Package: RoBMA (via r-universe)

October 29, 2026

Title Robust Bayesian Meta-Analyses

Version 4.0.0

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, 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 (https://mcmc-jags.sourceforge.io/)

NeedsCompilation yes

Depends R (>= 4.0.0)

Imports BayesTools (>= 0.3.0), bridgesampling, loo, MASS, parallel, runjags, rjags, stats, graphics, mvtnorm, scales, Rdpack, rlang, coda, ggplot2

Suggests metafor, posterior, weightr, lme4, fixest, metaBMA, emmeans, metadat, testthat, vdiffr, knitr, rmarkdown, covr

LinkingTo 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.)

Config/pak/sysreqs jags

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

Date/Publication 2026-05-07 15:07:58 UTC

RemoteUrl https://github.com/cran/RoBMA

RemoteRef HEAD

RemoteSha 8f5194f3d8764a18f7de390b474b00f254f3b969

Contents

RoBMA-package 5
add_loo.brma 6
add_marglik.brma 7
add_waic.brma 8
Anderson2010 9
Andrews2021 10
as.matrix.brma_samples 11
as_draws.brma 11
as_draws.brma_samples 13
as_zplot.brma 14
Bem2011 16
bf.brma 17
blup 18
blup.brma 19
BMA 20
BMA.glmm 24
bPEESE 29
bPET 33
bridge_sampler.brma 36
brma 37
brma.glmm 44
bselmodel 48
check_loo.brma 52
coef.brma 53
contr.BayesTools 53

cooks.distance.brma . . . 54
covratio.brma . . . 56
data_input . . . 57
dfbetas.brma . . . 58
dffits.brma . . . 60
estimate_unit_information_sd . . . 61
fitted.brma . . . 62
fitting_specification . . . 64
funnel.brma . . . 65
hatvalues.brma . . . 68
Havrankova2025 . . . 69
hist.zplot_brma . . . 70
Hoppen2025 . . . 71
influence.brma . . . 72
interpret . . . 73
Johnides2025 . . . 74
Kroupova2021 . . . 75
lines.zplot_brma . . . 76
logLik.brma . . . 77
logml.brma . . . 78
loo.brma . . . 79
loo_compare.brma . . . 80
loo_compare.loo . . . 81
loo_weights.brma . . . 81
Lui2015 . . . 82
ManyLabs16 . . . 83
marginal_means . . . 83
marginal_means.brma . . . 84
nobs.brma . . . 85
plot.brma . . . 86
plot.marginal_means.brma . . . 88
plot.zplot_brma . . . 89
plot_diagnostic . . . 91
plot_pet_peese . . . 93
plot_prior . . . 94
plot_weightfunction . . . 96
pooled_effect . . . 98
pooled_effect.brma . . . 98
pooled_heterogeneity . . . 100
pooled_heterogeneity.brma . . . 101
post_prob.brma . . . 102
Poulsen2006 . . . 103
predict.brma . . . 104
print.brma_samples . . . 107
print.marginal_means.brma . . . 108
print.RoBMA_data . . . 108
print.summary.brma_samples . . . 109
print.summary.marginal_means.brma . . . 109

print.summary.zplot_brma . . . 110
print.summary_heterogeneity.brma . . . 110
print.vif.brma . . . 111
print_prior . . . 111
prior . . . 113
prior_factor . . . 114
prior_informed . . . 115
prior_none . . . 116
prior_PEESE . . . 116
prior_PET . . . 117
prior_specification . . . 118
prior_weightfunction . . . 121
publication_bias_prior_specification . . . 123
qqnorm.brma . . . 124
radial.brma . . . 127
ranef . . . 130
ranef.brma . . . 131
regplot.brma . . . 132
residuals.brma . . . 136
RoBMA . . . 139
RoBMA_control . . . 144
RoBMA_options . . . 146
RoBMA_prior_specification . . . 147
rstandard.brma . . . 153
rstudent.brma . . . 155
summary.brma . . . 156
summary.brma_samples . . . 158
summary.marginal_means.brma . . . 158
summary.zplot_brma . . . 159
summary_heterogeneity . . . 160
summary_heterogeneity.brma . . . 161
summary_models . . . 163
true_effects . . . 164
true_effects.brma . . . 164
update.brma . . . 166
vif . . . 168
vif.brma . . . 168
waic.brma . . . 170
Wang2025 . . . 171
Weingarten2018 . . . 172
zplot.brma . . . 173

Index **175**

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`, `BMA`, `brma`, `brma.glmm`, `bselmodel`, `bPET`, and `bPEESE`.

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.
`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.

With `unit = "cluster"`, LOO-CV is computed with one joint contribution per cluster. For unweighted normal models without selection this uses the analytic cluster block covariance. Selection, data-weighted normal, and GLMM models integrate the held-out cluster effect with Gauss-Hermite quadrature.

For selection models, the LOO evaluates the weighted likelihood, conditioning on the posterior omega samples.

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 conditioning-depth targets.

Important for model comparison: When comparing models via `loo_compare`, the selection is based on expected out-of-sample predictive performance. This evaluates how well models predict *new* observations, not how well they fit the observed data.

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, ...)
```

Arguments

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

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`.

Product-space model-averaging objects (`BMA.norm`, `BMA.glmm`, and `RoBMA`) do not expose a bridge-sampling marginal likelihood; use predictive comparison methods such as `loo.brma` instead.

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.

Value

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

See Also

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

`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.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, ...)

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

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

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

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

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

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.

Details

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. 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.

Value

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

See Also

draws, as_draws.brma

---

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,
  ...
)
```

Arguments

`object`	a normal-outcome `brma` object.
`significance_level`	z-value threshold for significance. Defaults to `qnorm(0.975)` (two-sided alpha = 0.05).
`max_samples`	maximum number of posterior samples for estimation. Defaults to 10000. Use `Inf` to use all posterior samples.
`...`	additional arguments (currently unused).

Details

Zplot analysis estimates the Expected Discovery Rate (EDR), the posterior mean probability that an exact replication would be statistically significant at the supplied threshold. This provides insight into the replicability of findings in a literature.

The implementation uses extrapolation mode by default, which removes selection-model weights when evaluating the model-implied z-value density. PET/PEESE regression terms remain part of the fitted location model. 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.

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 missing-study `weights`

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

See Also

`summary.zplot_brma()`, `plot.zplot_brma()`, `hist.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")

  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, weights and cluster identifiers.

Value

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.

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.

Details

This function is a convenience wrapper around `predict.brma(..., type = "effect", newdata = NULL)`.

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.

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()` to obtain the summary table directly. 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 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.

ni — an optional vector of sample sizes. Used for measure = "GEN" or when estimating "UISD").

mods — an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in data.

scale — an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in data.

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.

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.

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.

prior_heterogeneity_allocation — prior distribution(s) for the alternative cluster-level heterogeneity allocation component(s).

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 single prior applies to all terms; a named list can specify term-specific components.

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.

`prior_heterogeneity_allocation_null`
: prior distribution(s) for the null cluster-level heterogeneity allocation component(s).

`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"`, while model-averaging constructors use `"meandif"`.

`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. For constructors with publication-bias prior distributions, `rescale_priors` does not rescale them except for the default PEESE prior's UISD adjustment. 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. 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 = "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 = 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 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.
mods	an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data`.
scale	an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data`.
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.
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.

`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.

`prior_heterogeneity_allocation`
prior distribution(s) for the alternative cluster-level heterogeneity allocation component(s).

