws from the deletion posterior; local-effect calculations require the corresponding retained-data reconstruction.

Value

An `S` x `K` matrix for estimate-unit LOO, or `S` x `G` matrix for cluster-unit LOO, with posterior samples in rows and LOO targets in columns. Columns are normalized to sum to one.

See Also

`loo.brma`, `loo_compare.brma`, `loo_model_weights.brma`

`Lui2015` *18 studies of a relationship between acculturation mismatch and intergenerational cultural conflict collected by Lui (2015)*

Description

The data set contains correlation coefficients r, sample sizes n, and labels for each study assessing the relationship between acculturation mismatch (that is the result of the contrast between the collectivist cultures of Asian and Latin immigrant groups and the individualist culture in the United States) and intergenerational cultural conflict (Lui 2015) which was used as an example in Bartoš et al. (2022).

Usage

```
Lui2015
```

Format

A data.frame with 3 columns and 18 observations:

`r` Correlation coefficient.

`n` Sample size.

`study` Study label.

References

Bartoš F, Maier M, Quintana DS, Wagenmakers E (2022). "Adjusting for publication bias in JASP and R — Selection models, PET-PEESE, and robust Bayesian meta-analysis." *Advances in Methods and Practices in Psychological Science*, **5**(3), 1–19. doi:10.1177/25152459221109259.

Lui PP (2015). "Intergenerational cultural conflict, mental health, and educational outcomes among Asian and Latino/a Americans: Qualitative and meta-analytic review." *Psychological Bulletin*, **141**(2), 404–446. doi:10.1037/a0038449.

`ManyLabs16` *55 effect sizes from Many Labs 2 replication studies of Tversky and Kahneman (1981) framing effects*

Description

The data set contains standardized mean differences (y) and standard errors (se) from 55 replication studies of the framing effect originally described by Tversky and Kahneman (1981). These studies were part of the Many Labs 2 project examining variation in replicability across samples and settings Klein et al. (2018).

Usage

```
ManyLabs16
```

Format

A data.frame with 2 columns and 55 observations:

`y` Standardized mean difference.

`se` Standard error of y.

References

Klein RA, Vianello M, Hasselman F, Adams BG, Adams Jr RB, Alper S, Aveyard M, Axt JR, Babalola MT, Bahník Š, others (2018). "Many Labs 2: Investigating variation in replicability across samples and settings." *Advances in Methods and Practices in Psychological Science*, **1**(4), 443–490. doi:10.1177/2515245918810225.

Tversky A, Kahneman D (1981). "The framing of decisions and the psychology of choice." *Science*, **211**(4481), 453–458. doi:10.1126/science.7455683.

`marginal_means` *Estimated Marginal Means*

Description

S3 generic for estimated marginal means. The `brma` method works with fitted moderator models and stores the BayesTools marginal-inference object for `summary()` and `plot()`.

Usage

```
marginal_means(object, ...)
```

Arguments

`object`	a fitted model object.
`...`	additional arguments passed to methods.

Value

A method-specific estimated marginal means object.

See Also

`summary()`, `plot()`, `summary.brma()`, `regplot()`

`marginal_means.brma` *Estimated Marginal Means for brma Objects*

Description

Computes estimated marginal means for a fitted `brma` object with moderators using `BayesTools::as_marginal_inference(`. Alternative-conditioned marginal means condition on the union of every nonzero formula coefficient contributing to the corresponding output cell. A location intercept fixed at zero is omitted whenever moderators are present.

Usage

```
## S3 method for class 'brma'
marginal_means(
  object,
  null_hypothesis = 0,
  normal_approximation = FALSE,
  n_samples = 10000,
  output_measure = NULL,
  transform = NULL,
  bf = NULL,
  parameter = NULL,
  type = NULL,
  levels = NULL,
  density_method = c("KDE", "qCMDE", "IWMDE"),
  density_control = NULL,
  ...
)
```

Arguments

`object`	a fitted `brma` object with moderators.
`null_hypothesis`	point null hypothesis used for inclusion Bayes factors, specified on the fitted linear-predictor scale. Defaults to `0`. This remains on the fitted scale when `output_measure` or `transform` changes the displayed scale. A marginal mean whose Savage-Dickey Bayes factor at the null hypothesis is undefined (BayesTools refuses it with a condition of class `BayesTools_hypothesis_ordinate`: its posterior has a point mass at the null, or its prior density there is a point mass, zero (for example a null outside a truncated prior), infinite, undefined, or without an exact value) has an `NA` inclusion Bayes factor, whose reason the summary prints below the table; the other marginal means are computed. A point hypothesis on that mean with `hypothesis()` stops with the BayesTools class. Declared unavailable formula measures (class `BayesTools_formula_measure_unavailable`) likewise give an `NA` row with the original reason, while available siblings are computed.

normal_approximation
: whether the prior and posterior density at the null hypothesis should be approximated with a normal distribution (the BayesTools normal approximation of the Savage-Dickey density ratio) instead of a kernel density estimate. Defaults to FALSE. It requires density_method = "KDE".

n_samples
: number of samples/grid points used by BayesTools for marginal prior densities. Defaults to 10000.

output_measure
: effect-size measure for location/effect predictions. Defaults to the fitted measure. Supported conversions are among "SMD", "COR", "ZCOR", and "OR"; "RR", "HR", "IRR", "RD", and "GEN" can only be returned on their fitted measure. Use transform = "EXP" for ratio-scale output from log-scale measures.

transform
: optional display transformation. Currently "EXP" exponentiates log-scale measures "OR", "RR", "HR", and "IRR".

bf
: whether inclusion Bayes factors should be shown by default in summaries. Defaults to TRUE for RoBMA/BMA objects and FALSE for single-model brma objects. With qCMDE/IWMDE, setting bf = FALSE also skips point-ordinate precomputation; point hypotheses can still compute requested ordinates on demand. With KDE, summary(..., bf = TRUE) computes previously skipped inclusion Bayes factors from the stored conditional marginal posteriors.

parameter, type, levels
: optional selectors restricting qCMDE/IWMDE density precomputation to a marginal-means parameter, output type, or factor levels. parameter and levels also restrict the conditional ordinates used for inclusion Bayes factors. These ordinates are always computed from conditional marginal means, even when type excludes "conditional".

density_method
: posterior density method. "KDE" uses the standard BayesTools kernel density estimate. "qCMDE" attaches RoBMA row-normalized q-grid conditional densities. "IWMDE" attaches Chen-style moment-matched IWMDE densities for plotting. RoBMA stores separate precomputed posterior ordinates at null_hypothesis; these ordinates do not alter the plotting grid. qCMDE is preferred when its additional normalization cost is acceptable; IWMDE can be faster but is more sensitive to its fitted conditional weights. Matching is case-insensitive. Selected weightfunction models near zero cumulative ordered effects can take more than 10 minutes with qCMDE or IWMDE. Ordered effects use normalized conditional-source contributions for both methods, with their respective row budgets; regular exact offsets can require positive conditional blocks. Their numerical proof is reported separately from the chain MCSE. When an ordinate is used for a Bayes factor, BF_error is a conditional Monte Carlo error estimate. It excludes uncertainty in the prior ordinate and, for IWMDE, uncertainty from estimating the conditional weight function. Use density_diagnostics() on a subsequent hypothesis() result to inspect the ordinate diagnostics. IWMDE is unavailable for binomial and Poisson GLMMs because its high-dimensional conditional weights did not meet the bridge-sampling certification tolerance. Use qCMDE for GLMM density curves and Bayes factors.

density_control
: named list of density-estimation settings for precomputed qCMDE/IWMDE methods. It is rejected with an error when density_method is "KDE". Supported

entries are `n_points` (default `100`), `samples` for density curves and point ordinates, `target_relative_mcse` (default `0.05`), `display_grid` (default `"adaptive"`), `normalization_points` (default `NULL`, resolved to `max(50, n_points)`), and `normalization_prob` (default `0.999`; see `hypothesis()` for the per-row truncation it permits and the ordinate bound reported for it). `integration_control` (default `NULL`) retains the fitted selection-integration settings. Supply a control created by `set_selection_likelihood_control()` to change those settings for this post-fit calculation, for example `list(integration_control = set_selection_likelihood_control(max_points_per_scramble = 32768))`. This entry is available for Gaussian selection models with a fitted integration plan. The fitted object and posterior draws are unchanged. The maximum point budget controls factor QMC fallback; analytic and deterministic quadrature rules remain unchanged. `samples` defaults to `500` for qCMDE and `1000` for IWMDE density curves and point ordinates. Point ordinates use one fixed state-independent simple random sample chosen before contributions are evaluated. Sample diagnostics do not decide whether a finite ordinate is returned. An unmet relative-MCSE target produces a warning; increase `samples` or use `Inf` for the census. The normalization entries are used with `density_method = "qCMDE"` and `density_method = "IWMDE"`. Curve diagnostics apply local reliability gates over the empirical 5–95 percent bulk, report the 5 and 95 percent tail checkpoints, and retain a whole-curve absolute MCSE safeguard relative to the density peak. Point ordinates retain separate numerical-stability gates; their sample precision, effective-sample-size, and contribution-concentration checks are warnings. qCMDE/IWMDE are unavailable for non-known-`V` `brma.mv()` random-formula models, for `brma.mv()` models with component-specific scale formulas (a named `scale` list), and for derived semantic random-effect quantities. A qCMDE/IWMDE request that the fitted model does not support stops before any density is estimated with an error of class `RoBMA_density_method_unavailable` and one class naming the cause: `RoBMA_density_method_random_unknown_v` (`brma.mv()` random-formula models without known `V`), `RoBMA_density_method_scale_components` (component-specific scale formulas), or `RoBMA_density_method_glmm` (IWMDE for binomial and Poisson GLMMs). A density whose construction fails stops with an error of class `RoBMA_density_construction_error` (a `RoBMA_density_error`); use `density_method = "KDE"` to estimate it from the posterior draws. A fitted chain with no eligible row of the target adds the class `RoBMA_density_chain_coverage`, and the message names longer chains and their convergence as the other remedy.

`...` unused additional arguments. Supplied arguments trigger a warning.

Value

A list of class `marginal_means.brma`. Its stable fields are `inference`, `parameters`, `term_map`, `formula`, `null_hypothesis`, `normal_approximation`, `n_samples`, `conditional_rule`, `input_measure`, `effect_transform`, `model_averaged`, `density_method`, `point_ordinate_supported`, `bf`, `name`, and `source_object`. qCMDE/IWMDE objects additionally contain `density_diagnostics`, `ordinate_diagnostics`, `density_settings`, and `ordinate_settings`.

See Also

`summary()`, `plot()`, `summary.brma()`, `regplot()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE) &&
    requireNamespace("metafor", quietly = TRUE)) {
  data(dat.bcg, package = "metadat")
  dat <- metafor::escalc(
    measure = "RR",
    ai      = tpos,
    bi      = tneg,
    ci      = cpos,
    di      = cneg,
    data    = dat.bcg
  )

  fit <- brma(yi = yi, vi = vi, mods = ~ alloc, data = dat, measure = "RR")
  mm  <- marginal_means(fit)
  summary(mm)
  plot(mm, parameter = "alloc")
}

## End(Not run)
```

---

nobs.brma *Number of Observations for brma Objects*

Description

Extract the number of observations (studies/estimates) from a fitted brma object.

Usage

```
## S3 method for class 'brma'
nobs(object, ...)
```

Arguments

`object`	a fitted brma object
`...`	additional arguments (currently ignored)

Value

An integer giving the number of observations.

See Also

`brma()`, `brma.glmm()`

plot.brma *Plots brma Object*

Description

plot.brma visualizes posterior (and prior) distribution a brma object.

Usage

```
## S3 method for class 'brma'
plot(
  x,
  parameter,
  parameter_mods,
  parameter_scale,
  prior = FALSE,
  standardized_coefficients = FALSE,
  conditional = FALSE,
  output_measure = NULL,
  transform = NULL,
  plot_type = "base",
  dots_prior = NULL,
  density_method = c("KDE", "qCMDE", "IWMDE"),
  density_control = NULL,
  component = "auto",
  ...
)

## S3 method for class 'brma'
lines(
  x,
  parameter = NULL,
  parameter_mods = NULL,
  parameter_scale = NULL,
  prior = FALSE,
  standardized_coefficients = FALSE,
  conditional = FALSE,
  output_measure = NULL,
  transform = NULL,
  plot_type = "base",
  dots_prior = NULL,
  density_method = c("KDE", "qCMDE", "IWMDE"),
  density_control = NULL,
  component = "auto",
  ...
)
```

Arguments

x	a fitted brma, BMA, or RoBMA object.
parameter	a parameter to be plotted. Defaults to "mu" for the effect size, or to the meta-regression intercept when moderators are present. Additional options are "tau", "rho" for multilevel models, "PET", "PEESE", and "omega" or "weightfunction" for selection models. Use plot_pet_peese() for PET/PEESE regression plots. Factor terms select all coefficient cells; use a semantic selector such as "group[level]" to plot one cell. Structural point masses are retained. A quantity of the fitted model that is no model parameter (an inclusion indicator such as "mu_x_indicator", "bias_indicator", a weight-function coordinate such as "omega[1]", or a latent cluster effect) stops with the classes BayesTools_hypothesis_target and BayesTools_hypothesis_unavailable, with which hypothesis() refuses it. Ordered level cells retain their semantic marginal draws and prior metadata on the requested predictor scale, including conditional row alignment.
parameter_mods	legacy moderator selector. Prefer parameter with component = "mods". Use "intercept" for the adjusted effect in meta-regression models.
parameter_scale	legacy scale-regression selector. Prefer parameter with component = "scale". Use "intercept" for the heterogeneity intercept in location-scale models.
prior	whether prior distribution should be added to figure. Defaults to FALSE.
standardized_coefficients	whether to plot moderator and scale-regression coefficients on the standardized predictor scale. Defaults to FALSE.
conditional	whether to plot the conditional posterior distribution for RoBMA product-space objects. Defaults to FALSE.
output_measure	effect-size measure for location/effect predictions. Defaults to the fitted measure. Supported conversions are among "SMD", "COR", "ZCOR", and "OR"; "RR", "HR", "IRR", "RD", and "GEN" can only be returned on their fitted measure. Use transform = "EXP" for ratio-scale output from log-scale measures.
transform	optional plotting transformation. "EXP" exponentiates effect-size location and individual meta-regression coefficients for fitted log-scale OR, RR, HR, and IRR models. For scale-regression coefficients, "EXP" displays the multiplicative change in heterogeneity. "LOG" displays a positive heterogeneity intercept or random-effect variance or SD multiplier on the log scale. The transformation is applied to KDE and precomputed qCMDE/IWMDE densities with the corresponding Jacobian. For automatically unscaled scale-formula intercepts, use standardized_coefficients = TRUE with qCMDE/IWMDE.
plot_type	whether to use a base plot "base" or ggplot2 "ggplot" for plotting. Defaults to "base".
dots_prior	list of additional graphical arguments to be passed to the plotting function of the prior distribution. Supported arguments are lwd, lty, col, and col.fill, to adjust the line thickness, line type, line color, and fill color of the prior distribution respectively. For ordered predictor targets, omitted line colors and types match each posterior level's color with a dashed prior curve, including selected cells. For selected ordered cells with prior = TRUE, color/colour aliases resolve to

col, with priority col, color, then colour; linetype takes precedence over lty in both renderers. Explicit col, lty, or linetype values take precedence; the ordinary prior defaults are retained for other targets. Ordered metadata validation is restricted to the selected terms, including selected group members; unrelated ordered terms do not affect prior styles.

density_method posterior density method. "KDE" uses the standard BayesTools kernel density estimate. "qCMDE" attaches RoBMA row-normalized q-grid conditional densities. "IWMDE" attaches Chen-style moment-matched IWMDE densities. qCMDE is preferred when its additional normalization cost is acceptable; IWMDE can be faster but is more sensitive to its fitted conditional weights. Matching is case-insensitive. Ordered effects use the same normalized conditional-source estimator for both methods, with their respective posterior-row budgets. Selected weightfunction models near zero cumulative ordered effects can take more than 10 minutes with qCMDE or IWMDE. Exact regular ordered offsets that need a finite supplier integral use a positive conditional block; offsets classified as nonfinite or unsupported are omitted only at that exact value from a density curve. Point hypotheses retain their structural or precision refusal. For semantic random-effect quantities, qCMDE/IWMDE support direct scalar fitted sources, allocated component SDs backed by a scalar aggregate (tau_total or tau_common), and allocation proportions or multipliers backed by a fitted simplex coordinate. Other nonlinear derived quantities remain KDE-only. Shared inclusion gates are supported for aggregate and component SDs and allocation proportions. SDs retain their excluded zero branches; proportions condition on positive total heterogeneity. Allocated component SDs in a predictor-scaled block require standardized_coefficients = TRUE or density_method = 'KDE' for an original-scale request. This block-wide limitation also covers a centered intercept whose value happens to be unchanged by scaling. A positive point source multiplied by stochastic allocation weights has a continuous target law, but its source-held-weight chart cannot sweep that target; use density_method = 'KDE'. Source-zero and inclusion-gate atoms retain their declared locations. In independently gated allocations, realized tau_total/tau2_total and tau2_prop(...) are KDE-only because they combine multiple gate and allocation coordinates. Their structural point masses are displayed separately from the continuous density; tau2_prop(...) is conditioned on positive realized total heterogeneity. qCMDE/IWMDE are not available for non-known-V brma.mv() random-formula models, brma.mv() models with component-specific scale formulas (a named scale list), or selection-weightfunction coordinates requiring joint replacement. IWMDE is also unavailable for binomial and Poisson GLMMs; use qCMDE for GLMM density plots. qCMDE/IWMDE densities are evaluated on the fitted coefficient coordinate. If automatic predictor scaling changes the requested display coordinate, use standardized_coefficients = TRUE; RoBMA does not infer a coefficient transformation from posterior draws.

density_control

named list of density-estimation settings. Supported entries are n_points (default 100), samples (default 500 for qCMDE and 1000 for IWMDE density curves), target_relative_mcse (default 0.05), display_grid (default "adaptive"), normalization_points (default NULL, resolved to max(50, n_points)), and normalization_prob (default 0.999; see hypothesis() for the per-row trun-

cation it permits). integration_control (default NULL) retains the fitted selection-integration settings. Supply a control created by set_selection_likelihood_control() to change those settings for this post-fit calculation, for example list(integration_control = set_selection_likelihood_control(max_points_per_scramble = 32768)). This entry is available for Gaussian selection models with a fitted integration plan. The fitted object and posterior draws are unchanged. The maximum point budget controls factor QMC fallback; analytic and deterministic quadrature rules remain unchanged. samples controls the fixed posterior-row budget for the density curve. target_relative_mcse is a point-ordinate diagnostic target and does not alter this fixed-budget density plot. The normalization entries are used with density_method = "qCMDE" and density_method = "IWMDE". Curve diagnostics apply local relative-MCSE, effective-sample-size, and contribution-concentration gates over the empirical 5–95 percent bulk and record the 5 and 95 percent tail checkpoints. The entire display retains an absolute MCSE safeguard relative to the density peak. Requested point ordinates used for Bayes factors retain separate strict local diagnostics. Increase the row and normalization budgets and compare results when density diagnostics report low effective sample size, concentrated contributions, or unstable normalization.

component — parameter component. Defaults to "auto", which infers the component when possible. Use "mods" (alias "location"), "scale", "random", or "bias" to disambiguate terms used in multiple model components. The random component selects semantic standard deviation, correlation, and allocation parameters from brma.mv() random formulas. A parameter_mods or parameter_scale selection with another component stops with an error of class RoBMA_component_mismatch.

... — list of additional graphical arguments to be passed to the plotting function. Supported arguments are lwd, lty, col, col.fill, xlab, ylab, ylab2, main, xlim, ylim, and ylim2 to adjust the line thickness, line type, line color, fill color, x-label, density-axis label, probability-axis label, title, x-axis range, density-axis range, and probability-axis range respectively. Set ylim2 on the initial plot() call; subsequent base lines() calls reuse that probability mapping.

Details

A qCMDE/IWMDE request that the fitted model does not support stops before any density is estimated with an error of class RoBMA_density_method_unavailable and one class naming the cause: RoBMA_density_method_random_unknown_v (brma.mv() random-formula models without known V), RoBMA_density_method_scale_components (component-specific scale formulas), or RoBMA_density_method_glmm (IWMDE for binomial and Poisson GLMMs). Requests that are unavailable for the plotted quantity stop with the same parent class and RoBMA_density_method_conditional_random (conditional random-effect plots, which are KDE-only), RoBMA_density_method_random_target (a random-effect quantity without a supported scalar random-component coordinate), RoBMA_density_method_bias_target (a composite weightfunction target; use density_method = 'KDE'), or RoBMA_density_method_original_scale (a coefficient or factor cell whose original-scale value is not a linear combination of its fitted coefficients, such as the exponentiated intercept of a scale formula with standardized predictors; use standardized_coefficients = TRUE). A density whose construction fails stops with an error of class RoBMA_density_construction_error (a RoBMA_density_error); use density_method = "KDE" to estimate it from the posterior draws. A fitted chain with no eligible row of the target adds the class RoBMA_density_chain_coverage, and the message names longer chains and their con-

vergence as the other remedy. hypothesis() refuses qCMDE/IWMDE for the same causes with these classes after its own. plot_prior() stops with RoBMA_density_method_original_scale for the prior of a term with several fitted coordinates whose original-scale coordinates the standardization changes (see there).

A returned qCMDE/IWMDE estimate that fails the density availability checks raises a RoBMA_density_plot_error identifying the plotted parameter. Its density_diagnostics field contains the returned diagnostic records, including computed density values and selected rows, without executable estimate plans that could retain the fitted object.

lines.brma() adds the posterior density to an existing base plot. Omit add: this method always adds, and supplying add triggers an error. With plot_type = "ggplot", it returns ggplot2 layer(s) that can be added to a plot.brma(..., plot_type = "ggplot") object with +. For base plots containing point masses, the initial plot.brma() call establishes the secondary probability axis and lines.brma() reuses it across fitted objects. An overlaid point mass outside the initial ylim2 is clipped with a warning.

Value

plot.brma returns either NULL if plot_type = "base" or a ggplot2 object (a named list of plots for a whole factor term) if plot_type = "ggplot". lines.brma returns NULL for plot_type = "base" and ggplot2 layer(s) for plot_type = "ggplot".

See Also

RoBMA()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- bPET(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")

  plot(fit, parameter = "mu")
  lines(fit, parameter = "mu", col = "blue", lwd = 2)

  ggplot_fit <- plot(fit, parameter = "mu", plot_type = "ggplot")
  ggplot_fit + lines(fit, parameter = "mu", plot_type = "ggplot", col = "blue")

  plot(fit, parameter = "tau", prior = TRUE)
  plot(fit, parameter = "PET")
}

## End(Not run)
```

plot.marginal_means.brma *Plot Estimated Marginal Means*

Description

Plots estimated marginal means stored in a `marginal_means.brma` object using `BayesTools::plot_marginal()`.

Usage

```
## S3 method for class 'marginal_means.brma'
plot(
  x,
  parameter,
  type = NULL,
  prior = FALSE,
  plot_type = "base",
  dots_prior = NULL,
  output_measure = NULL,
  transform = NULL,
  density_method = NULL,
  density_control = NULL,
  add = FALSE,
  ...
)

## S3 method for class 'marginal_means.brma'
lines(x, parameter, prior = FALSE, ...)
```

Arguments

`x`	a `marginal_means.brma` object.
`parameter`	moderator term to plot. Use the original term name, for example `"measure"`, `"intercept"` for the intercept when available, `"mu"` as an intercept alias, or the internal parameter name, for example `"mu_measure"`.
`type`	for RoBMA product-space objects, whether to plot model-averaged (`"averaged"`) or conditional (`"conditional"`) marginal means. Defaults to `"averaged"` and is available only for RoBMA marginal means.
`prior`	whether the marginal prior distribution should be added to the plot. Defaults to `FALSE`.
`plot_type`	whether to use base R graphics (`"base"`) or ggplot2 (`"ggplot"`). Defaults to `"base"`.
`dots_prior`	list of additional graphical arguments passed to the prior plotting function.

`output_measure` effect-size measure for location/effect predictions. Defaults to the fitted measure. Supported conversions are among `"SMD"`, `"COR"`, `"ZCOR"`, and `"OR"`; `"RR"`, `"HR"`, `"IRR"`, `"RD"`, and `"GEN"` can only be returned on their fitted measure. Use `transform = "EXP"` for ratio-scale output from log-scale measures.

`transform` optional display transformation. Currently `"EXP"` exponentiates log-scale measures `"OR"`, `"RR"`, `"HR"`, and `"IRR"`.

`density_method` posterior density method. The default `NULL` reuses the method stored on `x`; specify `"KDE"` to override a stored qCMDE/IWMDE method. `"qCMDE"` and `"IWMDE"` compute any missing densities for the plotted marginal means before plotting and never silently mix with KDE. Plot-time qCMDE/IWMDE computation can be slow and is used only for the current plot call; precompute with `marginal_means(..., density_method = "qCMDE")` or `marginal_means(..., density_method = "IWMDE")` to reuse densities across plots. Prefer qCMDE for likelihood-aware final results; IWMDE can be faster but is more sensitive to its fitted conditional weights. A qCMDE/IWMDE calculation for selected weight-function models near zero cumulative ordered effects can take more than 10 minutes. A qCMDE/IWMDE request that the source model does not support stops before any density is estimated with an error of class `RoBMA_density_method_unavailable` and the class naming its cause (see `marginal_means.brma()`).

`density_control`
named list of qCMDE/IWMDE density-estimation settings. Supported entries are `n_points` (default `100`), `samples` (default `500` for qCMDE and `1000` for IWMDE density curves), `target_relative_mcse` (default `0.05`), `display_grid` (default `"adaptive"`), `normalization_points` (default `NULL`, resolved to `max(50, n_points)`), and `normalization_prob` (default `0.999`; see `hypothesis()` for the per-row truncation it permits). `integration_control` (default `NULL`) accepts `set_selection_likelihood_control()` settings for Gaussian selection models; see `marginal_means.brma()` for an example. The fitted object and posterior draws are unchanged. `samples` controls the fixed posterior-row budget for the density curve. `target_relative_mcse` is a point-ordinate diagnostic target and does not alter this fixed-budget plot. Curve diagnostics use the empirical 5–95 percent bulk, report the 5 and 95 percent tail checkpoints, and retain a whole-curve absolute MCSE safeguard relative to the density peak. Supplying `density_control` forces recomputation for the plotted marginal means instead of reusing stored densities. Increase row and normalization budgets when diagnostics report concentrated contributions or unstable normalization.

`add` whether to add the densities to an existing plot. This is used by `lines.marginal_means.brma()` and defaults to `FALSE`.

`...` additional graphical arguments passed to `BayesTools::plot_marginal()`.

Details

`lines.marginal_means.brma()` adds posterior densities to an existing base plot. With `plot_type = "ggplot"`, it returns ggplot2 layer(s) that can be added to a marginal-means plot with `+`.

Value

`NULL` invisibly if `plot_type = "base"` or a ggplot object if `plot_type = "ggplot"`.

plot.zplot_brma *Plot Zplot Results*

Description

Plots a zplot visualization showing the histogram of observed z-statistics overlaid with model-implied densities.

Usage

```
## S3 method for class 'zplot_brma'
plot(
  x,
  plot_type = "base",
  probs = c(0.025, 0.975),
  max_samples = if (inherits(x, "brma.mv")) 1000 else 10000,
  plot_fit = TRUE,
  plot_extrapolation = FALSE,
  plot_ci = TRUE,
  plot_thresholds = TRUE,
  from = -6,
  to = 6,
  by.hist = 0.5,
  length.out.hist = NULL,
  by.lines = 0.05,
  length.out.lines = NULL,
  dots_hist = NULL,
  dots_fit = NULL,
  dots_extrapolation = NULL,
  dots_thresholds = NULL,
  integration_control = x[["zplot"]][["data"]][["integration_control"]],
  parallel = FALSE,
  cores = min(4, RoBMA.get_option("max_cores")),
  ...
)
```

Arguments

x	a zplot_brma object.
plot_type	graphics system: "base" or "ggplot". Defaults to "base".
probs	quantiles for credible intervals. Defaults to c(.025, .975). The bands are pointwise Harrell-Davis quantiles of the posterior density draws (see Details).
max_samples	maximum posterior samples for density estimation. Defaults to 1000 for brma.mv models and 10000 for other models. Use Inf to use all posterior samples. This plotting budget does not change the EDR and missing-study summary budget in as_zplot().

`plot_fit` whether to show fitted density (with bias adjustments). Defaults to `TRUE`.

`plot_extrapolation` whether to show extrapolated density (bias removed). Defaults to `FALSE`.

`plot_ci` whether to show credible interval bands. Defaults to `TRUE`.

`plot_thresholds` whether to show significance threshold lines. Defaults to `TRUE`.

`from, to` z-value range for plotting. Defaults to `-6` and `6`.

`by.hist` bin width for histogram. Defaults to 0.5.

`length.out.hist` number of histogram bins (alternative to `by.hist`).

`by.lines` step size for density lines. Defaults to 0.05.

`length.out.lines` number of density points (alternative to `by.lines`).

`dots_hist` graphical parameters for histogram (list).

`dots_fit` graphical parameters for fit lines (list).

`dots_extrapolation` graphical parameters for extrapolation lines (list).

`dots_thresholds` graphical parameters for threshold lines (list).

`integration_control` numerical integration settings created by `set_selection_likelihood_control()`. Defaults to the settings stored by `as_zplot()`; an explicit value overrides them for this plot. Legacy objects without stored settings require an explicit control, or recreation with `as_zplot()`, when their selected density needs integration.

`parallel` whether to distribute multivariate selection-model density calculations across local workers. Defaults to `FALSE`. Only posterior draws are divided; the full publication event and integration settings are retained. Scalar models, Gaussian-reference-only curves, and EDR summaries remain serial.

`cores` number of local workers when `parallel = TRUE`. Defaults to the smaller of 4 and `RoBMA.get_option("max_cores")`. One core or fewer posterior draws than workers uses serial computation. Parallel workers require matching installed package/native builds and share the configured selection-cache budget. Worker count is reported when parallel work starts.

`...` additional graphical parameters passed to components, such as `xlim` and `ylim` for the displayed axis ranges.

Details

All model-implied curves use the `conditioning_depth` stored by `as_zplot()`. Recreate the zplot object to change the predictive target.

The curves are posterior means of the density at each z value. Their credible bands (`plot_ci = TRUE`) are pointwise Harrell-Davis quantiles of the posterior density draws at the probabilities `probs`, computed by `BayesTools::harrell_davis_quantile()`. They estimate the same quantiles as the empirical quantiles of the draws but smooth the kinks that the empirical quantiles show

along z. The bands remain pointwise, and, because every draw contributes, they can differ visibly from the empirical quantiles when few draws are used (`max_samples` below a few hundred). Near the edge of a point mass in the draws (e.g., draws of models that fix the plotted quantity in a model-averaged posterior) the band blends the point mass and the remaining draws, and extreme probabilities (e.g., `probs = c(.0005, .9995)`) are not estimated more accurately than by the empirical quantiles; see `BayesTools::harrell_davis_quantile()` for these caveats.

The plot can display two density curves:

Fit (black) Model-implied density including publication bias adjustments. This represents the expected distribution of z-statistics given the estimated selection process. Shown by default.

Extrapolation (blue) The bias-adjusted reference without selective reporting. Multivariate models, retained estimate-level or sampling-error sources, applicable integrated contexts, and best-rule selection models use the normalized pre-selection Gaussian marginal. Released univariate product targets retain their inverse-weight scaling convention. Shown only when `plot_extrapolation = TRUE`, since the extrapolated curve rescales the display by an inferred selection convention rather than describing the observed literature.

Value

`NULL` invisibly for base graphics, or a ggplot2 object.

See Also

`hist.zplot_brma()`, `lines.zplot_brma()`, `summary.zplot_brma()`

---

plot_diagnostic *Plot MCMC Diagnostics*

Description

`plot_diagnostic` creates visual MCMC diagnostics for a fitted brma object. Convenience wrappers are available for trace, density, and autocorrelation plots.

