Package ‘RoBMA’

February 4, 2025

Title Robust Bayesian Meta-Analyses

Version 3.4.0

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

Description A framework for estimating ensembles of meta-analytic and meta-regression models (assuming either presence or absence of the effect, heterogeneity, publication bias, and moderators). The RoBMA framework uses Bayesian model-averaging to combine the competing meta-analytic models into a model ensemble, weights the posterior parameter distributions based on posterior model probabilities and uses Bayes factors to test for the presence or absence of the individual components (e.g., effect vs. no effect; Bartoš et al., 2022, <doi:10.1002/jrsm.1594>; Maier, Bartoš & Wagenmakers, 2022, <doi:10.1037/met0000405>). Users can define a wide range of prior distributions for + the effect size, heterogeneity, publication bias (including selection models and PET-PEESE), and moderator components. The package provides convenient functions for summary, visualizations, and fit diagnostics.

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

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

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

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

NeedsCompilation yes

Depends R (>= 4.0.0)

Imports BayesTools (>= 0.2.18), runjags, rjags, stats, graphics, mvtnorm, scales, Rdpack, rlang, coda, ggplot2

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

LinkingTo mvtnorm

RdMacros Rdpack

VignetteBuilder knitr

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

Repository CRAN

Date/Publication 2025-02-04 09:20:02 UTC

Contents

`RoBMA-package` *RoBMA: Robust Bayesian meta-analysis*

Description

RoBMA: Bayesian model-averaged meta-analysis with adjustments for publication bias and ability to specify informed prior distributions and draw inference with inclusion Bayes factors.

User guide

See Bartoš et al. (2023), Maier et al. (2023), and Bartoš et al. (2022) for details regarding the RoBMA methodology.

More details regarding customization of the model ensembles are provided in the **Reproducing BMA**, **BMA in Medicine**, and **Fitting Custom Meta-Analytic Ensembles** vignettes. Please, use the "Issues" section in the GitHub repository to ask any further questions.

Author(s)

František Bartoš <`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, 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`

`Anderson2010` *27 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 27 experimental studies focusing on the effect of violent video games on aggressive behavior. The full original data can found at https://github.com/Joe-Hilgard/Anderson-meta.

Usage

```
Anderson2010
```

Format

A data.frame with 3 columns and 23 observations.

Value

a data.frame.

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` *36 estimates of the effect of household chaos on child executive functions with the mean age and assessment type covariates from a meta-analysis by Andrews et al. (2021)*

Description

The data set contains correlation coefficients `r`, standard errors `se`, executive functioning assessment type `measure`, and the mean age of the children in each study `age`. The original data set assessed the effect of household chaos on child executive functions (Andrews et al. 2021) which was used as an example in Bartoš et al. (2022).

Usage

```
Andrews2021
```

Format

A data.frame with 4 columns and 36 observations.

Value

a data.frame.

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, 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.

`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.

Value

a data.frame.

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.

`BiBMA` *Estimate a Bayesian Model-Averaged Meta-Analysis of Binomial Data*

Description

`BiBMA` estimate a binomial-normal Bayesian model-averaged meta-analysis. The interface allows a complete customization of the ensemble with different prior (or list of prior) distributions for each component.

Usage

```
BiBMA(
  x1,
  x2,
  n1,
  n2,
  study_names = NULL,
  study_ids = NULL,
  rescale_priors = 1,
 priors_effect = set_default_binomial_priors("effect", rescale = rescale_priors),
  priors_heterogeneity = set_default_binomial_priors("heterogeneity", rescale =
    rescale_priors),
  priors_effect_null = set_default_binomial_priors("effect", null = TRUE),
 priors_heterogeneity_null = set_default_binomial_priors("heterogeneity", null = TRUE),
  priors_baseline = set_default_binomial_priors("baseline"),
  priors_baseline_null = set_default_binomial_priors("baseline", null = TRUE),
  chains = 3,
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  thin = 1,
```

```
    parallel = FALSE,
    autofit = TRUE,
    autofit_control = set_autofit_control(),
    convergence_checks = set_convergence_checks(),
    algorithm = "bridge",
    save = "all",
    seed = NULL,
    silent = TRUE,
    ...
)
```