`prior_baserate` prior distribution for the estimate-specific midpoint base-rate probability in binomial GLMM models. If omitted or `NULL`, defaults to independent `Beta(1, 1)` priors.

`prior_lograte` prior distribution for the estimate-specific midpoint log-rate in Poisson GLMM models. If omitted or `NULL`, a data-based unit-information normal prior is used independently for each estimate.

`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 single prior applies to all terms; a named list can specify term-specific components.

`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.

`prior_heterogeneity_allocation_null`
prior distribution(s) for the null cluster-level heterogeneity allocation component(s).

`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"`, while model-averaging constructors use `"meandif"`.

`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. For constructors with publication-bias prior distributions, `rescale_priors` does not rescale them except for the default PEESE prior's UISD adjustment. 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. 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.

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)
```

---

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 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.
ni	an optional vector of sample sizes. Used for `measure = "GEN"` or when estimating `"UISD"`).
mods	an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data`.
scale	an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data`.
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.
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, 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, a default prior is constructed. In single-model functions, explicit `NULL` or `FALSE` sets a spike at zero.

`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.

`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.

`prior_heterogeneity_allocation` prior distribution for the fraction of heterogeneity allocated to the cluster-level component in multilevel models (ρ). If omitted or `NULL`, defaults to `Beta(1, 1)`.

`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"`, while model-averaging constructors use `"meandif"`.

`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. For constructors with publication-bias prior distributions, `rescale_priors` does not rescale them except for the default PEESE prior's UISD adjustment. 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. 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

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)
```

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 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.
ni	an optional vector of sample sizes. Used for `measure = "GEN"` or when estimating `"UISD"`).
mods	an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data`.
scale	an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data`.
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.
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, 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, a default prior is constructed. In single-model functions, explicit `NULL` or `FALSE` sets a spike at zero.
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.
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.
prior_heterogeneity_allocation	prior distribution for the fraction of heterogeneity allocated to the cluster-level component in multilevel models (ρ). If omitted or `NULL`, defaults to `Beta(1, 1)`.

`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"`, while model-averaging constructors use `"meandif"`.

`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. For constructors with publication-bias prior distributions, `rescale_priors` does not rescale them except for the default PEESE prior's UISD adjustment. 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. 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

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)
```

---

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 bridge sampling object that was previously computed and stored using add_marglik. If the marginal likelihood has not been computed, an error is thrown. Product-space model-averaging objects (BMA.norm, BMA.glmm, and RoBMA) do not expose bridge-sampling marginal likelihoods.

The returned object can be used for Bayesian model comparison via bf and post_prob.

Value

An object of class "bridge" 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 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.

`ni` an optional vector of sample sizes. Used for `measure = "GEN"` or when estimating `"UISD"`).

`mods` an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data`.

`scale` an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data`.

`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.

`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, 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, a default prior is constructed. In single-model functions, explicit `NULL` or `FALSE` sets a spike at zero.

`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.

`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.

`prior_heterogeneity_allocation`
prior distribution for the fraction of heterogeneity allocated to the cluster-level component in multilevel models (ρ). If omitted or `NULL`, defaults to `Beta(1, 1)`.

`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"`, while model-averaging constructors use `"meandif"`.

`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. For constructors with publication-bias prior distributions, `rescale_priors` does not rescale them except for the default PEESE prior's UISD adjustment. 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. 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 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.
mods	an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data`.
scale	an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data`.
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.

`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, 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, a default prior is constructed. In single-model functions, explicit `NULL` or `FALSE` sets a spike at zero.

`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.

`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.

`prior_heterogeneity_allocation` prior distribution for the fraction of heterogeneity allocated to the cluster-level component in multilevel models (ρ). If omitted or `NULL`, defaults to `Beta(1, 1)`.

`prior_baserate` prior distribution for the estimate-specific midpoint base-rate probability in binomial GLMM models. If omitted or `NULL`, defaults to independent `Beta(1, 1)` priors.

`prior_lograte` prior distribution for the estimate-specific midpoint log-rate in Poisson GLMM models. If omitted or `NULL`, a data-based unit-information normal prior is used independently for each estimate.

`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"`, while model-averaging constructors use `"meandif"`.

`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. For constructors with publication-bias prior distributions, `rescale_priors` does not rescale them except for the default PEESE prior's UISD adjustment. 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. 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)
```

---

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,
  ...
)
```

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 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.
ni	an optional vector of sample sizes. Used for `measure = "GEN"` or when estimating `"UISD"`).
mods	an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data`.
scale	an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data`.
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.

`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, 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, a default prior is constructed. In single-model functions, explicit `NULL` or `FALSE` sets a spike at zero.

`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.

`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.

`prior_heterogeneity_allocation`
prior distribution for the fraction of heterogeneity allocated to the cluster-level component in multilevel models (ρ). If omitted or `NULL`, defaults to `Beta(1, 1)`.

`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"`, while model-averaging constructors use `"meandif"`.

`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. For constructors with publication-bias prior distributions, `rescale_priors` does not rescale them except for the default PEESE prior's UISD adjustment. 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. 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.

`...`
: 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.

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)
```

`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)`. The method warns when any Pareto k diagnostic is greater than 0.7.

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. Code is based on `stanova::contr.bayes` and corresponding to 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 brma object.
...	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 posterior Mahalanobis distance between the full-data and leave-one-out fitted-value vectors:

$$D_i = \frac{\Delta_i' V_\mu^+ \Delta_i}{P}$$

where $\Delta_i = \hat{\mu} - \hat{\mu}_{(-i)}$, V_μ^+ is the generalized inverse of the full-posterior fitted-value covariance, and P is the rank of the fixed-effect model matrix.

Value

A numeric vector of Cook's distance values, one for each observation.

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", ...)
```

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.
`...`	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). If any included parameter has zero posterior variance, or if a full or LOO covariance determinant is zero or non-finite, COVRATIO is undefined. In that case, values are reported as `NaN` with a printed note when available.

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)
```

`data_input` *Input Data Specification*

Description

Shared data-input arguments used by the RoBMA fitting functions.

Normal models use approximate effect-size estimates supplied through `yi` with either `vi` or `sei`. GLMM models use the raw two-arm count arguments for binomial (`measure = "OR"`) or Poisson (`measure = "IRR"`) outcomes.

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 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.
`ni`	an optional vector of sample sizes. Used for `measure = "GEN"` or when estimating `"UISD"`).
`mods`	an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data`.
`scale`	an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data`.
`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.

`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.

`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.

`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,
  ...
)
```

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 coefficients. Defaults to `FALSE`. When set to `TRUE`, standardized meta-regression coefficients are returned for the intercept and continuous predictors. These coefficients correspond to the standardized scale on which prior distributions are specified by default (i.e., `standardize_continuous_predictors = TRUE`).
`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`.
`...`	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 `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.

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 near zero, the corresponding DFFITS value is returned as `NA`.

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 complete numeric vector of strictly positive standard errors for each study.
ni	a complete numeric vector of 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"` 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()`

`fitting_specification` *Fitting specification*

Description

The `brma` family of functions uses the following arguments to specify the MCMC sampling and fitting settings.

Arguments

`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`.

See Also

`brma`, `set_autofit_control`, `set_convergence_checks`

`funnel.brma` *Funnel Plot for brma Object*

Description

`funnel.brma` creates a funnel plot for a fitted brma object. For intercept-only models without scale regression, the default outcome mode displays observed effect sizes against the fitted sampling distribution. For models with location or scale moderators, the default residual mode displays residuals against a standard-error funnel.

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",
  ...
)
```

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: For intercept-only models without scale regression, displays observed effect sizes against the fitted sampling distribution funnel; for models with moderators or scale regression, automatically uses residual mode. • `FALSE`: explicitly requests outcome mode. • `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"`.