Usage

```
plot_diagnostic(x, ...)

## S3 method for class 'brma'
plot_diagnostic(
  x,
  parameter = NULL,
  parameter_mods = NULL,
  parameter_scale = NULL,
  type,
  plot_type = "base",
  lags = 30,
  component = "auto",
```

```
  ...
)

plot_diagnostic_autocorrelation(x, ...)

## S3 method for class 'brma'
plot_diagnostic_autocorrelation(
  x,
  parameter = NULL,
  parameter_mods = NULL,
  parameter_scale = NULL,
  type = "autocorrelation",
  plot_type = "base",
  lags = 30,
  component = "auto",
  ...
)

plot_diagnostic_trace(x, ...)

## S3 method for class 'brma'
plot_diagnostic_trace(
  x,
  parameter = NULL,
  parameter_mods = NULL,
  parameter_scale = NULL,
  type = "trace",
  plot_type = "base",
  lags = 30,
  component = "auto",
  ...
)

plot_diagnostic_density(x, ...)

## S3 method for class 'brma'
plot_diagnostic_density(
  x,
  parameter = NULL,
  parameter_mods = NULL,
  parameter_scale = NULL,
  type = "density",
  plot_type = "base",
  lags = 30,
  component = "auto",
  ...
)
```

Arguments

x	a fitted brma object
...	additional graphical arguments passed through RoBMA's diagnostic setup to BayesTools::JAGS_diagnostics().
parameter	base parameter to plot. Defaults to NULL, which uses "mu" or the meta-regression intercept. Valid values include "mu", "tau", "rho", "PET", "PEESE", and "omega" or "weightfunction" when present. A quantity of the fitted model that is no model parameter (e.g. an inclusion indicator or a weight-function coordinate such as "omega[1]") stops with the classes BayesTools_hypothesis_target and BayesTools_hypothesis_unavailable, as in plot.brma().
parameter_mods	legacy moderator selector. Prefer parameter with component = "mods".
parameter_scale	legacy scale-regression selector. Prefer parameter with component = "scale".
type	diagnostic plot type. Convenience wrappers set a type-specific default but still forward this argument to plot_diagnostic.brma().
plot_type	whether to use a base plot "base" or ggplot2 "ggplot" for plotting. Defaults to "base".
lags	number of lags for autocorrelation plots. Defaults to 30.
component	parameter component. Defaults to "auto", which infers the component when possible. Use "mods" (alias "location"), "scale", "random", or "bias" to disambiguate terms used in multiple model components. The random component selects semantic standard deviation, correlation, and allocation parameters from brma.mv() random formulas. A parameter_mods or parameter_scale selection with another component stops with an error of class RoBMA_component_mismatch.

Value

plot_diagnostic returns the object returned by BayesTools::JAGS_diagnostics(), including a list when several coordinate plots are produced, invisibly for base graphics.

See Also

summary.brma()

---

plot_pet_peese *Plot PET-PEESE Fit of brma Object*

Description

plot_pet_peese visualizes posterior (and prior) PET-PEESE fit of a brma object.

Usage

```
plot_pet_peese(x, ...)

## S3 method for class 'brma'
plot_pet_peese(
  x,
  show_data = TRUE,
  prior = FALSE,
  plot_type = "base",
  dots_prior = NULL,
  ...
)
```

Arguments

x	a fitted brma object with a PET or PEESE component.
...	list of additional graphical arguments to be passed to the plotting function. Supported arguments are lwd, lty, col, col.fill, xlab, ylab, main, xlim, ylim, pch, bg, cex, and size to adjust the line thickness, line type, line color, fill color, x-label, y-label, title, x-axis range, y-axis range, and observed data point style respectively.
show_data	whether the observed effect sizes should be added to figure. Defaults to TRUE.
prior	whether prior distribution should be added to figure. Defaults to FALSE.
plot_type	whether to use a base plot "base" or ggplot2 "ggplot" for plotting. Defaults to "base".
dots_prior	list of additional graphical arguments to be passed to the plotting function of the prior distribution. Supported arguments are lwd, lty, col, and col.fill, to adjust the line thickness, line type, line color, and fill color of the prior distribution respectively.

Details

The plot shows observed yi values against sei. PET regression uses $\mu + PET \cdot se_i$; PEESE regression uses $\mu + PEESE \cdot se_i^2$, with the fitted effect direction. Models with moderators are not supported because the fitted curve depends on the moderator values at which it is evaluated.

Value

plot_pet_peese returns either NULL invisibly if plot_type = "base" or a ggplot2 object if plot_type = "ggplot".

See Also

RoBMA()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")

  fit <- bPET(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )

  plot_pet_peese(fit)
}

## End(Not run)
```

---

plot_prior *Plot Prior Distributions*

Description

plot_prior visualizes prior distributions stored in brma, BMA, and RoBMA objects. This is especially useful for objects created with only_priors = TRUE.

Usage

```
plot_prior(x, ...)

## S3 method for class 'prior'
plot_prior(x, plot_type = "base", ...)

## S3 method for class 'brma'
plot_prior(
  x,
  parameter = "mu",
  parameter_mods,
  parameter_scale,
  standardized_coefficients = TRUE,
  output_measure = NULL,
  transform = NULL,
  plot_type = "base",
  component = "auto",
  ...
)
```

Arguments

x	a brma, BMA, or RoBMA object, or a prior distribution object.
...	additional arguments passed to the prior plotting method. For mixed continuous and point priors, ylim controls the density axis, ylim2 controls the probability-mass axis, and ylab2 controls its label.
plot_type	whether to use a base plot "base" or ggplot2 "ggplot" for plotting. Defaults to "base".
parameter	character. Base parameter to plot. Defaults to "mu". Common options are "mu", "tau", "rho", "PET", "PEESE", "omega", and "bias", with aliases "effect" = "mu", "heterogeneity" = "tau", and "weightfunction" = "omega". "bias" plots only non-mixed or homogeneous bias priors; for mixed weightfunction and PET/PEESE mixtures use "omega", "PET", or "PEESE". Moderator and scale terms can also be selected by name when unambiguous. A character vector requests multiple base parameters. A quantity of the fitted model that is no model parameter (e.g. an inclusion indicator or a weight-function coordinate such as "omega[1]") stops with the classes BayesTools_hypothesis_target and BayesTools_hypothesis_unavailable, as in plot.brma().
parameter_mods	legacy moderator selector. Prefer parameter with component = "mods". Use "intercept" for the adjusted effect in meta-regression models.
parameter_scale	legacy scale-regression selector. Prefer parameter with component = "scale". Use "intercept" for the heterogeneity intercept in location-scale models.
standardized_coefficients	whether to plot moderator and scale-regression priors on the standardized predictor scale. Defaults to TRUE, which shows the priors as specified. Set to FALSE to transform them to the original predictor scale when continuous predictors were standardized (see Details for terms without an original-scale prior).
output_measure	effect-size measure for location/effect predictions. Defaults to the fitted measure. Supported conversions are among "SMD", "COR", "ZCOR", and "OR"; "RR", "HR", "IRR", "RD", and "GEN" can only be returned on their fitted measure. Use transform = "EXP" for ratio-scale output from log-scale measures.
transform	optional display transformation. Currently "EXP" exponentiates log-scale measures "OR", "RR", "HR", and "IRR".
component	parameter component. Defaults to "auto", which infers the component when possible. Use "mods" (alias "location"), "scale", "bias", or "random" (the random-effect quantities of fitted objects) to disambiguate terms used in multiple prior components. A parameter_mods or parameter_scale selection with another component stops with an error of class RoBMA_component_mismatch.

Details

output_measure and transform transform the prior plotting scale only for effect-size location priors ("mu" or the meta-regression intercept). For base plots, the probability mapping created by plot_prior() is reused by subsequent posterior lines.brma() calls, including overlays from other fitted objects. PET and PEESE priors, single or mixed, show bias-only effect-versus-standard-error curves by default, with a zero location contribution, the effect direction of the object (override

it with effect_direction), and the declared model weights. Use individual = TRUE for coefficient densities, or supply prior_list_mu to request full regression curves with explicit location priors, as supported by BayesTools::plot_prior_list().

The random-effect quantities of fitted brma.mv, BMA.mv, and RoBMA.mv objects are selected by their names in parameter, such as "tau", "tau_total", "tau2", "tau2_prop(study)", or "study:tau" ("tau" stops as ambiguous when several blocks have an SD). A quantity that is one declared prior (the SD of a block, the total SD of a variance allocation, or a correlation with its own prior) shows that prior. Any other quantity (a variance, a share of the total variance, the SD of a component of an allocation, and the quantities of gated allocations) shows the prior law it follows from the declared priors, the density that plot() overlays with prior = TRUE, drawn without a posterior; point masses (for example of inclusion gates) are drawn as arrows. For these laws, xlim, n_points, par_name, and graphical arguments apply as for other priors, while x_seq and x_range_quant stop. A quantity without a prior density stops with the classes BayesTools_hypothesis_target and BayesTools_hypothesis_unavailable. standardized_coefficients selects the fitted (default) or the original predictor scale of these laws. Objects created with only_priors = TRUE have no fitted random-effect quantities; print their random-effect priors with print_prior(x, parameter = "random").

Inconsistent random-allocation parent metadata stop with RoBMA_random_prior_derivation_error; cycles additionally carry RoBMA_random_allocation_cycle, and unknown parents carry RoBMA_random_allocation_pare
A parent chain beyond the supported depth of 16 allocations carries RoBMA_random_allocation_depth_unavailable. The condition retains its reason and visited allocation_chain. Other source errors propagate with their original classes and causes.

A term with several fitted coordinates (a factor with three or more levels or with independent contrasts, and the interactions of such a factor) has an original-scale prior only when the standardization of continuous predictors leaves its coordinates unchanged. With standardized_coefficients = FALSE, the prior of such a term whose coordinates the standardization changes (a factor or factor-by-factor interaction next to its interaction with a standardized predictor, and that interaction term) stops with an error of class RoBMA_density_method_original_scale and its parent RoBMA_density_method_unavailable; use standardized_coefficients = TRUE. The other terms (including factors and factor-by-factor interactions without an interaction with a standardized predictor) are plotted as before.

Value

plot_prior returns the selected prior plotting method's result, invisibly for base plots (which may include plot-data metadata), or a ggplot2 object for plot_type = "ggplot". If multiple parameters are requested, a named list is returned, invisibly for base plots.

See Also

BMA() RoBMA() brma() prior()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")

  priors <- BMA(
    yi           = yi,
```

```
    vi           = vi,
    mods         = ~ Preregistered,
    data         = dat.lehmann2018,
    measure      = "SMD",
    only_priors  = TRUE
  )

  plot_prior(priors, parameter = "mu")
  plot_prior(priors, parameter = "tau")
  plot_prior(priors, parameter = "Preregistered", component = "mods")
}

## End(Not run)
```

`plot_weightfunction` *Plots Weight Function of brma Object*

Description

`plot_weightfunction.brma` visualizes the posterior (and optionally prior) publication-bias weight function of a brma object.

Usage

```
plot_weightfunction(x, ...)

## S3 method for class 'brma'
plot_weightfunction(
  x,
  rescale_p_values = TRUE,
  prior = FALSE,
  plot_type = "base",
  show_data = TRUE,
  dots_data = NULL,
  dots_prior = NULL,
  ...
)
```

Arguments

`x`	a fitted `brma` object with a weightfunction/selection component, such as `bselmodel()` or a `RoBMA()` object with weightfunction priors.
`...`	list of additional graphical arguments to be passed to the plotting function. Supported arguments are `lwd`, `lty`, `col`, `col.fill`, `xlab`, `ylab`, `main`, `xlim`, `ylim` to adjust the line thickness, line type, line color, fill color, x-label, y-label, title, x-axis range, and y-axis range respectively.

`rescale_p_values`
: whether to rescale p-values to the interval shown by the weightfunction plot. Defaults to `TRUE`.

`prior`
: whether prior distribution should be added to figure. Defaults to `FALSE`.

`plot_type`
: whether to use a base plot `"base"` or ggplot2 `"ggplot"` for plotting. Defaults to `"base"`.

`show_data`
: whether observed p-values should be shown as rug marks on the weightfunction axis, using its one-sided or two-sided convention. Defaults to `TRUE`.

`dots_data`
: list of additional graphical arguments for observed p-value rug marks. Supported arguments include `col`/`color`, `alpha`, `lwd`/`linewidth`/`size`, `side`/`rug_side`, and `height`/`rug_height`/`ticksize`.

`dots_prior`
: list of additional graphical arguments to be passed to the plotting function of the prior distribution. Supported arguments are `lwd`, `lty`, `col`, and `col.fill`, to adjust the line thickness, line type, line color, and fill color of the prior distribution respectively.

Value

`plot_weightfunction.brma` returns either `NULL` if `plot_type = "base"` or a `ggplot2` object if `plot_type = "ggplot"`. The method errors for fitted objects without a weightfunction component.

See Also

`RoBMA()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- bselmodel(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )

  plot_weightfunction(fit)
  plot_weightfunction(fit, prior = TRUE)
}

## End(Not run)
```

pooled_effect *Pooled Effect Size*

Description

Computes the pooled (aggregated) effect size estimate from a fitted model.

Usage

```
pooled_effect(object, ...)
```

Arguments

object	a fitted model object
...	additional arguments passed to methods

Value

Method-specific return value, typically a summary table or posterior samples of the pooled effect size.

See Also

predict.brma()

pooled_effect.brma *Pooled Effect Size for brma Objects*

Description

Computes the pooled effect and prediction interval at the average expanded moderator design of a fitted brma object.

Usage

```
## S3 method for class 'brma'
pooled_effect(
  object,
  bias_adjusted = TRUE,
  output_measure = NULL,
  transform = NULL,
  probs = c(0.025, 0.975),
  conditional = FALSE,
  ...
)
```

Arguments

object	a fitted brma object
bias_adjusted	whether to adjust for publication bias. Defaults to TRUE. For PET/PEESE models this removes the regression bias term from the pooled location effect. This reference declares no new-publication standard error or selection event; explicit unadjusted predict() rows instead target a selected new publication. Selection-model weighting affects response predictions, not this fixed-location posterior summary.
output_measure	effect-size measure for location/effect predictions. Defaults to the fitted measure. Supported conversions are among "SMD", "COR", "ZCOR", and "OR"; "RR", "HR", "IRR", "RD", and "GEN" can only be returned on their fitted measure. Use transform = "EXP" for ratio-scale output from log-scale measures.
transform	optional display transformation. Currently "EXP" exponentiates log-scale measures "OR", "RR", "HR", and "IRR".
probs	quantiles of the posterior distribution to be displayed. Defaults to c(.025, .975) for 95% credible intervals.
conditional	whether to return the pooled effect conditional on the effect component for RoBMA product-space objects. Defaults to FALSE.
...	additional arguments passed to predict.brma (as_measure, V_new, and conditioning_depth); wrapper arguments such as newdata, type, and quiet are fixed (supplying one is R's error for an argument matched twice, as for predict()), and other unknown arguments are unused and trigger a warning.

Details

This function is a convenience wrapper around predict.brma(...,type = "terms", newdata = TRUE, bias_adjusted = TRUE, quiet = TRUE) that adds the prediction interval columns.

The fixed-location model is evaluated at the average expanded moderator design. Because the location predictor is linear in its design, this equals averaging its fitted-design posterior draws directly.

For meta-regression models, the pooled effect averages the effect size estimate across moderator levels proportionately to the levels observed in the data. This provides an estimate representative of the sample of studies.

For brma.mv() models and models with factor-specific or random-slope terms, the average expanded design can be synthetic and need not correspond to a single realizable outcome or study. The pooled effect and its prediction interval then describe that linear composite and should be interpreted only when the composite has a meaningful scientific interpretation. Use predict(..., newdata = ...) for a specified realizable design.

For models without moderators, this returns the single mu parameter. The prediction interval is marginal over one new true effect at the same average location and random-effect design, with the scale given by the mean of the observation-specific tau, as in pooled_heterogeneity(). It therefore uses pooled_heterogeneity(component = "total"), rather than the observed-design RMS heterogeneity reported by summary_heterogeneity(). For known-R random effects this is a unit-kernel reference interval; use predict() for an interval tied to a fitted group and its known covariance multiplier.

Value

A `brma_samples` object containing posterior samples. When printed, displays a one-row summary table whose `PI` columns contain posterior prediction quantiles. They are computed exactly from the posterior draws of the pooled location and heterogeneity, as quantiles of the normal mixture over the draws, so they involve no random numbers. Use `summary()`, or `as.data.frame(summary(x))`, to obtain the table; `as.data.frame(x)` returns the posterior draws. The samples can be converted to **posterior** draws formats using `as_draws()`.

See Also

`predict.brma()`, `pooled_heterogeneity()`, `blup()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- brma(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )

  pooled_effect(fit)
}

## End(Not run)
```

`pooled_heterogeneity` *Pooled Heterogeneity*

Description

Computes method-specific pooled (aggregated) heterogeneity from a fitted model.

Usage

```
pooled_heterogeneity(object, ...)
```

Arguments

`object`	a fitted model object
`...`	additional arguments passed to methods

Value

Method-specific return value, typically a summary table or posterior samples of the pooled heterogeneity.

See Also

`predict.brma()`

`pooled_heterogeneity.brma`

Pooled Heterogeneity for brma Objects

Description

Computes heterogeneity (tau) of a fitted brma object at the average fitted scale design (the mean of the observation-specific tau) and, for random-formula models, at the average expanded random-effect design.

Usage

```
## S3 method for class 'brma'
pooled_heterogeneity(
  object,
  probs = c(0.025, 0.975),
  conditional = FALSE,
  component = "all",
  ...
)
```

Arguments

`object`	a fitted brma object
`probs`	quantiles of the posterior distribution to be displayed. Defaults to `c(.025, .975)` for 95% credible intervals.
`conditional`	whether to return the pooled heterogeneity conditional on the heterogeneity component for RoBMA product-space objects. Defaults to `FALSE`.
`component`	heterogeneity component to return for `brma.mv()` models. Defaults to `"all"`. Use `"total"` for the variance-additive total heterogeneity.
`...`	additional arguments passed to `predict.brma` (`as_measure`, `V_new`, and `conditioning_depth`); wrapper arguments such as `newdata`, `type`, and `quiet` are fixed (supplying one is R's error for an argument matched twice, as for `predict()`), and other unknown arguments are unused and trigger a warning.

Details

For location-scale models, the function returns the arithmetic mean over the fitted rows of the observation-specific tau, as in 4.0.0; it equals `rowMeans(predict(object, type = "terms.scale"))`. It is not the root mean square heterogeneity across the observed rows. Use `summary_heterogeneity()` for the latter.

For models without scale regression, this returns the single tau parameter.

For multilevel (3-level) models, the returned tau is the total heterogeneity: `tau = sqrt(tau_within^2 + tau_between^2)`.

For `brma.mv()` models, one ordinary selected component is named `tau`; a genuine variance-additive total is named `tau_total`; and a mean-variance allocation anchor is named `tau_common`. These quantities are available when a random formula is declared. For random-formula models, `component = "all"` returns one `brma_samples` object when there is a single heterogeneity component and a named list when there are multiple components. `component = "total"` computes the variance-additive total at the average random-effect design. Random slopes are evaluated as $\bar{q}^T G \bar{q}$, including within-block correlations. Default and named allocation components retain their declared scale anchors; explicit totals and pooled-effect prediction intervals use the full covariance of the average design, including allocation-owned blocks. With a known group covariance R, the reported `sd` is the fitted multiplier of that kernel. It need not equal every row's marginal standard deviation when $\mathrm{diag}(R)$ is not one. Row-specific marginal standard deviations are available from `predict(type = "terms.scale")`. A scale formula of a random-formula model enters the average random-effect design as the arithmetic mean of the observation-specific tau, as for the univariate models. `predict(object, newdata = TRUE, type = "terms.scale")` returns the same component result: a named list for several components and one `brma_samples` object for a single component.

For `brma.mv()` models and models with factor-specific or random-slope terms, the average expanded random-effect design can be synthetic and need not correspond to a single realizable outcome or study. The result is the heterogeneity of that linear composite, not typical heterogeneity across the observed designs. Interpret it only when the composite has a meaningful scientific interpretation; use `summary_heterogeneity()` for observed-design heterogeneity or `predict(..., newdata = ...)` for a specified realizable design.

Value

A `brma_samples` object containing posterior samples. When printed, displays a summary table. For decomposed `brma.mv()` models, a named list of `brma_samples` objects is returned. Use `summary()` on an individual object (`as.data.frame()` of an individual object returns its draws) or `as.data.frame()` on the complete result to obtain a summary data frame with component and parameter identifiers. Use `as.data.frame(format = "list")` to retain separate component tables. The samples can be converted to **posterior** draws formats using `as_draws()`.

See Also

`predict.brma()`, `pooled_effect()`, `blup()`

Examples

```
## Not run:
```

```
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- brma(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )

  pooled_heterogeneity(fit)
}

## End(Not run)
```

`post_prob.brma` *Posterior Model Probabilities for brma Objects*

Description

Compute posterior model probabilities from marginal likelihoods of brma models.

Usage

```
## S3 method for class 'brma'
post_prob(x, ..., prior_prob = NULL, model_names = NULL)
```

Arguments

`x`	a brma model object.
`...`	additional brma model objects.
`prior_prob`	numeric vector with prior model probabilities or weights. Values must be finite, nonnegative, have the same length as the retained models, and have a positive total. If omitted, a uniform prior is used. Finite comparisons combine the original weights with marginal likelihoods in log space and normalize the resulting probabilities. An exactly zero prior weight excludes that model.
`model_names`	character vector with model names. If `NULL` (the default), names will be derived from deparsing the call.

Details

The marginal likelihoods must first be computed using `add_marglik`. `x` and at least one additional `brma` model must be supplied. Non-`brma` objects in `...` are ignored with a warning. All retained models must be fitted to the same outcome target/data, including outcome type and, when present,

likelihood weights. Cluster identifiers define model structure and are not part of the outcome-data identity. A known sampling covariance without correlated blocks (e.g., `V = diag(sei^2)` in `brma.mv`) is the outcome data of its standard errors, whereas a correlated `V` is a different target.

If any model contains repeated bridge estimates, comparisons retain those repetitions regardless of model order. Models with fewer repetitions are recycled to the largest repetition count with a warning, following **bridgesampling**. Standardized marginal-likelihood results must be finite. Legacy raw `"bridge_list"` results retain non-finite repetitions and use upstream comparison behavior through **bridgesampling**, with normalized prior probabilities and its missing-value propagation, warnings, and errors.

Value

A named numeric vector with posterior model probabilities for single estimates, or a repetition-by-model numeric matrix when any model has repeated estimates. Each successfully evaluated comparison sums to one.

See Also

`add_marglik`, `bridge_sampler.brma`, `bf.brma`, `logml.brma`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit1 <- brma(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")
  fit2 <- brma(
    yi           = yi,
    vi           = vi,
    data         = dat.lehmann2018,
    measure      = "SMD",
    prior_effect = FALSE
  )

  fit1 <- add_marglik(fit1)
  fit2 <- add_marglik(fit2)

  post_prob(fit1, fit2)
}

## End(Not run)
```

`Poulsen2006` *5 studies with a tactile outcome assessment from Poulsen et al. (2006) of the effect of potassium-containing toothpaste on dentine hypersensitivity*

Description

The data set contains Cohen's d effect sizes, standard errors, and labels for 5 studies assessing the tactile outcome from a meta-analysis of the effect of potassium-containing toothpaste on dentine hypersensitivity (Poulsen et al. 2006) which was used as an example in Bartoš et al. (2021).

Usage

```
Poulsen2006
```

Format

A data.frame with 3 columns and 5 observations:

`d` Cohen's d effect size.

`se` Standard error of d.

`study` Study label.

References

Bartoš F, Gronau QF, Timmers B, Otte WM, Ly A, Wagenmakers E (2021). "Bayesian model-averaged meta-analysis in medicine." *Statistics in Medicine*, **40**(30), 6743–6761. doi:10.1002/sim.9170.

Poulsen S, Errboe M, Mevil YL, Glenny A (2006). "Potassium containing toothpastes for dentine hypersensitivity." *Cochrane Database of Systematic Reviews*. doi:10.1002/14651858.cd001476.pub2.

---

predict.brma *Predict From brma Object*

Description

`predict.brma` predicts values

Usage

```
## S3 method for class 'brma'
predict(
  object,
  newdata = NULL,
  type = "terms",
  as_measure = TRUE,
  output_measure = NULL,
  transform = NULL,
  probs = c(0.025, 0.975),
  bias_adjusted = FALSE,
  quiet = FALSE,
  conditional = FALSE,
```

```
  V_new = NULL,
  conditioning_depth = "marginal",
  ...
)
```

Arguments

object	a fitted brma object
newdata	prediction data. Defaults to NULL, which predicts the observed design rows. TRUE returns a single aggregated prediction at the average fitted design (the scalar parameter for models without moderators or scale predictors, otherwise the arithmetic mean across the fitted rows of the location and of the observation-specific heterogeneity): the pooled location (type = "terms"), the pooled heterogeneity ("terms.scale"), a new true effect ("estimate"), or a new cluster effect added to the pooled location ("cluster", multilevel models only). It is the quantity reported by pooled_effect and pooled_heterogeneity without the prediction interval columns, requires marginal depth, and is not available for type = "response", which needs observation-level sampling variance. Marginal new-effect predictions may use an explicit data frame or named list with one row per target effect and every variable used by the requested fixed, scale, and random-effect designs. Missing ordinary grouping-only identifiers are synthesized as row-specific new levels; known-R grouping identifiers must be supplied. Outcome columns are optional for type = "terms" unless PET/PEESE bias terms are included via bias_adjusted = FALSE. Missing values are rejected; prediction never drops rows silently. Binomial GLMM response predictions require arm totals n1i and n2i; supply the totals alone, optionally with event counts ai and ci, or supply the four cell counts ai, bi, ci, and di. Supplied cells and totals must agree. Poisson GLMM response predictions require exposures t1i and t2i; event counts x1i and x2i are optional. For known-V brma.mv() response prediction, explicit newdata also requires sampling covariance through V_new. Product-only selection models require no publication-group column. Unadjusted normal selected latent predictions (type = "estimate") require sei or vi in explicit newdata. If an active selection branch uses weight_rule = "best", its publication groups must also be resolvable in the prediction data.
type	type of prediction to be performed. Options are: • "terms" (alias: "marginal"): fixed location parameters only, $X\beta$. • "estimate" (alias: "effect"): latent true effects. The random-effect distribution is selected by conditioning_depth. • "response" (alias: "outcome"): Predicted observed values (yi). Adds the outcome sampling distribution to the corresponding latent-effect target. • "terms.scale": Scale parameter (tau), incorporating scale regression if present. For brma.mv() random-formula models, returns a named list of component-specific brma_samples matrices, whether or not scale formulas are present. The released "cluster" and "blup" spellings are retained as compatibility shortcuts for conditional location samples: they fix conditioning_depth =

"cluster" and "estimate", respectively. Use fitted() or blup() when conditional means are the intended quantity.

as_measure — logical; whether to convert GLMM response predictions from simulated counts to continuity-corrected effect-size estimators (logOR for binomial, logIRR for Poisson). Defaults to TRUE. Only relevant for GLMM models with type = "response". When FALSE, returns raw frequency data (counts). Use type = "estimate" for latent logOR/logIRR predictions.

output_measure — effect-size measure for location/effect predictions. Defaults to the fitted measure. Supported conversions are among "SMD", "COR", "ZCOR", and "OR"; "RR", "HR", "IRR", "RD", and "GEN" can only be returned on their fitted measure. Use transform = "EXP" for ratio-scale output from log-scale measures.

transform — optional display transformation. Currently "EXP" exponentiates log-scale measures "OR", "RR", "HR", and "IRR".

probs — quantiles of the posterior samples to be displayed when the returned brma_samples object is printed. Defaults to c(.025, .975).

bias_adjusted — whether predictions should adjust for publication bias. Defaults to FALSE. When TRUE:

- PET/PEESE terms are NOT added to the mean parameter (mu), returning the bias-corrected effect estimate.
- For type = "response" with selection models, samples from the ordinary normal predictive distribution instead of the selected-normal distribution, simulating what would be observed without publication bias.

When FALSE:

- PET/PEESE terms ARE added to mu, returning predictions that include the expected bias (i.e., what we expect to observe given publication bias).
- For type = "response" with selection models, samples from the selected-normal distribution reflecting the selective publishing process.

For new outcomes, selection predictions draw retained contexts from their original Gaussian law, then sample each full publication event conditional on that context. Selected latent predictions additionally reconstruct the integrated true effects conditional on the selected response. At estimate depth, response prediction is a new replication event conditional on the fitted latent true effects; LOO-PIT instead uses deletion within the original publication event. Retained estimate-level effects use their fitted posterior values at estimate depth and fresh Gaussian draws for a new estimate within a fitted cluster. The fitted selection specification and prediction conditioning depth are separate choices.

quiet — logical; whether to suppress the informational message when conditional posterior samples are flattened to one chain.

conditional — whether to return conditional posterior predictions for RoBMA product-space objects. For location predictions, samples are conditioned on the effect component; for type = "terms.scale", samples are conditioned on the heterogeneity component. Conditional samples are flattened to one chain after subsetting posterior rows.

V_new — optional sampling covariance for explicit newdata response or selected latent predictions from known-V brma.mv() models. Accepts a vector of diagonal

variances, a square covariance matrix, a list of block covariance matrices, or a known_v_factor declaration. Its total dimension must match nrow(newdata). It is required for explicit-newdata response predictions and unadjusted selected latent predictions. Cross-covariance with observed rows is not supported. Numerical symmetry is handled as for V: accepted off-diagonal pairs are averaged internally; the supplied object and diagonal are unchanged.

conditioning_depth

random-effect conditioning depth for type = "estimate" and "response". "marginal" (default) predicts new latent effects, "cluster" conditions on the fitted cluster effect and predicts a new estimate within that cluster, and "estimate" predicts the fitted latent effect. Cluster depth is available only for specialized multilevel models fitted with cluster. Non-marginal prediction is currently restricted to newdata = NULL, where fitted identities are unambiguous.

... unused additional arguments. Supplied arguments trigger a warning.

Details

Prediction has two independent axes. type selects the quantity: fixed location ("terms"), latent true effect ("estimate"), or observed outcome ("response"). conditioning_depth selects how much of the fitted random-effect hierarchy is retained. In the specialized multilevel model fitted with cluster, $y_{ij} = X_{ij}\beta + u_j + v_{ij} + \epsilon_{ij}$, the targets are:

- "marginal": condition on neither u_j nor v_{ij};
- "cluster": condition on fitted u_j, but draw a new v_{ij};
- "estimate": draw the fitted latent effect from its posterior.

type = "response" adds ϵ to the selected latent-effect target. For normal fitted effects this includes the conditional latent uncertainty, not only the BLUP mean. blup() and fitted(...,conditioning_depth = "estimate") remain the conditional-mean interfaces.

For admitted weighted Gaussian selection models, reconstruction of fitted integrated true effects uses the normalized conditional kernel $N(u; 0, \tau^2)N(y; c + u, s^2)^w$. Its conditioning variance is s^2/w; the fitted global posterior and retained latent contributions remain the supplied posterior. This post-fit augmentation does not assert that likelihood powering commutes with integrating a latent effect. Estimate-depth estimate and response targets record this policy and the row-aligned likelihood weights in RoBMA_target$latent_augmentation. Response replication adds the original sampling variance s^2.

Ordinary independent Gaussian selection predictions use exact diagonal-cell chunk pricing after conservative fixed-geometry and output reserves. Other model structures retain the dense covariance charge under the same budget. The RoBMA.known_v_covariance_max_bytes option controls covariance chunk memory. Selected predictions are reproducible for a fixed seed and chunk budget, but changing this option can change the returned seeded draw sequence because chunking changes the order of random-number consumption.

newdata supplies design values and grouping identities; it does not select an estimand. Consequently, marginal prediction has the same law for newdata = NULL and an equivalent explicit design. Matching fitted grouping labels retains the requested dependence among new marginal draws but does not reuse fitted random effects. Cluster- and estimate-depth predictions require fitted identities and therefore currently reject explicit newdata.

For binomial and Poisson GLMM response predictions, estimate depth uses the fitted posterior nuisance base rate π_i or rate ϕ_i. Marginal and cluster depths describe a new estimate and therefore draw a new nuisance rate independently from its model prior. Those predictions contain a prior-predictive nuisance-rate component, which can dominate their width when the nuisance prior is diffuse; fitted π_i/ϕ_i values are never silently reused for a marginal or cluster target.

For RoBMA product-space objects, conditional posterior predictions subset posterior rows according to model indicators. This removes the original chain structure, so returned `brma_samples` objects are intentionally stored as one flattened chain.

Note that in contrast to predict, the `type = "response"` produces predictions for the new effect size estimates. To obtain results corresponding to metafor's predict function, use `type = "terms"` for the mean effect size and `type = "estimate"` for true effects (prediction interval).

Likelihood weights: If the model was fitted with `weights`, the weights affect the posterior fit, log-likelihood, LOO/WAIC, and existing-data conditional diagnostics such as BLUP shrinkage and leverage. They do not change the observation-level sampling error used by `type = "response"` for normal models: response predictions simulate raw future effect-size estimates with the supplied `sei`, not `sei / sqrt(weight)`.

For known-`V` `brma.mv()` models, explicit `newdata` response and unadjusted selected latent prediction require `V_new`. Random-formula models preserve the joint random-effect structure selected by `conditioning_depth` and then add `V` or `V_new` sampling noise. Cross-covariance with observed rows is not supported. Explicit `newdata` estimate/response predictions are not available for `brma.mv()` objects with marginalized known-`R` blocks, because their row multipliers are fitted-row metadata. `brma.mv()` prediction results carry a `brma_mv_prediction_target` attribute recording the formula, random-effect, and response-generation target used.

For known-`V` models, the conditional random-effect predictions require applying the inverse of the marginal covariance. A covariance that is too ill-conditioned for that (a reciprocal condition number of about `2e-10` or less in any posterior draw used) is unavailable: the call stops with an error of class `RoBMA_known_v_precision_unavailable`, whose parent class is `RoBMA_numerical_unavailable`, instead of returning a rounded number. The condition carries the draw positions (`rows`) and their `reciprocal_condition`.

Value

A `brma_samples` object containing posterior samples. For `brma.mv()` random-formula `type = "terms.scale"` predictions, returns a named list of component-specific `brma_samples` objects; with `newdata = TRUE` this component result is the `component = "all"` result of pooled_heterogeneity (a named list for several components, one `brma_samples` object for a single component), computed with the arithmetic mean of the observation-specific tau of a scale formula; its components are titled "Pooled Heterogeneity (<block>)" and "Total Pooled Heterogeneity". When printed, each `brma_samples` object displays a summary table via `BayesTools::ensemble_estimates_table`. The underlying samples matrix can be accessed directly because the object inherits from matrix. For a single `brma_samples` object, use `summary()`, or `as.data.frame(summary(x))`, to obtain the summary table with component and parameter identifiers; `as.data.frame(x)` returns the posterior draws. A list of results (a `brma_samples_list`) coerces directly: `as.data.frame()` of the list gives the summary table, and `as.data.frame(format = "list")` retains separate tables. The samples can also be converted to **posterior** draws formats using `as_draws()` and related functions.

See Also

pooled_effect(), pooled_heterogeneity(), blup()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE) &&
    requireNamespace("metafor", quietly = TRUE)) {
  data(dat.bcg, package = "metadat")
  dat <- metafor::escalc(
    measure = "RR",
    ai      = tpos,
    bi      = tneg,
    ci      = cpos,
    di      = cneg,
    data    = dat.bcg
  )

  fit <- brma(
    yi      = yi,
    vi      = vi,
    mods    = ~ ablat + year,
    data    = dat,
    measure = "RR",
    seed    = 1,
    silent  = TRUE
  )

  predict(fit, type = "terms")
  predict(fit, newdata = TRUE, type = "terms")
  predict(fit, newdata = data.frame(ablat = 50, year = 1970),
          type = "estimate")
  predict(fit, type = "estimate")
}

## End(Not run)
```

---

print.brma_samples *Print brma_samples Object*

Description

Prints a summary table of posterior samples using BayesTools::ensemble_estimates_table. Returns the samples invisibly for method chaining.