Arguments

`x1`	a vector with the number of successes in the first group
`x2`	a vector with the number of successes in the second group
`n1`	a vector with the number of observations in the first group
`n2`	a vector with the number of observations in the second group
`study_names`	an optional argument with the names of the studies
`study_ids`	an optional argument specifying dependency between the studies (for using a multilevel model). Defaults to `NULL` for studies being independent.
`rescale_priors`	a re-scaling factor for the prior distributions. The re-scaling factor allows to adjust the width of all default priors simultaneously. Defaults to `1`.
`priors_effect`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the alternative hypothesis. Defaults to `prior(distribution = "student", parameters = list(location = 0, scale = 0.58, df = 4))`, based on logOR meta-analytic estimates from the Cochrane Database of Systematic Reviews (Bartoš et al. 2023).
`priors_heterogeneity`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the alternative hypothesis. Defaults to `prior(distribution = "invgamma", parameters = list(shape = 1.77, scale = 0.55))` that is based on heterogeneities of logOR estimates from the Cochrane Database of Systematic Reviews (Bartoš et al. 2023).
`priors_effect_null`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero, `prior(distribution = "point", parameters = list(location = 0))`.
`priors_heterogeneity_null`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero (a fixed effect meta-analytic models), `prior(distribution = "point", parameters = list(location = 0))`.
`priors_baseline`	prior distributions for the alternative hypothesis about intercepts (`pi`) of each study. Defaults to `NULL`.

`priors_baseline_null`	prior distributions for the null hypothesis about intercepts (`pi`) for each study. Defaults to an independent uniform prior distribution for each intercept `prior("beta", parameters = list(alpha = 1, beta = 1), contrast = "independent")`.
`chains`	a number of chains of the MCMC algorithm.
`sample`	a number of sampling iterations of the MCMC algorithm. Defaults to `5000`.
`burnin`	a number of burnin iterations of the MCMC algorithm. Defaults to `2000`.
`adapt`	a number of adaptation iterations of the MCMC algorithm. Defaults to `500`.
`thin`	a thinning of the chains of the MCMC algorithm. Defaults to `1`.
`parallel`	whether the individual models should be fitted in parallel. Defaults to `FALSE`. The implementation is not completely stable and might cause a connection error.
`autofit`	whether the model should be fitted until the convergence criteria (specified in `autofit_control`) are satisfied. Defaults to `TRUE`.
`autofit_control`	allows to pass autofit control settings with the `set_autofit_control()` function. See `?set_autofit_control` for options and default settings.
`convergence_checks`	automatic convergence checks to assess the fitted models, passed with `set_convergence_checks()` function. See `?set_convergence_checks` for options and default settings.
`algorithm`	a string specifying the algorithm used for the model averaging. Defaults to `"bridge"` which results in estimating individual models using JAGS and computing the marginal likelihood using bridge sampling. An alternative is `"ss"` which uses spike and slab like parameterization to approximate the Bayesian model averaging with a single model. Note that significantly more `sample`, `burnin`, and `adapt` iterations are needed for the `"ss"` algorithm.
`save`	whether all models posterior distributions should be kept after obtaining a model-averaged result. Defaults to `"all"` which does not remove anything. Set to `"min"` to significantly reduce the size of final object, however, some model diagnostics and further manipulation with the object will not be possible.
`seed`	a seed to be set before model fitting, marginal likelihood computation, and posterior mixing for reproducibility of results. Defaults to `NULL` - no seed is set.
`silent`	whether all print messages regarding the fitting process should be suppressed. Defaults to `TRUE`. Note that `parallel = TRUE` also suppresses all messages.
`...`	additional arguments.

Details

The `BiBMA()` function estimates the binomial-normal Bayesian model-averaged meta-analysis described in Bartoš et al. (2023). See `vignette("MedicineBiBMA", package = "RoBMA")` vignette for a reproduction of the Oduwole et al. (2018) example. Also `RoBMA()` for additional details.

Generic `summary.RoBMA()`, `print.RoBMA()`, and `plot.RoBMA()` functions are provided to facilitate manipulation with the ensemble. A visual check of the individual model diagnostics can be obtained using the `diagnostics()` function. The fitted model can be further updated or modified by `update.RoBMA()` function.

Value

`NoBMA` returns an object of class 'RoBMA'.

References

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.

Oduwole O, Udoh EE, Oyo-Ita A, Meremikwu MM (2018). "Honey for acute cough in children." *Cochrane Database of Systematic Reviews*. doi:10.1002/14651858.CD007094.pub5.