Only used when funnel is in residual mode.

`unit` output unit for residual mode. Only `"estimate"` is implemented in this pass.

`conditioning_depth`
residual conditioning depth for residual mode. Options are `"marginal"`, `"cluster"`, and `"estimate"`. The default LOO-PIT residual path is available only with marginal conditioning depth.

`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. When `TRUE` and the model includes PET/PEESE, incorporates the expected skew from these regression adjustments.

`max_samples` maximum number of posterior samples used for model-averaged publication-bias funnel contours. Defaults to `10000`. Use `Inf` to use all posterior samples.

`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 funnel plot has two modes. If `residual` is not specified, the mode is chosen automatically from the fitted model: intercept-only models without scale regression use outcome mode, whereas models with location or scale moderators use residual mode.

Outcome mode (intercept-only models without scale regression): 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 pooled effect; 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 sampling distribution, optionally incorporating heterogeneity and publication bias.

Residual mode (models with moderators or scale regression): Displays residuals on the x-axis and standard errors on the y-axis. The funnel region represents the central 95\ $N(0, \mathrm{SE}^2)$. With `type = "LOO-PIT"`, the plotted residuals and standard errors are the raw-scale LOO predictive companions returned by `rstudent.brma`; the PIT-normalized `z` values are used by `qqnorm.brma` and influence diagnostics. With `type = "rstandard"`, the plotted values are internally standardized residual companions from `rstandard.brma`. With `type = "outcome"`, these are raw outcome residuals. Under a correctly specified model, most points should fall within this region.

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, observed effect sizes are computed from the raw frequency data using formulas equivalent to `metafor::escalc`. Residual-mode GLMM funnels use approximate effect-size-scale residual/PIT companions, not exact PIT diagnostics for the raw count likelihood.

Value

If `as_data = TRUE`, `funnel.brma` returns a list with the data used for plotting, including the plotted points, funnel polygons, plotting limits, labels, and reference line. 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()`

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)
  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, ...)
```

Arguments

`model`	a fitted brma object
`...`	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 weight matrix inverse to the marginal variance matrix.

The hat matrix is computed for each posterior draw and then averaged over draws, matching the vector output shape used by `metafor`.

This method is available only for normal outcome models without weightfunction selection.

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 existing plot. Defaults to `FALSE`.
`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.

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.

Value

`NULL` invisibly for base graphics, or a ggplot2 object.

See Also

`plot.zplot_brma()`, `lines.zplot_brma()`

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.

`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).

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 aggregate heterogeneity. For scale models, aggregation is over the remaining scale-model rows after each deletion.
`dfbs`	A data frame with DFBETAS values for the model coefficients.

Undefined determinant- or variance-standardized diagnostics are reported as `NaN` and printed with an explanatory note.

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.

`Kroupova2021` *881 estimates from 69 studies of a relationship between employment and educational outcomes collected by Kroupova et al. (2021)*

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

http://meta-analysis.cz/students

References

Kroupova K, Havranek T, Irsova Z (2021). "Student employment and education: A meta-analysis." *CEPR Discussion Paper*. https://www.ssrn.com/abstract=3928863.

`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 = 10000,
  plot_ci = TRUE,
  extrapolate = FALSE,
  from = -6,
  to = 6,
  by = 0.05,
  length.out = NULL,
  col = "black",
  as_data = FALSE,
  ...
)
```

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)`.
`max_samples`	maximum posterior samples for density. Defaults to 10000. Use `Inf` to use all posterior samples.
`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`.
`...`	additional graphical parameters.

Details

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. Under selection models, the curve is scaled by inverse selection probability and 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).

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.

Value

An S x K or S x G matrix of log-likelihood values.

See Also

`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, and RoBMA) do not expose bridge-sampling marginal likelihoods.

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)
```

`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 out-of-sample predictive performance. This evaluates how well models predict *new* observations, not how well they fit the observed data. RoBMA rejects comparisons with different outcome targets/data, `unit`, or implied `conditioning_depth`.

Value

A matrix of class `"compare.loo"` as returned by `loo_compare`.

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

A matrix of class `"compare.loo"` as returned by `loo_compare`.

See Also

`loo.brma`, `loo_compare`

`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.

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.

`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(

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,
  ...
)
```

Arguments

object	a fitted brma object with moderators.
null_hypothesis	point null hypothesis used for inclusion Bayes factors. Defaults to 0.
normal_approximation	whether prior and posterior density at the null should be approximated with a normal distribution. Defaults to FALSE.
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.
...	additional arguments (currently ignored).

Value

A list of class marginal_means.brma containing the BayesTools marginal_inference object and parameter metadata.

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,
  ...
)
```

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.
parameter_mods	character. Moderator term to plot. Use "intercept" for the adjusted effect in meta-regression models.
parameter_scale	character. Scale-regression term to plot. 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 display transformation. Currently "EXP" exponentiates log-scale measures "OR", "RR", "HR", and "IRR".
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.
...	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.

Value

plot.brma returns either NULL 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")

  plot(fit, parameter = "mu")
  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,
  ...
)
```

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".
...	additional graphical arguments passed to BayesTools::plot_marginal().

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 = 10000,
  plot_fit = TRUE,
  plot_extrapolation = TRUE,
  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,
  ...
)
```

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).
max_samples	maximum posterior samples for density estimation. Defaults to 10000. Use Inf to use all posterior samples.
plot_fit	whether to show fitted density (with bias adjustments). Defaults to TRUE.
plot_extrapolation	whether to show extrapolated density (bias removed). Defaults to TRUE.
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).
...	additional graphical parameters passed to components.

Details

The plot displays 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.

Extrapolation (blue) Bias-corrected curve representing the hypothetical distribution without selective reporting, scaled by the expected suppressed studies under selection models. This curve is not normalized to integrate to one when selection implies missing studies.

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,
  ...
)

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,
  ...
)

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,
  ...
)

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,
  ...
)
```

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.
parameter_mods	moderator term for location regression.
parameter_scale	term for scale regression.
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.

Value

plot_diagnostic returns the object returned by BayesTools::JAGS_diagnostics(), 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.