Usage

```
## S3 method for class 'brma_samples'
print(x, probs = NULL, ...)
```

Arguments

x	a brma_samples object
probs	quantiles for credible intervals. If NULL, uses the default stored in the object (typically c(.025, .975))
...	additional arguments passed to BayesTools::ensemble_estimates_table

Value

Returns x invisibly.

---

print.marginal_means.brma *Print Estimated Marginal Means*

Description

Prints the estimated marginal means summary.

Usage

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

Arguments

x	a marginal_means.brma object.
...	additional arguments passed to summary().

Value

Returns x invisibly.

---

print.RoBMA_data *Print method for RoBMA_data objects*

Description

Prints a concise summary of an RoBMA_data object.

Usage

```
## S3 method for class 'RoBMA_data'
print(x, n = 6, ...)
```

Arguments

x	an RoBMA_data object
n	maximum number of rows to display for each component. Defaults to 6.
...	additional arguments (ignored)

Value

Invisibly returns the input object.

`print.summary.brma_samples`

Print summary.brma_samples Object

Description

Prints a summary table of posterior samples using `BayesTools::ensemble_estimates_table`. Returns the summary table invisibly.

Usage

```
## S3 method for class 'summary.brma_samples'
print(x, probs = NULL, ...)
```

Arguments

x	a `summary.brma_samples` object
probs	quantiles for credible intervals. If `NULL`, uses the default stored in the object (typically `c(.025, .975)`)
...	additional arguments passed to `BayesTools::ensemble_estimates_table`

Value

Returns the summary table invisibly.

`print.summary.marginal_means.brma`
Print Summary of Estimated Marginal Means

Description

Prints a summary table of estimated marginal means.

Usage

```
## S3 method for class 'summary.marginal_means.brma'
print(x, ...)
```

Arguments

`x`	a `summary.marginal_means.brma` object.
`...`	unused additional arguments. Supplied arguments trigger a warning.

Value

Returns `x` invisibly.

`print.summary.zplot_brma`
Print Zplot Summary

Description

Print Zplot Summary

Usage

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

Arguments

`x`	a summary.zplot_brma object.
`...`	additional arguments (currently unused).

Value

Invisibly returns the summary object.

print.summary_heterogeneity.brma *Print Summary of Heterogeneity*

Description

Prints the heterogeneity summary table.

Usage

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

Arguments

x	a summary_heterogeneity.brma object
...	additional arguments (currently ignored)

Value

Returns x invisibly.

print.vif.brma *Print VIF Results*

Description

Prints variance inflation factors and optional posterior correlation matrix.

Usage

```
## S3 method for class 'vif.brma'
print(x, digits = 3, ...)
```

Arguments

x	a vif.brma object
digits	number of decimal places. Defaults to 3.
...	additional arguments (currently ignored)

Value

Returns x invisibly.

print_prior *Print Prior Distributions*

Description

print_prior prints prior distributions stored in brma, BMA, and RoBMA objects. This is a user-facing helper for inspecting priors without extracting them from the internal $priors list.

Usage

```
print_prior(x, ...)

## S3 method for class 'prior'
print_prior(x, ...)

## S3 method for class 'brma'
print_prior(
  x,
  parameter,
  parameter_mods,
  parameter_scale,
  component = "auto",
  ...
)
```

Arguments

x	a brma, BMA, or RoBMA object, or a prior distribution object.
...	additional arguments passed to the prior printing method. Use silent = TRUE for programmatic inspection without console output.
parameter	character. Base parameter to print. If omitted, all stored prior distributions are printed. Common options are "mu", "tau", "rho", "PET", "PEESE", "omega", and "bias", with aliases "effect" = "mu", "heterogeneity" = "tau", and "weightfunction" = "omega". GLMM outcome priors "pi" and "phi" are available when present. Moderator and scale terms can also be selected by name when unambiguous. For brma.mv() objects with random-effect formulas, "random" prints the structured random-effect prior specification, and the random-effect quantities of fitted objects (such as "tau", "tau_total", "tau2", "tau2_prop(study)", or "study: tau") print their declared prior, or, for a quantity that follows from several declared priors, one line giving the derivation (for example tau2 = tau^2 or study: tau = sigma_total * sqrt(w[1]), in the notation of the "random" specification) followed by the declared priors it refers to. A quantity without a prior density stops as in plot_prior(). A character vector requests multiple base parameters. A quantity of the fitted model that is no model parameter (e.g. an inclusion indicator or a weight-function coordinate such as "omega[1]") stops with the classes BayesTools_hypothesis_target and BayesTools_hypothesis_unavailable, as in plot.brma().

parameter_mods — legacy moderator selector. Prefer parameter with component = "mods". Use "intercept" for the adjusted effect in meta-regression models.

parameter_scale — legacy scale-regression selector. Prefer parameter with component = "scale". Use "intercept" for the heterogeneity intercept in location-scale models.

component — parameter component. Defaults to "auto", which infers the component when possible. Use "mods" (alias "location"), "scale", "bias", or "random" (the random-effect quantities of fitted objects) to disambiguate terms used in multiple prior components. A parameter_mods or parameter_scale selection with another component stops with an error of class RoBMA_component_mismatch.

Value

print_prior invisibly returns the selected prior distribution (for a random-effect quantity that is not one declared prior, its prior density, a BayesTools prior_linear_density). If multiple parameters are requested, a named list of prior distributions is returned invisibly.

See Also

plot_prior() BMA() RoBMA() brma() prior()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  priors <- BMA(
    yi          = yi,
    vi          = vi,
    mods        = ~ Preregistered,
    data        = dat.lehmann2018,
    measure     = "SMD",
    only_priors = TRUE
  )

  print_prior(priors)
  print_prior(priors, parameter = "mu")
  print_prior(priors, parameter = "tau")
  print_prior(priors, parameter = "Preregistered", component = "mods")
}

## End(Not run)
```

---

prior *Prior Distribution*

Description

Create prior distribution objects used by RoBMA and brma fitting functions.

Usage

```
prior(
  distribution,
  parameters,
  truncation = list(lower = -Inf, upper = Inf),
  prior_weights = 1
)
```

Arguments

`distribution`	character. Prior distribution name.
`parameters`	list. Distribution parameters.
`truncation`	list with `lower` and `upper` truncation bounds.
`prior_weights`	numeric prior model weight.

Details

This is RoBMA's re-export of `BayesTools::prior()`; supported distribution names and parameter lists follow BayesTools.

Value

An object inheriting from `prior`.

Examples

```
prior("normal", list(mean = 0, sd = 0.5))
prior(
  "normal",
  list(mean = 0, sd = 0.5),
  truncation = list(lower = 0, upper = Inf)
)
```

`prior_factor` *Factor Prior*

Description

Create priors on factor contrasts.

Usage

```
prior_factor(
  distribution,
  parameters,
  truncation = list(lower = -Inf, upper = Inf),
  prior_weights = 1,
  contrast = "meandif"
)
```

Arguments

distribution	character. Prior distribution name.
parameters	list. Distribution parameters.
truncation	list with lower and upper truncation bounds.
prior_weights	numeric prior model weight.
contrast	character. Contrast coding used for factor levels. Common RoBMA model options are "treatment", "meandif", and "orthonormal"; the standalone helper also accepts BayesTools contrast aliases such as "independent".

Details

Mean-difference and orthonormal contrasts require vector or multivariate priors. Treatment/dummy and independent contrasts use univariate simple priors per contrast coefficient.

Value

An object inheriting from prior, prior.factor, and a contrast-specific marker class.

---

prior_informed *Informed Prior*

Description

Create empirical informed prior distributions.

Usage

```
prior_informed(name, parameter = NULL, type = "smd")
```

Arguments

name	character. Informed prior name. Supported examples include "van Erp", "Oosterwijk", "Cochrane", and medicine subfields from BayesTools::prior_informed_medicine_names.
parameter	character. Parameter subset. Required for medicine priors and not needed for psychology priors.
type	character. Effect-size type. Medicine priors use values such as "smd", "logOR", "logRR", "RD", and "logHR".

Details

Further details can be found in van Erp et al. (2017), Gronau et al. (2017), Bartoš et al. (2021), and Bartoš et al. (2023).

Value

An object of class `prior`.

References

Bartoš F, Gronau QF, Timmers B, Otte WM, Ly A, Wagenmakers E (2021). "Bayesian model-averaged meta-analysis in medicine." *Statistics in Medicine*, **40**(30), 6743–6761. doi:10.1002/sim.9170.

Bartoš F, Otte WM, Gronau QF, Timmers B, Ly A, Wagenmakers E (2023). "Empirical prior distributions for Bayesian meta-analyses of binary and time-to-event outcomes." doi:10.48550/arXiv.2306.11468. Preprint available at https://doi.org/10.48550/arXiv.2306.11468.

Gronau QF, Van Erp S, Heck DW, Cesario J, Jonas KJ, Wagenmakers E (2017). "A Bayesian model-averaged meta-analysis of the power pose effect with informed and default priors: The case of felt power." *Comprehensive Results in Social Psychology*, **2**(1), 123–138. doi:10.1080/23743603.2017.1326760.

van Erp S, Verhagen J, Grasman RP, Wagenmakers E (2017). "Estimates of between-study heterogeneity for 705 meta-analyses reported in Psychological Bulletin from 1990–2013." *Journal of Open Psychology Data*, **5**(1), 1–5. doi:10.5334/jopd.33.

`prior_none` *Empty Prior*

Description

Create an empty prior used to omit a model component.

Usage

```
prior_none(prior_weights = 1)
```

Arguments

`prior_weights`	numeric prior model weight.

Value

An object inheriting from `prior` and `prior.none`.

prior_ordered *Ordered Factor Prior*

Description

Create priors on ordered factor effects by separating a scalar total effect from its allocation across ordered-level increments.

Usage

```
prior_ordered(
  total,
  allocation = NULL,
  contrast = "cumulative",
  id = NULL,
  prior_weights = 1
)
```

Arguments

total	scalar prior for the sum of ordered increments. With "cumulative", this is the first-to-last level difference; with "cumulative_levels", it is the last level's effect relative to the zero baseline preceding the first level.
allocation	allocation prior or fixed split. NULL creates a late-bound flat Dirichlet allocation with dimension determined by the formula term. Numeric vectors specify fixed splits. Dirichlet priors specify random allocations. For terms with multiple ordered factors, use a named list.
contrast	ordered contrast. "cumulative" uses nlevels(f) - 1 increments and sets the first level effect to zero. "cumulative_levels" uses nlevels(f) increments and gives the first level a non-zero allocation share.
id	optional allocation-sharing id. Equal ids share compatible allocation weights, not the total-effect prior. For terms with multiple ordered factors, use a named character vector.
prior_weights	numeric prior model weight.

Details

This is RoBMA's re-export of BayesTools::prior_ordered(). Spike-and-slab or mixture behavior belongs on total, e.g., prior_ordered(prior_spike_and_slab(...)). Unsupported ordered combinations raise typed BayesTools_ordered_unavailable conditions; constructor support does not imply support for every post-fit operation.

Value

An object inheriting from prior.

See Also

prior_factor

`prior_PEESE` *PEESE Prior*

Description

Create PEESE publication-bias regression priors.

Usage

```
prior_PEESE(
  distribution,
  parameters,
  truncation = list(lower = 0, upper = Inf),
  prior_weights = 1
)
```

Arguments

`distribution`	character. Prior distribution name.
`parameters`	list. Distribution parameters.
`truncation`	list with `lower` and `upper` truncation bounds.
`prior_weights`	numeric prior model weight.

Details

This forwards to `BayesTools::prior_PEESE()` and uses the same distribution and parameter conventions as `prior()`.

Value

An object inheriting from `prior` with the `prior.PEESE` marker class.

See Also

publication_bias_prior_specification

prior_PET *PET Prior*

Description

Create PET publication-bias regression priors.

Usage

```
prior_PET(
  distribution,
  parameters,
  truncation = list(lower = 0, upper = Inf),
  prior_weights = 1
)
```

Arguments

distribution	character. Prior distribution name.
parameters	list. Distribution parameters.
truncation	list with lower and upper truncation bounds.
prior_weights	numeric prior model weight.

Details

This forwards to BayesTools::prior_PET() and uses the same distribution and parameter conventions as prior(). By default, PET priors are truncated to the positive half-line.

Value

An object inheriting from prior with the prior.PET marker class.

See Also

publication_bias_prior_specification

prior_random *Random-effect prior constructors*

Description

Constructors for specifying priors and policies for the random effects in `brma.mv()`. These functions are re-exported from BayesTools so a partial or complete random-effect prior can be written with the RoBMA namespace alone.

`prior_random()` collects the specification supplied through `prior_heterogeneity`. `random_block()` overrides priors or policies for one named random-effect block. `random_covariance()` supplies correlation priors, with `prior_lkj()` used for an unstructured correlation matrix. `random_variance_allocation()` distributes an aggregate random-effect standard deviation across blocks or across the SD components of one heterogeneous block. Its required `name` is a stable internal identifier, while `display_name` and `component_names` control public semantic labels. `allocation_ref()` connects a child allocation to a component of an earlier named allocation. `random_monitor()` controls saved random-effect quantities, and `random_new_levels()` controls prediction for previously unseen grouping levels. `random_sd_source()` is the advanced adapter for an existing scalar or row-shaped JAGS node that supplies a random-effect SD without owning a new prior for that node.

Usage

```
prior_random(
  ...,
  sd = NULL,
  covariance = NULL,
  cor = NULL,
  monitor = random_monitor(),
  new_levels = random_new_levels(),
  allocation = NULL,
  parameterization = "noncentered"
)

random_block(
  sd = NULL,
  covariance = NULL,
  cor = NULL,
  monitor = NULL,
  new_levels = NULL,
  terms = NULL,
  contrasts = NULL,
  sd_source = NULL,
  parameterization = NULL
)

random_variance_allocation(
  name,
```

```
  terms = NULL,
  sd = NULL,
  weights = NULL,
  parent = NULL,
  target = c("block", "sd_component"),
  scale = c("total_variance", "mean_variance"),
  sd_source = NULL,
  inclusion = NULL,
  display_name = name,
  component_names = NULL
)

allocation_ref(allocation_name, component)

random_covariance(
  structure = NULL,
  sd = NULL,
  cor = NULL,
  eta = NULL,
  cor_scale = c("fisher_z", "cor", "logit")
)

prior_lkj(eta = 1, include_correlation = TRUE, include_primitives = FALSE)

random_monitor(
  latent = TRUE,
  coefficients = FALSE,
  correlation = TRUE,
  lkj_primitives = FALSE
)

random_new_levels(method = c("error", "zero", "sample"))

random_sd_source(source, shape = c("scalar", "row"))
```

Arguments

`...`	named `random_block()` overrides and, optionally, `random_variance_allocation()` objects. Override names must match resolved block names.
`sd`	an SD prior with nonnegative support. In `prior_random()`, it is the top-level default inherited by blocks. In `random_block()` and `random_covariance()`, it is a more specific block or covariance SD prior. In a root `random_variance_allocation()`, it is the aggregate-SD prior.
`covariance`	a `random_covariance()` object. A top-level value is inherited by blocks without a block-specific covariance override.
`cor`	an optional correlation prior. Use `prior_lkj()` for an unstructured correlation matrix and an ordinary scalar prior for CS, HCS, AR1, HAR, or CAR. It is a convenience alternative to `covariance = random_covariance(cor = ...)`.

monitor — a random_monitor() object. In prior_random(), it supplies the top-level default; in random_block(), NULL inherits that default.

new_levels — a random_new_levels() object controlling prediction for grouping levels absent from the fitted data. In random_block(), NULL inherits the top-level policy.

allocation — a random_variance_allocation() object or list of such objects. Allocations can equivalently be supplied through ..., but the two routes cannot be combined.

parameterization — requested sampled representation: "noncentered", "centered", or "auto". The top-level value is inherited by blocks whose random_block() value is NULL. A named random_block() can additionally request "mean_centered"; see the details.

terms — in random_block(), an optional named list of term-specific SD overrides. Each entry is an SD prior or random_block(sd = ...), and its name must match a resolved term such as "intercept" or a slope label. In random_variance_allocation(), a character vector of targeted block names; NULL lets a single root block allocation target all remaining blocks at resolution time.

contrasts — for random_block(), an optional named character vector assigning contrast families to factor predictors in that block. Accepted values are "treatment", "independent", "orthonormal", "meandif", "cumulative", and "cumulative_levels", with or without the contr. prefix. Structure-owned index families do not accept this argument.

sd_source — an external SD source created with random_sd_source(). In random_block(), it conflicts with block-local sd, covariance-owned sd, and term-specific SD overrides. In a root allocation, exactly one of sd and sd_source is required; child allocations inherit from parent.

name — required stable allocation identifier used internally and by allocation_ref().

weights — an optional Dirichlet prior over allocation weights. For a block allocation with explicit terms, omission creates a symmetric Dirichlet(1, ..., 1) prior.

parent — an optional allocation_ref() selecting one component of an earlier named allocation. A child allocation must not specify sd or sd_source.

target — allocation target: "block" divides a variance budget among blocks or symbolic components; "sd_component" divides it among the resolved SD components of exactly one heterogeneous block.

scale — variance normalization. "total_variance" uses sd_child = sd_parent * sqrt(w). "mean_variance" uses sd_child = sd_parent * sqrt(K * w) and is available only for target = "sd_component".

inclusion — an optional named list of scalar probability priors. For a block allocation, each named component receives an independent Bernoulli inclusion gate.

display_name — public semantic owner used as the parameter-name prefix. It defaults to name; use "" when a single model component needs no disambiguating prefix.

component_names — optional public names for allocation components, in the same order as terms. Internal component identifiers remain the sanitized names of terms.

`allocation_name`
: name of an earlier allocation referred to by `allocation_ref()`.

`component`
: component label within the referenced allocation. Named allocation `terms` define labels; unnamed terms use sanitized block names.

`structure`
: optional covariance-family label for `random_covariance()`: `"ID"`, `"DIAG"`, `"CS"`, `"HCS"`, `"AR"` / `"AR1"`, `"HAR"`, `"CAR"`, or `"UN"` / `"US"`. Matching is case-insensitive. In RoBMA it must agree with the formula-owned family.

`eta`
: positive LKJ concentration. Values above one increasingly favor correlations near zero; `eta = 1` is uniform over correlation matrices.

`cor_scale`
: scale for an explicitly supplied scalar `cor` prior. `"fisher_z"` places it on `atanh(cor)` and is the default; `"cor"` uses raw correlation; `"logit"` uses a logit transform of the admissible raw interval. An omitted `cor` receives `LKJ(1)` for US/UN and the structure-specific raw-uniform default for scalar correlations after the structure and dimension are resolved.

`include_correlation`
: whether `prior_lkj()` requests monitoring of the semantic correlation matrix.

`include_primitives`
: whether `prior_lkj()` requests monitoring of its primitive beta coordinates. These coordinates remain internal and are not added to the map's public quantities.

`latent`
: whether to monitor standardized latent random effects. These are required for bridge sampling and random-coefficient reconstruction.

`coefficients`
: whether to monitor realized group-level coefficients. This can be convenient for inspection but memory intensive.

`correlation`
: whether semantic correlation matrices are included in monitored output. Scalar structures may retain an internal backend coordinate during sampling and derive the requested correlation matrix afterward.

`lkj_primitives`
: whether to retain LKJ primitive beta coordinates for backend inspection. These coordinates remain internal and are not public plotting, density, or hypothesis targets.

`method`
: new-level prediction method. `"error"` rejects unseen levels, `"zero"` assigns zero random contribution, and `"sample"` draws from the fitted random-effect distribution.

`source`
: existing JAGS node name, or a BayesTools `parameter_source` object, used by `random_sd_source()`.

`shape`
: source shape for `random_sd_source()`: `"scalar"` for one node or `"row"` for a vector aligned with the formula rows.

Details

Formula structure is separate from prior structure:

The `random` formula owns the grouping variables, coefficient or index design, and covariance family. These prior constructors configure the already parsed blocks; they do not reinterpret the formula. The `id()`, `diag()`, and `us()` / `un()` families use a coefficient formula: `1`, `0`, and `-1` control the random intercept, and continuous slopes, factor slopes, and interactions are supported. Plain `|` syntax defaults to US and `||` to DIAG.

Factor coding in a random-coefficient block is owned by resolved contrast metadata, not by the presence of an intercept. Existing fixed-factor contrasts are reused when available; `random_block(contrasts = ...)` supplies an explicit block-specific override. Thus `us(0 + group | study)` combined with `contrasts = c(group = "independent")` gives one correlated coefficient per factor level and no random intercept. Omitting the intercept alone does not force a level-indicator basis.

The `cs()` / `hcs()`, `ar1()` / `ar()` / `har()`, and `car()` structures instead own an index basis. Their blocks reject intercept controls and `random_block(contrasts = ...)`. HCS estimates one common pairwise correlation; US estimates an unrestricted correlation matrix among the selected coefficient columns, except within an exchangeable mean-difference or orthonormal factor unit. See `random_effect_formula_tags()` for the full formula and index-type contract.

RoBMA completion of partial specifications:

A named `random_block()` must match the block name produced by the parsed formula. Use `only_priors = TRUE` when constructing a `brma.mv()` object and then `print_prior(object, parameter = "random")` to inspect the resolved names and completed specification. An ordinary positive prior supplied to `prior_heterogeneity` changes RoBMA's base heterogeneity prior while retaining its automatic allocation and covariance rules. A `prior_random()` object with no SD or allocation specification is completed in the same way: RoBMA adds its UISD-scaled base SD and any required Dirichlet allocation while retaining the supplied block contrasts, covariance priors, and policies. If any global or block SD, SD source, term-specific SD override, covariance-owned SD, or allocation is supplied, the user owns the complete scale architecture and RoBMA does not merge in scale defaults.

Consequently, covariance-only, contrast-only, monitoring-only, prediction-only, and parameterization-only objects are partial and self-finishing in RoBMA. For example, `prior_random(study = random_block(contrasts = c(group = "independent")))` needs no explicit SD, allocation, or LKJ prior for a US block. In contrast, adding `sd = ...` makes the object complete; every other required scale or allocation must then be supplied explicitly.

Standard-deviation and correlation priors:

Standard-deviation priors must have nonnegative support. Scalar-correlation structures use `cor` in `random_covariance()`; `cor_scale = "fisher_z"` places the prior on Fisher's z, whereas `cor_scale = "cor"` places it on the raw correlation. When omitted, CS/HCS receive a uniform prior over `(-1 / (K - 1), 1)`, AR1/HAR over `(-1, 1)`, and CAR over `[0, 1)`. Unstructured `us()` blocks default to `prior_lkj(eta = 1)`. Values of `eta` above one shrink correlations toward zero; `eta = 1` is uniform over correlation matrices.

The covariance family remains formula-owned. `random_covariance(structure = ...)` is mainly useful in direct BayesTools workflows; when used with `brma.mv()`, an explicit structure must match the parsed formula family. Use `prior_lkj()` as `cor` for an unstructured matrix prior and an ordinary scalar prior as `cor` for a scalar-correlation structure.

Variance allocation:

A block allocation with `scale = "total_variance"` exposes RoBMA quantities `tau_total` and `tau2_total` and splits the slab variance across multiple random-effect blocks. With optional inclusion gates,

$$\sigma_j = I_j \sigma_{\text{slab}} \sqrt{w_j}, \qquad \sigma^2_{\text{total}} = \sigma^2_{\text{slab}} \sum_j I_j w_j.$$

The public `tau2_prop(j)` is $I_j w_j / \sum_k I_k w_k$, conditional on positive total variance. The slab scale and raw weights remain internal. An SD-component allocation with `scale = "mean_variance"`

instead exposes tau_common and tau2_common and gives each component the common-SD scale at equal weights:

$$\sigma_j = \sigma_{\text{common}}\sqrt{Kw_j}, \qquad K^{-1}\sum_j \sigma_j^2 = \sigma_{\text{common}}^2.$$

The corresponding public multipliers are tau2_mult(j) = K * w[j] and tau_mult(j) = sqrt(K * w[j]). This is useful for heterogeneous level-specific SDs. Root allocations own an SD prior or external SD source. Child allocations inherit one named parent component through allocation_ref() and therefore must not supply another SD source. inclusion optionally places independent Bernoulli gates on named block-allocation components.

Public parameter names:

Random-effect quantities are exposed through the semantic catalog view of BayesTools' fitted parameter map. Concrete posterior coordinates are a linked view of the same map and are not additional public names. BayesTools canonical names use (formula) owner: quantity(arguments) with general sd, var, and cor terminology. RoBMA maps these to meta-analytic I/O names: a sole random intercept is tau for a bare block and study: tau for a named block; a non-intercept coefficient remains explicit, for example study: tau(x). An owner-free tau is accepted only when it resolves uniquely. Public RoBMA correlations use rho; compact covariance-construction coordinates remain internal backend dependencies.

A bare formula or unnamed one-entry list omits a redundant component owner. An explicitly named one-entry list retains its owner. For multiple components, public component names come from the formula list; missing names are generated as component 1, component 2, and so on. Total-variance allocations expose realized tau_total, tau2_total, and tau2_prop(...). Mean-variance allocations expose tau_common, tau2_common, tau2_mult(...), and tau_mult(...).

Monitoring, prediction, and parameterization:

Keep the default random_monitor(latent = TRUE) when bridge sampling or conditional prediction is required. For new grouping levels, random_new_levels("error") is the safe default; "zero" removes their random contribution and "sample" draws from the fitted random-effect distribution.

RoBMA creates external SD sources automatically for its location-scale random-effect models. Use random_sd_source() directly only when a custom complete specification intentionally references a JAGS node created elsewhere in the model being fitted.

parameterization changes only the sampled backend representation. "noncentered" samples standardized latent effects, "centered" samples realized group coefficients, and "auto" asks BayesTools to choose from the resolved design. It does not change any SD, allocation, or correlation prior. Unsupported centered configurations are rejected; "auto" can fall back to noncentering.

A named random_block() may instead request "mean_centered" for a sampled block; the top-level parameterization does not accept it and "auto" never selects it. It samples the group effects around the compiled contribution of the fixed terms the block absorbs (the intercept and any moderator its random-effect design spans or that is constant within its groups, and the group locations of an enclosing mean-centered block), so the original effect, SD, allocation, and correlation priors are unchanged; see BayesTools::prior_random(). A block that absorbs nothing is rejected. The mode needs the positive SD coordinates of centered blocks and is rejected for marginalized blocks, known between-group covariance, and selection models in which the block is integrated rather than retained in the selection normalizer.

Value

`prior_random()` returns a `prior_random` object. The remaining functions return the corresponding list-like helper objects consumed by `prior_random()`.

See Also

`random_effect_prior_specification()`, `random_effect_formula_tags()`, `prior()`, `brma.mv()`

Examples

```
sd_prior <- prior(
  "normal",
  parameters = list(mean = 0, sd = 0.5),
  truncation = list(lower = 0, upper = Inf)
)

# One random coefficient per group level, with unrestricted correlation.
random_formula <- ~ us(0 + group | study)
unstructured_prior <- prior_random(
  study = random_block(
    contrasts = c(group = "independent")
  )
)

# The same coefficient basis can instead have a diagonal covariance.
diagonal_formula <- ~ diag(0 + group | study)
diagonal_prior <- prior_random(
  study = random_block(
    contrasts = c(group = "independent")
  )
)

# Correlation-only customization remains partial: RoBMA supplies the SDs.
cor_prior <- prior("normal", list(mean = 0, sd = 1))
hcs_prior <- prior_random(
  study = random_block(
    covariance = random_covariance(
      cor = cor_prior,
      cor_scale = "fisher_z"
    )
  )
)

# Allocate one common SD scale across a block's level-specific SD components.
component_allocation <- random_variance_allocation(
  name = "heterogeneity",
  display_name = "",
  terms = "study",
  sd = sd_prior,
  weights = prior("dirichlet", list(alpha = c(1, 1))),
  target = "sd_component",
  scale = "mean_variance"
```

```
)
allocated_prior <- prior_random(
  study = random_block(
    covariance = random_covariance(cor = prior_lkj(eta = 1)),
    contrasts = c(group = "independent")
  ),
  allocation = component_allocation
)

# A complete AR prior owns both its SD and its scalar correlation prior.
ar_cor_prior <- prior("normal", list(mean = 0, sd = 0.5))
autoregressive_prior <- prior_random(
  study = random_block(
    sd = sd_prior,
    covariance = random_covariance(
      cor = ar_cor_prior,
      cor_scale = "fisher_z"
    )
  )
)

# Monitoring and new-level prediction policies are explicit when customized.
prediction_prior <- prior_random(
  sd = sd_prior,
  monitor = random_monitor(latent = TRUE, coefficients = FALSE),
  new_levels = random_new_levels(method = "sample")
)

# Advanced: refer to an existing row-shaped model node as the SD source.
external_sd <- random_sd_source("tau", shape = "row")
```

`prior_weightfunction` *Weightfunction Prior*

Description

Create weightfunction publication-bias priors and their weight-prior helper objects.

Usage