See Also

`RoBMA()`, `summary.RoBMA()`, `update.RoBMA()`, `check_setup()`

Examples

```
## Not run:
# using the example data from Oduwole (2018) and reproducing the example from
# Bartos et al. (2023) with domain specific informed prior distributions

fit <- BiBMA(
  x1          = c(5, 2),
  x2          = c(0, 0),
  n1          = c(35, 40),
  n2          = c(39, 40),
  priors_effect        = prior_informed(
      "Acute Respiratory Infections",
      type = "logOR", parameter = "effect"),
  priors_heterogeneity = prior_informed(
      "Acute Respiratory Infections",
      type = "logOR", parameter = "heterogeneity")
 )

 summary(fit)

 # produce summary on OR scale
 summary(fit, output_scale = "OR")


## End(Not run)
```

`BiBMA.reg` *Estimate a Robust Bayesian Meta-Analysis Meta-Regression*

Description

`RoBMA` is used to estimate a robust Bayesian meta-regression. The interface allows a complete customization of the ensemble with different prior (or list of prior) distributions for each component.

Usage

```
BiBMA.reg(
  formula,
  data,
  test_predictors = TRUE,
  study_names = NULL,
  study_ids = NULL,
  standardize_predictors = TRUE,
  priors = NULL,
  rescale_priors = 1,
  priors_effect = set_default_binomial_priors("effect", rescale = rescale_priors),
  priors_heterogeneity = set_default_binomial_priors("heterogeneity", rescale =
    rescale_priors),
  priors_effect_null = set_default_binomial_priors("effect", null = TRUE),
  priors_heterogeneity_null = set_default_binomial_priors("heterogeneity", null = TRUE),
  prior_covariates = set_default_binomial_priors("covariates", rescale = rescale_priors),
  prior_covariates_null = set_default_binomial_priors("covariates", null = TRUE),
  prior_factors = set_default_binomial_priors("factors", rescale = rescale_priors),
  prior_factors_null = set_default_binomial_priors("factors", null = TRUE),
  priors_baseline = set_default_binomial_priors("baseline"),
  priors_baseline_null = set_default_binomial_priors("baseline", null = TRUE),
  algorithm = "bridge",
  chains = 3,
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  thin = 1,
  parallel = FALSE,
  autofit = TRUE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  save = "all",
  seed = NULL,
  silent = TRUE,
  ...
)
```

Arguments

`formula`	a formula for the meta-regression model
`data`	a data.frame containing the data for the meta-regression. Note that the column names have to correspond to the effect sizes (`d`, `logOR`, `OR`, `r`, `z`), a measure of sampling variability (`se`, `v`, `n`, `lCI`, `uCI`, `t`), and the predictors. See

	`combine_data()` for a complete list of reserved names and additional information about specifying input data.
`test_predictors`	vector of predictor names to test for the presence of moderation (i.e., assigned both the null and alternative prior distributions). Defaults to `TRUE`, all predictors are tested using the default prior distributions (i.e., `prior_covariates`, `prior_covariates_null`, `prior_factors`, and `prior_factors_null`). To only estimate and adjust for the effect of predictors use `FALSE`. If `priors` is specified, any settings in `test_predictors` is overridden.
`study_names`	an optional argument with the names of the studies
`study_ids`	an optional argument specifying dependency between the studies (for using a multilevel model). Defaults to `NULL` for studies being independent.
`standardize_predictors`	whether continuous predictors should be standardized prior to estimating the model. Defaults to `TRUE`. Continuous predictor standardization is important for applying the default prior distributions for continuous predictors. Note that the resulting output corresponds to standardized meta-regression coefficients.
`priors`	named list of prior distributions for each predictor (with names corresponding to the predictors). It allows users to specify both the null and alternative hypothesis prior distributions for each predictor by assigning the corresponding element of the named list with another named list (with `"null"` and `"alt"`). If only one prior is specified for a given parameter, it is assumed to correspond to the alternative hypotheses and the default null hypothesis is specified (i.e., `prior_covariates_null` or `prior_factors_null`). If a named list with only one named prior distribution is provided (either `"null"` or `"alt"`), only this prior distribution is used and no default distribution is filled in. Parameters without specified prior distributions are assumed to be only adjusted for using the default alternative hypothesis prior distributions (i.e., `prior_covariates` or `prior_factors`). If `priors` is specified, `test_predictors` is ignored.
`rescale_priors`	a re-scaling factor for the prior distributions. The re-scaling factor allows to adjust the width of all default priors simultaneously. Defaults to `1`.
`priors_effect`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the alternative hypothesis. Defaults to `prior(distribution = "student", parameters = list(location = 0, scale = 0.58, df = 4))`, based on logOR meta-analytic estimates from the Cochrane Database of Systematic Reviews (Bartoš et al. 2023).
`priors_heterogeneity`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the alternative hypothesis. Defaults to `prior(distribution = "invgamma", parameters = list(shape = 1.77, scale = 0.55))` that is based on heterogeneities of logOR estimates from the Cochrane Database of Systematic Reviews (Bartoš et al. 2023).
`priors_effect_null`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero, `prior(distribution = "point", parameters = list(location = 0))`.