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",
  ...
)
```

Arguments

x	a brma, BMA, or RoBMA object, or a prior distribution object.
...	additional arguments passed to the prior plotting method.
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.
parameter_mods	character. Moderator term to plot. Use "intercept" for the adjusted effect in meta-regression models.
parameter_scale	character. Scale-regression term to plot. 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.
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".

Details

output_measure and transform transform the prior plotting scale only for effect-size location priors ("mu" or the meta-regression intercept).

Value

plot_prior returns either NULL invisibly if plot_type = "base" or a ggplot2 object if 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_mods = "Preregistered")
}

## 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 one-sided p-values should be shown as rug marks on the weightfunction axis. 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 (aggregated) effect size estimate from a fitted brma object by averaging across the moderator model matrix.

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. Selection-model weighting affects response predictions, not this type = "terms" wrapper.
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; wrapper arguments such as newdata, type, and quiet are fixed.

Details

This function is a convenience wrapper around predict.brma(...,type = "terms", newdata = TRUE, bias_adjusted = TRUE, quiet = TRUE).

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 models without moderators, this returns the single mu parameter.

Value

A brma_samples object containing posterior samples. When printed, displays a summary table. Use summary() to obtain the summary table directly. 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 the pooled (aggregated) heterogeneity estimate (tau) 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 the pooled (aggregated) heterogeneity estimate (tau) from a fitted brma object by averaging across the scale model matrix.

Usage

```
## S3 method for class 'brma'
pooled_heterogeneity(object, probs = c(0.025, 0.975), conditional = FALSE, ...)
```

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.
...	additional arguments passed to predict.brma; wrapper arguments such as newdata, type, and quiet are fixed.

Details

This function is a convenience wrapper around predict.brma(..., type = "terms.scale", newdata = TRUE).

For location-scale models (with scale regression), the pooled heterogeneity averages tau across the scale model matrix proportionately to the levels observed in the data.

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).

Value

A brma_samples object containing posterior samples. When printed, displays a summary table. Use summary() to obtain the summary table directly. 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. Supplied values are normalized internally.
`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, weights and cluster identifiers.

Value

A named numeric vector with posterior model probabilities (i.e., which sum 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,
  ...
)
```

Arguments

`object`	a fitted brma object
`newdata`	specification for prediction data. Defaults to `NULL` which corresponds to prediction for the observed data. Alternatives are:

- TRUE returns a single aggregated prediction (either the scalar parameter for models without moderators/scale, or the average across the model matrix for regression models). Not available for type = "response" since observation-level sampling variance is required.
- A data.frame (for meta-regression) or a named list with effect size measure and variability metrics (for meta-analysis) for new studies. The input must contain the variables used by the requested prediction type. Outcome columns are optional for type = "terms" unless PET/PEESE bias terms are included via bias_adjusted = FALSE.

type — type of prediction to be performed. Options are:

- "terms" (alias: "marginal"): Fixed-effect parameters only (mu). Produces the mean effect size estimate at the given predictor levels, not accounting for random effects.
- "cluster": Fixed effects plus cluster-level random effects (mu + gamma). Only available for multilevel (3-level) models.
- "estimate" (aliases: "effect", "blup"): True effects (mu + gamma + theta). For existing normal data, returns posterior draws of conditional BLUP means, not simulated latent-effect draws. For new data, samples estimate-level random effects.
- "response" (alias: "outcome"): Predicted observed values (yi). Incorporates both heterogeneity and sampling variability.
- "terms.scale": Scale parameter (tau), incorporating scale regression if present.

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.

`quiet` logical; whether to suppress informational messages about prediction scale and bias adjustment.

`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.

`...` additional arguments

Details

Type hierarchy:

- `"terms"`: mu (fixed effects only)
- `"cluster"`: mu + gamma (adds cluster-level random effect)
- `"estimate"`: mu + gamma + theta (adds estimate-level random effect)
- `"response"`: mu + gamma + theta + epsilon (adds sampling error)

For existing normal observations, `type = "estimate"` reports the conditional BLUP mean $E(\theta_i \mid y_i, \mu_i, \tau_i)$ for each posterior draw. It is therefore an empirical-Bayes summary, not a draw from the full latent-effect posterior $\theta_i \mid y_i$. For new data, estimate-level random effects are sampled from their model distribution.

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)`.

Value

A `brma_samples` object containing posterior samples. When printed, displays a summary table via `BayesTools::ensemble_estimates_table`. The underlying samples matrix can be accessed directly (the object inherits from matrix) or via `summary()` to obtain the summary table. 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, 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.
`...`	additional arguments (currently ignored).

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, ...)
```

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. A character vector requests multiple base parameters.
parameter_mods	character. Moderator term to print. Use "intercept" for the adjusted effect in meta-regression models.
parameter_scale	character. Scale-regression term to print. Use "intercept" for the heterogeneity intercept in location-scale models.

Value

print_prior invisibly returns the selected prior distribution. 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_mods = "Preregistered")
}

## 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_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_specification` *Prior specification*

Description

Prior distributions can be supplied for all model parameters. When omitted, the fitting functions construct defaults from the effect-size measure, sample sizes, or informed-prior settings when available.

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.

Arguments

`prior_effect`	prior distribution for the effect size (μ) parameter (the intercept). If omitted, 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, a default prior is constructed. In single-model functions, explicit `NULL` or `FALSE` sets a spike at zero.
`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.
`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.
`prior_heterogeneity_allocation`	prior distribution for the fraction of heterogeneity allocated to the cluster-level component in multilevel models (ρ). If omitted or `NULL`, defaults to `Beta(1, 1)`.
`prior_baserate`	prior distribution for the estimate-specific midpoint base-rate probability in binomial GLMM models. If omitted or `NULL`, defaults to independent `Beta(1, 1)` priors.
`prior_lograte`	prior distribution for the estimate-specific midpoint log-rate in Poisson GLMM models. If omitted or `NULL`, a data-based unit-information normal prior is used independently for each estimate.
`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. For constructors with publication-bias prior distributions, `rescale_priors` does not rescale them except for the default PEESE prior's UISD adjustment. Defaults to 1.

`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"`, while model-averaging constructors use `"meandif"`.

`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. Omit to use the field-specific default, such as `"Cochrane"` for `prior_informed_field = "medicine"`; explicit `NULL` is invalid.

Details

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_priors.sc` 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. For publication-bias prior distributions, see `publication_bias_prior_specification`.

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

`publication_bias_prior_specification`, `prior`, `RoBMA.options`, `brma`

`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
)

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.
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))
)
```

`publication_bias_prior_specification`

Publication-bias prior specification

Description

Publication-bias prior distributions are specified separately from the meta-analytic prior distributions documented in `prior_specification`. They define selection-model weights, PET/PEESE regression coefficients, or the publication-bias model space in `RoBMA`.

Arguments

`prior_bias`	prior distribution for publication-bias adjustment. For `bselmodel`, this is usually a weightfunction prior created by `prior_weightfunction`. For `bPET`, use `prior_PET`. For `bPEESE`, use `prior_PEESE`. For `RoBMA`, this can be a single publication-bias prior distribution or a list of publication-bias prior distributions. In the single-model bias-adjustment constructors, omitted or `NULL` uses the corresponding default prior distribution.
`prior_bias_null`	prior distribution(s) for null publication-bias component(s) in `RoBMA`, usually `prior_none()`. A single prior distribution object creates one null component, a list creates multiple null components, and `NULL` or `FALSE` omits the null publication-bias component.
`model_type`	character string specifying predefined publication-bias model ensembles for `RoBMA`. One of `"PSMA"`, `"6w"`, `"2w"`, or `"PP"`.
`steps`	numeric vector of one-sided p-value cut points for the default `bselmodel` weightfunction prior. If `prior_bias` is supplied, the prior distribution carries its own p-value cut points.

Details

Single-model bias-adjustment priors:

`bselmodel`, `bPET`, and `bPEESE` fit one publication-bias adjustment at a time. The `prior_bias` argument must match the fitted bias-adjustment type.

Constructor	Prior constructor	Default prior distribution
`bselmodel()`	`prior_weightfunction`	one-sided cumulative weightfunction with `steps = 0.025`

bPET()	prior_PET	positive Cauchy centered at 0
bPEESE()	prior_PEESE	positive Cauchy centered at 0, with UISD-adjusted scale

The default PET prior distribution uses RoBMA.get_option("default_bias_PET.scale"). The default PEESE prior distribution uses RoBMA.get_option("default_bias_PEESE.scale") after rescaling to the analyzed effect-size measure. The default weightfunction prior distribution uses RoBMA.get_option("default_bias_weightfunction.alpha") for each cumulative-Dirichlet alpha parameter.