```
prior_weightfunction(
  side = "one-sided",
  steps = c(0.025, 0.05),
  weights = wf_cumulative(),
  reference = "most_significant",
  prior_weights = 1,
  model = selection_model()
)
```

```
wf_cumulative(alpha = NULL)

wf_fixed(omega)

wf_independent(prior, scale = "omega")
```

Arguments

side	character. Either "one-sided" or "two-sided".
steps	numeric vector of p-value cut points. These define length(steps) + 1 p-value bins and must be ordered values in (0, 1).
weights	a weight-prior object created by wf_cumulative(), wf_fixed(), or wf_independent().
reference	character. Reference bin, currently "most_significant".
prior_weights	numeric prior model weight.
model	fixed likelihood choices from selection_model(). The default integrates estimate random effects and the complete sampling error, conditions on other random effects, and uses product weights. These choices do not change the weight-height prior or prior_weights.
alpha	optional positive cumulative-Dirichlet concentration parameters, one per p-value bin. If NULL, prior_weightfunction() uses rep(1, length(steps) + 1). Cumulative weights encode monotone decreasing publication weights relative to the most-significant bin.
omega	fixed publication weights, one per bin; values must be non-missing, nonnegative, and match length(steps) + 1 when used in prior_weightfunction().
prior	continuous simple prior distribution for each non-reference weight. Point, discrete, mixture, and other non-simple priors are invalid.
scale	latent scale for independent weights; either "omega", "log_omega", or the "log" alias. Direct "omega" priors need nonnegative support; "log" is normalized to "log_omega".

Details

Fixed weights must have one value per p-value bin (length(steps) + 1), and the reference bin must have weight 1.

Value

prior_weightfunction() returns an object inheriting from prior and prior.weightfunction; the wf_*() helpers return weightfunction_weights helper objects with subclass markers.

See Also

publication_bias_prior_specification

Examples

```
prior_weightfunction("one-sided", steps = 0.025)
prior_weightfunction(
  side    = "one-sided",
  steps   = c(0.025, 0.5),
  weights = wf_fixed(c(1, 0.8, 0.6))
)
```

---

qqnorm.brma *Normal QQ Plot for brma Object*

Description

qqnorm.brma creates a normal QQ plot for a fitted brma object. The plot displays sorted standardized residuals on the vertical axis against the theoretical quantiles of a standard normal distribution on the horizontal axis. Under a correctly specified model, the points should fall approximately along the diagonal reference line.

Usage

```
## S3 method for class 'brma'
qqnorm(
  y,
  type = "rstudent",
  unit = "estimate",
  conditioning_depth = "marginal",
  envelope = TRUE,
  conf_level = 95,
  bonferroni = FALSE,
  reps = 1000,
  smooth = TRUE,
  xlim,
  ylim,
  xlab,
  ylab,
  plot_type = "base",
  max_samples = Inf,
  ...
)
```

Arguments

y	a fitted brma object.
type	the type of standardized residuals to use. Options are:

- `"rstudent"` (alias: `"LOO-PIT"`; default): Leave-one-out probability integral transform residuals. Works for normal outcome models, including selection models, and, as a descriptive normal approximation, for binomial and Poisson GLMMs. This is the recommended type for Bayesian models as it properly accounts for estimation uncertainty and leverage. Note: requires that loo has been computed (see `add_loo`).
- `"rstandard"`: Internally standardized residuals using the hat matrix. Only available for normal outcome models without selection (weightfunction) bias adjustment.

`unit` output unit. Only `"estimate"` is implemented currently.

`conditioning_depth`
residual conditioning depth. Options are `"marginal"`, `"cluster"`, and `"estimate"`. It does not apply to LOO-PIT residuals, whose metadata record the deletion unit and retained context instead; leave it omitted for those residuals.

`envelope` logical indicating whether to add a pseudo confidence envelope to the plot. The deterministic envelope uses Beta order-statistic quantiles transformed to the standard normal scale, providing a reference under a correctly specified model. Defaults to `TRUE`.

`conf_level` finite numeric value strictly between 0 and 100 specifying the confidence level for the envelope bounds. Defaults to `95`.

`bonferroni` logical indicating whether to apply Bonferroni correction to the envelope, so it can be regarded as a simultaneous confidence region for all k residuals. Defaults to `FALSE`.

`reps` retained for compatibility with earlier simulation-based envelopes. The current envelope is deterministic and does not use this argument.

`smooth` logical indicating whether to smooth the envelope bounds using Friedman's SuperSmoother (`supsmu`). Defaults to `TRUE`.

`xlim` x-axis limits. If not specified, limits are computed from data.

`ylim` y-axis limits. If not specified, limits are computed from data.

`xlab` title for the x-axis. If not specified, defaults to `"Theoretical Quantiles"`.

`ylab` title for the y-axis. If not specified, defaults to `"Sample Quantiles"`.

`plot_type` whether to use a base plot `"base"` or ggplot2 `"ggplot"` for plotting. Defaults to `"base"`.

`max_samples` maximum posterior draws used to compute known-`V` internally standardized residuals. Defaults to `Inf`. Finite values select draws deterministically and are ignored for other residual types.

`...` additional graphical arguments to customize the plot appearance:

pch point symbol (default: 21, filled circle)

col point border color (default: "black")

bg point fill/background color (default: "#A6A6A6")

cex point size multiplier for base graphics (default: 1)

size point size for ggplot2 (default: 2)

shade fill color for the envelope region (default: "grey90"). Set to `NA` to suppress shading.

col.envelope color for envelope boundary lines (default: "grey50")

col.refline color of the diagonal reference line (default: "black")

main character string for plot title (default: no title)

as_data if TRUE, returns plot data instead of creating the plot

Details

The normal QQ plot is a diagnostic tool for assessing whether the standardized residuals follow a standard normal distribution, as expected under a correctly specified model. Each point represents one estimate, plotted at coordinates $(\Phi^{-1}(p_i), z_{(i)})$ where p_i are the plotting positions and $z_{(i)}$ are the sorted standardized residuals. Each retained input row contributes one point, irrespective of its likelihood weight. Weights affect fitting and the residual calculation; they do not replicate points or multiply residual values in this plot.

The default residual type is `"rstudent"` (LOO-PIT), which differs from `metafor::qqnorm` (which defaults to `"rstandard"`). LOO-PIT residuals are preferred for normal outcome models because they properly account for estimation uncertainty via leave-one-out cross-validation.

The confidence envelope is computed in closed form from the distribution of normal order statistics. For the i-th sorted residual, $\Phi(Z_{(i)})$ follows a beta distribution with shape parameters i and $k+1-i$. Pointwise quantile bounds are transformed back with Φ^{-1}. When `smooth = TRUE`, bounds are smoothed using Friedman's SuperSmoother. When `bonferroni = TRUE`, the confidence level is adjusted for simultaneous inference across all k comparisons.

For correlated known-V `brma.mv()` `"rstandard"` residuals, this envelope remains an independent order-statistic reference band. It is descriptive and does not model residual correlation induced by known sampling covariance.

For binomial and Poisson GLMMs the LOO-PIT residuals are a descriptive normal approximation on the effect-size scale rather than a PIT of the fitted discrete likelihood (see `residuals.brma`); the default y-axis label says so.

Value

For `plot_type = "base"`, returns an invisible list with components `x` (theoretical quantiles) and `y` (sorted residuals). For `plot_type = "ggplot"`, returns a ggplot object.

If `as_data = TRUE`, returns a list with all computed plot data including: `x`, `y`, `points`, `refline`, `envelope`, `xlim`, `ylim`, `xlab`, and `ylab`.

See Also

`rstudent.brma()`, `rstandard.brma()`, `residuals.brma()`, `funnel.brma()`, `radial.brma()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- brma(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
```

```
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )
  fit <- add_loo(fit)

  qqnorm(fit)
  qqnorm(fit, type = "rstandard")
  qqnorm(fit, envelope = FALSE)
  qqnorm(fit, plot_type = "ggplot")
  qqnorm(fit, pch = 19, col = "blue", shade = "lightblue")
}

## End(Not run)
```

`radial.brma` *Radial (Galbraith) Plot for brma Object*

Description

`radial.brma` creates a radial (Galbraith) plot for a fitted brma object. The plot displays each study as a point with precision on the x-axis and the standardized effect size on the z-axis. Without centering, a line from the origin through any point has slope equal to the observed effect size. With centering, slopes are relative to the pooled estimate. An arc on the right side maps standardized values back to the effect size scale.

Usage

```
## S3 method for class 'brma'
radial(
  x,
  center = FALSE,
  xlim,
  zlim,
  xlab,
  zlab,
  atz,
  aty,
  steps = 7,
  level = 95,
  digits = 2,
  transf,
  targs,
  plot_type = "base",
  ...
)
```

```
## S3 method for class 'brma'
galbraith(x, ...)
```

Arguments

x	a fitted brma object. Must be an intercept-only model (no moderators or scale regression).
center	logical indicating whether to center the plot at the pooled estimate. When TRUE, the pooled estimate is shifted to zero on the z-axis. Defaults to FALSE.
xlim	x-axis limits. If not specified, limits are computed from data.
zlim	z-axis limits. If not specified, limits are computed from data.
xlab	title for the x-axis. If not specified, a default label is used.
zlab	title for the z-axis. If not specified, a default label is used.
atz	numeric vector of positions for tick marks on the z-axis (left vertical axis). If not specified, positions are chosen automatically.
aty	numeric vector of effect size values to mark on the y-axis arc (right side). If not specified, values are chosen automatically.
steps	integer specifying the number of tick marks on the y-axis arc when aty is not specified. Defaults to 7.
level	numeric value between 0 and 100 specifying the confidence level for the confidence band and pooled-effect CI arc. Defaults to 95.
digits	integer specifying the number of decimal places for y-axis arc labels. Defaults to 2.
transf	optional transformation function applied to the y-axis arc labels (e.g., transf = exp when effect sizes are on a log scale).
targs	optional list passed to transf as its second argument, transf(x, targs), as in **metafor** (e.g., transf = metafor::transf.exp.int, targs = list(tau2 = 0)).
plot_type	whether to use a base plot "base" or ggplot2 "ggplot" for plotting. Defaults to "base".
...	additional graphical arguments to customize the plot appearance: **pch** point symbol (default: 21, filled circle) **col** point border color (default: "black") **bg** point fill/background color (default: "#A6A6A6") **cex** point size multiplier for base graphics (default: 1) **size** point size for ggplot2 (default: 2) **las** axis-label style for base graphics (default: 1) **back** background color for the confidence band (default: "grey90"). Set to NA to suppress. **arc.res** integer specifying the number of line segments used to draw arcs (default: 100) **main** character string for plot title (default: no title) **as_data** if TRUE, returns plot data instead of creating the plot

Details

The radial (Galbraith) plot transforms each study's effect size and standard error into a point in precision-standardized space:

- x-axis: precision = $1/\sqrt{v_i + \hat{\tau}_i^2}$
- z-axis: standardized effect = $y_i/\sqrt{v_i + \hat{\tau}_i^2}$

For ordinary and specialized multilevel models, the marginal heterogeneity is common and $\hat{\tau}_i = \hat{\tau}$. Random-formula models use each row's marginal random-effect standard deviation. Known sampling covariance uses its diagonal for v_i; plotted points can therefore be dependent.

For a single normal model, the sloped parallelogram is the usual marginal Gaussian reference band around the pooled-effect line. For GLMMs it is a continuity-corrected effect-size approximation; for selection, PET/PEESE, and product-space models it remains a bias-adjusted posterior-mean reference rather than an exact envelope of the fitted observation law. The arc on the right side allows reading individual effect sizes by projecting from the origin through a point to the arc; when `center = TRUE`, the plotted slope is relative to the pooled effect.

This function requires an intercept-only model without scale regression or likelihood weights. Likelihood-weighted fits do not have a single radial precision target because the observational variance and power-likelihood information scales differ.

`galbraith()` is a same-argument alias for `radial()`.

Value

`radial.brma` returns `NULL` invisibly if `plot_type = "base"` or a ggplot object if `plot_type = "ggplot"`.

If `as_data = TRUE`, returns a list with the computed plot data including: `points`, `refline`, `band`, `arc`, `arc_ticks`, `ci_arc`, `ci_ticks`, `xlim`, and `zlim`. Returned arc coordinates use plain trigonometry; both renderers recompute arcs for their aspect ratio, so the data do not encode final device geometry.

References

Galbraith, R. F. (1988). Graphical display of estimates having differing standard errors. *Technometrics*, 30(3), 271-281.

Galbraith, R. F. (1994). Some applications of radial plots. *Journal of the American Statistical Association*, 89(428), 1232-1242.

See Also

`funnel.brma()`, `pooled_effect.brma()`, `pooled_heterogeneity.brma()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE) &&
    requireNamespace("metafor", quietly = TRUE)) {
  data(dat.bcg, package = "metadat")
  dat <- metafor::escalc(
```

```
    measure = "RR",
    ai      = tpos,
    bi      = tneg,
    ci      = cpos,
    di      = cneg,
    data    = dat.bcg
  )

  fit <- brma(yi = yi, vi = vi, data = dat, measure = "RR")
  radial(fit)
  radial(fit, center = TRUE)
  radial(fit, plot_type = "ggplot")
  galbraith(fit)
}

## End(Not run)
```

`random_effect_formula_tags` *Random-Effect Formula Structure Tags*

Description

`brma.mv()` random-effect formulas use BayesTools random-effect formula structure tags to describe the covariance structure of random coefficients. These tags are formula specials parsed by `BayesTools::random_effects_formula()`, not ordinary user-facing functions.

Details

Use tags as bare names inside the `random` formula supplied to `brma.mv()`, for example `random = ~ cs(comp | study)`. They are parsed from the formula call and are not called as normal R functions, so users should not write `RoBMA::cs()` or `BayesTools::cs()`. RoBMA forwards the formula to `BayesTools::random_effects_formula()`, stores the parsed BayesTools random-effect metadata, and uses that metadata for JAGS compilation, prior matching, prediction, and summary labels.

BayesTools distinguishes random-coefficient formulas from structured index specifications. For `id()`, `diag()`, and `us()` / `un()`, the left side of `|` is a coefficient formula. It can contain an intercept, continuous or factor slopes, and interactions. As in an ordinary random-slope formula, `1`, `0`, and `-1` include or remove the random intercept. Plain `(expr | group)` defaults to `us(expr | group)`, while `expr || group` is the diagonal shorthand.

Factor coding inside a random-coefficient block is not selected by removing the intercept. BayesTools reuses an already established fixed-factor contrast by default and lets a `random_block` set an explicit block-specific basis through `contrasts`. For example, `us(0 + group | study)` together with `random_block(contrasts = c(group = "independent"))` creates one correlated random coefficient per `group` level and no random intercept. Without the independent contrast override, `0 + group` preserves the resolved contrast basis instead of forcing raw level indicators.

In contrast, `cs()`, `hcs()`, `ar1()` / `ar()`, `har()`, and `car()` own a structure-specific index basis. Their left side names index columns rather than coefficients. These tags reject explicit `1`, `0`, and `-1`, and their blocks do not accept `random_block(contrasts = ...)`.

Supported structure tags:

- `id()`: independent random-coefficient columns with one shared standard deviation.
- `diag()`: independent random-coefficient columns with one standard deviation per column. `||` is shorthand for this structure.
- `us()` / `un()`: one standard deviation per random-coefficient column and an unrestricted correlation matrix. Plain `|` syntax defaults to this structure.
- `cs()`: compound symmetry over one or more discrete index columns; index levels share one standard deviation and one common pairwise correlation.
- `hcs()`: heterogeneous compound symmetry over one or more discrete index columns; levels have separate standard deviations and one common pairwise correlation.
- `ar1()` / `ar()`: homogeneous autoregressive order-1 covariance over exactly one discrete index; levels share one standard deviation and correlations decay as $\mathrm{cor}^{|i-j|}$ over the resolved order.
- `har()`: heterogeneous autoregressive order-1 covariance; index levels have separate standard deviations and AR(1) correlations.
- `car()`: continuous-time autoregressive covariance; use exactly one finite numeric/integer coordinate or an ordered factor with numeric level labels, as in `car(time | study)`. Correlations are cor^{d} based on actual absolute distances, and cor is constrained to $[0, 1)$.
- `random()` / `re()`: explicit random-effect block wrapper. By default this uses an unstructured covariance, and it can be useful when specifying an explicit `name` for prior matching and summaries.

Under `diag()`, `us()`, and plain bars, the contrast columns of a mean-difference or orthonormal factor term are one exchangeable unit: they share one standard deviation and are uncorrelated with each other, while `us()` keeps the unit's correlations with the other columns. A `us()` block that is a single such unit therefore has no correlation parameter, and a `us()` block cannot hold two such terms with three or more levels.

The discrete structured tags `cs()`, `hcs()`, `ar1()`, and `har()` accept factor, character, numeric/integer, or logical index columns. Existing factor level order is retained; otherwise sorted unique values determine the levels and, for AR(1), their order. `cs()` and `hcs()` may combine multiple columns with `+`; BayesTools forms the observed interaction in lexicographic order. `ar1()` and `har()` require exactly one index column. AR1/HAR warns when numeric labels in a character or unordered factor resolve to a non-increasing numeric order, such as `1, 10, 2`, while preserving that order. Use numeric values for numeric ordering, or an ordered factor with explicit levels to confirm another intended order.

`hcs(index | group)` and `us(0 + index | group)` are different models even when both produce one standard deviation per displayed level. HCS constrains every pair to the same correlation. US estimates an unrestricted correlation matrix among the selected coefficient columns and its factor basis must be specified through contrast metadata when it matters. Alias tags are normalized before model compilation: `un()` to `us()`, `ar()` to `ar1()`, and unqualified `random()` / `re()` to `us()` unless a covariance override changes the structure. In these wrappers, `covariance = "cs", hom = FALSE` resolves to `hcs()`, and `covariance = "ar1", hom = FALSE` resolves to `har()`. Any explicit

covariance structure in `prior_random()` must match the resolved structure. Direct `cs()` / `ar1()` tags retain their homogeneous meaning.

`brma.mv()` also accepts metafor-style known group covariance through its `R` and `Rscale` arguments. This covariance is attached to random-effect grouping levels and is distinct from the known sampling covariance `V`. RoBMA translates `R` internally to BayesTools random-effect group covariance metadata; users should continue to use `R`/`Rscale` in `brma.mv()` rather than BayesTools-native `group_covariance` arguments. Current support is limited to random-intercept blocks such as `random = ~ 1 | study`. Supported one-to-one known-`R` blocks in known-`V` normal models can be compiled as marginalized diagonal row-variance terms; other supported known-`R` blocks remain sampled.

The scalar-correlation structures `cs()`, `hcs()`, `ar1()` / `ar()`, and `har()` use uniform defaults over their complete valid raw-correlation intervals: `(-1 / (K - 1), 1)` for CS/HCS and `(-1, 1)` for AR1/HAR, where `K` is the number of resolved index levels. CAR uses `Uniform(0, 1)` because its correlation-decay parameter is one-sided. These defaults therefore do not silently exclude negative correlations when the structure permits them.

A list of formulas defines top-level random-effect components:

```
random = list(
  study  = ~ cs(comp | study),
  design = ~ cs(comp | design)
)
```

A bare formula or unnamed one-entry list has no redundant user-facing component prefix. An explicitly named one-entry list retains its name. In a list with two or more formulas, missing names are generated as `component 1`, `component 2`, and so on. These component names are used by RoBMA summaries and by component-specific scale models. Explicit `name =` arguments inside `random()` or `re()` define the final random-effect block name used by `prior_random()` block overrides.

Public parameter names:

BayesTools stores concrete coordinates, selectable semantic quantities, and aliases in one versioned parameter map. Its canonical random-effect names use `(formula) owner: quantity(arguments)`. RoBMA maps these to meta-analytic I/O names: a sole random intercept is `tau` for a bare block and `study: tau` for a named block, while a non-intercept coefficient remains explicit, for example `study: tau(x)`. The owner-free `tau` shorthand is accepted only when it resolves uniquely. Bare formulas and unnamed one-entry lists omit a redundant owner; explicitly named one-entry lists and multi-component models retain it.

Public correlations are named `rho`. Automatic total-variance allocations expose `tau_total`, `tau2_total`, and `tau2_prop(...)`. Mean-variance allocations within heterogeneous blocks expose `tau_common`, `tau2_common`, `tau2_mult(...)`, and `tau_mult(...)`. BayesTools' general names and compact covariance-construction coordinates remain internal to RoBMA. These meta-analytic names are the supported selectors for summaries, plotting, density estimation, and hypotheses.

See Also

`brma.mv()`, `random_effect_prior_specification`, `prior_random`, `random_covariance`, `BayesTools::random_effect`

Examples

```
## Not run:
fit <- brma.mv(
  yi      = yi,
  V       = V,
  random  = ~ cs(outcome | study),
  data    = dat,
  measure = "GEN",
  prior_unit_information_sd = 1
)

fit_network <- brma.mv(
  yi = yi,
  V  = V,
  random = list(
    study  = ~ cs(comp | study),
    design = ~ cs(comp | design)
  ),
  data    = dat,
  measure = "OR"
)

group_prior <- prior_factor(
  distribution = "normal",
  parameters   = list(mean = 0, sd = 1),
  contrast     = "independent"
)
random_prior <- prior_random(
  study = random_block(
    contrasts = c(group = "independent")
  )
)
fit_group <- brma.mv(
  yi = yi, V = V, ni = ni,
  mods = ~ 0 + group,
  random = ~ us(0 + group | study),
  prior_mods = group_prior,
  prior_heterogeneity = random_prior,
  data = dat, measure = "GEN",
  prior_unit_information_sd = 1
)

## End(Not run)
```

---

ranef *Random Effects*

Description

Extracts the estimated random effects (deviations from the fixed-effect predictions) from a fitted model.

Usage

```
ranef(object, ...)
```

Arguments

object	a fitted model object
...	additional arguments passed to methods

Value

Method-specific return value, typically posterior samples of the random effect deviations.

See Also

blup(), predict.brma()

---

ranef.brma *Random Effects for brma Objects*

Description

Extracts random effect deviations from a fitted brma object. These are posterior-sample offsets from the fixed-effect predictions, analogous to random-effect deviations returned by metafor::ranef().

Usage

```
## S3 method for class 'brma'
ranef(
  object,
  bias_adjusted = FALSE,
  probs = c(0.025, 0.975),
  component = "all",
  simplify = TRUE,
  expand = TRUE,
  ...
)
```

Arguments

object	a fitted brma object
bias_adjusted	whether to adjust for publication bias. Defaults to FALSE. See blup.brma for details.
probs	quantiles of the posterior distribution to be displayed. Defaults to c(.025, .975) for 95% credible intervals.

`component` random-effect component to return. Defaults to `"all"`. For decomposed models, use a component name such as `"cluster"`, `"estimate"`, or a `brma.mv()` random-effect block name. Use `"total"` for the summed random-effect deviation.

`simplify` whether `component = "all"` should return a single `brma_samples` object instead of a one-element list. Defaults to `TRUE`, matching standard 2-level `brma()` behavior.

`expand` whether to repeat random-effect contributions for every fitted observation. Defaults to `TRUE` (one column per estimate; the cluster block of a multilevel model labels its columns `u_cluster[<cluster>|<estimate>]`), except for `brma.mv()` random-formula models, where it defaults to `FALSE`. `FALSE` returns one column per unique grouping-level contribution in the fitted grouping order used by `metafor::ranef()` (`u_cluster[<cluster>]` for a multilevel model). Nested groups follow their original outer-to-inner order. Indicator-coded random coefficients retain one column per observed grouping-level and coefficient combination. Set to `TRUE` for observation-aligned output.

`...` additional arguments forwarded to `predict.brma` for supported options such as `conditional`. `newdata`, `type`, `quiet`, `output_measure`, and `transform` are controlled by this method.

Details

Random effects are computed as the difference between the true effects (BLUPs) and the fixed-effect predictions:

$$u_i = \hat{\theta}_i - \hat{\mu}_i$$

For standard (2-level) models, returns a single `brma_samples` object with the estimate-level random effects.

For multilevel (3-level) models, returns a list with two `brma_samples` matrices:

`cluster` Cluster-level random effects ($\gamma_j \cdot \tau_{between}$), representing between-cluster deviations from the fixed effects.

`estimate` Estimate-level random effects ($\theta_i - \mu_i - \gamma_j \cdot \tau_{between}$), representing within-cluster deviations from the cluster means.

For `brma.mv()` random-formula models, decomposes the Gaussian conditional means by random-effect block. The result is invariant to whether each block was sampled or marginalized during fitting. If there is only one block and `simplify = TRUE`, returns a single `brma_samples` object. Multiple blocks are returned in one flat list under their canonical fitted block names; no additional location layer or component-name prefix is added. Unique-level output is available when a block's contribution is constant within each grouping level or observed grouping-level and indicator- coefficient combination. Other random slopes generally require `expand = TRUE`. When the random-effect standard deviation of a block varies across the rows of one returned cell (a grouping level, or a grouping level and indicator coefficient) through a `scale` formula (also for the cluster component of a clustered model with a `scale` formula, whether normal, GLMM, or with a weight function), the unique-level output is refused with an error of class `RoBMA_ranef_expand_required` (also of class `RoBMA_unsupported_request`); use `expand = TRUE`. Only the components that are returned are judged: `component = "estimate"` of a clustered model and a block name whose own scale

does not vary are returned, whereas "all" and "total" include every block. Summed component = "total" output across multiple blocks also requires expand = TRUE because different blocks need not share grouping levels.

Value

A brma_samples object for a single selected/simplified component, or a named list of brma_samples objects for decomposed component = "all" output. Use summary() on a single component (as.data.frame() of it returns its draws), or as.data.frame() on the list, to obtain the displayed summary table in long form, or as.data.frame(format = "list") to retain separate component tables.

See Also

blup.brma(), predict.brma(), pooled_effect()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- brma(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )

  ranef(fit)
}

## End(Not run)
```

---

regplot.brma *Regression Plot (Bubble Plot) for brma Object*

Description

regplot.brma creates a regression plot (also known as bubble plot) for a fitted brma object with location or scale moderators. The plot displays observed effect sizes against a moderator variable, with point sizes proportional to study precision.

Usage

```
## S3 method for class 'brma'
regplot(
  x,
  mod = NULL,
  pred = TRUE,
  ci = TRUE,
  pi = FALSE,
  si = FALSE,
  level = 95,
  at = NULL,
  digits = 2,
  transf = NULL,
  atransf = NULL,
  targs = NULL,
  refline = NULL,
  psize = NULL,
  plim = c(0.5, 3),
  by = NULL,
  legend = TRUE,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  sampling_bias = TRUE,
  sei = NULL,
  max_samples = 10000,
  plot_type = "base",
  as_data = FALSE,
  ...
)
```

Arguments

x	a fitted brma object with location or scale moderators
mod	index or name of the moderator variable to plot on the x-axis. If not specified and only one moderator is present, that moderator is used. If multiple moderators are present, this argument is required.
pred	logical; whether to show the prediction line. Defaults to TRUE.
ci	logical; whether to show credible interval bands. Defaults to TRUE. See Details for the quantile estimator of the bands.
pi	logical; whether to show prediction interval bands. Defaults to FALSE.
si	logical; whether to show sampling interval bands. Defaults to FALSE. The sampling interval shows the expected range of observed effect sizes, incorporating both heterogeneity (tau) and a representative level of sampling error (median SE across studies by default). When the model includes publication bias ad-

	justments and sampling_bias = TRUE, the sampling interval incorporates the expected distortion from the selection process.
level	numeric; credible/prediction interval level in percent, from 50 to 99.9. Defaults to 95.
at	numeric vector; for continuous moderators, values at which to evaluate the prediction. If not specified, uses a sequence across the observed range. Continuous evaluation values are sorted with all associated predictions and intervals; duplicate evaluations are retained.
digits	integer; number of decimal places for labels. Defaults to 2.
transf	function; transformation to apply to the y-axis (effect sizes). Defaults to NULL (no transformation).
atransf	reserved for axis-label transformations. Currently not implemented and must be NULL.
targs	reserved for additional transformation arguments. Currently not implemented and must be NULL.
refline	numeric; position of horizontal reference line. Defaults to NULL (no reference line).
psize	numeric vector or NULL; point sizes for each study. If scalar, it is recycled to all studies. If NULL (default), sizes are computed from the inverse sampling variance, mapped to plim. Equal precisions use the lower endpoint; zero SEs receive the established maximum priority when positive SEs are also present.
plim	numeric vector of length 2; range for point sizes. Defaults to c(0.5, 3).
by	character; name of a moderator variable to use for separate lines/colors. Defaults to NULL. If omitted and the selected moderator enters exactly one two-way interaction, the other variable in the interaction is used automatically. Continuous by moderators are shown at mean - SD, mean, and mean + SD. Grouping by sei or vi supplies the pointwise precision when neither precision measure is plotted; an explicit sei reference is then not allowed. The companion precision measure (sei versus vi) is not selected automatically; an explicitly inconsistent companion grouping grid is unavailable.
legend	logical; whether to show legend when by is specified. Defaults to TRUE.
xlab	character; x-axis label. Defaults to the moderator name.
ylab	character; y-axis label. Defaults to "Observed Effect Size".
xlim	numeric vector of length 2; x-axis limits. Defaults to data range.
ylim	numeric vector of length 2; y-axis limits. Defaults to data range.
sampling_bias	whether publication bias should be incorporated into plotted predictions and sampling intervals. Defaults to TRUE. For PET/PEESE models, this includes the expected regression bias in predictions. For selection models, sampling-bias adjustment applies to sampling intervals when si = TRUE; the mean prediction is unchanged. Selected sampling intervals require a supported scalar selection law; joint configurations needing retained-source or multivariate-event integration are unavailable. Use sampling_bias = FALSE for bias-adjusted intervals. Conditioning every applicable selection source cancels positive weights and uses the ordinary Gaussian interval law. This Gaussian case permits zero pointwise SE; active selection kernels still require strictly positive pointwise SE.

`sei` single positive numeric value used as the reference standard error for sampling-bias and sampling-interval calculations. Defaults to the median observed standard error. When `mod = "sei"` or `mod = "vi"`, the plotted moderator supplies the pointwise standard error and an explicit `sei` override is not allowed. A grouping moderator `by = "sei"` or `by = "vi"` also supplies pointwise precision and disallows an explicit `sei` override. The companion precision column is synchronized with the plotted or grouped values. When precision is neither plotted nor grouped and no override is supplied, its existing representative predictor value supplies the precision.

`max_samples` maximum number of posterior samples used for prediction summaries and interval bands. Defaults to `10000`. Use `Inf` to use all posterior samples.

`plot_type` character; whether to use base R graphics (`"base"`) or ggplot2 (`"ggplot"`). Defaults to `"base"`.

`as_data` logical; if `TRUE`, returns plot data instead of creating plot. Defaults to `FALSE`.

`...` additional graphical arguments:

main character string for plot title

pch point symbol (default: 21)

col point border color (default: "black")

bg point fill color (default: "#A6A6A6")

lcol line color (default: "black")

lwd line width (default: 2)

shade CI/PI/SI band shading (default: TRUE)

col.ci CI band color (default: "gray70")

col.pi PI band color (default: "gray85")

col.si SI band color (default: "gray92")

alpha.ci CI band transparency (default: 0.4)

alpha.pi PI band transparency (default: 0.2)

alpha.si SI band transparency (default: 0.15)

jitter jitter amount for categorical moderators (default: 0.2)

box.width box width for categorical interval summaries (default: 0.5)

Details

The regression plot (bubble plot) is a standard visualization for meta-regression results. It displays:

- Observed effect sizes (y-axis) against moderator values (x-axis)
- Point sizes proportional to study precision (inverse variance)
- Prediction line showing the estimated regression relationship
- Confidence bands showing uncertainty in the mean prediction
- Optional marginal prediction bands showing the expected range of new true effects
- Optional sampling interval bands showing expected range of observed outcomes

For continuous moderators, predictions are computed across the observed range of the moderator. For categorical moderators (factors), predictions are computed at each factor level with optional jittering of points. The other predictors are held at a reference design: continuous predictors of the location and scale formulas at their means and factors at their reference levels, predictors of random slopes in the same way (a random slope whose predictor is held at zero, such as a centered predictor at its mean, adds no variance to the prediction and sampling bands), and grouping variables at new row-unique levels. The line and bands describe the regression at this reference design, not an average over the observed predictor values, and the prediction and sampling bands are pointwise rather than a joint band for one random regression curve. For PET/PEESE models with `sampling_bias = TRUE`, the line and its credible band include the expected bias at the reference standard error (`sei`); `sampling_bias = FALSE` shows the bias-adjusted line. A moderator that appears only in the scale formula produces a constant mean prediction while prediction and sampling intervals reflect its effect on heterogeneity.