`priors_heterogeneity_null`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero (a fixed effect meta-analytic models), `prior(distribution = "point", parameters = list(location = 0))`.
`prior_covariates`	a prior distributions for the regression parameter of continuous covariates on the effect size under the alternative hypothesis (unless set explicitly in `priors`). Defaults to a relatively wide normal distribution `prior(distribution = "normal", parameters = list(mean = 0, sd = 0.25))`.
`prior_covariates_null`	a prior distributions for the regression parameter of continuous covariates on the effect size under the null hypothesis (unless set explicitly in `priors`). Defaults to a no effect `prior("spike", parameters = list(location = 0))`.
`prior_factors`	a prior distributions for the regression parameter of categorical covariates on the effect size under the alternative hypothesis (unless set explicitly in `priors`). Defaults to a relatively wide multivariate normal distribution specifying differences from the mean contrasts `prior_factor("mnormal", parameters = list(mean = 0, sd = 0.25), contrast = "meandif")`.
`prior_factors_null`	a prior distributions for the regression parameter of categorical covariates on the effect size under the null hypothesis (unless set explicitly in `priors`). Defaults to a no effect `prior("spike", parameters = list(location = 0))`.
`priors_baseline`	prior distributions for the alternative hypothesis about intercepts (`pi`) of each study. Defaults to `NULL`.
`priors_baseline_null`	prior distributions for the null hypothesis about intercepts (`pi`) for each study. Defaults to an independent uniform prior distribution for each intercept `prior("beta", parameters = list(alpha = 1, beta = 1), contrast = "independent")`.
`algorithm`	a string specifying the algorithm used for the model averaging. Defaults to `"bridge"` which results in estimating individual models using JAGS and computing the marginal likelihood using bridge sampling. An alternative is `"ss"` which uses spike and slab like parameterization to approximate the Bayesian model averaging with a single model. Note that significantly more `sample`, `burnin`, and `adapt` iterations are needed for the `"ss"` algorithm.
`chains`	a number of chains of the MCMC algorithm.
`sample`	a number of sampling iterations of the MCMC algorithm. Defaults to `5000`.
`burnin`	a number of burnin iterations of the MCMC algorithm. Defaults to `2000`.
`adapt`	a number of adaptation iterations of the MCMC algorithm. Defaults to `500`.
`thin`	a thinning of the chains of the MCMC algorithm. Defaults to `1`.
`parallel`	whether the individual models should be fitted in parallel. Defaults to `FALSE`. The implementation is not completely stable and might cause a connection error.
`autofit`	whether the model should be fitted until the convergence criteria (specified in `autofit_control`) are satisfied. Defaults to `TRUE`.

`autofit_control`	allows to pass autofit control settings with the `set_autofit_control()` function. See `?set_autofit_control` for options and default settings.
`convergence_checks`	automatic convergence checks to assess the fitted models, passed with `set_convergence_checks()` function. See `?set_convergence_checks` for options and default settings.
`save`	whether all models posterior distributions should be kept after obtaining a model-averaged result. Defaults to `"all"` which does not remove anything. Set to `"min"` to significantly reduce the size of final object, however, some model diagnostics and further manipulation with the object will not be possible.
`seed`	a seed to be set before model fitting, marginal likelihood computation, and posterior mixing for reproducibility of results. Defaults to `NULL` - no seed is set.
`silent`	whether all print messages regarding the fitting process should be suppressed. Defaults to `TRUE`. Note that `parallel = TRUE` also suppresses all messages.
`...`	additional arguments.

Details

`BiBMA.reg()` function estimates the Bayesian model-averaged binomial meta-regression. See `vignette("/MetaRegression` `package = "RoBMA")` vignette describes how to use the similar `RoBMA.reg()` function to fit Bayesian meta-regression ensembles. See Bartoš et al. (2023) for more details about the methodology and `BiBMA()` for more details about the function options. By default, the function standardizes continuous predictors. As such, the output should be interpreted as standardized meta-regression coefficients.