Model-averaged publication-bias priors:

RoBMA averages over publication-bias components. By default, prior_bias_null is prior_none() and model_type determines the alternative components:

"PSMA"	six weight functions, PET, and PEESE
"6w"	six weight functions
"2w"	two two-sided weight functions
"PP"	PET and PEESE

Custom prior_bias replaces the preset alternative components. If prior_bias is omitted, model_type still supplies the default alternative components even when prior_bias_null is customized or omitted. Setting prior_bias = NULL or prior_bias = FALSE omits all alternative bias-adjustment models. Setting prior_bias_null = NULL or prior_bias_null = FALSE omits the no-bias component.

Each publication-bias component has a prior model weight. User-specified components receive equal weights unless their prior objects set prior_weights. The "2w" and "6w" presets split the alternative bias mass equally across their weight functions, "PP" splits it equally across PET and PEESE, and "PSMA" assigns half of the alternative bias mass to the six weight functions combined and one quarter each to PET and PEESE.

Publication-bias prior distributions are not rescaled by rescale_priors. The exception is the default PEESE prior distribution, whose scale is adjusted to the effect-size measure via the unit-information standard deviation.

See Also

prior_weightfunction, prior_PET, prior_PEESE, prior_none, RoBMA_prior_specification, prior_specification

---

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",
  ...
)
```

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 all model types including GLMMs and selection models. 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".
envelope	logical indicating whether to add a pseudo confidence envelope to the plot. The envelope is created by simulating sets of standard normal quantiles, providing a reference for the expected variability under a correctly specified model. Defaults to TRUE.
conf_level	numeric value 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".
...	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.

The default residual type is "rstudent" (LOO-PIT), which differs from metafor::qqnorm (which defaults to "rstandard"). LOO-PIT residuals are preferred because they are available for all model types (including GLMMs and selection models) and 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 GLMM models, LOO-PIT residuals and the QQ plot are computed on the approximate effect-size scale used by loo; they are not exact count-scale PIT diagnostics.

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 of additional arguments passed to the transf function.
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 = $1/\sqrt{v_i + \hat{\tau}^2}$
- z-axis: standardized effect = $y_i/\sqrt{v_i + \hat{\tau}^2}$

Under the random-effects model, studies consistent with the pooled effect should fall within the sloped parallelogram confidence band around the pooled-effect line. 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; radial plots are not meaningful for meta-regression or location-scale models where the pooled effect varies across studies.

`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`.

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)
```

---

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), ...)
```

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.
`...`	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 observation-aligned `brma_samples` matrices, one column per estimate row:

`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.

Value

For 2-level models, a `brma_samples` object. For 3-level models, a named list of `brma_samples` objects (one per variance component).

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 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 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.
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 adjustments and sampling_bias = TRUE, the sampling interval incorporates the expected distortion from the selection process.
level	numeric; credible/prediction interval level in percent. 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.
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 based on inverse sampling variance.
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.
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.
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.
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 prediction bands showing expected range of 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 by argument allows displaying separate regression lines for different levels of a second moderator, useful for visualizing interactions.

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)
```

`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,
  ...
)
```

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 marginal standard error $\sqrt{v_i + \tau^2}$. Only available for normal outcome models without selection (weightfunction) bias adjustment.
- `"rstandard"`: Internally standardized residuals, dividing raw residuals by their standard errors 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 all model types. 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.

`...` 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. For models without moderators, this simplifies to the Pearson formula. Only available for normal outcome models without selection (weightfunction) bias adjustment.

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 predictive distribution used by LOO. 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 GLMM models (binomial or Poisson), observed effect sizes and their sampling variances are computed from the raw frequency data using the same formulas as `metafor::escalc` with the default zero-cell adjustment (adding 0.5 to all cells when any cell is zero). GLMM residuals and LOO-PIT values are therefore approximate effect-size-scale diagnostics, not exact PIT diagnostics for the raw count likelihood.

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,
  ...
)
```

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 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.
ni	an optional vector of sample sizes. Used for `measure = "GEN"` or when estimating `"UISD"`).
mods	an optional matrix, data.frame, or formula specifying location moderators (meta-regressors). Formula input is evaluated in `data`.
scale	an optional matrix, data.frame, or formula specifying scale predictors for location-scale models. Formula input is evaluated in `data`.
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.
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.

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.

prior_heterogeneity_allocation
: prior distribution(s) for the alternative cluster-level heterogeneity allocation component(s).

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 single prior applies to all terms; a named list can specify term-specific components.

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.

prior_heterogeneity_allocation_null
: prior distribution(s) for the null cluster-level heterogeneity allocation component(s).

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", while model-averaging constructors use "meandif".

`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. For constructors with publication-bias prior distributions, rescale_priors does not rescale them except for the default PEESE prior's UISD adjustment. 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. 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.

...
: 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().

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, 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_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
)

set_convergence_checks(
  max_Rhat = 1.05,
  min_ESS = 500,
  max_error = NULL,
  max_SD_error = NULL
)
```

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.

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. 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, and delegated check_indicators. set_convergence_checks returns a list with convergence thresholds max_Rhat, min_ESS, max_error, max_SD_error, and delegated check_indicators.

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)

`check_scaling` whether to check scaling of predictors (default `TRUE`)

`silent` whether to suppress output (default `FALSE`)

`autocompute.loo` whether to automatically compute LOO (default `FALSE`)

`autocompute.waic` whether to automatically compute WAIC (default `FALSE`)

`autocompute.marglik` whether to automatically compute marginal likelihood (default `FALSE`)

`cluster_likelihood.n_gamma` number of Gauss-Hermite nodes used for cluster-unit log-likelihoods (default `15`)

`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_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. 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.

`RoBMA_prior_specification`

Prior specification for model-averaging

Description

The `RoBMA` function extends the prior specification described in `prior_specification` by allowing mixture priors that define competing hypotheses for Bayesian model-averaging. This enables multimodel inference via the product space method (Carlin and Chib 1995; Lodewyckx et al. 2011), where a single MCMC chain explores a joint model space containing all competing models.

Arguments

`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.

`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.

`prior_heterogeneity_allocation`
prior distribution(s) for the alternative cluster-level heterogeneity allocation component(s).

`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 single prior applies to all terms; a named list can specify term-specific components.

`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.

`prior_heterogeneity_allocation_null`
prior distribution(s) for the null cluster-level heterogeneity allocation component(s).

`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.

`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.

Details

The product space method for model-averaging:

RoBMA implements Bayesian model-averaging using the product space method (Carlin and Chib 1995; Lodewyckx et al. 2011). Rather than fitting each model separately and combining results after the fitting is complete, this approach embeds all competing models within a single joint model space. A discrete model indicator variable selects which model component is "active" at each MCMC iteration, and the posterior probability of each model is estimated from the proportion of iterations spent in that model's subspace.

This is achieved by specifying **mixture priors** that combine:

- **Null hypothesis component(s)**: Typically point masses (spikes) representing the absence of an effect, heterogeneity, or publication bias
- **Alternative hypothesis component(s)**: Continuous distributions representing the presence of the parameter

The mixture structure enables direct computation of:

- **Inclusion Bayes factors**: Evidence for the presence vs. absence of each parameter (e.g., effect, heterogeneity, bias)
- **Model-averaged estimates**: Posterior estimates that account for model uncertainty by weighting across all models
- **Conditional estimates**: Posterior estimates given that a particular hypothesis is true

Default mixture prior structure:

By default, `RoBMA` creates the following mixture priors:

Parameter	Null component	Alternative component
Effect (μ)	Spike(0)	Normal(0, UISD-scaled)
Heterogeneity (τ)	Spike(0)	Normal+(0, UISD-scaled)
Publication bias	No bias (prior_none)	Weight functions + PET + PEESE
Allocation (ρ)	(none by default)	Beta(1, 1)

See `prior_specification` for details on how the UISD scaling is determined.

Specifying custom mixture priors:

Single prior vs. list of priors:
Each `prior_*` and `prior_*_null` argument accepts either a single prior or a list:

- **Single prior**: Creates one component in the mixture

- **List of priors**: Creates multiple components, enabling model-averaging across different prior specifications (e.g., different effect size scales)

When multiple priors are specified in a list, they are all treated as alternative (or null) hypothesis components for the main inclusion Bayes factor. The inclusion Bayes factor tests the combined evidence for all alternative components against all null components. The detailed summary output shows posterior probabilities and inclusion Bayes factors for each individual component separately, allowing finer-grained inference about which specific prior specification is most supported by the data.

Omitting hypothesis components:
For top-level mixture components, setting a prior argument to `NULL` or `FALSE` removes that hypothesis component:

- `prior_effect_null = NULL`: No null hypothesis for effect (assumes effect exists)
- `prior_effect = NULL`: No alternative hypothesis (assumes no effect)
- `prior_bias_null = NULL` or `FALSE`: No "no bias" model (assumes some bias mechanism)
- `prior_bias = NULL` or `FALSE`: No bias model (assumes no bias mechanism)

For moderator and scale regression terms, an omitted or whole-argument `NULL` `prior_mods` / `prior_scale` means "use defaults". To omit a component for a specific term, use a named list entry such as `prior_mods_null = list(x1 = NULL)`.

Parameter-specific customization for moderators:
For moderator priors (`prior_mods`, `prior_mods_null`), you can specify different priors for different predictors using named lists. A single prior object applies to all moderator terms:

```
RoBMA(...,
  mods = ~ x1 + x2,
  prior_mods_null = list(x1 = NULL)  # No null for x1, default null for x2
)
```

This allows testing whether specific moderators have effects while assuming others are always included or excluded.

Mixture priors for different parameter types:

Effect size (μ) and heterogeneity (τ):
Effect and heterogeneity priors combine spike-and-slab components:

```
# Default: spike(0) null + normal alternative
# Custom: multiple alternatives with different scales
RoBMA(...,
  prior_effect = list(
    prior("normal", list(mean = 0, sd = 0.5)),
    prior("normal", list(mean = 0, sd = 1.0))
  )
)
```

Publication bias:
Bias priors combine different publication bias adjustment mechanisms:

- No publication bias (null hypothesis)
- Weight functions: Selection models with different step functions
- PET/PEESE: Regression-based bias adjustment

The `model_type` argument provides convenient presets:

```
RoBMA(..., model_type = "PSMA")  # Full ensemble (default)
RoBMA(..., model_type = "PP")    # Only PET-PEESE models
```

See publication_bias_prior_specification for the bias-prior constructors and preset model spaces.

Heterogeneity allocation (ρ) for multilevel models:
When `cluster` is specified, ρ controls the cluster-level variance allocation. The decomposition is $\tau_{between} = \tau\sqrt{\rho}$ and $\tau_{within} = \tau\sqrt{1-\rho}$. By default, only an alternative (Beta(1,1)) is specified, but null hypotheses can be added:

```
RoBMA(...,
  cluster = study,
  prior_heterogeneity_allocation_null = prior("spike", list(location = 0.5))
)
```

Moderator and scale regression terms:
When `mods` is specified, the effect prior becomes the intercept prior, and each moderator receives a mixture prior. The **intercept** inherits the effect mixture structure, while **regression coefficients** get separate spike-and-slab mixtures.
Similarly, when `scale = ~ ...` is specified for location-scale models, the heterogeneity prior becomes the scale intercept prior, and each scale predictor receives a mixture prior following the same pattern as moderator terms. Note that the scale intercept (i.e., baseline heterogeneity) is always positive and the multiplicative scale coefficients require non-zero prior distribution on the intercept.

Model weights and prior odds:

Each component in a mixture prior has an associated prior weight (prior model probability). User-specified components receive equal weights unless their prior objects set `prior_weights`. Publication-bias presets use fixed weights: `"2w"` and `"6w"` split the alternative bias mass equally across their weight functions, `"PP"` splits it equally across PET and PEESE, and `"PSMA"` assigns half of the alternative bias mass to the six weight functions combined and one quarter each to PET and PEESE. Custom weights can be specified via the `prior_weights` argument in individual prior objects.

The prior odds between null and alternative hypotheses affect the Bayes factor interpretation. With equal prior weights on null and alternative, the posterior odds equal the Bayes factor.

References

Carlin BP, Chib S (1995). "Bayesian model choice via Markov chain Monte Carlo methods." *Journal of the Royal Statistical Society: Series B (Methodological)*, **57**(3), 473–484. doi:10.1111/j.25176161.1995.tb02042.x.

Lodewyckx T, Kim W, Lee MD, Tuerlinckx F, Kuppens P, Wagenmakers E (2011). "A tutorial on Bayes factor estimation with the product space method." *Journal of Mathematical Psychology*, **55**(5), 331–347. doi:10.1016/j.jmp.2011.06.001.

See Also

prior_specification for base prior specification options, publication_bias_prior_specification for publication-bias priors, RoBMA for the main model-averaging function, prior for creating prior distribution objects

Examples

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

  # Default mixture priors (spike-and-slab for effect and heterogeneity)
  fit1 <- RoBMA(
    yi      = yi,
    vi      = vi,
    data    = dat.lehmann2018,
    measure = "SMD",
    seed    = 1,
    silent  = TRUE
  )

  # Custom alternative priors (two effect size scales)
  fit2 <- RoBMA(
    yi           = yi,
    vi           = vi,
    data         = dat.lehmann2018,
    measure      = "SMD",
    prior_effect = list(
      prior("normal", list(mean = 0, sd = 0.35)),
      prior("normal", list(mean = 0, sd = 0.70))
    ),
    seed         = 1,
    silent       = TRUE
  )

  # No null hypothesis for effect (assume effect exists)
  fit3 <- RoBMA(
    yi                = yi,
    vi                = vi,
    data              = dat.lehmann2018,
    measure           = "SMD",
    prior_effect_null = NULL,
    seed              = 1,
    silent            = TRUE
  )

  # Only selection model bias adjustment (no PET-PEESE)
  fit4 <- RoBMA(
    yi         = yi,
    vi         = vi,
    data       = dat.lehmann2018,
    measure    = "SMD",
    model_type = "6w",
    seed       = 1,
    silent     = TRUE
  )

  # Meta-regression with different null specifications per moderator
```

```
  fit5 <- RoBMA(
    yi              = yi,
    vi              = vi,
    mods            = ~ Preregistered,
    data            = dat.lehmann2018,
    measure         = "SMD",
    prior_mods_null = list(Preregistered = NULL),
    seed            = 1,
    silent          = TRUE
  )
}

## End(Not run)
```