The credible interval band of a continuous moderator (`ci = TRUE`) is formed by pointwise Harrell-Davis quantiles (at the probabilities implied by `level`) of the posterior draws of the mean prediction at each grid value; they are computed by `BayesTools::harrell_davis_quantile()` on the model scale, before `transf` is applied. The Harrell-Davis quantiles estimate the same quantiles as the empirical quantiles of the draws but smooth the kinks that the empirical quantiles show along the moderator. The bands remain pointwise, and, because every draw contributes, they can differ visibly from the empirical quantiles when few draws are used (`max_samples` below a few hundred). Near the edge of a point mass in the draws (e.g., draws of models that fix the plotted quantity in a model-averaged posterior) the band blends the point mass and the remaining draws, and extreme probabilities (e.g., `level = 99.9`) are not estimated more accurately than by the empirical quantiles; see `BayesTools::harrell_davis_quantile()` for these caveats. The credible intervals of categorical moderators use the empirical quantiles of the draws of each level. Prediction and sampling intervals are quantiles of a different distribution, the posterior predictive mixture over the draws, and are computed by a different estimator (mixture quantiles, not Harrell-Davis quantiles of the draws).

Prediction intervals marginalize newly realized random effects. Sampling intervals use the corresponding marginal response distribution by also adding representative sampling error; evaluating a fitted moderator value does not condition these bands on its fitted latent effect. For marginalized known-`R` random-effect blocks, `pi` and `si` are unavailable because the group-covariance multipliers belong to the fitted layout and `R_new` is unsupported. Use `pi = FALSE, si = FALSE` for the mean and credible-interval plot. For binomial and Poisson GLMMs, prediction intervals retain this latent-effect interpretation, but sampling intervals are only descriptive normal approximations on the derived effect-size scale. They are not intervals from the fitted discrete likelihood and do not use the prior-nuisance marginal GLMM response target from `predict()`.

The `by` argument allows displaying separate regression lines for different levels of a second moderator, useful for visualizing interactions. For `brma.mv()` known-`V` models, prediction and sampling bands are pointwise descriptive bands. They summarize row-wise marginal random-effect variation and, for sampling intervals, a representative sampling standard error. They are not full joint predictive regions for correlated known-`V` outcomes.

Value

`regplot.brma` returns `NULL` invisibly if `plot_type = "base"` or a ggplot object if `plot_type = "ggplot"`. If `as_data = TRUE`, returns a list with plot data components.

See Also

funnel.brma(), predict.brma()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE) &&
    requireNamespace("metafor", quietly = TRUE)) {
  data(dat.bcg, package = "metadat")
  dat <- metafor::escalc(
    measure = "RR",
    ai      = tpos,
    bi      = tneg,
    ci      = cpos,
    di      = cneg,
    data    = dat.bcg
  )

  fit <- brma(
    yi      = yi,
    vi      = vi,
    mods    = ~ ablat + year,
    data    = dat,
    measure = "RR"
  )
  regplot(fit, mod = "ablat")
  regplot(fit, mod = "year", pi = TRUE, si = TRUE)
  regplot(fit, mod = "ablat", plot_type = "ggplot")

  fit_cat <- brma(yi = yi, vi = vi, mods = ~ alloc, data = dat, measure = "RR")
  regplot(fit_cat)
}

## End(Not run)
```

`remove_selection_cache` *Remove a fitted model's selection cache*

Description

Removes retained selection-normalizer cache entries from a fitted object without changing its draws, inference, or ability to obtain more draws.

Usage

```
remove_selection_cache(object)
```

Arguments

object	a fitted brma object, including bselmodel.mv() and RoBMA() fits.

Details

Set RoBMA.options(selection.cache_retain = TRUE) before fitting or extending to retain exact and coarse cache entries. The default is FALSE. This option is checked for each RoBMA fit or update() call. Existing retained entries are used during extension even when retention is turned off; in that case the returned fit does not retain a new cache.

Retained entries are ordinary serialized data and survive saveRDS(). They are restored only with matching package code, native builds, R and JAGS. Incompatible or damaged cache data are discarded with a warning; extension proceeds with ordinary likelihood evaluation. Cache contents never replace the model, numerical integration controls, or JAGS random-number state. Reconstructed JAGS models can still require adaptation before collecting new samples; retaining the normalizer cache does not retain sampler tuning.

Each worker saves its own entries. Extension with the same worker count restores the corresponding worker cache. With a different worker count, previous worker caches are assigned round-robin without duplication, and entries that do not fit the destination's share of the memory budget are dropped. Increasing the worker count can leave some workers with an empty cache initially. Cache reuse can improve speed but is not guaranteed.

Successful capture releases the native cache. A retained object's compact cache payload is bounded by the fitted total native budget, apart from small R metadata overhead. The byte budget is not a limit on total R memory: capture, serialization and extension can temporarily hold both a snapshot and native storage, and other copies or saved fits can retain their own snapshots. Save or load operations can therefore require substantial memory.

Assign the result back to the object to release its reference to the retained data. Memory remains in use if another object references the same snapshot. This helper does not clear the current process's independent native cache; use selection_cache_info(clear = TRUE) for that. Future retention is controlled by selection.cache_retain.

Value

The fitted object with its retained selection cache removed.

Examples

```
## Not run:
RoBMA.options(selection.cache_retain = TRUE)
fit <- bselmodel.mv(yi, V = V, random = ~ 1 | study / esid, data = dat)
saveRDS(fit, "fit.rds")
fit <- update(readRDS("fit.rds"), sample_extend = 1000)
fit <- remove_selection_cache(fit)
RoBMA.options(selection.cache_retain = FALSE)

## End(Not run)
```

`residuals.brma` *Residuals for brma Objects*

Description

Computes residuals (observed minus fitted values) from a fitted brma object, with options for standardization.

Usage

```
## S3 method for class 'brma'
residuals(
  object,
  type = "outcome",
  unit = "estimate",
  conditioning_depth = "marginal",
  bias_adjusted = FALSE,
  max_samples = Inf,
  ...
)
```

Arguments

`object`	a fitted brma object
`type`	the type of residuals to compute. Options are: • `"outcome"` (default): Raw residuals (observed - fitted). Available for all model types. • `"pearson"`: Pearson (semi-standardized) residuals, dividing raw residuals by the SD of the corresponding outcome target: marginal includes all applicable heterogeneity, cluster depth includes within- cluster heterogeneity, and estimate depth uses sampling variance only. Only available for normal outcome models without selection (weightfunction) bias adjustment. • `"rstandard"`: Internally standardized projection residuals, divided by their corresponding SDs computed using the hat matrix. Only available for normal outcome models without selection (weightfunction) bias adjustment. • `"LOO-PIT"` (alias: `"rstudent"`): Leave-one-out probability integral transform (PIT) residuals computed via Pareto smoothed importance sampling. The LOO-CDF value for each observation is computed and transformed to standard normal quantiles via $\Phi^{-1}(u_i)$. Under a correctly specified model, these residuals should follow a standard normal distribution. This is the recommended standardized residual for Bayesian models as it properly accounts for estimation uncertainty and leverage. Available for normal outcome models and, as a descriptive normal approximation on the effect-size scale, for binomial and Poisson GLMMs (see Details). Note: This requires that the loo has been computed previously (see `add_loo()` function).
`unit`	output unit. Only `"estimate"` is implemented currently.

`conditioning_depth`
: conditioning depth for non-LOO residuals. `"marginal"` uses fixed effects only, `"cluster"` conditions on cluster-level random effects, and `"estimate"` conditions on the full estimate-level fitted value. LOO-PIT residuals always use the estimate-unit LOO target.

`bias_adjusted`
: whether residuals should be computed from bias-adjusted fitted values. Defaults to `FALSE`, which means residuals are computed as the difference between observed values and raw (biased) predictions including PET/PEESE terms. Set to `TRUE` to compute residuals from bias-corrected fitted values. Applies to outcome and Pearson residuals. Note that bias-adjusted residuals are not residuals in the traditional sense.

`max_samples`
: maximum posterior draws used for known-`V` `brma.mv()` marginal covariance computations in Pearson and internally standardized residuals. Defaults to `Inf`. Finite values deterministically thin draws across posterior row order.

`...`
: additional arguments.

Details

Raw residuals (`type = "outcome"`) are computed as:

$$e_i = y_i - \hat{\mu}_i$$

where y_i is the observed effect size and $\hat{\mu}_i$ is the fitted value (prediction from the fixed effects).

Pearson residuals (`type = "pearson"`) divide raw residuals by the marginal standard error:

$$r_i^{Pearson} = \frac{e_i}{\sqrt{v_i + \tau^2}}$$

where v_i is the sampling variance and τ^2 is the relevant heterogeneity component. Only available for normal outcome models without selection (weightfunction) bias adjustment.

Standardized residuals (`type = "rstandard"`) use the hat matrix to compute residual standard errors that account for the uncertainty in estimated coefficients:

$$z_i = \frac{e_i}{\sqrt{[(I - H)M(I - H)']_{ii}}}$$

where H is the hat matrix and M is the marginal variance-covariance matrix. Without additional likelihood weights, the GLS projection satisfies $HM = MH' = HMH'$, so the denominator equals $M_{ii} - [X(X'M^{-1}X)^{-1}X']_{ii}$, which is strictly smaller than the Pearson denominator $\sqrt{M_{ii}}$. This holds with no moderators too, where fitting the pooled effect alone removes $1/(1'M^{-1}1)$ from every residual variance, so standardized and Pearson residuals do not coincide. Only available for normal outcome models without selection (weightfunction) bias adjustment.

Likelihood `weights` affect fitting, the fitted projection, and PSIS deletion weights; they do not divide the observation variance used for diagnostic scaling. Pearson residuals and the predictive CDF/moments in `rstudent()` use the original outcome law. Internally standardized residuals combine the fitted weighted projection H with that law's covariance M, as in **metafor** diagnostics with custom fitting weights. RoBMA retains its Bayesian posterior averaging and LOO-PIT transformation; these are not identical to frequentist deleted-residual z-statistics. Each retained row produces

one residual, without multiplying or repeating it by its observation weight. Estimate-unit LOO removes the row's entire weighted likelihood contribution, for integer and non-integer weights alike.

LOO-PIT residuals (`type = "LOO-PIT"`) are the Bayesian equivalent of studentized deleted residuals (Vehtari et al. 2017). They are computed via leave-one-out probability integral transformation:

$$r_i = \Phi^{-1}(u_i)$$

where $u_i = \sum_s w_{is} F(y_i|\theta^{(s)})$ is the LOO-weighted CDF value, w_{is} are the normalized PSIS weights, and F is the cumulative distribution function of the estimate-unit outcome distribution under the leave-one-out fit. With likelihood weights, PSIS removes the weighted likelihood contribution while this CDF remains unpowered. For weighted Gaussian cluster models, the cluster effect is integrated given the retained weighted rows. These retained moments are paired with the integrated powered-likelihood PSIS ratios; full-data sampled cluster effects are not reused. Numerically unresolved retained moments raise `RoBMA_weighted_gaussian_unavailable`, a `RoBMA_numerical_unavailable` condition. Lower and upper probabilities are accumulated separately on the log scale, and the smaller tail is passed directly to the normal quantile function. Therefore extreme residuals are not truncated by probability clipping. Under a correctly specified model, LOO-PIT residuals should follow a standard normal distribution. Unlike traditional standardized residuals, LOO-PIT residuals properly account for estimation uncertainty and leverage without requiring a hat matrix. This is the recommended method for standardized residuals in Bayesian meta-analysis.

For meta-regression models, fitted values incorporate moderator effects. For models without moderators, all fitted values equal the pooled effect.

For binomial and Poisson GLMMs the fitted predictive distributions are discrete, so the LOO-PIT residuals are a descriptive normal approximation on the effect-size scale: F is the normal distribution of the continuity-corrected log odds ratio or log incidence rate ratio with its approximate sampling standard error, as **metafor** computes residuals of GLMMs. They are not probability integral transforms of the fitted count likelihood, and the functions say so in a message (plots label the approximation).

For correlated known-`V` `brma.mv()` models, LOO-PIT residuals use the same conditional estimate-unit target as LOO, $p(y_i \mid y_{-i}, \theta)$. Random-formula `brma.mv()` models support marginal Pearson and internally standardized residuals through the marginal covariance $V + ZGZ'$ and estimate-depth residuals through the fitted existing-level random-effect target. Selection models retain their declared context and original publication event during deletion. Retained product weights cancel; best selection also retains the best p-value among the observed outcomes in that event. Pearson and hat-matrix standardized residuals remain unavailable for selection models; use LOO-PIT residuals for a likelihood-aware standardized diagnostic.

The residuals are computed separately for each posterior sample, naturally propagating uncertainty in model parameters to the residuals.

Value

A numeric vector of residual means, one per estimate.

References

Vehtari A, Gelman A, Gabry J (2017). "Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC." *Statistics and computing*, **27**(5), 1413–1432. doi:10.1007/s11222016-96964.

See Also

predict.brma(), blup.brma(), pooled_effect(), rstandard.brma(), rstudent.brma()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")

  fit <- bPET(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )

  # raw residuals (default)
  residuals(fit)

  # Pearson and internally standardized residuals
  residuals(fit, type = "pearson")
  residuals(fit, type = "rstandard")

  # LOO-PIT residuals require stored LOO
  fit <- add_loo(fit)
  residuals(fit, type = "LOO-PIT")

  # residuals from bias-adjusted predictions
  residuals(fit, bias_adjusted = TRUE)

  # check Pareto k diagnostics
  plot(loo(fit))
}

## End(Not run)
```

RoBMA *Robust Bayesian Model-Averaged Meta-Analysis*

Description

Fits a robust Bayesian model-averaged meta-analysis. The default ensemble averages across models with and without an effect, heterogeneity, and publication-bias adjustment.

Usage

```
RoBMA(
  yi,
  vi,
  sei,
  weights,
  ni,
  mods,
  scale,
  cluster,
  data,
  slab,
  subset,
  measure,
  effect_direction = "detect",
  prior_effect,
  prior_heterogeneity,
  prior_mods,
  prior_scale,
  prior_heterogeneity_allocation,
  prior_bias,
  prior_effect_null,
  prior_heterogeneity_null,
  prior_mods_null,
  prior_scale_null,
  prior_heterogeneity_allocation_null,
  prior_bias_null,
  standardize_continuous_predictors = TRUE,
  set_contrast_factor_predictors = "meandif",
  prior_unit_information_sd,
  rescale_priors = 1,
  prior_informed_field,
  prior_informed_subfield,
  model_type = "PSMA",
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  chains = 3,
  thin = 1,
  parallel = FALSE,
  autofit = FALSE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  seed = NULL,
  silent = TRUE,
  selection = BayesTools::selection_model(),
  selection_control = set_selection_likelihood_control(),
  ...
```

)

Arguments

yi	a vector of effect sizes, or a formula with the effect size on the left-hand side and location moderators on the right-hand side (for example yi ~ x1 + x2). If a formula is supplied, mods must not be specified.
vi	a vector of sampling variances. Either vi or sei must be supplied for normal models.
sei	a vector of standard errors. Either vi or sei must be supplied for normal models.
weights	an optional vector of finite positive likelihood weights. For normal/effect-size models, each weight powers the estimate likelihood. For constructors with GLMM raw-count input, each weight powers the paired two-arm likelihood for one study. An integer weight w repeats that conditional likelihood contribution w times, with the declared latent-effect and grouping structure held fixed. Positive fractional likelihood powers are also supported.
ni	an optional vector of sample sizes. Used for measure = "GEN" or when estimating the unit information standard deviation.
mods	an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in data and supports explicit columns and standard formula operators, including ^ as R's crossing operator with a literal whole exponent of at least 2: (x + z)^2 expands to x + z + x:z, and x^2 is not a quadratic term (x + x^2 is x). Precompute transformed predictors, such as a squared predictor, as columns in data; inline calls such as I(), poly(), spline constructors, and user functions are not supported.
scale	an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in data and follows the same explicit-column formula grammar as mods. Random-formula models also accept a named list targeting top-level random components or concrete random-effect blocks; see brma.mv().
cluster	an optional vector of cluster identifiers for multilevel meta-analysis.
data	an optional data frame containing the variables.
slab	an optional vector of study labels.
subset	an optional logical or numeric vector specifying a subset of data to be used. Subsetting precedes missing-row removal; retained-row outcome and predictor checks follow these steps. Type, shape, and existing extraction checks can still reject malformed inputs before subsetting. Row selection (by subset or missing-row removal) drops factor levels left without rows from mods, scale, and random-effect grouping factors. The index variable of a structured random-effect term (cs(), hcs(), ar1(), har() with one index variable) keeps its declared levels, which define the covariance structure; the variables of a multivariable cs(a + b | group) index follow the ordinary rule, because that index is built from their observed combinations. If an index variable that keeps its levels is also used in mods or scale and loses one of its levels to the selection, the fit stops with an error of class RoBMA_structured_index_level_removed (also RoBMA_data_error); remove the level from the data with droplevels() or use a separate column for the moderator.

`measure` a character string specifying the effect size measure. Normal/effect-size constructors require an explicit value and support `"SMD"`, `"ZCOR"`, `"RR"`, `"OR"`, `"HR"`, `"RD"`, `"IRR"`, and `"GEN"`. Use `"GEN"` only for general effect sizes without a known unit information standard deviation. GLMM raw-count constructors support only `"OR"` and `"IRR"` and default to `"OR"`.

`effect_direction`
direction used by publication-bias adjustments. `"positive"` assumes statistically significant positive estimates are more likely to be selected; `"negative"` mirrors the selection direction; `"detect"` infers the direction from the fitted data.

`prior_effect` prior distribution(s) for the alternative effect component(s).

`prior_heterogeneity`
prior distribution(s) for the alternative heterogeneity component(s).

`prior_mods` prior distribution(s) for alternative moderator components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `mods` is specified. For a formula without an intercept (`~ 0 + x`), an `intercept` entry other than a point prior at zero stops with an error of class `RoBMA_prior_intercept_conflict` (also of class `RoBMA_prior_error`). A named alternative `intercept` must be a single simple or point prior; it is the complete intercept prior and replaces the global effect mixture. Specify it in one place: null-list intercepts and explicit global effect declarations left without a consumer are refused. Ordinary term entries replace only their alternative component.

`prior_scale` prior distribution(s) for alternative scale-regression components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `scale` is specified. A named alternative `intercept` must be a single positive simple or point prior; it is the complete SD prior used by `log(intercept)`. Global heterogeneity declarations remain meaningful when they supply another random root or its inclusion weights; declarations replaced without a consumer are refused. Null-list intercept declarations are unavailable.

`prior_heterogeneity_allocation`
prior distribution(s) for the alternative cluster-level heterogeneity allocation component(s). This argument can be specified only when `cluster` is specified.

`prior_bias` prior distribution(s) for alternative publication-bias component(s), such as weight functions, PET, or PEESE.

Alternative prior arguments can be:

- A single prior distribution object (creates a mixture with one alternative)
- A list of prior distributions (creates a mixture with multiple alternatives)
- `NULL` or `FALSE` (omits the alternative hypothesis component)

See `publication_bias_prior_specification` for details on specifying publication-bias priors and `prior_specification` for details on specifying meta-analytic parameter priors.

`prior_effect_null`
prior distribution(s) for the null effect component(s).

prior_heterogeneity_null
: prior distribution(s) for the null heterogeneity component(s).

prior_mods_null
: prior distribution(s) for null moderator components. A named `intercept` entry, including `NULL` or `FALSE`, is unavailable: put the complete intercept prior in `prior_mods`, or use global prior arguments without a formula-list intercept. The same rule applies to `prior_scale_null`. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `mods` is specified. The null and the alternative prior of a factor term need the same contrast; a conflict stops with an error of class `RoBMA_prior_contrast_conflict` (also of class `RoBMA_prior_error`) that names the term and the remedies. The same rule applies to `prior_scale_null`.

prior_scale_null
: prior distribution(s) for null scale-regression components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `scale` is specified.

prior_heterogeneity_allocation_null
: prior distribution(s) for the null cluster-level heterogeneity allocation component(s). This argument can be specified only when `cluster` is specified.

prior_bias_null
: prior distribution(s) for null publication-bias component(s), usually `prior_none()`. See `publication_bias_prior_specification`.

  Null prior arguments can be:

  - A single prior distribution object (creates a mixture with one null)
  - A list of prior distributions (creates a mixture with multiple nulls)
  - `NULL` or `FALSE` (omits the null hypothesis component)

  Defaults to a point mass (spike) at zero for effect and heterogeneity parameters.

standardize_continuous_predictors
: logical. Whether to standardize continuous predictors. Defaults to `TRUE`.

set_contrast_factor_predictors
: character. How to set contrast for factor predictors. Defaults are constructor-specific and shown in each function usage. Single-model constructors use `"treatment"`; model-averaging constructors use `"meandif"`. Accepted values are `"treatment"`, `"meandif"`, `"orthonormal"`, and `"independent"`. Independent coding creates one coefficient per factor level. Random-coefficient blocks reuse the resolved fixed-factor contrast unless `random_block(contrasts = ...)` overrides it. A warning is issued when default or inherited independent contrasts make a fixed or random design overspecified. With `"meandif"` or `"orthonormal"` contrasts, a formula term that interacts a factor with a predictor whose own term is missing, such as `g:x` in `~ g + g:x` or `~ g / x`, stops the fit: the term codes `g` by level indicators, with one coefficient per level, on which mean-difference and orthonormal priors are not defined. Specify an independent prior for that term, e.g., `prior_mods = list("g:x" = prior_factor("normal", list(0, 1), contrast = "independent"))` (or `prior_scale` for a scale formula), include the missing term (`~ g * x`), or use `set_contrast_factor_predictors = "treatment"`.

`prior_unit_information_sd`
: numeric. The unit information standard deviation (σ_{unit}). Cannot be used together with `prior_informed_field`.

`rescale_priors`
: numeric. A scaling factor for supported prior distributions. Point and none priors are unchanged; supplied `prior_random()` scale architectures and publication-bias priors are not rescaled. The default PEESE prior separately uses its documented UISD adjustment, independently of `rescale_priors`. Defaults to 1.

`prior_informed_field`
: character. The field of the informed prior distributions. Omit to use the standard default prior specification; explicit `NULL` is invalid.

`prior_informed_subfield`
: character. The subfield of the informed prior distributions. Can only be specified together with `prior_informed_field`. Omit to use the field-specific default, such as `"Cochrane"` for `prior_informed_field = "medicine"`; explicit `NULL` is invalid.

`model_type`
: character string specifying predefined publication-bias model ensembles. One of:

  - `"PSMA"` (default): Full RoBMA-PSMA ensemble with 6 weight functions + PET + PEESE
  - `"6w"`: Six weight function models
  - `"2w"`: Two weight function models
  - `"PP"`: PET-PEESE models only

  Custom `prior_bias` replaces the preset alternative bias components. If `prior_bias` is omitted, `model_type` determines the default alternatives even when `prior_bias_null` is customized or omitted.

`sample`
: numeric. Number of MCMC samples to save. Defaults to `5000`.

`burnin`
: numeric. Number of burn-in iterations. Defaults to `2000`.

`adapt`
: numeric. Number of adaptation iterations. Defaults to `500`.

`chains`
: numeric. Number of MCMC chains. Defaults to `3`.

`thin`
: numeric. Thinning interval. Defaults to `1`.

`parallel`
: logical. Whether to run MCMC chains in parallel. Defaults to `FALSE`.

`autofit`
: logical. Whether to automatically extend the MCMC chains if convergence is not met. Defaults to `FALSE`.

`autofit_control`
: list of autofit control settings. See `set_autofit_control()` for details.

`convergence_checks`
: list of convergence check settings. See `set_convergence_checks()` for details.

`seed`
: numeric. Random seed for reproducibility. Defaults to `NULL`.

`silent`
: logical. Whether to suppress output. Constructors with no explicit default use `RoBMA.get_option("silent")` when `silent` is omitted. Model-averaging wrappers default to `TRUE` unless explicitly changed.

`selection` specification created by `selection_model()` for automatically constructed weight-function priors. The default integrates estimate-level random effects and the complete sampling error, and conditions on other random effects. Each source can instead be conditioned upon or integrated through the corresponding selection-model setting. The default `weight_rule = "product"` needs no publication groups and ignores `group`. Publication grouping is active only for `weight_rule = "best"`. Explicit `prior_bias` objects retain their own selection specification; this argument does not overwrite it. Conditioning on sampling variation retains the entire sampling-error realization, including in ordinary univariate models. This sampling setting requires omitted or unit observation `weights`.

`selection_control` numerical integration settings created by `set_selection_likelihood_control()`. These affect numerical evaluation, without changing the conditioning model or the vector weighting rule.

`...` additional advanced arguments. Fitting functions reject unused arguments; currently recognized internal arguments include `only_data`, `only_priors`, `is_JASP`, and `is_JASP_prefix`.

Details

`RoBMA()` uses product-space Bayesian model averaging. Inclusion Bayes factors and model-averaged estimates are obtained from mixture priors for effect, heterogeneity, moderators, scale regression, and publication-bias components.

By default, `model_type = "PSMA"` includes selection-model weight functions together with PET and PEESE publication-bias adjustments. Use `BMA()` for model averaging without publication-bias adjustment, or `brma()` for fitting a single meta-analytic model.

`RoBMA()` uses normal/effect-size input (`yi` with `vi` or `sei`). Raw-count GLMM model averaging is provided by `BMA.glmm()`.

`selection` configures every generated default weightfunction. Its default integrates estimate-level random effects and the complete sampling error, and conditions on other random effects. Explicit priors retain their nested `selection_model()` settings. Active selection branches must share the same conditioning choices. Only `weight_rule = "best"` branches require and must share a publication partition. Product branches ignore `group`; an ensemble without `"best"` needs no publication groups and accepts `group = NULL`. Branches can differ in weight-height priors, bins, and `weight_rule`. These fixed model choices do not introduce new inclusion parameters.

Product-space objects support predictive comparison with `add_loo()` and `add_waic()`. Bridge-sampling marginal likelihood via `add_marglik()` is not available for product-space model-averaging objects.

Value

A fitted object of class `c("RoBMA", "brma")`. The object contains checked `data`, checked `priors`, the JAGS `fit`, cached `summary`, and cached `coefficients`. It can be passed to `summary()`, `plot()`, `predict()`, `funnel()`, `add_loo()`, and related methods.

See Also

publication_bias_prior_specification, RoBMA.mv(), BMA(), brma(), bselmodel(), bPET(), bPEESE(), summary.brma(), plot.brma()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")

  fit <- RoBMA(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )

  summary(fit)
  plot(fit)
}

## End(Not run)
```

RoBMA.mv *Robust Bayesian Model-Averaged Multivariate Meta-Analysis*

Description

Fits robust Bayesian model-averaged multivariate and multilevel meta-analytic models with known sampling covariance and formula random effects.

Usage

```
RoBMA.mv(
  yi,
  V,
  ni,
  mods,
  scale,
  random,
  R = NULL,
  Rscale = "cor",
  data,
  slab,
  subset,
```