Generic `summary.RoBMA()`, `print.RoBMA()`, and `plot.RoBMA()` functions are provided to facilitate manipulation with the ensemble. A visual check of the individual model diagnostics can be obtained using the `diagnostics()` function. The fitted model can be further updated or modified by `update.RoBMA()` function. Estimated marginal means can be computed by `marginal_summary()` function and visualized by the `marginal_plot()` function.

Value

`RoBMA.reg` returns an object of class 'RoBMA.reg'.

References

Bartoš F, Maier M, Stanley TD, Wagenmakers E (2023). "Robust Bayesian meta-regression: Model-averaged moderation analysis in the presence of publication bias." doi:10.31234/osf.io/98xb5, Preprint available at https://doi.org/10.31234/osf.io/98xb5.

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.

See Also

`BiBMA()` `summary.RoBMA()`, `update.BiBMA()`, `check_setup.reg()`

`check_RoBMA` *Check fitted RoBMA object for errors and warnings*

Description

Checks fitted RoBMA object for warnings and errors and prints them to the console.

Usage

```
check_RoBMA(fit)

check_RoBMA_convergence(fit)
```

Arguments

`fit`	a fitted RoBMA object.

Value

`check_RoBMA` returns a vector of error and warning messages. `check_RoBMA_convergence` returns a logical vector indicating whether the models have converged.

`check_setup` *Prints summary of* `"RoBMA"` *ensemble implied by the specified priors*

Description

`check_setup` prints summary of `"RoBMA"` ensemble implied by the specified prior distributions. It is useful for checking the ensemble configuration prior to fitting all of the models.

Usage

```
check_setup(
   model_type = NULL,
  priors_effect = prior(distribution = "normal", parameters = list(mean = 0, sd = 1)),
  priors_heterogeneity = prior(distribution = "invgamma", parameters = list(shape = 1,
      scale = 0.15)),
  priors_bias = list(prior_weightfunction(distribution = "two.sided", parameters =
      list(alpha = c(1, 1), steps = c(0.05)), prior_weights = 1/12),
    prior_weightfunction(distribution = "two.sided", parameters = list(alpha = c(1, 1,
    1), steps = c(0.05, 0.1)), prior_weights = 1/12), prior_weightfunction(distribution =
    "one.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_weights =
    1/12), prior_weightfunction(distribution = "one.sided", parameters = list(alpha =
      c(1, 1, 1), steps = c(0.025, 0.05)), prior_weights = 1/12),

    prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1,
```

```
    1), steps = c(0.05, 0.5)), prior_weights = 1/12), prior_weightfunction(distribution =
    "one.sided", parameters = list(alpha = c(1, 1, 1, 1), steps = c(0.025, 0.05, 0.5)),
    prior_weights = 1/12), prior_PET(distribution = "Cauchy", parameters = list(0, 1),
    truncation = list(0, Inf), prior_weights = 1/4), prior_PEESE(distribution = "Cauchy",
     parameters = list(0, 5), truncation = list(0, Inf), prior_weights = 1/4)),
  priors_effect_null = prior(distribution = "point", parameters = list(location = 0)),
  priors_heterogeneity_null = prior(distribution = "point", parameters = list(location =
     0)),
  priors_bias_null = prior_none(),
  priors_hierarchical = prior("beta", parameters = list(alpha = 1, beta = 1)),
  priors_hierarchical_null = NULL,
  models = FALSE,
  silent = FALSE
)

check_setup.RoBMA(
  model_type = NULL,
  priors_effect = prior(distribution = "normal", parameters = list(mean = 0, sd = 1)),
  priors_heterogeneity = prior(distribution = "invgamma", parameters = list(shape = 1,
     scale = 0.15)),
  priors_bias = list(prior_weightfunction(distribution = "two.sided", parameters =
     list(alpha = c(1, 1), steps = c(0.05)), prior_weights = 1/12),
    prior_weightfunction(distribution = "two.sided", parameters = list(alpha = c(1, 1,
    1), steps = c(0.05, 0.1)), prior_weights = 1/12), prior_weightfunction(distribution =
    "one.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_weights =
    1/12), prior_weightfunction(distribution = "one.sided", parameters = list(alpha =
     c(1, 1, 1), steps = c(0.025, 0.05)), prior_weights = 1/12),

    prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1,
    1), steps = c(0.05, 0.5)), prior_weights = 1/12), prior_weightfunction(distribution =
    "one.sided", parameters = list(alpha = c(1, 1, 1, 1), steps = c(0.025, 0.05, 0.5)),
    prior_weights = 1/12), prior_PET(distribution = "Cauchy", parameters = list(0, 1),
    truncation = list(0, Inf), prior_weights = 1/4), prior_PEESE(distribution = "Cauchy",
     parameters = list(0, 5), truncation = list(0, Inf), prior_weights = 1/4)),
  priors_effect_null = prior(distribution = "point", parameters = list(location = 0)),
  priors_heterogeneity_null = prior(distribution = "point", parameters = list(location =
     0)),
  priors_bias_null = prior_none(),
  priors_hierarchical = prior("beta", parameters = list(alpha = 1, beta = 1)),
  priors_hierarchical_null = NULL,
  models = FALSE,
  silent = FALSE
)
```