`rstandard.brma` *Internally Standardized Residuals for brma Objects*

Description

Computes internally standardized residuals from a fitted brma object using the hat matrix. 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", ...)
```

Arguments

`model`	a fitted brma object.
`unit`	output unit. Only `"estimate"` is implemented currently.
`conditioning_depth`	conditioning depth. Options are: • `"marginal"` (default): Residuals from fixed effects predictions ($observed - X\beta$). • `"cluster"`: Residuals from cluster-level predictions (observed - ($X\beta + gamma$)). Only available for multilevel (3-level) models. • `"estimate"`: Residuals from BLUPs, i.e., deviations of the observed effect sizes from the best linear unbiased predictions of the estimate-specific true effects (observed - theta).
`...`	additional arguments (currently ignored)

Details

This function returns a data frame with three columns matching the output of `metafor::rstandard`:

- `resid`: Raw residuals (observed - fitted values)
- `se`: Standard errors of the residuals
- `z`: Standardized residuals (resid / se)

Internally standardized residuals divide the observed residuals by their corresponding standard errors computed using the hat matrix. For a correctly specified model, these residuals should approximately follow a standard normal distribution.

This function is only available for normal outcome models without selection (weightfunction) bias adjustment. For other model types, use `rstudent.brma` which uses LOO-PIT.

Value

A data frame with columns:

- `resid`: Raw residuals
- `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. For GLMMs, they are computed on the approximate effect-size scale used by the LOO-PIT diagnostic; they are not exact PIT diagnostics for the raw count likelihood.

Unlike `rstandard.brma` (which uses the hat matrix), LOO-PIT residuals properly account for estimation uncertainty and leverage without requiring explicit hat matrix computation. This makes `rstudent.brma` suitable for all model types including selection models and GLMMs.

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)
```

---

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.

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 meta-regression coefficients. Defaults to FALSE. When set to TRUE, standardized meta-regression coefficients are returned for the intercept and continuous predictors. These coefficients correspond to the standardized scale on which prior distributions are specified by default (i.e., standardize_continuous_predictors = 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.
...	additional arguments
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. The printed form displays the non-empty tables.

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().
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".
...	additional arguments (currently ignored).

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 estimated missing studies.

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 - average power of significant studies

Soric FDR Expected proportion of false discoveries among significant results, computed from EDR following Soric (1989)

Missing N Estimated number of studies suppressed by publication bias, computed from the selection model weights

The footer reports the Observed Discovery Rate (ODR) with 95\ 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 the absolute heterogeneity (tau, tau^2) and relative measures of heterogeneity (I^2, H^2) 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), ...)
```

Arguments

`object`	a fitted brma object
`probs`	quantiles of the posterior distribution to be displayed. Defaults to `c(.025, .975)` for 95% credible intervals.
`...`	additional arguments (currently ignored)

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.

Value

A list of class `summary_heterogeneity.brma` containing:

- `estimates`: A `BayesTools_table` with heterogeneity statistics

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, BMA/BMA.norm, and BMA.glmm.
...	additional arguments
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

Only mixture-prior components are summarized; non-mixture components 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.

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() to obtain the summary table directly. 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,
  seed = 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.
seed	optional seed used before extending.
evaluate	unsupported; included for compatibility with update.

Details

Extending a fit adds posterior samples only. It does not rerun adaptation or burn-in. Prior, data, and model-structure updates are intentionally not supported by this method.

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, ...)
```

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`.
`...`	additional arguments (currently ignored)

Details

VIF is computed from the correlation matrix derived from the coefficient variance-covariance matrix $(X'WX)^{-1}$. For standard meta-regression models, $W = \text{diag}(w_i/(v_i + \hat{\tau}^2))$ with $\hat{\tau}$ equal to the posterior mean heterogeneity. For scale and multilevel models, the coefficient covariance is averaged across posterior heterogeneity draws, using observation-specific τ_i and block-structured multilevel covariance where applicable.

A VIF of 1 indicates no collinearity; values above 5 or 10 are commonly considered problematic.

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.

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`.

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,
  ...
)