```
  measure,
  effect_direction = "detect",
  prior_effect,
  prior_heterogeneity,
  prior_mods,
  prior_scale,
  prior_bias,
  prior_effect_null,
  prior_heterogeneity_null,
  prior_mods_null,
  prior_scale_null,
  prior_bias_null,
  standardize_continuous_predictors = TRUE,
  set_contrast_factor_predictors = "meandif",
  prior_unit_information_sd,
  rescale_priors = 1,
  prior_informed_field,
  prior_informed_subfield,
  model_type = "PSMA",
  selection = BayesTools::selection_model(),
  selection_control = set_selection_likelihood_control(),
  known_v_parameterization = "auto",
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  chains = 3,
  thin = 1,
  parallel = FALSE,
  autofit = FALSE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  seed = NULL,
  silent = TRUE,
  ...,
  vi = NULL,
  sei = NULL
)
```

Arguments

yi	a vector of effect sizes, or a formula with the effect size on the left-hand side and location moderators on the right-hand side (for example `yi ~ x1 + x2`). If a formula is supplied, `mods` must not be specified.
V	a known working variance-covariance matrix, a list of block variance-covariance matrices, or an exact diagonal-plus-factor declaration created by `known_v_factor()`, used by `brma.mv()`. Ordinary covariance matrices from `metafor::vcalc()` can be supplied directly. Product selection requires no publication groups. For best-p-value selection, declare groups separately with `selection_model(weight_rule`

= "best", group = ...). Matrix symmetry uses base R's numerical tolerance. Accepted matrices are stored symmetrically by averaging off-diagonal pairs; the supplied object and its diagonal are unchanged. Dependency blocks that are block constant on nested partitions are represented exactly as a diagonal-plus-factor structure without a known_v_factor() declaration. Dimensions and row alignment are checked on the complete input; when rows are dropped (subset or missing values), the numerical covariance checks apply to the retained rows only, so acceptance says nothing about the entries of the dropped rows.

ni — an optional vector of sample sizes. Used for measure = "GEN" or when estimating the unit information standard deviation.

mods — an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in data and supports explicit columns and standard formula operators, including ^ as R's crossing operator with a literal whole exponent of at least 2: (x + z)^2 expands to x + z + x:z, and x^2 is not a quadratic term (x + x^2 is x). Precompute transformed predictors, such as a squared predictor, as columns in data; inline calls such as I(), poly(), spline constructors, and user functions are not supported.

scale — an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in data and follows the same explicit-column formula grammar as mods. Random-formula models also accept a named list targeting top-level random components or concrete random-effect blocks; see brma.mv().

random — Optional BayesTools random-effect formula or list of formulas. Omission or NULL means a fixed-effect model with no implicit heterogeneity term. Known sampling covariance V remains unchanged. Declare random before supplying heterogeneity priors or scale formulas.

R — optional known random-effect group covariance/correlation matrix, or a named list of matrices. Matrix row and column names identify grouping levels. This is distinct from the known sampling covariance V.

Rscale — scaling applied to each supplied R matrix before fitting. Supported values are "cor", "none", "cor0", "cov0", with metafor-compatible aliases TRUE, FALSE, 0, 1, 2, and 3. An unnamed scalar is recycled across all supplied matrices. A vector or list can specify one scaling value per named R entry, with exactly matching names. An unnamed one-element list is supported only for a single R matrix.

data — an optional data frame containing the variables.

slab — an optional vector of study labels.

subset — an optional logical or numeric vector specifying a subset of data to be used. Subsetting precedes missing-row removal; retained-row outcome and predictor checks follow these steps. Type, shape, and existing extraction checks can still reject malformed inputs before subsetting. Row selection (by subset or missing-row removal) drops factor levels left without rows from mods, scale, and random-effect grouping factors. The index variable of a structured random-effect term (cs(), hcs(), ar1(), har() with one index variable) keeps its declared levels, which define the covariance structure; the variables of a multivariable cs(a + b | group) index follow the ordinary rule, because that index is

built from their observed combinations. If an index variable that keeps its levels is also used in `mods` or `scale` and loses one of its levels to the selection, the fit stops with an error of class `RoBMA_structured_index_level_removed` (also `RoBMA_data_error`); remove the level from the data with `droplevels()` or use a separate column for the moderator.

`measure` a character string specifying the effect size measure. Normal/effect-size constructors require an explicit value and support `"SMD"`, `"ZCOR"`, `"RR"`, `"OR"`, `"HR"`, `"RD"`, `"IRR"`, and `"GEN"`. Use `"GEN"` only for general effect sizes without a known unit information standard deviation. GLMM raw-count constructors support only `"OR"` and `"IRR"` and default to `"OR"`.

`effect_direction`
direction used by publication-bias adjustments. `"positive"` assumes statistically significant positive estimates are more likely to be selected; `"negative"` mirrors the selection direction; `"detect"` infers the direction from the fitted data.

`prior_effect` prior distribution(s) for the alternative effect component(s).

`prior_heterogeneity`
prior distribution(s) for the alternative heterogeneity component(s).

`prior_mods` prior distribution(s) for alternative moderator components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `mods` is specified. For a formula without an intercept (`~ 0 + x`), an `intercept` entry other than a point prior at zero stops with an error of class `RoBMA_prior_intercept_conflict` (also of class `RoBMA_prior_error`). A named alternative `intercept` must be a single simple or point prior; it is the complete intercept prior and replaces the global effect mixture. Specify it in one place: null-list intercepts and explicit global effect declarations left without a consumer are refused. Ordinary term entries replace only their alternative component.

`prior_scale` prior distribution(s) for alternative scale-regression components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `scale` is specified. A named alternative `intercept` must be a single positive simple or point prior; it is the complete SD prior used by `log(intercept)`. Global heterogeneity declarations remain meaningful when they supply another random root or its inclusion weights; declarations replaced without a consumer are refused. Null-list intercept declarations are unavailable.

`prior_bias` prior distribution(s) for alternative publication-bias component(s), such as weight functions, PET, or PEESE.

Alternative prior arguments can be:

- A single prior distribution object (creates a mixture with one alternative)
- A list of prior distributions (creates a mixture with multiple alternatives)
- `NULL` or `FALSE` (omits the alternative hypothesis component)

See `publication_bias_prior_specification` for details on specifying publication-bias priors and `prior_specification` for details on specifying meta-analytic parameter priors.

prior_effect_null
: prior distribution(s) for the null effect component(s).

prior_heterogeneity_null
: prior distribution(s) for the null heterogeneity component(s).

prior_mods_null
: prior distribution(s) for null moderator components. A named `intercept` entry, including `NULL` or `FALSE`, is unavailable: put the complete intercept prior in `prior_mods`, or use global prior arguments without a formula-list intercept. The same rule applies to `prior_scale_null`. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `mods` is specified. The null and the alternative prior of a factor term need the same contrast; a conflict stops with an error of class `RoBMA_prior_contrast_conflict` (also of class `RoBMA_prior_error`) that names the term and the remedies. The same rule applies to `prior_scale_null`.

prior_scale_null
: prior distribution(s) for null scale-regression components. A single prior applies to all terms; a named list can specify term-specific components. This argument can be specified only when `scale` is specified.

prior_bias_null
: prior distribution(s) for null publication-bias component(s), usually `prior_none()`. See publication_bias_prior_specification.

  Null prior arguments can be:

  - A single prior distribution object (creates a mixture with one null)
  - A list of prior distributions (creates a mixture with multiple nulls)
  - `NULL` or `FALSE` (omits the null hypothesis component)

  Defaults to a point mass (spike) at zero for effect and heterogeneity parameters.

standardize_continuous_predictors
: logical. Whether to standardize continuous predictors. Defaults to `TRUE`.

set_contrast_factor_predictors
: character. How to set contrast for factor predictors. Defaults are constructor-specific and shown in each function usage. Single-model constructors use `"treatment"`; model-averaging constructors use `"meandif"`. Accepted values are `"treatment"`, `"meandif"`, `"orthonormal"`, and `"independent"`. Independent coding creates one coefficient per factor level. Random-coefficient blocks reuse the resolved fixed-factor contrast unless `random_block(contrasts = ...)` overrides it. A warning is issued when default or inherited independent contrasts make a fixed or random design overspecified. With `"meandif"` or `"orthonormal"` contrasts, a formula term that interacts a factor with a predictor whose own term is missing, such as `g:x` in `~ g + g:x` or `~ g / x`, stops the fit: the term codes `g` by level indicators, with one coefficient per level, on which mean-difference and orthonormal priors are not defined. Specify an independent prior for that term, e.g., `prior_mods = list("g:x" = prior_factor("normal", list(0, 1), contrast = "independent"))` (or `prior_scale` for a scale formula), include the missing term (`~ g * x`), or use `set_contrast_factor_predictors = "treatment"`.

prior_unit_information_sd
: numeric. The unit information standard deviation (σ_{unit}). Cannot be used together with `prior_informed_field`.

`rescale_priors` numeric. A scaling factor for supported prior distributions. Point and none priors are unchanged; supplied `prior_random()` scale architectures and publication-bias priors are not rescaled. The default PEESE prior separately uses its documented UISD adjustment, independently of `rescale_priors`. Defaults to 1.

`prior_informed_field`
character. The field of the informed prior distributions. Omit to use the standard default prior specification; explicit `NULL` is invalid.

`prior_informed_subfield`
character. The subfield of the informed prior distributions. Can only be specified together with `prior_informed_field`. Omit to use the field-specific default, such as `"Cochrane"` for `prior_informed_field = "medicine"`; explicit `NULL` is invalid.

`model_type` character string specifying predefined publication-bias model ensembles. One of:

- `"PSMA"` (default): Full RoBMA-PSMA ensemble with 6 weight functions + PET + PEESE
- `"6w"`: Six weight function models
- `"2w"`: Two weight function models
- `"PP"`: PET-PEESE models only

Custom `prior_bias` replaces the preset alternative bias components. If `prior_bias` is omitted, `model_type` determines the default alternatives even when `prior_bias_null` is customized or omitted.

`selection` specification created by `selection_model()` for automatically constructed weight-function priors. The default integrates estimate-level random effects and the complete sampling error, and conditions on other random effects. Each source can instead be conditioned upon or integrated through the corresponding selection-model setting. The default `weight_rule = "product"` needs no publication groups and ignores `group`. Publication grouping is active only for `weight_rule = "best"`. Explicit `prior_bias` objects retain their own selection specification; this argument does not overwrite it. Conditioning on sampling variation retains the entire sampling-error realization, including in ordinary univariate models. This sampling setting requires omitted or unit observation `weights`.

`selection_control`
numerical integration settings created by `set_selection_likelihood_control()`. These affect numerical evaluation, without changing the conditioning model or the vector weighting rule.

`known_v_parameterization`
known-V backend used by `brma.mv()`. `"auto"` chooses an exact backend when feasible; `"latent"` uses a latent `D + BB'` decomposition; `"whitened"` uses an eigen-rotated normal likelihood; `"block_mvn"` uses an exact native block multivariate-normal likelihood.

`sample` numeric. Number of MCMC samples to save. Defaults to `5000`.

`burnin` numeric. Number of burn-in iterations. Defaults to `2000`.

`adapt` numeric. Number of adaptation iterations. Defaults to `500`.

`chains` numeric. Number of MCMC chains. Defaults to `3`.

thin	numeric. Thinning interval. Defaults to 1.
parallel	logical. Whether to run MCMC chains in parallel. Defaults to FALSE.
autofit	logical. Whether to automatically extend the MCMC chains if convergence is not met. Defaults to FALSE.
autofit_control	list of autofit control settings. See set_autofit_control() for details.
convergence_checks	list of convergence check settings. See set_convergence_checks() for details.
seed	numeric. Random seed for reproducibility. Defaults to NULL.
silent	logical. Whether to suppress output. Constructors with no explicit default use RoBMA.get_option("silent") when silent is omitted. Model-averaging wrappers default to TRUE unless explicitly changed.
...	additional advanced arguments. Fitting functions reject unused arguments; currently recognized internal arguments include only_data, only_priors, is_JASP, and is_JASP_prefix.
vi	a vector of sampling variances. Either vi or sei must be supplied for normal models.
sei	a vector of standard errors. Either vi or sei must be supplied for normal models.

Details

Omitted or NULL random specifies no heterogeneity or random-effect inclusion mixture. Publication-bias and fixed-effect alternatives remain available. Heterogeneity priors and scale formulas require an explicit random structure; the known sampling covariance remains unchanged.

RoBMA.mv() combines the publication-bias product space of RoBMA() with the known-covariance, formula, random-effect, and prediction machinery of brma.mv(). Selection models, PET, PEESE, and the unadjusted branch are averaged together according to model_type or custom prior_bias and prior_bias_null specifications. Selection is defined over the estimates in each publication event. PET uses sqrt(diag(V)) and PEESE uses diag(V) as their bias predictors while the likelihood retains the full known sampling covariance.

Random structures supplied through random use the same independent component gates as BMA.mv(). A Dirichlet prior allocates the slab total variance across multiple top-level components and each allocation is multiplied by its own inclusion indicator without renormalizing the remaining allocations. Component-specific or partial scale models instead have independent SD roots, with allocation within each component; unscaled components keep their own base SD prior. See random_effect_prior_specificatio
The default prior inclusion probability is 0.5 for every component. Public tau_total and tau2_total draws are the realized gated aggregate, including the all-off zero branch. Public tau2_prop(...) draws are realized shares conditional on positive total heterogeneity; excluded components have zero share, and the all-off branch is undefined. The positive slab scale and raw Dirichlet weights remain internal coordinates.

The constructor's selection specification applies to every generated weightfunction. It integrates estimate-level random effects and the complete sampling error by default, and conditions on other random effects. Default product weights require no publication groups: group = NULL is valid, including with a multilevel random formula and correlated V. Explicit priors carry their own selection_model() settings. Active selection branches must share one conditioning cell; only

`weight_rule = "best"` branches require and must share a publication partition. Product branches ignore `group`. Their bins, weight priors, and weighting rules may differ. Non-selection branches retain the corresponding Gaussian, PET, or PEESE contribution with the same contextual source representation.

Whole sampling-error selection, grouping, and the Gaussian source likelihood follow `bselmodel.mv()`. Numerical backends do not change these model choices. All three source choices are explicit in `selection`. Without an active selection branch, supported estimate-level Gaussian random intercepts are automatically integrated into the likelihood variance.

Product-space marginal likelihood and bridge-sampling methods are not available. Predictive comparison through `loo.brma()` and `waic.brma()` remains available. If `V` is singular, every allowed product-space branch must contain structural variance sufficient to regularize its null space.

Value

A fitted object of class `c("RoBMA.mv", "RoBMA", "brma.mv", "brma.norm", "brma")`.

See Also

`RoBMA()`, `BMA.mv()`, `brma.mv()`, `bselmodel.mv()`, `bPET.mv()`, `bPEESE.mv()`, `set_selection_likelihood_control()`, `summary.brma()`, `summary_models()`

Examples

```
## Not run:
data("dat.assink2016", package = "metadat")
V <- metafor::vcalc(
  vi,
  cluster = study,
  type = deltype,
  obs = esid,
  rho = c(0.7, 0.5),
  data = dat.assink2016
)
fit <- RoBMA.mv(
  yi = yi,
  V = V,
  mods = ~ deltype,
  random = ~ 1 | study / esid,
  data = dat.assink2016,
  measure = "SMD",
  seed = 1,
  silent = TRUE
)
summary(fit)
summary_models(fit)

## End(Not run)
```

RoBMA_control *Control MCMC fitting process*

Description

set_autofit_control and set_convergence_checks control settings for the autofit process of the MCMC JAGS sampler and specify termination criteria and convergence checks.

Usage

```
set_autofit_control(
  max_Rhat = 1.05,
  min_ESS = 500,
  max_error = NULL,
  max_SD_error = NULL,
  max_time = list(time = 60, unit = "mins"),
  sample_extend = 1000,
  restarts = 10,
  max_extend = 10,
  check_indicators = FALSE,
  monitor = NULL,
  allow_not_assessable = FALSE
)

set_convergence_checks(
  max_Rhat = 1.05,
  min_ESS = 500,
  max_error = NULL,
  max_SD_error = NULL,
  check_indicators = FALSE,
  monitor = NULL,
  allow_not_assessable = FALSE
)
```

Arguments

max_Rhat	maximum value of the R-hat diagnostic. Defaults to 1.05.
min_ESS	minimum estimated sample size. Defaults to 500.
max_error	maximum value of the MCMC error. Defaults to NULL. Be aware that PEESE publication bias adjustment can have estimates on different scale than the rest of the output, resulting in relatively large max MCMC error.
max_SD_error	maximum value of the proportion of MCMC error of the estimated SD of the parameter. Defaults to NULL.
max_time	list with the time and unit specifying the maximum autofitting process per model. Passed to difftime function (possible units are "secs", "mins", "hours", "days", and "weeks"). Defaults to list(time = 60, unit = "mins").

`sample_extend` number of samples to extend the fitting process if the criteria are not satisfied. Defaults to `1000`.

`restarts` number of times new initial values should be generated in case a model fails to initialize. Defaults to `10`.

`max_extend` number of times after which the automatic fitting function is stopped. Defaults to `10`.

`check_indicators` whether model indicator variables should be included in convergence checks. Binary indicators are checked as state occupancies and categorical indicators are checked separately for every observed state. Eligible indicators remain included when `monitor` selects a narrower parameter set; auxiliary inclusion-probability coordinates remain excluded unless requested explicitly. Defaults to `FALSE`.

`monitor` optional character vector selecting the parameters used for convergence checks. A base name selects all indexed elements. Defaults to `NULL`, which checks every eligible parameter.

`allow_not_assessable` whether requested sampled parameters with undefined diagnostics may be ignored. Defaults to `FALSE`.

Details

`set_autofit_control` controls the automatic model fitting process, determining when to stop iterating based on convergence criteria. `set_convergence_checks` sets thresholds for determining whether a model has converged adequately.

The autofit control manages computational resources by setting maximum time limits and determining how many additional samples to draw if convergence criteria are not met. The convergence checks determine the quality standards that fitted models must meet. Autofit thresholds `max_Rhat`, `min_ESS`, `max_error`, `max_SD_error`, `max_time`, `restarts`, and `max_extend` can be set to `NULL`; `sample_extend` must be a positive integer. The routing controls are delegated to `BayesTools::JAGS_check_convergence()`. Thresholds can be disabled with `NULL`; otherwise `max_Rhat` must be at least 1, `min_ESS` and `max_error` must be nonnegative, and `max_SD_error` must be between 0 and 1.

Value

`set_autofit_control` returns a list of autofit control settings including `max_Rhat`, `min_ESS`, `max_error`, `max_SD_error`, `max_time`, `sample_extend`, `restarts`, `max_extend`, `check_indicators`, `monitor`, and `allow_not_assessable`. `set_convergence_checks` returns a list with convergence thresholds `max_Rhat`, `min_ESS`, `max_error`, `max_SD_error`, `check_indicators`, `monitor`, and `allow_not_assessable`.

See Also

`RoBMA()`

Examples

```
# Set custom autofit control with shorter time limit
autofit_ctrl <- set_autofit_control(
```

```
  max_time = list(time = 30, unit = "mins"),
  sample_extend = 2000
)

# Set custom convergence checks with stricter criteria
conv_checks <- set_convergence_checks(
  max_Rhat = 1.01,
  min_ESS = 1000
)

## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")

  fit <- RoBMA(
    yi                = yi,
    vi                = vi,
    data              = dat.lehmann2018,
    measure           = "SMD",
    autofit_control   = autofit_ctrl,
    convergence_checks = conv_checks
  )
}

## End(Not run)
```

`RoBMA_options` *Options for the RoBMA package*

Description

Inspect or change package-level defaults used by RoBMA.

Usage

```
RoBMA.options(...)

RoBMA.get_option(name)
```

Arguments

`...`	named option(s) to change. Names must be exact public option names, nonempty, and unique.
`name`	a single non-missing character string matching one public option exactly; for available options, see details below.

Details

The available options are:

`max_cores` number of cores to use for parallel computing (default is one fewer than detected logical cores, with a minimum/fallback of 1)

`native_threads` maximum number of native threads used to evaluate independent posterior rows of the compiled selection-likelihood batches in post-processing such as likelihood-aware densities, z-plots, and predictions. The default `NA` inherits the model's fitting setup: models fitted with `parallel = TRUE` evaluate their post-processing rows with up to `max_cores` native threads, unless this option sets an explicit thread count. Parallel work is arranged so that every reduction sums in a fixed order, so results do not depend on the thread count; single-state fitting and bridge-sampling evaluations stay serial. A model fitted with `parallel = FALSE` does its post-processing serially unless this option sets an explicit thread count.

`check_scaling` whether to check scaling of predictors (default `TRUE`)

`silent` whether to suppress JAGS output when fitting or extending models (default `FALSE`); the model-averaging constructors `RoBMA()`, `BMA()`, `BMA.norm()`, `BMA.glmm()`, `RoBMA.mv()`, and `BMA.mv()` default to `silent = TRUE` and ignore the option

`jags.worker_output` file path for parallel JAGS worker stdout and stderr when fitting or extending. The parent directory must exist; workers append to the same file and messages may interleave. The default empty string disables capture. Set this option inside any background job that fits the model. It does not change sampling or numerical integration settings.

`autocompute.loo` whether to automatically compute LOO (default `FALSE`); a GLMM whose nuisance-prior likelihood the quadrature cannot certify warns and the fit is returned without it

`autocompute.waic` whether to automatically compute WAIC (default `FALSE`); a GLMM whose nuisance-prior likelihood the quadrature cannot certify warns and the fit is returned without it

`autocompute.marglik` whether to automatically compute marginal likelihood (default `FALSE`)

`cluster_likelihood.n_gamma` number of Gauss-Hermite nodes formerly used for weighted normal cluster-unit log-likelihoods (default `15`). Normal cluster-unit log-likelihoods, weighted or not, are now integrated in closed form, so the option is still accepted but no longer affects them; selected-normal likelihoods use their own quadrature schedules and integration controls.

`selection.cache_max_bytes` one shared storage budget for exact and coarse selection-normalizer caches across all chains in a fit. The default `"auto"` uses the smaller of 4 GiB and one quarter of currently available system RAM, resolved before fitting. Supply a nonnegative whole number of bytes to set an explicit total; `0` disables caching. Storage grows as needed. Chains receive equal shares, pooled within each worker process; sequential chains share the total.

`selection.cache_retain` whether to save selection-normalizer cache entries with the fitted object after fitting or extending (default `FALSE`). Retained entries survive `saveRDS()` and are restored when extending a compatible fit. See `remove_selection_cache` for memory costs and removal.

`selection.sampler` `"default"` retains ordinary JAGS sampling. `"coarse_corrected"` uses coarse slice proposals with a full-target Metropolis correction for eligible scalar updates involving multivariate, cluster or factor step selection blocks (see `selection_sampler_info` for the blocks that take part). The posterior target is unchanged. Settings are captured when each model is compiled.

`selection.coarse_grid` named nonnegative grid steps `c(mean = .05, variance = .05, log_weight = .01)` of the dense multivariate step blocks; cluster and factor blocks evaluate their coarse rule at the actual parameters and ignore the grid. Mean spacing is its step times the largest original sampling SE in the block; covariance-diagonal spacing is its step times that SE squared. Each diagonal is rounded upward on a grid whose origin is its own squared original sampling SE, retaining all off-diagonal entries. Positive weights are rounded on the natural-log scale; zero weights remain zero. A zero step retains that coordinate exactly but does not disable the coarse rule budget or fallback. These settings affect proposals, not the corrected target.

`selection.coarse_max_rules` coarse quadrature rule budget (default `3`, minimum `3`). Dense multivariate blocks try at most this many rules from the beginning of the fitted quadrature schedule for a coarse anchor; unsupported or unsuccessful coarse integration uses the prescribed unit normalizer, without QMC fallback. Cluster and factor blocks use the one rule of this number in their own schedule (the last rule of a shorter schedule) at the actual parameters, and the exact density where that rule is unavailable. The full-target evaluator retains its ordinary numerical controls and diagnostics.

`default_UISD.effect` default scaling of the unit information standard deviation for the effect size parameter (default `0.5`)

`default_UISD.heterogeneity` default scaling of the unit information standard deviation for the heterogeneity parameter (default `0.25`)

`default_UISD.mods` default scaling of the unit information standard deviation for the moderators (default `0.25`)

`default_UISD.scale` default scaling of the unit information standard deviation for the scale parameter (default `0.5`)

`default_lograte.sd` default standard deviation of the estimate-specific midpoint log-rate prior in Poisson GLMMs (default `1`)

`default_informed_priors.mods` default scaling of informed priors for moderators (default `0.5`)

`default_informed_priors.scale` default scaling of informed priors for the scale parameter (default `0.5`)

`default_bias_weightfunction.alpha` default alpha for the weightfunction (default `1`)

`default_bias_PET.scale` default scale for the PET (default `1`)

`default_bias_PEESE.scale` default scale for the PEESE (default `5`)

Boolean options require scalar `TRUE` or `FALSE` values. `max_cores` must be integer-like and at least 1; `cluster_likelihood.n_gamma` must be integer-like and at least 3. `native_threads` accepts `NA` or a positive integer. Scale and alpha defaults must be finite positive numbers.

Value

`RoBMA.options()` invisibly returns a named list with all current options after applying any changes. `RoBMA.get_option()` returns the current value of the requested option.

rstandard.brma *Internally Standardized Residuals for brma Objects*

Description

Computes internally standardized residuals from a fitted brma object using the hat matrix or the corresponding known-V GLS/BLUP projection. Returns a data frame with raw residuals, standard errors, and standardized residuals (z-values). Available for normal outcome models only.

Usage

```
## S3 method for class 'brma'
rstandard(
  model,
  unit = "estimate",
  conditioning_depth = "marginal",
  max_samples = Inf,
  ...
)
```

Arguments

model	a fitted brma object.
unit	output unit. Only "estimate" is implemented currently.
conditioning_depth	conditioning depth. Options are: • "marginal" (default): GLS projection residuals $(I - H)y$. • "cluster": Projection residuals after the fitted cluster contribution is removed. Only available for specialized multilevel models. • "estimate": Sampling residual projection after the fitted random-effect contribution is removed (a BLUP-style target).
max_samples	maximum posterior draws used for known-V brma.mv() marginal covariance computations. Defaults to Inf. Finite values deterministically thin draws across posterior row order.
...	additional arguments (currently ignored)

Details

This function returns a data frame with three columns matching the output of metafor::rstandard:

- resid: Projection residuals: marginal depth uses $(I - H)y$; cluster and estimate depths retain their corresponding random-effect target. Known-V estimate depth uses the sampling residual projection. Residuals and SDs come from the same projection. These are not the posterior-mean residuals returned by residuals(type = "outcome"), which subtract the posterior mean of the fitted values and therefore carry the prior's shrinkage; the two agree as the effect prior becomes diffuse.

- se: Standard errors of the residuals
- z: Posterior mean of draw-wise standardized residuals

Internally standardized residuals are computed draw by draw by dividing the observed residual by its corresponding residual standard error from the hat matrix. The returned z column is the posterior mean of those draw-wise ratios, so it is not generally equal to the displayed resid / se summaries. For correlated known-V models, marginal residuals use the GLS residual covariance and estimate-depth residuals use the BLUP-style sampling residual projection. For a correctly specified model, these residuals should approximately follow a standard normal distribution. A design-exactly fitted row is set to zero only when its covariance multiplier has no other partners; fully saturated designs remain exactly zero under correlated covariance as well. Such a row has residual and standard error zero and an undefined standardized residual, so its z is NA (also when it is exactly fitted in only some posterior draws); qqnorm() omits these rows with a message.

For ensembles whose members differ in their PET and PEESE terms, each draw uses the design of its member, as described in hatvalues.brma.

This function is only available for normal outcome models without selection (weightfunction) bias adjustment. Selection models can use rstudent.brma, which uses LOO-PIT. Internally standardized residuals are unavailable for binomial and Poisson GLMMs because no count-likelihood standardization has been implemented. Use rstudent() (the LOO-PIT residuals) for GLMMs.

For known-V models, the standardized residuals require applying the inverse of the marginal covariance. A covariance that is too ill-conditioned for that (a reciprocal condition number of about 2e-10 or less in any posterior draw used) is unavailable: the call stops with an error of class RoBMA_known_v_precision_unavailable, whose parent class is RoBMA_numerical_unavailable, instead of returning a rounded number. The condition carries the draw positions (rows) and their reciprocal_condition. These positions refer to the current request after deterministic thinning, including estimate-depth requests.

Value

A data frame with columns:

- resid: Posterior mean projection residuals at the requested depth
- se: Standard errors of the residuals
- z: Standardized residuals

See Also

rstudent.brma(), residuals.brma(), blup.brma(), predict.brma()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- brma(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")

  # marginal internally standardized residuals (default)
  rstandard(fit)
```

```
  # estimate-level (BLUP-based) residuals
  rstandard(fit, conditioning_depth = "estimate")
}

## End(Not run)
```

`rstudent.brma` *Externally Standardized (Studentized) Residuals for brma Objects*

Description

Computes externally standardized residuals (also called studentized residuals or standardized deleted residuals) from a fitted brma object using LOO-PIT (Leave-One-Out Probability Integral Transform). Returns a data frame with raw residuals, standard errors, and standardized residuals (z-values).

Usage

```
## S3 method for class 'brma'
rstudent(model, unit = "estimate", conditioning_depth = "marginal", ...)
```

Arguments

`model`	a fitted brma object.
`unit`	output unit. Only `"estimate"` is available for LOO-PIT residuals.
`conditioning_depth`	unused for LOO-PIT residuals. LOO-PIT residuals always use the estimate-unit LOO target.
`...`	additional arguments (currently ignored)

Details

This function returns a data frame with three columns matching the output of `metafor::rstudent`:

- `resid`: LOO predictive residuals (observed - fitted values)
- `se`: LOO predictive standard errors when available
- `z`: Externally standardized residuals (LOO-PIT transformed)

LOO-PIT residuals are the Bayesian equivalent of studentized deleted residuals. They are computed via leave-one-out probability integral transformation using Pareto smoothed importance sampling. For each observation, the LOO-weighted CDF is computed and transformed to a standard normal quantile.

Under a correctly specified model, LOO-PIT residuals should follow a standard normal distribution. Large absolute values may indicate outliers or model misspecification.

The z column is the primary standardized diagnostic. The resid and se columns are raw-scale companions computed from LOO predictive moments using the normalized PSIS weights. For selection models, these moments are computed from the fitted selected-normal predictive distribution.

Unlike rstandard.brma (which uses the hat matrix), LOO-PIT residuals properly account for estimation uncertainty and leverage without requiring explicit hat matrix computation. For binomial and Poisson GLMMs the residuals are a descriptive normal approximation on the effect-size scale (see residuals.brma), not a PIT of the fitted count likelihood, and a message says so. For correlated known-V brma.mv() models, the PIT and companion residual moments are computed from the Schur-complement conditional predictive distribution used by estimate-unit LOO.

Value

A data frame with columns:

- resid: Raw residuals
- se: Standard errors of the residuals
- z: Externally standardized residuals (LOO-PIT)

See Also

rstandard.brma(), residuals.brma(), loo.brma(), blup.brma()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- brma(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")
  fit <- add_loo(fit)

  # externally standardized residuals
  rstudent(fit)

  # check Pareto k values
  plot(loo(fit))
}

## End(Not run)
```

---

selection_cache_info *Inspect or clear the native selection cache*

Description

Reports the bounded native selection-normalizer cache in the current R process. Exact caching reuses identical numerical inputs and does not change the likelihood or its integration diagnostics. Post-fit requests using bounded CDF interpolation bypass these caches, including when they fall back to the ordinary primitive. Fitting retains its ordinary-primitive cache.

Usage

```
selection_cache_info(clear = FALSE, component = c("all", "exact", "coarse"))
```

Arguments

clear	clear the requested entries and reset their counters before reporting them. Clearing "all" also releases the shared storage.
component	report the "exact" cache, the "coarse" proposal cache, or "all" caches.

Details

The default selection.cache_max_bytes = "auto" chooses the smaller of 4 GiB and one quarter of currently available system RAM. Available RAM is reported by ps::ps_system_memory(), excluding swap. A numeric setting supplies an explicit total byte budget; 0 disables storage. If the available-memory query fails, a warning is issued and caching stays disabled until a budget can be resolved or supplied explicitly. Loading the package does not query available memory. The first selection computation or cache inspection initializes the current process's cache, including computations on a saved fit. Explicit option changes and fitting setup also initialize it. Exact and coarse entries share one byte budget for directly allocated key, control, list-node and index blocks, with global least-recently-used eviction. Storage is allocated lazily. The ordinary sampler does not use coarse entries.

The budget is resolved once before fitting and divided equally among chains. Shares are pooled for chains assigned to the same process: ten parallel workers receive one tenth each, while ten sequential chains share the total. Integer-byte rounding can leave a few bytes unused. The coordinator releases its cache while workers run and restores its local budget when they stop. Separate simultaneous fits are separate budgets; this is not a machine-wide memory manager. Stored fit extensions retain the resolved budget and sampler settings, redistributing the budget for their current worker topology.

This function reports only the current process, not separate workers or saved fitted-object caches. By default entries live in the native process. Set RoBMA.options(selection.cache_retain = TRUE) to retain them with the fit; see remove_selection_cache. Changing the budget or clearing all entries releases the shared storage. Clearing one component releases its keys, nodes and controls; shared index buckets can remain allocated. Switching sampler options does not clear existing entries or replace already compiled samplers.

The exact cache accelerates successful deterministic covariance-envelope normalizers for multivariate step selection. Unsupported numerical routes continue through their ordinary evaluator. The coarse cache can also store its prescribed unit-normalizer fallback. If cache storage cannot be allocated, the same exact or surrogate evaluator runs without storage; memory capacity never determines the returned normalizer or its numerical diagnostics.

Value

A data frame with the cache component, byte counts, entry count, and lookup, eviction and allocation counters. The "total" row reports the shared capacity and directly allocated blocks, including retained controls and index buckets; component byte counts are NA because storage is shared. Allocator metadata, the surrounding R process and JAGS model are excluded; this is not a process-RSS limit. Eviction releases entry blocks, while controls and index buckets can remain allocated.

Oversize lookup keys increment misses without incrementing allocation failures. Failed insertion, capacity or allocation attempts can increment allocation failures; these counts do not necessarily indicate exhaustion of system memory.

`selection_model` *Selection Model*

Description

Declare the contextual sources retained by a selection model, how estimate weights combine, and a publication-group column for best-p-value selection when needed.

Usage

```
selection_model(
  estimate_random_effects = "integrate",
  other_random_effects = "condition",
  known_sampling_variance = "integrate",
  weight_rule = "product",
  group = NULL
)
```

Arguments

`estimate_random_effects`
: Either `"condition"` or `"integrate"` (default), for the declared estimate-level true-effect source. One-to-one grouping determines this role in multivariate models; it does not imply independent covariance.

`other_random_effects`
: Either `"condition"` (default) or `"integrate"`, for all remaining declared random-effect terms.

`known_sampling_variance`
: Either `"condition"` or `"integrate"` (default), for the complete sampling-error vector $e \sim N(0, V)$. Conditioning retains its realized value during selection normalization; integration uses the full sampling covariance. This choice also applies to univariate and independent sampling errors supplied through `vi` or `sei`. It does not change `V`.

`weight_rule`
: Either `"product"` (default), which multiplies estimate weights without publication grouping, or `"best"`, which uses the weight at the smallest p-value within each publication group.

`group`
: Optional unresolved data-column name, supplied as a bare name, a backtick-quoted name, or one character string. Active only for `weight_rule = "best"`, where the default `NULL` requests automatic group resolution when data are bound. Product models ignore this argument and require no group column.

Details

This is RoBMA's re-export of `BayesTools::selection_model()`. The specification is stored in `prior_weightfunction(model = ...)`. Constructor `selection = selection_model(...)` settings apply only to generated default priors; explicit priors retain their own specification. A group name is captured without evaluating data; RoBMA binds it only for `weight_rule = "best"`. When `group = NULL`, best-rule groups come from a supported constructor's `cluster` input or, for independent models without a random structure, one group per estimate. Other best-rule models require an explicit group column. Conditioned sources remain unknown and estimated. Integrating estimate variation is the default; conditioning on it defines a different selection model. The specialized clustered interface retains its total heterogeneity, allocation and within-/between-cluster I2 summaries.

With positive sampling standard errors, the sampling law can be written as $e = Sz$, with $z \sim N(0, R)$ and $V = SRS$, where S contains sampling standard errors and R is their correlation matrix. Conditioning retains all of e; no residual sampling error is integrated separately. Covariance factorizations affect computation, not the selection model. Selection thresholds continue to use the original sampling standard errors.

When all outcome-generating sources are conditioned upon and weights are positive almost surely, selection weights cancel and the observed law is the ordinary Gaussian model. The likelihood then contains no information about the selection weights. A retained context with zero acceptance probability cannot complete the selection process. Integrating all sources instead gives population-level selection among completed studies. These choices are independent of prediction's `conditioning_depth`. Non-unit observation `weights` require `known_sampling_variance = "integrate"`; unit weights are equivalent to omitting `weights`. Other observation-weight restrictions still apply.

`weight_rule = "best"` uses the weight of the smallest p-value's bin, rather than the largest weight. `weight_rule = "product"` multiplies all estimate weights. Its joint normalizer retains the dependencies in the integrated covariance, regardless of grouping. In ensembles, only best-rule branches require matching publication partitions; product-only ensembles accept `group = NULL`.

See `bselmodel.mv()` for the supported conditioning cells, source and group requirements, and covariance boundaries.

Value

A `selection_model` object containing fixed mode choices and a deferred column name or `NULL`. The group is used only by the best rule.

See Also

`prior_weightfunction`, `bselmodel.mv()`, `RoBMA.mv()`

Examples