Arguments

`model_type`	string specifying the RoBMA ensemble. Defaults to `NULL`. The other options are `"PSMA"`, `"PP"`, and `"2w"` which override settings passed to the `priors_effect`, `priors_heterogeneity`, `priors_effect`, `priors_effect_null`, `priors_heterogeneity_null`,

	`priors_bias_null`, and `priors_effect`. See details for more information about the different model types.
`priors_effect`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the alternative hypothesis. Defaults to a standard normal distribution `prior(distribution = "normal", parameters = list(mean = 0, sd = 1))`.
`priors_heterogeneity`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the alternative hypothesis. Defaults to `prior(distribution = "invgamma", parameters = list(shape = 1, scale = .15))` that is based on heterogeneities estimates from psychology (van Erp et al. 2017).
`priors_bias`	list of prior distributions for the publication bias adjustment component that will be treated as belonging to the alternative hypothesis. Defaults to `list(prior_weightfunction(distribution = "two.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_weights = 1/12),prior_weightfunction(distribution = "two.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.05, 0.10)), prior_weights = 1/12),prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_weights = 1/12),prior_weightfu = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.025, 0.05)), prior_weights = 1/12),prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.05, 0.5)), prior_weights = 1/12),prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1, 1), steps = c(0.025, 0.05, 0.5)), prior_weights = 1/12),prior_PET(distribution = "Cauchy", parameters = list(0,1), truncation = list(0, Inf), prior_weights = 1/4),prior_PEESE(distribution = "Cauchy", parameters = list(0,5), truncation = list(0, Inf), prior_weights = 1/4))`, corresponding to the RoBMA-PSMA model introduce by Bartoš et al. (2023).
`priors_effect_null`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero, `prior(distribution = "point", parameters = list(location = 0))`.
`priors_heterogeneity_null`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero (a fixed effect meta-analytic models), `prior(distribution = "point", parameters = list(location = 0))`.
`priors_bias_null`	list of prior weight functions for the `omega` parameter that will be treated as belonging to the null hypothesis. Defaults no publication bias adjustment, `prior_none()`.
`priors_hierarchical`	list of prior distributions for the correlation of random effects (`rho`) parameter that will be treated as belonging to the alternative hypothesis. This setting allows users to fit a hierarchical (three-level) meta-analysis when `study_ids` are supplied. Note that this is an experimental feature and see News for more details. Defaults to a beta distribution `prior(distribution = "beta", parameters = list(alpha = 1, beta = 1))`.
`priors_hierarchical_null`	list of prior distributions for the correlation of random effects (`rho`) parameter that will be treated as belonging to the null hypothesis. Defaults to `NULL`.

`models`	should the models' details be printed.
`silent`	do not print the results.

Value

`check_setup` invisibly returns list of summary tables.

See Also

`check_setup.reg()` `RoBMA()`

`check_setup.BiBMA` *Prints summary of* `"BiBMA.reg"` *ensemble implied by the specified priors and formula*

Description

`check_setup` prints summary of `"RoBMA.reg"` ensemble implied by the specified prior distributions. It is useful for checking the ensemble configuration prior to fitting all of the models.