## S3 method for class 'zplot_brma'
zplot(object, ...)
```

Arguments

`object`	a normal-outcome `brma` object, or a `zplot_brma` object.
`significance_level`	z-value threshold for significance. Defaults to `qnorm(0.975)` (two-sided alpha = 0.05).
`summary_max_samples`	maximum number of posterior samples used for the EDR and missing-study summaries stored in the generated zplot object. This is separate from the plot-density `max_samples` argument accepted by `plot.zplot_brma()` through `...`. Defaults to 10000. Use `Inf` to use all posterior samples.
`...`	arguments passed to `plot.zplot_brma()`.

Details

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, 9
Andrews2021, 10
Bem2011, 16
Havrankova2025, 69
Hoppen2025, 71
Johnides2025, 74
Kroupova2021, 75
Lui2015, 82
ManyLabs16, 83
Poulsen2006, 103
Wang2025, 171
Weingarten2018, 172
∗ **package**
RoBMA-package, 5

add_loo, *9*, *52*, *60*, *77*, *79*, *82*, *125*, *170*
add_loo *(*add_loo.brma*)*, 6
add_loo.brma, 6
add_marglik, *17*, *36*, *37*, *78*, *102*, *103*
add_marglik *(*add_marglik.brma*)*, 7
add_marglik.brma, 7
add_waic, *170*, *171*
add_waic *(*add_waic.brma*)*, 8
add_waic.brma, 8
Anderson2010, 9
Andrews2021, 10
as.matrix.brma_samples, 11
as_draws *(*as_draws.brma*)*, 11
as_draws.brma, 11, *14*
as_draws.brma_samples, 13, *13*
as_draws.default *(*as_draws.brma*)*, 11
as_draws_array *(*as_draws.brma*)*, 11
as_draws_array.brma_samples *(*as_draws.brma_samples*)*, 13
as_draws_df *(*as_draws.brma*)*, 11
as_draws_df.brma_samples *(*as_draws.brma_samples*)*, 13
as_draws_list *(*as_draws.brma*)*, 11
as_draws_list.brma_samples *(*as_draws.brma_samples*)*, 13
as_draws_matrix *(*as_draws.brma*)*, 11
as_draws_matrix.brma_samples *(*as_draws.brma_samples*)*, 13
as_draws_rvars *(*as_draws.brma*)*, 11
as_draws_rvars.brma_samples *(*as_draws.brma_samples*)*, 13
as_zplot *(*as_zplot.brma*)*, 14
as_zplot.brma, 14
as_zplot.brma(), *160*, *174*

bayes_factor.brma *(*bf.brma*)*, 17
Bem2011, 16
bf, *37*
bf.brma, *8*, 17, *37*, *78*, *103*
bias_prior_specification *(*publication_bias_prior_specification*)*, 123
blup, 18
blup(), *100*, *101*, *106*, *130*, *164*
blup.brma, 19, *131*, *164*, *165*
blup.brma(), *64*, *132*, *138*, *154*, *156*, *165*
BMA, *5*, 20
BMA(), *43*, *95*, *112*, *143*
BMA.glmm, 24
BMA.glmm(), *47*
bPEESE, *5*, 29, *123*
bPEESE(), *36*, *51*, *143*
bPET, *5*, 33, *123*
bPET(), *32*, *51*, *143*
bridge_sampler, *37*
bridge_sampler.brma, *8*, *17*, 36, *78*, *103*
brma, *5*, *13*, 37, *64*, *121*
brma(), *24*, *47*, *62*, *86*, *95*, *112*, *143*, *157*
brma.glmm, *5*, *28*, 44
brma.glmm(), *28*, *43*, *86*, *157*
bselmodel, *5*, 48, *123*
bselmodel(), *32*, *36*, *143*

check_loo (check_loo.brma), 52
check_loo.brma, 52
coef.brma, 53
contr.BayesTools, 53
contr.independent (contr.BayesTools), 53
contr.meandif (contr.BayesTools), 53
contr.orthonormal (contr.BayesTools), 53
cooks.distance.brma, 54, *56*, *61*, *72*
covratio (covratio.brma), 56
covratio.brma, 56, *72*

data_input, 57
dfbetas.brma, 58, *72*
dffits (dffits.brma), 60
dffits.brma, *55*, *56*, 60, *72*
difftime, *145*
draws, *13*, *14*

estimate_unit_information_sd, 61
estimate_unit_information_sd(), *41*, *120*

fitted, *63*
fitted.brma, 62
fitting_specification, 64
funnel (funnel.brma), 65
funnel.brma, 65
funnel.brma(), *32*, *36*, *51*, *127*, *129*, *135*

galbraith (radial.brma), 127

hatvalues.brma, *55*, *61*, 68, *72*
Havrankova2025, 69
hist.zplot_brma, 70
hist.zplot_brma(), *15*, *77*, *91*
Hoppen2025, 71

influence.brma, *55*, *56*, *61*, 72
interpret, 73

Johnides2025, 74

Kroupova2021, 75

lines.zplot_brma, 76
lines.zplot_brma(), *71*, *91*
logLik.brma, 77
logml.brma, *8*, *17*, *37*, 78, *103*
loo, *7*, *79*
loo (loo.brma), 79
loo.brma, *7*, *8*, *77*, 79, *80*, *81*, *171*
loo.brma(), *156*
loo_compare, *6*, *7*, *79–81*
loo_compare (loo_compare.brma), 80
loo_compare.brma, 80
loo_compare.loo, 81
loo_weights (loo_weights.brma), 81
loo_weights.brma, *60*, 81
Lui2015, 82

ManyLabs16, 83
marginal_means, 83
marginal_means.brma, 84

nobs.brma, 85

pareto_k_ids, *7*, *79*
plot(), *84*, *85*
plot.brma, 86
plot.brma(), *24*, *28*, *43*, *143*
plot.marginal_means.brma, 88
plot.zplot_brma, 89
plot.zplot_brma(), *15*, *71*, *77*, *160*, *173*, *174*
plot_diagnostic, 91
plot_diagnostic.brma (plot_diagnostic), 91
plot_diagnostic_autocorrelation (plot_diagnostic), 91
plot_diagnostic_density (plot_diagnostic), 91
plot_diagnostic_trace (plot_diagnostic), 91
plot_pet_peese, 93
plot_prior, 94
plot_prior(), *112*
plot_weightfunction, 96
pooled_effect, 98
pooled_effect(), *20*, *101*, *106*, *132*, *138*, *165*
pooled_effect.brma, 98
pooled_effect.brma(), *129*
pooled_heterogeneity, 100
pooled_heterogeneity(), *20*, *100*, *106*, *161*, *162*, *165*
pooled_heterogeneity.brma, 101
pooled_heterogeneity.brma(), *129*
post_prob, *37*
post_prob.brma, *8*, *17*, *37*, *78*, 102
Poulsen2006, 103
predict, *106*
predict.brma, *19*, *63*, *99*, *101*, 104, *131*, *165*

predict.brma(), *18*, *20*, *43*, *47*, *64*, *67*, *98*, *100*, *101*, *130*, *132*, *135*, *138*, *154*, *164*, *165*
print.brma *(*summary.brma*)*, 156
print.brma_samples, 107
print.interpret.brma *(*interpret*)*, 73
print.marginal_means.brma, 108
print.RoBMA_data, 108
print.summary.brma *(*summary.brma*)*, 156
print.summary.brma_samples, 109
print.summary.marginal_means.brma, 109
print.summary.zplot_brma, 110
print.summary_heterogeneity.brma, 110
print.summary_models.RoBMA *(*summary_models*)*, 163
print.vif.brma, 111
print_prior, 111
prior, 113, *121*, *151*
prior(), *95*, *112*
prior_factor, 114
prior_informed, 115
prior_none, 116, *123*, *124*
prior_PEESE, 116, *123*, *124*
prior_PET, 117, *123*, *124*
prior_specification, 118, *123*, *124*, *141*, *147–149*, *151*
prior_weightfunction, 121, *123*, *124*
publication_bias_prior_specification, *32*, *36*, *51*, *117*, *121*, *122*, 123, *141*, *143*, *148*, *151*

qqnorm.brma, *67*, 124

radial *(*radial.brma*)*, 127
radial.brma, 127
radial.brma(), *127*
ranef, 130
ranef.brma, 131
regplot *(*regplot.brma*)*, 132
regplot(), *84*, *85*, *168*, *169*
regplot.brma, 132
residuals.brma, *65*, *67*, 136
residuals.brma(), *64*, *67*, *127*, *154*, *156*
RoBMA, *5*, *24*, *28*, *123*, *124*, 139, *147*, *149*, *151*
RoBMA(), *24*, *28*, *32*, *36*, *43*, *51*, *87*, *94*, *95*, *97*, *112*, *145*
RoBMA-package, 5
RoBMA.get_option *(*RoBMA_options*)*, 146
RoBMA.options, *121*
RoBMA.options *(*RoBMA_options*)*, 146
RoBMA.package *(*RoBMA-package*)*, 5
RoBMA_control, 144
RoBMA_options, 146
RoBMA_package *(*RoBMA-package*)*, 5
RoBMA_prior_specification, *124*, 147
rstandard.brma, *65*, *67*, 153, *155*
rstandard.brma(), *67*, *127*, *138*, *156*
rstudent.brma, *65*, *67*, *154*, 155
rstudent.brma(), *67*, *127*, *138*, *154*

set_autofit_control, *64*
set_autofit_control *(*RoBMA_control*)*, 144
set_autofit_control(), *23*, *28*, *32*, *35*, *40*, *47*, *51*, *64*, *143*
set_convergence_checks, *64*
set_convergence_checks *(*RoBMA_control*)*, 144
set_convergence_checks(), *23*, *28*, *32*, *35*, *40*, *47*, *51*, *64*, *143*
summary(), *84*, *85*
summary.brma, 156
summary.brma(), *24*, *28*, *32*, *36*, *43*, *47*, *51*, *53*, *84*, *85*, *92*, *143*, *161*, *162*, *168*, *169*
summary.brma_samples, 158
summary.marginal_means.brma, 158
summary.zplot_brma, 159
summary.zplot_brma(), *15*, *91*, *174*
summary_heterogeneity, 160
summary_heterogeneity.brma, 161
summary_models, 163
supsmu, *126*

true_effects, 164
true_effects(), *20*
true_effects.brma, 164

update, *167*
update.brma, 166

vif, 168
vif.brma, 168

waic, *9*, *171*
waic *(*waic.brma*)*, 170
waic.brma, *9*, 170
Wang2025, 171
Weingarten2018, 172

wf_cumulative *(*prior_weightfunction*)*, 121
wf_fixed *(*prior_weightfunction*)*, 121
wf_independent *(*prior_weightfunction*)*, 121

zplot *(*zplot.brma*)*, 173
zplot.brma, 173