```
prior_weightfunction(
  steps = .05,
  model = selection_model(known_sampling_variance = "condition")
)
prior_weightfunction(
  steps = .05,
  model = selection_model(weight_rule = "best", group = paper_id)
)
```

`selection_sampler_info` *Inspect corrected selection-sampler counters*

Description

Counts completed coarse slice proposals and their full-target acceptance decisions in the current R process.

Usage

```
selection_sampler_info(clear = FALSE)
```

Arguments

`clear`	reset counters before reporting them.

Details

Enable the alternative for future fits with `RoBMA.options(selection.sampler = "coarse_corrected")`. Every proposal is corrected against the full original likelihood and prior. Only the selection normalizer used to construct proposals is coarsened. A reversible slice transition for that surrogate is followed by a Metropolis correction using both original and surrogate densities at the current and proposed states. The original likelihood keeps its existing numerical integration controls and diagnostics; the correction does not replace those numerical integrals by symbolic exact values.

Dense multivariate step blocks. The surrogate rounds means and the logarithms of positive weights to fixed grids. Each covariance diagonal is rounded upward on a grid whose origin is that row's squared original sampling SE; off-diagonal covariances, original SEs, selection cutoffs, and zero weights are retained. These are proposal grids, not changes to fitted variance components or the input covariance. The Gaussian numerator and observed weights use the actual parameters. At most the first `selection.coarse_max_rules` rules of the fitted quadrature schedule are tried, with its existing integration criterion. If a valid anchor cannot be constructed or its covariance-envelope calculation is unsupported or unsuccessful, its prescribed normalizer is one; the proposal stage does not fall back to QMC. This fallback is independent of cache history and memory capacity and remains subject to the full-target correction. It does not replace full-target integration failures.

Rank-one cluster and factor step blocks. The surrogate is the exact numerator (Gaussian log density and observed log weights at the actual parameters) minus the log normalizer of one rule of the block's own Gauss-Hermite schedule: rule number `selection.coarse_max_rules`, or the last rule of a shorter schedule, evaluated at the actual parameters. No grid rounding, change test, certificate, refinement, sparse grid, QMC or cache enters, so the surrogate is a deterministic function of the sampled state; the correction evaluates the exact block, which keeps its ordinary numerical controls and diagnostics. Where that rule is unavailable at an update (a best-weight rule in the current state, or a non-finite normalizer) the surrogate is the exact density of the block. A block takes part unless its fixed controls rule it out: a weight rule fixed at a best-weight rule, a kernel mode fixed at the normal kernel, or, for a factor block, a rule beyond the node budget at the declared rank (rank two

to six at the default setting). In model-averaged fits the weight rule and the kernel mode are nodes of the bias indicator, so their blocks take part, with the exact density in the states of the indicator that have a best-weight rule. A model average whose selection branches all use the best-weight rule takes part without any state that can gain: its surrogate is the exact density everywhere and every update pays two extra exact evaluations, so run such fits with the default sampler. Blocks fixed at a best-weight rule, blocks of higher rank and blocks with a fixed normal kernel keep the ordinary sampler.

A scalar update is eligible when at least one stochastic child of the updated node is a dense multivariate block or a coarse-capable cluster or factor block; its other children enter the surrogate through their exact densities. Discrete, vector-valued and unrelated updates, and updates whose selection children are all without a coarse rule, retain ordinary JAGS samplers, so the option can be set without effect: check that the counters grow. Whatever the surrogate, the corrected chain targets the exact full conditional; a coarser rule changes the acceptance rate and the cost, not the posterior.

Already compiled samplers retain their grid and coarse rule-budget settings. Coarse settings can affect sampling efficiency even when correction acceptance is high. Compare effective samples per second for parameters of interest, not only iteration times or this acceptance rate. Counters combine all corrected samplers in this process, including warmup; clear them between phases for a sampling-only comparison. They do not report separate parallel workers. Inspect them between sampling calls.

Value

A data frame with proposal, acceptance and correction-rejection counts, and an acceptance rate (`NA` before any proposals).

`selection_sensitivity_diagnostics` *Diagnose Selection Conditioning Sensitivity*

Description

Describes how integrating the contexts retained by a fitted selection model would reweight their pre-selection Gaussian distribution. This compares reporting models; it is not a numerical approximation error bound.

Usage

```
selection_sensitivity_diagnostics(
  object,
  max_posterior_samples = 256L,
  latent_samples = 512L,
  seed = 1L,
  integration_control = NULL
)

add_selection_sensitivity_diagnostics(object, ...)
```

Arguments

`object`	A fitted normal-outcome model with a bound selection model.
`max_posterior_samples`	Maximum number of posterior draws to evaluate. A positive integer of at least `10`, or `Inf` for all posterior draws. Draws are deterministically thinned over posterior row order.
`latent_samples`	Number of fresh pre-selection retained-context draws per posterior draw. Must be one finite integer of at least `2`; defaults to `512`.
`seed`	Integer seed for the diagnostic simulation. The calling R session's random-number state is restored before returning.
`integration_control`	Optional settings from `set_selection_likelihood_control()` for diagnostic event integration. `NULL` uses the fitted model's settings.
`...`	Simulation arguments passed to `selection_sensitivity_diagnostics()`.

Details

At each posterior draw the fitted model defines a conditional Gaussian response law, retained contexts, and any best-rule publication groups. The diagnostic evaluates each Gaussian selection unit's normalizer at fresh retained contexts, preserving the fitted product or best weight rule and any best-rule publication groups. Product-only models require no grouping. Within a connected source unit, their product gives the relative reweighting when the retained contexts are integrated before selection normalization. Sampling-conditioned comparisons use the fitted candidate factors, including singular declared support. Integration diagnostics report maxima over the constituent event evaluations, rather than an error bound for their product.

`ess_fraction` estimates `mean(W)^2 / mean(W^2)`; this is a retained-context reweighting ESS fraction within each posterior parameter state, not the posterior sample ESS. `total_variation` estimates the distance between the pre-selection context law and its normalized `W`-reweighted law. The fitted and reference conditioning specifications, publication identities, and source roles are retained in the `target` attribute. With no retained context this comparison is inapplicable and its reweighting is identically constant.

The fitted weight rule is held fixed. These summaries do not compare arbitrary weight rules, identify the correct reporting mechanism, or reveal parameter regions absent from the fitted posterior.

Simulation budgets control diagnostic Monte Carlo precision separately from the fitted Gaussian event integration controls. The largest unit-level posterior-median total variation distance is silent up to `0.05`, produces a message above `0.05` and below `0.10`, and a warning from `0.10`. These notifications describe sensitivity rather than statistical adequacy.

This optional comparison is computed only when requested; fitting does not compute it. Calls without simulation arguments reuse an explicitly stored result, and explicit settings reuse it only when they match. Use `add_selection_sensitivity_diagnostics()` to compute and store a result. Chain extension silently refreshes only stored diagnostics using their stored simulation settings; label-only updates retain them. Failed refreshes remain stored as unavailable without discarding the fitted posterior. Call `selection_sensitivity_diagnostics()` to inspect the comparison or its error. Sensitivity notifications are emitted only by these explicit helper calls, not by fitting, extension, or ordinary model printing.

The substantial-sensitivity warning has class `RoBMA_selection_sensitivity_warning`, retaining `simpleWarning`, `warning`, and `condition`. Its fields are `maximum_total_variation`, `threshold` (`0.10`), and `comparison` (`"retained_context_reweighting"`). Own nonfinite normalizer refusals have class `RoBMA_selection_sensitivity_normalizer_unavailable` and fields `reason`, `posterior_draw` (the evaluated posterior-row index), `latent_draw`, `event`, and the observed `log_mass` values. Unsupported retained covariance has class `RoBMA_selection_sensitivity_covariance_unavailable` and `reason = "retained_covariance_unavailable"`. Both refusals inherit from `RoBMA_selection_sensitivity_error`, `simpleError`, `error`, and `condition`. Downstream integration failures retain their original conditions.

Value

A data frame with one row per connected source unit, identified by `block`, comma-separated observation `rows`, and `n_estimates`. The `ess_fraction`, `total_variation`, and `log_weight_iqr` summaries each have `_q05`, `_median`, and `_q95` columns across all evaluated posterior draws, including non-selection draws. The `ess_fraction_mcse` and `total_variation_mcse` columns have `_median` and `_max` suffixes for conditional Monte Carlo uncertainty. `integration_relative_mcse_max` and `integration_relative_quadrature_error_max` are maxima over constituent event evaluations, not an error bound for the multiplied unit weight. `posterior_samples`, `total_posterior_samples`, and `latent_samples` report the simulation budgets. The `settings` attribute records `max_posterior_samples`, `latent_samples`, `seed`, and `integration_control`. The `target` attribute records the comparison, fitted and reference models, applicability, source roles, fitted and reference sampling structures, publication groups, likelihood unit, row blocks, and resolved event integration settings. The add function returns the fitted object.

See Also

`bselmodel.mv()`, `set_selection_likelihood_control()`

`set_selection_likelihood_control`
Control numerical integration of selection likelihoods

Description

Creates numerical integration settings for the product and best-p-value selection likelihoods used by `bselmodel()`, `bselmodel.mv()`, `RoBMA()`, and `RoBMA.mv()`. Covariance factors use deterministic sequences of Gauss-Hermite rules. Certified factors of structural rank one through eight use deterministic interval refinement for rank-one product fallbacks and randomized quasi-Monte Carlo for other unresolved factor integrals. General covariance blocks with product weights first try same-diagonal covariance approximations with a common Gaussian factor and independent group factors. Monotone weights permit covariance bounds on the normalizer. Two-sided and other nonmonotone weights require a covariance-perturbation bound; agreement of auxiliary normalizers alone is insufficient. The absolute bound uses the total variation of the step weights. Strictly positive weights also permit a relative Gaussian covariance bound, including reconstructed diagonal and factor rounding, which remains useful for tiny normalizers. Their original full covariance remains authoritative. The covariance error bound, estimated quadrature error, and a bound on Gaussian mass outside the

quadrature nodes share one numerical error budget. Interior quadrature error is estimated, so this is not a rigorous probability interval. Unsupported or unresolved cases use randomized quasi-Monte Carlo in the dense conditional-normal representation. Under product weights, a deterministic rung of the factor or cluster quadrature ladder is accepted by a shift-amplitude certificate. Shifting the nodes of one latent axis by a fraction of the node gap and reweighting leaves the integral unchanged, so the change of the rule under half and quarter gap shifts of each axis measures its aliasing error. The rung's successive changes must be within `relative_tolerance`, the summed shift amplitudes must not exceed half of it, and twice their sum, together with the exterior Gaussian mass, must fit within it. A rung must also resolve the selection edges: the edge width (conditional residual SD divided by the absolute loading) is at least 0.15 times the node gap. When no rung is accepted, rank-one clusters, tensors of rank at most two and depth-one forests try the average of the finest rung over four shifts per axis before sparse grids and the randomized fallback. That average asks the edges for at least one eighth of the node gap, with a separate penalty for each axis's row count, and is used only when `relative_tolerance` is at least 0.005, the tolerance at which its checks were calibrated. For product weights with selection transitions, the ladder and this refinement tier use the smaller of `relative_tolerance` and 0.005 for their acceptance checks: a loose request must not relax the calibration of these deterministic rules. After these checks decline, rank-one product integrals refine bounds over the complete Gaussian line, including both tails. Exact row groups and nonnegative excess weights give interval extrema; the returned midpoint must satisfy the requested relative tolerance against both bounds. Floating arithmetic uses an explicit working-precision allowance, rather than a universal guarantee for library probability evaluations. Insufficient work or unresolved arithmetic is rejected without a QMC retry. Other randomized fallbacks retain the requested tolerance and their empirical diagnostics. The best-p-value rule and the other deterministic routes instead require that the node gap across an interior transition not exceed its conditional residual SD divided by the absolute loading; coupled axes use conservative central node spacing. Unresolved candidates refine or use the existing fallback; the numerical settings and likelihood target are unchanged. These checks are heuristics, not a bound on the total integration error, including reciprocal normalizers. Each fit or post-fit request uses a fixed integration design, so repeated likelihood evaluations are deterministic. One-observation blocks are evaluated analytically.

Post-fit dense two-bin product normalizers can use a bounded interpolation of the standard-normal CDF. Its deterministic error is included in the same numerical error budget, separately from Monte Carlo standard error. Other weights, arithmetic outside its supported range, or insufficient error budgets retain the ordinary scalar primitive. An unresolved interpolation attempt retries the ordinary covariance-envelope calculation before QMC. Fitting and random generation retain their ordinary scalar primitive.

The numerical defaults are package operating settings, not statistical estimands or literature-mandated constants. `points_per_scramble`, `max_points_per_scramble`, and `scrambles` set the fixed computational design, while `relative_tolerance` sets its diagnostic acceptance rule. Increasing a budget changes only the numerical evaluation of the same likelihood target.

For qCMDE/IWMDE post-fit densities and hypotheses in Gaussian selection models, pass this object as `density_control$integration_control`, for example `density_control = list(integration_control = set_selection_likelihood_control(max_points_per_scramble = 32768))`. The default `NULL` retains the fitted integration settings. An explicit control rebuilds a temporary execution plan from the stored covariance metadata without changing the fitted object or its posterior draws. In fitting and qCMDE/IWMDE likelihood evaluation, the maximum point budget controls factor QMC fallback and the rank-one product refinement work ceiling. For rank-one refinement, `max_points_per_scramble` and `scrambles` allow at most their product (with `max_points_per_scramble` rounded up to an even number) times the active exact row-group count times the positive excess-

weight bin count in scalar probability evaluations. This includes latent interval masses and bound preparation. Active intervals are limited to the same product and to the existing native node ceiling. `points_per_scramble` initializes QMC routes only. Post-fit full-event and partial-event calculations, including z-plots and sensitivity diagnostics, also refine dense QMC evaluations up to that budget. Analytic and deterministic quadrature rules remain unchanged.

Usage

```
set_selection_likelihood_control(
  points_per_scramble = 512L,
  max_points_per_scramble = 8192L,
  scrambles = 8L,
  relative_tolerance = 0.005,
  seed = 1L
)
```

Arguments

`points_per_scramble`
: nominal number of integrand evaluations per randomized quasi-Monte Carlo scramble for certified factor fallbacks, and the number of shifted Halton points for general covariance blocks. Factor likelihood-normalizer fallbacks under product weights combine the prior Gaussian with a Gaussian at the tilted mean and covariance of the finest quadrature rung and a copy with 1.5 times its standard deviations, where the quadrature provides them (tensors and depth-one forests with at least three finite rules, exterior mass covered and a positive definite covariance). Otherwise, and for the best-p-value rule, they combine the prior Gaussian with proposals optimized from zero and selection-boundary starting points. Every finite optimized candidate is retained, including candidates that locate the same mode. The total count is divided as evenly as possible across the proposals, and importance weights use the actual allocation. Odd likelihood-normalizer factor budgets use the next even total because the fixed design uses two streams. This count is the initial nested-design budget, not an accuracy guarantee. The integration diagnostics determine whether it is adequate for each evaluated likelihood state. Selected-density projection fallbacks instead retain the prior Gaussian and one proposal optimized from zero.

`max_points_per_scramble`
: largest nested-design budget used for a certified factor fallback, including fitting-time refinement, or a post-fit Gaussian event calculation, when the initial `points_per_scramble` budget fails its integration diagnostic. Refinement doubles the point count up to this value while reusing the same fixed design. Rank-one product fallback instead uses this maximum and `scrambles` as the deterministic scalar-probability work ceiling described above.

`scrambles`
: number of independent randomized shifts used to estimate integration error for certified factors and general covariance blocks. The stopping multiplier of `relative_tolerance` (2.36) is that of the default eight scrambles and does not change with this number. It also scales the rank-one product deterministic work ceiling.

`relative_tolerance`
: largest accepted successive relative quadrature change for factor integration, or the largest nested-design relative change and 2.36 times the relative Monte Carlo standard error for a factor fallback and for the conditioned-sampling normalizer. Rank-one product interval refinement instead checks the conservative relative radius of the actual returned midpoint against both bounds. General covariance blocks and Gaussian event integrals use 2.36 times the relative Monte Carlo standard error alone. Dense covariance-envelope quadrature additionally requires the sum of covariance-bound width, twice the successive quadrature change, the relative exterior-mass bound, and any post-fit CDF interpolation error to satisfy this tolerance. Factor quadrature also checks exterior Gaussian mass before accepting rule agreement; an unresolved rule sequence uses the existing QMC fallback. The returned deterministic candidate also requires its shift-amplitude certificate under product weights (summed shift amplitudes within half of this tolerance and twice their sum, with the exterior mass, within the tolerance; the four-shift average that follows a rejected ladder is used only for tolerances of at least 0.005) or, on the other routes, selection-transition resolution; coarser comparison rules retain their actual values without an added resolution requirement. The randomized fallback stops when the larger of its relative change and 2.36 times its relative Monte Carlo standard error (the 97.5% quantile of Student's t distribution with seven degrees of freedom, for eight scrambles) is within this tolerance, and a result that does not pass that test is rejected; the reported standard error stays unscaled, and the criterion does not bound the error of the accepted value. Zplot projections additionally check density error relative to the peak within each posterior draw and raw normalizer-mass diagnostics, including their bounded approximation and omission terms. These diagnostics do not certify the total integrated error of a compressed displayed curve; see `as_zplot.brma()`.

`seed`
: non-negative integer used only to create fixed quasi-Monte Carlo designs. It does not alter R's random-number state.

Value

A `RoBMA_selection_likelihood_control` object.

`summary.brma` *Summarize brma Object*

Description

`summary.brma` creates summary tables for a brma object. For RoBMA objects, inclusion summaries are printed before parameter estimates. Random-effect inclusion rows are included in `Component Inclusion`, immediately before `Publication Bias` when present. Their labels prefix the corresponding SD quantity with `Heterogeneity:`, retaining component names where needed. Data-frame exports use the same layout.

Usage

```
## S3 method for class 'brma'
summary(
  object,
  probs = c(0.025, 0.5, 0.975),
  include_mcmc_diagnostics = TRUE,
  standardized_coefficients = FALSE,
  conditional = FALSE,
  logBF = FALSE,
  BF01 = FALSE,
  ...
)

## S3 method for class 'summary.brma'
print(x, ...)

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

Arguments

object	a fitted brma object
probs	quantiles of the posterior samples to be displayed. Defaults to c(.025, .50, .975)
include_mcmc_diagnostics	whether to include MCMC diagnostics in the output. Defaults to TRUE.
standardized_coefficients	whether to show standardized formula coefficients. Defaults to FALSE. When set to TRUE, meta-regression coefficients and random-component summaries are returned on the standardized scale on which prior distributions are specified by default (i.e., standardize_continuous_predictors = TRUE). A random-effect row whose original-scale value the fitted random block cannot represent (for example the intercept of a block with factor-specific slopes of a standardized predictor) is NA in the default original-scale table, with a footnote; its value on the fitted scale is shown with standardized_coefficients = TRUE.
conditional	whether to include conditional estimates for RoBMA product-space objects. Defaults to FALSE.
logBF	whether to show inclusion Bayes factors on the log scale. Defaults to FALSE.
BF01	whether to show inverse inclusion Bayes factors. Defaults to FALSE.
...	unused additional arguments. Supplied arguments trigger a warning.
x	a summary.brma or fitted brma object.

Value

A list of class summary.brma with model name, optional RoBMA inclusion tables, common estimates, moderator estimates, scale estimates, publication-bias estimates, and optional conditional estimates. known_v_backend records the known-sampling-covariance backend. inclusion_random

contains random-effect inclusion tables; selection_model, selection_sampling, and selection_sensitivity_diagnos retain selection-target metadata and optional sensitivity diagnostics. The printed form displays the non-empty tables. Objects created with only_data = TRUE return the resolved data, and objects created with only_priors = TRUE return a summary.brma object that prints only the model title, as at 4.0.0, and holds the resolved prior list in its priors element, because neither carries a posterior. Intercept-only multivariate location models label their sole coefficient mu, consistently with ordinary meta-analysis models. Once location moderators are present, the coefficient is labeled intercept; an intercept fixed at zero is omitted. Multivariate models without mods or scale display location and random estimates together in one Estimates table. With predictors, random estimates join Common Estimates, location coefficients appear in Meta-Regression (or Location when scale is present), and scale coefficients appear in Scale. Without location moderators, mu remains in the common table. Conditional estimates follow the same layout. Data-frame conversion follows these displayed sections; the returned list retains its separate raw tables. Scale tables label the baseline SD as exp(intercept); its estimates are already exponentiated. Other scale coefficients remain on the log-SD scale. Multivariate scale rows identify their targeted random-effect heterogeneity (tau), total heterogeneity (tau_total), or mean-variance allocation scale (tau_common). The baseline corresponds to zero non-intercept design columns under the fitted contrasts and the requested predictor standardization. The random table reports the quantities aligned with prior specification; use summary_heterogeneity() for aggregate variances and the complete family of deterministic allocation transforms. This reporting rule is the same for brma.mv() and BMA.mv(): component tau values derived from a variance allocation appear only in summary_heterogeneity(). Optional selection-conditioning comparisons are inspected explicitly with selection_sensitivity_diagnostics(). Printing a model or its summary does not emit sensitivity notifications. Fitted objects store their default summary, so print() and summary() without further arguments return it without recomputation; update() and add_selection_sensitivity_diagnostics() store it again.

See Also

brma(), brma.glmm()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")

  fit <- bPET(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )

  summary(fit)
}

## End(Not run)
```

summary.brma_samples *Summarize brma_samples Object*

Description

Creates and returns a summary table of posterior samples using `BayesTools::ensemble_estimates_table`.

Usage

```
## S3 method for class 'brma_samples'
summary(object, probs = NULL, ...)
```

Arguments

`object`	a `brma_samples` object
`probs`	quantiles for credible intervals. If `NULL`, uses the default stored in the object (typically `c(.025, .975)`)
`...`	additional arguments passed to `BayesTools::ensemble_estimates_table`

Value

A `BayesTools_table` object containing the summary statistics.

summary.marginal_means.brma *Summarize Estimated Marginal Means*

Description

Summarizes estimated marginal means stored in a `marginal_means.brma` object using `BayesTools::marginal_estimates`

Usage

```
## S3 method for class 'marginal_means.brma'
summary(
  object,
  type = NULL,
  probs = c(0.025, 0.5, 0.975),
  logBF = FALSE,
  BF01 = FALSE,
  bf = NULL,
  output_measure = NULL,
  transform = NULL,
  ...
)
```

Arguments

object	a marginal_means.brma object.
type	for RoBMA product-space objects, whether to summarize model-averaged ("averaged") or conditional ("conditional") marginal means. Defaults to "averaged" and is available only for RoBMA marginal means.
probs	quantiles of the posterior distribution to be displayed. Defaults to c(.025, .50, .975).
logBF	whether to show inclusion Bayes factors on the log scale. Defaults to FALSE.
BF01	whether to show inverse inclusion Bayes factors. Defaults to FALSE.
bf	whether to show inclusion Bayes factors. Defaults to the setting stored by marginal_means(). An explicit TRUE computes previously skipped KDE Bayes factors from the stored conditional posteriors at the stored null hypothesis.
output_measure	effect-size measure for location/effect predictions. Defaults to the fitted measure. Supported conversions are among "SMD", "COR", "ZCOR", and "OR"; "RR", "HR", "IRR", "RD", and "GEN" can only be returned on their fitted measure. Use transform = "EXP" for ratio-scale output from log-scale measures.
transform	optional display transformation. Currently "EXP" exponentiates log-scale measures "OR", "RR", "HR", and "IRR".
...	unused additional arguments. Supplied arguments trigger a warning.

Value

A BayesTools_table of class summary.marginal_means.brma.

---

summary.zplot_brma *Summarize Zplot Results*

Description

Creates summary tables for zplot estimates including EDR, Soric FDR, and the conventional missing-count diagnostic when applicable.

Usage

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

Arguments

object	a zplot_brma object.
probs	quantiles of the posterior distribution to display. Defaults to c(.025, .975).
...	additional arguments (currently unused).

Details

The summary includes:

EDR Expected Discovery Rate - the posterior average probability of significance for a new outcome from the bias-adjusted reference

Soric FDR Expected proportion of false discoveries among significant results, computed from EDR following Soric (1989)

Missing N The retained reference-bin-normalized inverse-weight diagnostic for independent univariate and conditional-cluster product models. See `as_zplot.brma()` for the additional assumptions required by a literal count interpretation. Unavailable for multivariate models, retained estimate-level or sampling-error sources, applicable integrated contexts, or best-rule selection models because relative weights do not identify a missing count

The footer reports the Observed Discovery Rate (ODR) with 95% CI for comparison with the model-estimated EDR.

Value

An object of class `"summary.zplot_brma"` containing the estimates table.

See Also

`as_zplot.brma()`, `plot.zplot_brma()`

`summary_heterogeneity` *Summary of Heterogeneity*

Description

Computes method-specific absolute and relative heterogeneity summaries for a fitted model.

Usage

```
summary_heterogeneity(object, ...)
```

Arguments

`object`	a fitted model object
`...`	additional arguments passed to methods

Value

Method-specific return value containing heterogeneity estimates.

See Also

`pooled_heterogeneity()`, `summary.brma()`

summary_heterogeneity.brma *Summary of Heterogeneity for brma Objects*

Description

Computes the absolute heterogeneity (tau, tau^2) and relative measures of heterogeneity (I^2, H^2) for a fitted brma object.

Usage

```
## S3 method for class 'brma'
summary_heterogeneity(object, probs = c(0.025, 0.975), component = "all", ...)

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

Arguments

object	a fitted brma object
probs	quantiles of the posterior distribution to be displayed. Defaults to c(.025, .975) for 95% credible intervals.
component	heterogeneity component to return for brma.mv() models. Defaults to "all". Use "total" for the variance-additive total heterogeneity. Random-formula allocation nodes can be selected by their displayed name, such as "study/esid".
...	additional arguments (currently ignored)
x	a summary_heterogeneity.brma_list object

Details

For standard (2-level) random-effects models, the function reports:

- tau: Between-study standard deviation
- tau2: Between-study variance
- I2: Percentage of total variance due to heterogeneity
- H2: Ratio of total to sampling variance

For multilevel (3-level) models with nested effects, the function additionally partitions heterogeneity into estimate-level and cluster-level components:

- rho: Proportion of heterogeneity variance allocated to clusters
- tau [within]: Estimate-level standard deviation
- tau [between]: Cluster-level standard deviation
- tau2 [within]: Estimate-level variance
- tau2 [between]: Cluster-level variance

- `I2 [within]`: Percentage of variance due to estimate-level heterogeneity
- `I2 [between]`: Percentage of variance due to cluster-level heterogeneity

For location-scale models, `tau2` aggregates the observation-specific heterogeneity variances τ_i^2; the corresponding `tau` summary is the square root of this aggregate variance. The relative I^2 and H^2 measures average the observation-specific indices.

The I^2 and H^2 statistics are computed following the metafor package implementation, using the "typical" sampling variance formula from Higgins and Thompson (2002). For multilevel models, the partitioned I^2 follows the approach described in the metafor documentation. For `RoBMA()` ensembles whose members differ in their PET and PEESE terms, each draw uses the typical sampling variance of its member's design (see `hatvalues.brma`).

For `brma.mv()` models, the method reports `tau` and `tau2` for each selected random-effect component. A genuine variance-additive aggregate is reported as `tau_total` and `tau2_total`; a mean-variance allocation scale is reported as `tau_common` and `tau2_common`. These RoBMA I/O names map to BayesTools' general random-effect names without changing the fitted quantities. Formula-random summaries are present only when a `random` formula is declared. A random-effect quantity that the fitted block cannot represent on the original predictor scale (for example the intercept SD, variance and correlations of a block with factor-specific slopes of a standardized predictor) is `NA`, with a row note, in the row that shows it. This table shows the SDs, variances and correlations, so all of them can be `NA` here; `summary.brma` shows the correlations and the fitted-scale multipliers, so only its correlation rows can be `NA`. Public random-effect correlations (`rho`) are included in their component's table, including homogeneous structures such as `ar()`, `cs()`, and `car()`. A variance-additive total across multiple components has no single correlation parameter and reports only its total SD and variance. When a known group covariance R is supplied, `tau` is the fitted base multiplier of that covariance kernel. It need not equal every row's marginal standard deviation when $\mathrm{diag}(R)$ is not one. Row-specific marginal standard deviations remain available from `predict(type = "terms.scale")` and in covariance-based prediction and diagnostics. I^2 and H^2 use the original sampling variances and the fitted location design to define the typical sampling variance. Likelihood weights affect these summaries through the posterior heterogeneity but do not replace the sampling variances v_i by v_i/w_i, so rescaling all weights does not change the reference variance (as in **metafor**). Relative I^2 and H^2 summaries are not reported for general known-V covariance structures, nor for a meta-regression whose location design leaves no residual degrees of freedom (as many estimable coefficients as estimates), where the typical sampling variance is undefined. For `component = "total"`, independent component variances are summed before reporting `tau2_total`; `tau_total` is the square root of this variance draw. When a random-formula model uses a shared total-SD plus variance-allocation node, `component = "all"` also includes an allocation-node table with the appropriate aggregate SD and variance plus `tau2_prop(<block>)` rows. For gated allocations, the aggregate is the realized model-averaged total and includes the all-off zero branch; proportions are realized shares conditional on positive total variance. For nested formulas such as `random = ~ 1 | study / esid`, this table is displayed under the user-facing path `"study/esid"`. For heterogeneous structured random effects, such as `random = ~ har(time | study)`, the allocation table reports the exhaustive semantic allocation family: aggregate SD and variance, level-specific component SDs and variances, correlations, SD multipliers, and variance multipliers. For row-shaped scale sources, absolute level SDs and variances use the observed rows assigned to each level; their aggregate uses the observed-source RMS. A redundant block owner is omitted for a bare formula or unnamed one-entry list and retained for explicitly named one-entry lists and multiple components.

Value

A list of class `summary_heterogeneity.brma` containing:

- `estimates`: A `BayesTools_table` with heterogeneity statistics
- `component`: The public heterogeneity-component name

For decomposed `brma.mv()` models with multiple selected components, a named list of such summary objects is returned.

References

Higgins JP, Thompson SG (2002). "Quantifying heterogeneity in a meta-analysis." *Statistics in Medicine*, **21**(11), 1539–1558. doi:10.1002/sim.1186.

See Also

`pooled_heterogeneity()`, `summary.brma()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- brma(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )

  summary_heterogeneity(fit)
}

## End(Not run)
```

---

summary_models *Summarize Model-Averaged Component Weights*

Description

Creates marginal or individual model-weight summaries for RoBMA-class product-space objects.

Usage

```
summary_models(object, ...)

## S3 method for class 'RoBMA'
summary_models(object, type = "marginal", include_mcmc_diagnostics = TRUE, ...)

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

Arguments

object	a fitted RoBMA-class product-space object, including RoBMA, RoBMA.mv, BMA/BMA.norm, BMA.glmm, and BMA.mv.
...	unused additional arguments. Supplied arguments trigger a warning.
type	whether to summarize marginal component prior distributions ("marginal") or individual model combinations ("individual").
include_mcmc_diagnostics	whether to include Bayes factor MCMC diagnostics in the output. Defaults to TRUE.
x	a summary_models.RoBMA object.

Details

Model families include ordinary mixtures, spike-and-slab priors, and declared Bernoulli inclusion gates of random-effect components. Fixed components without a model choice are omitted.

Value

A list of class summary_models.RoBMA with elements name and type. For type = "marginal", element marginal contains component tables with columns such as prior_prob, post_prob, and inclusion_BF. For type = "individual", element individual contains individual model combinations and posterior probabilities.