Usage

```
check_setup.BiBMA(
  priors_effect = prior(distribution = "student", parameters = list(location = 0, scale =
    0.58, df = 4)),
  priors_heterogeneity = prior(distribution = "invgamma", parameters = list(shape = 1.77,
    scale = 0.55)),
  priors_effect_null = prior(distribution = "point", parameters = list(location = 0)),
  priors_heterogeneity_null = prior(distribution = "point", parameters = list(location =
    0)),
   priors_baseline = NULL,
  priors_baseline_null = prior_factor("beta", parameters = list(alpha = 1, beta = 1),
    contrast = "independent"),
   models = FALSE,
   silent = FALSE,
   ...
)
```

Arguments

`priors_effect`	list of prior distributions for the effect size (mu) parameter that will be treated as belonging to the alternative hypothesis. Defaults to `prior(distribution = "student", parameters = list(location = 0, scale = 0.58, df = 4))`, based on logOR meta-analytic estimates from the Cochrane Database of Systematic Reviews (Bartoš et al. 2023).

`priors_heterogeneity`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the alternative hypothesis. Defaults to `prior(distribution = "invgamma", parameters = list(shape = 1.77, scale = 0.55))` that is based on heterogeneities of logOR estimates from the Cochrane Database of Systematic Reviews (Bartoš et al. 2023).
`priors_effect_null`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero, `prior(distribution = "point", parameters = list(location = 0))`.
`priors_heterogeneity_null`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero (a fixed effect meta-analytic models), `prior(distribution = "point", parameters = list(location = 0))`.
`priors_baseline`	prior distributions for the alternative hypothesis about intercepts (`pi`) of each study. Defaults to `NULL`.
`priors_baseline_null`	prior distributions for the null hypothesis about intercepts (`pi`) for each study. Defaults to an independent uniform prior distribution for each intercept `prior("beta", parameters = list(alpha = 1, beta = 1), contrast = "independent")`.
`models`	should the models' details be printed.
`silent`	whether all print messages regarding the fitting process should be suppressed. Defaults to `TRUE`. Note that `parallel = TRUE` also suppresses all messages.
`...`	additional arguments.

Value

`check_setup.reg` invisibly returns list of summary tables.

See Also

`check_setup()` `BiBMA()`

`check_setup.reg` *Prints summary of* `"RoBMA.reg"` *ensemble implied by the specified priors and formula*

Description

`check_setup` prints summary of `"RoBMA.reg"` ensemble implied by the specified prior distributions. It is useful for checking the ensemble configuration prior to fitting all of the models.

`check_setup` prints summary of `"RoBMA.reg"` ensemble implied by the specified prior distributions. It is useful for checking the ensemble configuration prior to fitting all of the models.