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- RoBMA(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")

  summary_models(fit)
  summary_models(fit, type = "individual")
}

## End(Not run)
```

true_effects *True Effects*

Description

Computes the estimated true effects (theta) from a fitted model. This is a separate S3 generic whose `brma` method delegates to `blup.brma`.

Usage

```
true_effects(object, ...)
```

Arguments

`object`	a fitted model object
`...`	additional arguments passed to methods

Value

Method-specific return value, typically a summary table or posterior samples of BLUP or empirical-Bayes true-effect summaries.

See Also

`blup()`, `predict.brma()`

true_effects.brma *True Effects for brma Objects*

Description

Computes the estimated true effects (theta) for a fitted brma object. This is an alias for `blup.brma`.

Usage

```
## S3 method for class 'brma'
true_effects(
  object,
  bias_adjusted = FALSE,
  output_measure = NULL,
  transform = NULL,
  probs = c(0.025, 0.975),
  ...
)
```

Arguments

object	a fitted brma object
bias_adjusted	whether to adjust for publication bias. Defaults to FALSE, which returns estimates including publication bias effects (i.e., what we expect the true effects to be given the biased observations). Set to TRUE to obtain bias-corrected estimates.
output_measure	effect-size measure for location/effect predictions. Defaults to the fitted measure. Supported conversions are among "SMD", "COR", "ZCOR", and "OR"; "RR", "HR", "IRR", "RD", and "GEN" can only be returned on their fitted measure. Use transform = "EXP" for ratio-scale output from log-scale measures.
transform	optional display transformation. Currently "EXP" exponentiates log-scale measures "OR", "RR", "HR", and "IRR".
probs	quantiles of the posterior distribution to be displayed. Defaults to c(.025, .975) for 95% credible intervals.
...	additional arguments passed to predict.brma; wrapper arguments such as newdata, type, quiet, output_measure, and transform are fixed by this method (supplying one is R's error for an argument matched twice, as for predict()), and other unknown arguments are unused and trigger a warning.

Details

This function is identical to blup.brma. See that function for full details on how true effects are computed.

Value

A brma_samples object containing posterior draws of BLUP or empirical-Bayes true-effect summaries with one column per estimate. For existing normal data, these are conditional BLUP means, not simulated latent-effect draws. When printed, displays a summary table. Use summary(), or as.data.frame(summary(x)), to obtain the summary table; as.data.frame(x) returns the posterior draws. The samples can be converted to **posterior** draws formats using as_draws().

See Also

blup.brma(), predict.brma(), pooled_effect(), pooled_heterogeneity()

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- brma(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )
```

```
  true_effects(fit)
}

## End(Not run)
```

`update.brma` *Update a brma Fit*

Description

Extends an existing fitted `brma` object by additional MCMC samples, updates study labels, and optionally recomputes cached fit-dependent quantities.

Usage

```
## S3 method for class 'brma'
update(
  object,
  formula. = NULL,
  ...,
  sample_extend = NULL,
  slab = NULL,
  autofit_control = NULL,
  convergence_checks = NULL,
  recompute = c("all", "drop"),
  parallel = NULL,
  cores = NULL,
  silent = NULL,
  evaluate = TRUE
)
```

Arguments

`object`	a fitted `brma` object.
`formula.`	unsupported; included for compatibility with `update`.
`...`	unsupported additional arguments.
`sample_extend`	integer. Number of additional samples per chain.
`slab`	optional character vector of study labels. Updating labels does not refit or extend the model.
`autofit_control`	list of autofit control settings. Values are merged with the existing settings before extending.
`convergence_checks`	list of convergence check settings. Values are merged with the existing settings and used to re-check the fit.

recompute	whether cached loo, waic, and marglik values already stored in object should be recomputed after extension ("all") or dropped with a warning ("drop").
parallel	logical. Whether to extend chains in parallel.
cores	integer. Number of cores to use when parallel = TRUE.
silent	logical. Whether to suppress JAGS output during extension.
evaluate	unsupported; included for compatibility with update.

Details

Extending a fit preserves the fitted model and adds posterior samples. No additional burn-in is requested. Rebuilding the compiled JAGS model, for example after loading a saved fit, for parallel workers, or under runjags' repeatable-method settings, can rerun adaptation before collecting those samples. Retained selection caches store normalizer calculations, not JAGS model pointers or sampler tuning. Prior, data, and model-structure updates are intentionally not supported by this method. For brma.mv() objects, the stored V, R, known-V backend, and marginalized random-effect metadata are preserved because the fitted model structure is reused.

Extensions continue the random-number generator state stored by JAGS. They cannot be restarted from a new seed without changing the meaning of chain continuation, so update() has no seed argument.

Retained selection-normalizer caches are restored when compatible. Set RoBMA.options(selection.cache_retain = TRUE) to retain the cache after extension, or use remove_selection_cache() to remove saved cache entries.

Only explicitly stored selection-sensitivity diagnostics are refreshed after chain extension, silently and using their stored simulation settings, independently of recompute. Models without a stored comparison do not compute one. Label-only updates retain the stored diagnostic. Use selection_sensitivity_diagnostic to inspect the refreshed result or error.

Named NULL elements in autofit_control and convergence_checks explicitly disable or clear the corresponding nullable setting. Omitted elements, a top-level NULL, and an empty list() inherit the setting stored in object.

Value

The updated brma object.

Examples

```
## Not run:
fit <- update(fit, sample_extend = 1000)
fit <- update(fit, slab = paste("Study", seq_len(nobs(fit))))

## End(Not run)
```

vif *Variance Inflation Factors*

Description

Computes variance inflation factors (VIF) for a fitted model with moderators, to assess multicollinearity among predictors.

Usage

```
vif(object, ...)
```

Arguments

object	a fitted model object
...	additional arguments passed to methods

Value

Method-specific return value containing VIF diagnostics.

References

(Fox and Monette 1992)

See Also

regplot(), summary.brma()

vif.brma *Variance Inflation Factors for brma Objects*

Description

Computes variance inflation factors (VIF) and generalized VIF (GVIF) for a fitted brma meta-regression model. Also optionally returns the posterior correlation matrix of regression coefficients.

Usage

```
## S3 method for class 'brma'
vif(
  object,
  posterior_correlation = TRUE,
  max_samples = Inf,
  ...,
  probs = NULL,
  include_samples = FALSE
)
```

Arguments

`object`	a fitted brma object with moderators
`posterior_correlation`	logical; whether to also compute and return the posterior correlation matrix of regression coefficients. Defaults to `TRUE`.
`max_samples`	maximum posterior draws used for the VIF/GVIF computation in every model class. Defaults to `Inf`. Finite values deterministically thin draws across posterior row order. The optional posterior coefficient correlation still uses all draws.
`...`	additional arguments (currently ignored)
`probs`	optional credible-interval probabilities for GVIF (ordinary VIF for one-df terms). Defaults to `NULL`, which retains the point-summary table. Requested empirical quantiles appear as `CI_<probability>` columns. These intervals describe GVIF, not its adjusted root.
`include_samples`	whether to include a `samples` data frame with posterior `draw` row indices before thinning, `term_index` (the row of the `vif` table, also distinguishing repeated display labels), `term`, `df`, `GVIF`, and `GVIF^(1/(2*df))`. Defaults to `FALSE`.

Details

VIF is computed from the correlation matrix derived from the coefficient variance-covariance matrix $(X'WX)^{-1}$. For standard meta-regression models, $W = \mathrm{diag}(w_i/(v_i + \tau^2))$ at each posterior heterogeneity draw. Scale and multilevel models use observation-specific τ_i and block-structured covariance where applicable. VIF/GVIF and its adjusted root are computed in each draw and averaged separately. These posterior means are generally different from applying VIF to an averaged covariance or to a posterior-mean heterogeneity. The mean adjusted GVIF is likewise generally different from transforming the mean GVIF. The diagnostic retains its GLS design-based meaning; it is not calculated from the empirical posterior covariance of the coefficients.

A VIF of 1 indicates no collinearity; values above 5 or 10 are commonly considered problematic.

The members of a `RoBMA()` ensemble can differ in their bias terms (PET adds a `PET` column, PEESE a `PEESE` column, and the null and weight-function members add none), so each posterior draw uses the design of its member. A location term summarizes all selected posterior draws; `PET` and `PEESE` summarize only draws in their own class (`NA` when no selected draw belongs to it). Their samples are `NA` outside that class, not zero, and their credible intervals are conditional on class membership. Location terms keep the specified location design, including terms with spike-and-slab priors.

For multi-column terms, such as factor contrasts, the Generalized VIF (GVIF) of Fox and Monette (1992) is reported. GVIF captures the joint inflation for all coefficients belonging to the same term. To enable comparison across terms with different degrees of freedom, $GVIF^{1/(2 \cdot df)}$ is also reported; this value can be compared against the usual VIF thresholds (after squaring).

When `posterior_correlation = TRUE`, the function also returns the posterior correlation matrix of the regression coefficients. This Bayesian diagnostic complements VIF: while VIF diagnoses the *potential* for collinearity problems (a data property), the posterior correlation shows the *realized* identification given the data and priors. Informative priors can mitigate collinearity, reducing posterior correlations even when VIF is high.

Data-only objects created with `only_data = TRUE` are rejected because VIF depends on fitted heterogeneity information, not just the parsed data.

For `brma.mv()` known-`V` models, VIF is computed from the full marginal GLS covariance. Formula random effects are marginalized through BayesTools' random-effect covariance metadata rather than treated as conditioned fitted effects.

For known-`V` models, the VIF requires applying the inverse of the marginal covariance. A covariance that is too ill-conditioned for that (a reciprocal condition number of about `2e-10` or less in any posterior draw used) is unavailable: the call stops with an error of class `RoBMA_known_v_precision_unavailable`, whose parent class is `RoBMA_numerical_unavailable`, instead of returning a rounded number. The condition carries the draw positions (`rows`) and their `reciprocal_condition`.

For generalized linear mixed models, this is a working effect-size-scale diagnostic based on the fitted model's approximate sampling variances. It is not a VIF derived directly from the binomial or Poisson information matrix. When the marginal covariance is fixed, the calculation reduces to the conventional fixed-covariance VIF used by **metafor**.

Value

An object of class `vif.brma` containing:

`vif` A data frame with columns `term`, `df`, `GVIF`, and `GVIF^(1/(2*df))` (= $GVIF^{1/(2 \cdot df)}$). For single-df terms, GVIF equals the standard VIF.

`posterior_correlation` A correlation matrix of posterior regression coefficient samples when requested and available; otherwise `NULL`.

`samples` The optional long data frame of draw-wise diagnostics when `include_samples = TRUE`. Printing and data-frame coercion of the result retain the summary-table view; extract this element explicitly for draws.

References

Fox J, Monette G (1992). "Generalized collinearity diagnostics." *Journal of the American Statistical Association*, **87**(417), 178–183. doi:10.1080/01621459.1992.10475190.

See Also

`regplot()`, `summary.brma()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE) &&
    requireNamespace("metafor", quietly = TRUE)) {
  data(dat.bcg, package = "metadat")
  dat <- metafor::escalc(
    measure = "RR",
    ai      = tpos,
    bi      = tneg,
    ci      = cpos,
    di      = cneg,
    data    = dat.bcg
  )
  fit <- brma(
```

```
    yi      = yi,
    vi      = vi,
    mods    = ~ ablat + year,
    data    = dat,
    measure = "RR",
    seed    = 1,
    silent  = TRUE
  )

  vif(fit)
}

## End(Not run)
```

`waic.brma` *WAIC for brma Objects*

Description

Extract the WAIC object from a brma model object. The WAIC must first be computed using `add_waic`.

Usage

```
## S3 method for class 'brma'
waic(x, unit = "estimate", ...)
```

Arguments

`x`	a brma model object.
`unit`	output/deletion unit. See `add_loo`.
`...`	additional arguments (currently unused).

Details

This function extracts the WAIC object that was previously computed and stored using `object <- add_waic(object, unit = unit)`. If WAIC has not been computed for the requested unit, an error is thrown.

This is the RoBMA S3 generic and `brma` method. The method is also registered for `waic`, so `loo::waic(fit)` extracts the cached WAIC object for `brma` fits. Use `waic` directly for raw log-likelihood arrays or matrices.

In most cases, LOO-PSIS (via `loo.brma`) is preferred over WAIC because it provides better estimates and includes diagnostics (Pareto k values) that indicate when the approximation may be unreliable.

Value

An object of class "waic" as returned by waic.

See Also

add_waic, loo.brma, waic

Wang2025 *70 effect sizes from a meta-analysis of ChatGPT's impact on student learning by Wang and Fan (2025)*

Description

The data set contains Hedges' g effect sizes, standard errors, sample sizes for experimental and control groups, and various study characteristics including grade level, type of course, duration, learning model, role of ChatGPT, and area of ChatGPT application. The meta-analysis examined the effect of ChatGPT on students' learning performance, learning perception, and higher-order thinking (Wang and Fan 2025).

Usage

```
Wang2025
```

Format

A data.frame with 12 columns and 70 observations:

Learning_effect Learning-effect outcome category.

Author_year Author-year study label.

N_EG Experimental-group sample size.

N_CG Control-group sample size.

g Hedges' g effect size.

Grade_level Grade level.

Type_of_course Course type.

Duration Study duration.

Learning_model Learning model.

Role_of_ChatGPT Role of ChatGPT in the intervention.

Area_of_ChatGPT_application Application area.

se Standard error of g.

References

Wang J, Fan W (2025). "The effect of ChatGPT on students' learning performance, learning perception, and higher-order thinking: insights from a meta-analysis." *Humanities and Social Sciences Communications*, **12**(1), 1–21. doi:10.1057/s4159902504787y.

`Weingarten2018` *582 effect sizes examining the ease-of-retrieval effect from a meta-analysis by Weingarten and Hutchinson (2018)*

Description

The data set contains correlation coefficients between the manipulation and outcome variable (r_xy), sample sizes (N), and various study characteristics including publication status, country (USA vs. other), number of few and many examples requested, whether the trial targeted episodic memory, paradigm type (standard vs. other), dataset type (proximal vs. distal), and mediation variables (r_xm, r_my). The meta-analysis examined whether subjective ease mediates the ease-of-retrieval effect, where participants list either few or many examples and then make judgments (Weingarten and Hutchinson 2018).

Usage

```
Weingarten2018
```

Format

A data.frame with 12 columns and 582 observations:

`r_xy` Correlation between manipulation and outcome.

`N` Sample size.

`paper_id` Paper identifier.

`published` Publication-status indicator.

`USA` Whether the study was conducted in the USA.

`number_of_few` Number of examples in the few-examples condition.

`number_of_many` Number of examples in the many-examples condition.

`episodic_memory` Whether the trial targeted episodic memory.

`standard_paradigm` Whether the standard paradigm was used.

`proximal_dataset` Whether the data set was proximal.

`r_xm` Correlation between manipulation and mediator.

`r_my` Correlation between mediator and outcome.

References

Weingarten E, Hutchinson JW (2018). "Does ease mediate the ease-of-retrieval effect? A meta-analysis." *Psychological Bulletin*, **144**(3), 227–283. doi:10.1037/bul0000122.

zplot.brma *Plot Zplot Diagnostics Directly*

Description

Convenience wrapper for creating and plotting zplot diagnostics from a fitted brma object.

Usage

```
## S3 method for class 'brma'
zplot(
  object,
  significance_level = stats::qnorm(0.975),
  summary_max_samples = 10000,
  conditioning_depth = "marginal",
  integration_control = set_selection_likelihood_control(),
  ...
)

## S3 method for class 'zplot_brma'
zplot(object, ...)
```

Arguments

object	a normal-outcome brma object, or a zplot_brma object.
significance_level	positive z-value threshold for significance. Defaults to qnorm(0.975) (two-sided alpha = 0.05). The threshold must be finite and its two-sided alpha, 2 * pnorm(significance_level, lower.tail = FALSE), strictly between 0 and 1. Zero (alpha = 1), a threshold so small that its alpha rounds to 1 (below about 1e-16), and one so large that its alpha underflows to 0 (above about 38.4) are refused because the Soric false discovery rate is undefined there.
summary_max_samples	retained for compatibility and validated as a posterior-sample budget, but unused by the default conversion path: direct plotting does not compute EDR or missing-study summaries. Use as_zplot(max_samples = ...) to compute those summaries. The plot-density max_samples argument is accepted through ...; it defaults to 1000 for brma.mv models and 10000 for other models. Use Inf to use all posterior samples.
conditioning_depth	predictive conditioning depth of the displayed densities. Defaults to "marginal"; see as_zplot() for details.
integration_control	numerical integration settings created by set_selection_likelihood_control(). These control integration within each posterior draw, independently of max_samples.

Factor blocks and supported full-covariance product-selection blocks use deterministic Gauss-Hermite quadrature where possible, with the existing adaptive randomized quasi-Monte Carlo fallback. Density-error checks use the peak within each posterior draw and also check the selection normalizer. Bounded CDF interpolation, Gaussian-mixture approximation, and omitted contributions are charged to the same numerical error budget.

`...` arguments passed to `plot.zplot_brma()`.

Details

For models using the default `as_zplot.brma()` method, direct plotting prepares the observed z-statistics and predictive target without computing EDR or missing-study summary integrals. It uses the same density computation and numerical checks as `plot(as_zplot(object))`. A failure confined to a summary integral therefore does not prevent a plot; density-integration failures still stop plotting. Use `as_zplot()` followed by `summary()` to obtain the summaries.

Custom `as_zplot()` methods for subclasses retain their conversion and plotting dispatch, including any summaries those methods compute.

When `object` already inherits from `zplot_brma`, `zplot()` dispatches directly to `plot.zplot_brma()` without recomputing stored summaries.

Value

`NULL` invisibly for base graphics, or a ggplot2 object.

See Also

`as_zplot.brma()`, `plot.zplot_brma()`, `summary.zplot_brma()`

Examples

```
## Not run:
if (requireNamespace("metadat", quietly = TRUE)) {
  data(dat.lehmann2018, package = "metadat")
  fit <- bPET(yi = yi, vi = vi, data = dat.lehmann2018, measure = "SMD")

  zplot(fit)
}

## End(Not run)
```

Index

∗ **datasets**
Anderson2010, 13
Andrews2021, 13
Bem2011, 33
Havrankova2025, 135
Hoogeveen2023, 137
Hoppen2025, 138
Johnides2025, 151
Kroupova2021, 153
Lui2015, 165
ManyLabs16, 165
Poulsen2006, 195
Wang2025, 296
Weingarten2018, 297
∗ **package**
RoBMA-package, 5

add_loo, *11, 12*, *111*, *121*, *156*, *158–160*, *163, 164*, *224*, *295*
add_loo *(*add_loo.brma*)*, 6
add_loo.brma, 6
add_marglik, *34, 35*, *76*, *157*, *194, 195*
add_marglik *(*add_marglik.brma*)*, 8
add_marglik.brma, 8
add_selection_sensitivity_diagnostics *(*selection_sensitivity_diagnostics*)*, 273
add_selection_sensitivity_diagnostics(), *109*, *280*
add_waic, *159*, *295, 296*
add_waic *(*add_waic.brma*)*, 11
add_waic.brma, 11
addpoly.default, *29*, *126, 127*
AIC.brma, 12, *156*
allocation_ref *(*prior_random*)*, 214
Anderson2010, 13
Andrews2021, 13
as.data.frame.brma, 15
as.data.frame.brma_samples, 16
as.data.frame.brma_samples_list *(*as.data.frame.brma_samples*)*, 16
as.data.frame.covratio.brma, 17
as.data.frame.dfbetas.brma, 18
as.data.frame.infl.brma, 18
as.data.frame.marginal_means.brma, 19
as.data.frame.summary.brma *(*as.data.frame.brma*)*, 15
as.data.frame.summary.brma_samples *(*as.data.frame.brma_samples*)*, 16
as.data.frame.summary.marginal_means.brma *(*as.data.frame.marginal_means.brma*)*, 19
as.data.frame.summary.zplot_brma *(*as.data.frame.zplot_brma*)*, 23
as.data.frame.summary_heterogeneity.brma, 20
as.data.frame.summary_heterogeneity.brma_list *(*as.data.frame.summary_heterogeneity.brma*)*, 20
as.data.frame.summary_models.RoBMA, 21
as.data.frame.vif.brma, 22
as.data.frame.zplot_brma, 23
as.matrix.brma_samples, 24
as_draws *(*as_draws.brma*)*, 24
as_draws.brma, 24, *28*
as_draws.brma_samples, *27*, 27
as_draws_array *(*as_draws.brma*)*, 24
as_draws_array.brma_samples *(*as_draws.brma_samples*)*, 27
as_draws_df *(*as_draws.brma*)*, 24
as_draws_df.brma_samples *(*as_draws.brma_samples*)*, 27
as_draws_list *(*as_draws.brma*)*, 24
as_draws_list.brma_samples *(*as_draws.brma_samples*)*, 27
as_draws_matrix *(*as_draws.brma*)*, 24

as_draws_matrix.brma_samples (as_draws.brma_samples), 27
as_draws_rvars (as_draws.brma), 24
as_draws_rvars.brma_samples (as_draws.brma_samples), 27
as_metafor_forest, 28
as_metafor_forest(), *128*
as_zplot (as_zplot.brma), 30
as_zplot.brma, 30
as_zplot.brma(), *278*, *283*, *299*

bayes_factor.brma (bf.brma), 34
BayesTools::harrell_davis_quantile(), *155*, *179*, *180*, *239*
BayesTools::plot_prior_list(), *186*
BayesTools::prior_random(), *54*, *219*
Bem2011, 33
bf, *76*, *157*
bf.brma, *10*, 34, *76*, *157*, *195*
BF_hypothesis (hypothesis), 139
bf_hypothesis (hypothesis), 139
bfunnel (funnel), 129
BIC.brma (AIC.brma), 12
blup, 35
blup(), *110*, *191*, *193*, *201*, *233*, *288*
blup.brma, 36, *233*, *288*, *289*
blup.brma(), *125*, *235*, *245*, *266*, *268*, *289*
BMA, *5*, 38
BMA(), *54*, *55*, *83*, *186*, *207*, *252*
BMA.glmm, 43
BMA.glmm(), *89*
BMA.mv, *5*, 49
BMA.mv(), *97*, *258*, *259*
bPEESE, *5*, 55
bPEESE(), *65*, *70*, *103*, *252*
bPEESE.mv, *5*, 60
bPEESE.mv(), *60*, *75*, *97*, *259*
bPET, *5*, 66
bPET(), *60*, *75*, *103*, *252*
bPET.mv, *5*, 70
bPET.mv(), *65*, *70*, *97*, *259*
bridge_sampler, *76*, *157*
bridge_sampler.brma, *9*, *10*, *35*, 76, *157*, *195*
bridge_sampler.brma(), *147*
brma, *5*, *27*, 77
brma(), *42*, *89*, *97*, *124*, *170*, *186*, *207*, *252*, *280*
brma.glmm, *5*, *48*, 84
brma.glmm(), *49*, *83*, *170*, *280*
brma.mv, *5*, *34*, *58*, *63*, *68*, *73*, *79*, *87*, 90, *93*, *100*, *106*, *161*, *195*
brma.mv(), *39*, *45*, *51*, *54*, *55*, *57*, *62*, *65*, *67*, *72*, *75*, *78*, *86*, *92*, *99*, *105*, *108*, *110*, *152*, *214*, *218*, *220*, *231*, *247*, *254*, *258*, *259*
bselmodel, *5*, 98
bselmodel(), *60*, *70*, *108*, *110*, *252*, *275*
bselmodel.mv, *5*, 103
bselmodel.mv(), *97*, *102*, *259*, *271*, *275*

check_loo (check_loo.brma), 111
check_loo.brma, 111
coef.brma, 112
contr.BayesTools, 112
contr.independent (contr.BayesTools), 112
contr.meandif (contr.BayesTools), 112
contr.orthonormal (contr.BayesTools), 112
cooks.distance.brma, 113, *116*, *123*, *149*
covratio (covratio.brma), 115
covratio.brma, 115, *149*

density_diagnostics, 116
density_diagnostics(), *144*, *147*
dfbetas.brma, 120, *149*
dffits (dffits.brma), 122
dffits.brma, *114*, *116*, 122, *149*
difftime, *260*
draws, *27*, *28*

estimate_unit_information_sd, 123
estimate_unit_information_sd(), *82*

fitted, *125*
fitted.brma, 124
fitted.brma(), *110*
forest, *29*
forest.brma, 126
forest.default, *29*, *126–128*
forest.metafor_forest.brma (forest.brma), 126
funnel, 129
funnel.brma(), *33*, *60*, *70*, *103*, *225*, *228*, *240*

galbraith (radial.brma), 226

hatvalues.brma, *114*, *123*, 134, *148*, *149*, *266*, *285*

Havrankova2025, 135
hist.zplot_brma, 136
hist.zplot_brma(), *33*, *156*, *180*
Hoogeveen2023, 137
Hoppen2025, 138
hypothesis, 139
hypothesis(), *116*, *119*, *167*, *169*, *172*, *173*, *175*, *177*
hypothesis_quantities(), *141*, *142*

influence.brma, *114*, *116*, *123*, 148
interpret, 149

Johnides2025, 151

known_v_factor, 151, *199*
known_v_factor(), *50*, *61*, *62*, *72*, *91*, *104*, *105*, *109*, *253*, *254*
Kroupova2021, 153

lines.brma *(*plot.brma*)*, 171
lines.marginal_means.brma *(*plot.marginal_means.brma*)*, 176
lines.zplot_brma, 154
lines.zplot_brma(), *137*, *180*
log_lik, *12*, *156*, 158
logLik.brma, *12*, 156, *158*
logml, *76*
logml.brma, *10*, *35*, *76*, 157, *195*
loo, *8*, *160*
loo *(*loo.brma*)*, 159
loo.brma, *8*, *9*, *12*, *156*, *159*, 159, *161–164*, *295*, *296*
loo.brma(), *55*, *259*, *268*
loo_compare, *8*, *160–162*
loo_compare *(*loo_compare.brma*)*, 160
loo_compare.brma, 160, *163*, *164*
loo_compare.loo, 161
loo_model_weights, *163*
loo_model_weights.brma, 162, *164*
loo_weights *(*loo_weights.brma*)*, 164
loo_weights.brma, *121*, 164
Lui2015, 165

ManyLabs16, 165
marginal_means, 166
marginal_means(), *142*, *147*
marginal_means.brma, 167
marginal_means.brma(), *177*
metafor::vcalc(), *50*, *61*, *72*, *91*, *104*, *109*, *152*, *253*

nobs.brma, 170

pareto_k_ids, *8*, *160*
plot(), *166*, *169*
plot.brma, 171
plot.brma(), *42*, *49*, *83*, *140*, *142*, *147*, *182*, *185*, *206*, *252*
plot.marginal_means.brma, 176
plot.zplot_brma, 178
plot.zplot_brma(), *33*, *137*, *156*, *283*, *299*
plot_diagnostic, 180
plot_diagnostic_autocorrelation *(*plot_diagnostic*)*, 180
plot_diagnostic_density *(*plot_diagnostic*)*, 180
plot_diagnostic_trace *(*plot_diagnostic*)*, 180
plot_pet_peese, 182
plot_prior, 184
plot_prior(), *175*, *206*, *207*
plot_weightfunction, 187
pooled_effect, *127*, 189, *197*
pooled_effect(), *37*, *128*, *193*, *201*, *235*, *245*, *289*
pooled_effect.brma, 189
pooled_effect.brma(), *228*
pooled_heterogeneity, 191, *197*, *200*
pooled_heterogeneity(), *37*, *191*, *201*, *283*, *286*, *289*
pooled_heterogeneity.brma, 192
pooled_heterogeneity.brma(), *228*
post_prob, *76*, *157*
post_prob.brma, *10*, *35*, *76*, *157*, 194
Poulsen2006, 195
predict, *200*
predict.brma, *36*, *125*, *127*, *190*, *192*, 196, *234*, *289*
predict.brma(), *33*, *36*, *37*, *83*, *89*, *97*, *110*, *125*, *128*, *133*, *189*, *191–193*, *233*, *235*, *240*, *245*, *266*, *288*, *289*
print.brma *(*summary.brma*)*, 278
print.brma_samples, 201
print.interpret.brma *(*interpret*)*, 149
print.marginal_means.brma, 202
print.RoBMA_data, 202
print.summary.brma *(*summary.brma*)*, 278

print.summary.brma_samples, 203
print.summary.marginal_means.brma, 204
print.summary.zplot_brma, 204
print.summary_heterogeneity.brma, 205
print.summary_heterogeneity.brma_list (summary_heterogeneity.brma), 284
print.summary_models.RoBMA (summary_models), 286
print.vif.brma, 205
print_prior, 206
prior, 207
prior(), *186*, *207*, *220*
prior_factor, 208, *212*
prior_informed, 209
prior_lkj (prior_random), 214
prior_none, 210
prior_ordered, 211
prior_PEESE, 212
prior_PET, 213
prior_random, *58*, *63*, *68*, *73*, *79*, *87*, *93*, *100*, *106*, 214, *231*
prior_specification, *46*, *87*, *248*, *255*
prior_weightfunction, 221, *271*
publication_bias_prior_specification, *60*, *70*, *103*, *212, 213*, *222*, *248, 249*, *252*, *255, 256*

qqnorm.brma, 223

radial (radial.brma), 226
radial.brma, 226
radial.brma(), *225*
random-effect formula structure tags, *90*, *95*
random_block, *229*
random_block (prior_random), 214
random_covariance, *231*
random_covariance (prior_random), 214
random_effect_formula_tags, *97*, 229
random_effect_formula_tags(), *218*, *220*
random_effect_prior_specification, *54*, *58*, *63*, *68*, *73*, *79*, *87*, *93*, *97*, *100*, *106*, *231*, *258*
random_effect_prior_specification(), *220*
random_monitor (prior_random), 214
random_new_levels (prior_random), 214
random_sd_source (prior_random), 214
random_variance_allocation (prior_random), 214
ranef, 232
ranef(), *110*
ranef.brma, 233
regplot (regplot.brma), 235
regplot(), *166*, *169*, *292*, *294*
regplot.brma, 235
regplot.brma(), *33*
remove_selection_cache, 240, *263*, *269*
remove_selection_cache(), *291*
residuals.brma, *130*, *132*, *225*, 242, *268*
residuals.brma(), *125*, *133*, *225*, *266*, *268*
RoBMA, *5*, *42, 43*, *48*, 245
RoBMA(), *42*, *49*, *60*, *70*, *83*, *103*, *175*, *183*, *186*, *188*, *207*, *258, 259*, *261*, *275*
RoBMA-package, 5
RoBMA.get_option (RoBMA_options), 262
RoBMA.mv, *5*, 252
RoBMA.mv(), *55*, *97*, *252*, *271*, *275*
RoBMA.options (RoBMA_options), 262
RoBMA.package (RoBMA-package), 5
RoBMA_control, 260
RoBMA_options, 262
RoBMA_package (RoBMA-package), 5
rstandard.brma, *130*, *132*, 265, *268*
rstandard.brma(), *133*, *225*, *245*, *268*
rstudent.brma, *130*, *132*, *149*, *266*, 267
rstudent.brma(), *133*, *225*, *245*, *266*

selection_cache_info, 268
selection_model, 270
selection_model(), *97*, *102*, *107, 108*, *251*, *257, 258*
selection_sampler_info, *263*, 272
selection_sensitivity_diagnostics, 273
selection_sensitivity_diagnostics(), *109*, *280*, *291*
set_autofit_control (RoBMA_control), 260
set_autofit_control(), *42*, *48*, *53*, *59*, *64*, *69*, *75*, *81*, *89*, *94*, *102*, *108*, *250*, *258*
set_convergence_checks (RoBMA_control), 260
set_convergence_checks(), *42*, *48*, *54*, *59*, *65*, *69*, *75*, *81*, *89*, *94*, *102*, *108*, *250*, *258*
set_selection_likelihood_control, 275
set_selection_likelihood_control(), *102*, *107*, *110*, *143*, *169*, *174*, *177*,

251, *257*, *259*, *274*, *275*
summary(), *166*, *169*
summary.brma, 278, *285*
summary.brma(), *42*, *49*, *55*, *60*, *65*, *70*, *75*, *83*, *89*, *97*, *103*, *110*, *112*, *166*, *169*, *182*, *252*, *259*, *283*, *286*, *292*, *294*
summary.brma_samples, 281
summary.marginal_means.brma, 281
summary.zplot_brma, 282
summary.zplot_brma(), *33*, *180*, *299*
summary_heterogeneity, *148*, 283
summary_heterogeneity(), *280*
summary_heterogeneity.brma, 284
summary_models, 286
summary_models(), *55*, *259*
supsmu, *224*

true_effects, 288
true_effects(), *37*
true_effects.brma, 288

update, *290*, *291*
update(), *280*
update.brma, 290

vif, 292
vif.brma, 292

waic, *11*, *12*, *295*, *296*
waic *(*waic.brma*)*, 295
waic.brma, *12*, 295
waic.brma(), *55*, *259*
Wang2025, 296
Weingarten2018, 297
wf_cumulative *(*prior_weightfunction*)*, 221
wf_fixed *(*prior_weightfunction*)*, 221
wf_independent *(*prior_weightfunction*)*, 221

zplot *(*zplot.brma*)*, 298
zplot.brma, 298
zplot.brma(), *133*