Usage

```
check_setup.reg(
  formula,
  data,
  test_predictors = TRUE,
  study_names = NULL,
  study_ids = NULL,
  transformation = if (any(colnames(data) != "y")) "fishers_z" else "none",
  prior_scale = if (any(colnames(data) != "y")) "cohens_d" else "none",
  standardize_predictors = TRUE,
  effect_direction = "positive",
  priors = NULL,
  model_type = NULL,
  priors_effect = prior(distribution = "normal", parameters = list(mean = 0, sd = 1)),
  priors_heterogeneity = prior(distribution = "invgamma", parameters = list(shape = 1,
    scale = 0.15)),
  priors_bias = list(prior_weightfunction(distribution = "two.sided", parameters =
    list(alpha = c(1, 1), steps = c(0.05)), prior_weights = 1/12),
   prior_weightfunction(distribution = "two.sided", parameters = list(alpha = c(1, 1,
   1), steps = c(0.05, 0.1)), prior_weights = 1/12), prior_weightfunction(distribution =
   "one.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_weights =
   1/12), prior_weightfunction(distribution = "one.sided", parameters = list(alpha =
    c(1, 1, 1), steps = c(0.025, 0.05)), prior_weights = 1/12),

   prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1,
   1), steps = c(0.05, 0.5)), prior_weights = 1/12), prior_weightfunction(distribution =
   "one.sided", parameters = list(alpha = c(1, 1, 1, 1), steps = c(0.025, 0.05, 0.5)),
   prior_weights = 1/12), prior_PET(distribution = "Cauchy", parameters = list(0, 1),
   truncation = list(0, Inf), prior_weights = 1/4), prior_PEESE(distribution = "Cauchy",
    parameters = list(0, 5), truncation = list(0, Inf), prior_weights = 1/4)),
  priors_effect_null = prior(distribution = "point", parameters = list(location = 0)),
  priors_heterogeneity_null = prior(distribution = "point", parameters = list(location =
    0)),
  priors_bias_null = prior_none(),
  priors_hierarchical = prior("beta", parameters = list(alpha = 1, beta = 1)),
  priors_hierarchical_null = NULL,
  prior_covariates = prior("normal", parameters = list(mean = 0, sd = 0.25)),
  prior_covariates_null = prior("spike", parameters = list(location = 0)),
  prior_factors = prior_factor("mnormal", parameters = list(mean = 0, sd = 0.25),
    contrast = "meandif"),
  prior_factors_null = prior("spike", parameters = list(location = 0)),
  models = FALSE,
  silent = FALSE,
  ...
)

check_setup.RoBMA.reg(
  formula,
```

```
    data,
    test_predictors = TRUE,
    study_names = NULL,
    study_ids = NULL,
    transformation = if (any(colnames(data) != "y")) "fishers_z" else "none",
    prior_scale = if (any(colnames(data) != "y")) "cohens_d" else "none",
    standardize_predictors = TRUE,
    effect_direction = "positive",
    priors = NULL,
    model_type = NULL,
   priors_effect = prior(distribution = "normal", parameters = list(mean = 0, sd = 1)),
   priors_heterogeneity = prior(distribution = "invgamma", parameters = list(shape = 1,
      scale = 0.15)),
   priors_bias = list(prior_weightfunction(distribution = "two.sided", parameters =
      list(alpha = c(1, 1), steps = c(0.05)), prior_weights = 1/12),
     prior_weightfunction(distribution = "two.sided", parameters = list(alpha = c(1, 1,
     1), steps = c(0.05, 0.1)), prior_weights = 1/12), prior_weightfunction(distribution =
     "one.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_weights =
     1/12), prior_weightfunction(distribution = "one.sided", parameters = list(alpha =
      c(1, 1, 1), steps = c(0.025, 0.05)), prior_weights = 1/12),

     prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1,
     1), steps = c(0.05, 0.5)), prior_weights = 1/12), prior_weightfunction(distribution =
     "one.sided", parameters = list(alpha = c(1, 1, 1, 1), steps = c(0.025, 0.05, 0.5)),
     prior_weights = 1/12), prior_PET(distribution = "Cauchy", parameters = list(0, 1),
     truncation = list(0, Inf), prior_weights = 1/4), prior_PEESE(distribution = "Cauchy",
      parameters = list(0, 5), truncation = list(0, Inf), prior_weights = 1/4)),
   priors_effect_null = prior(distribution = "point", parameters = list(location = 0)),
   priors_heterogeneity_null = prior(distribution = "point", parameters = list(location =
      0)),
    priors_bias_null = prior_none(),
    priors_hierarchical = prior("beta", parameters = list(alpha = 1, beta = 1)),
    priors_hierarchical_null = NULL,
    prior_covariates = prior("normal", parameters = list(mean = 0, sd = 0.25)),
    prior_covariates_null = prior("spike", parameters = list(location = 0)),
    prior_factors = prior_factor("mnormal", parameters = list(mean = 0, sd = 0.25),
      contrast = "meandif"),
    prior_factors_null = prior("spike", parameters = list(location = 0)),
    models = FALSE,
    silent = FALSE,
    ...
)

check_setup.reg(
  formula,
  data,
  test_predictors = TRUE,
  study_names = NULL,
```

```
  study_ids = NULL,
  transformation = if (any(colnames(data) != "y")) "fishers_z" else "none",
  prior_scale = if (any(colnames(data) != "y")) "cohens_d" else "none",
  standardize_predictors = TRUE,
  effect_direction = "positive",
  priors = NULL,
  model_type = NULL,
 priors_effect = prior(distribution = "normal", parameters = list(mean = 0, sd = 1)),
 priors_heterogeneity = prior(distribution = "invgamma", parameters = list(shape = 1,
    scale = 0.15)),
 priors_bias = list(prior_weightfunction(distribution = "two.sided", parameters =
    list(alpha = c(1, 1), steps = c(0.05)), prior_weights = 1/12),
   prior_weightfunction(distribution = "two.sided", parameters = list(alpha = c(1, 1,
   1), steps = c(0.05, 0.1)), prior_weights = 1/12), prior_weightfunction(distribution =
   "one.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_weights =
   1/12), prior_weightfunction(distribution = "one.sided", parameters = list(alpha =
    c(1, 1, 1), steps = c(0.025, 0.05)), prior_weights = 1/12),

   prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1,
   1), steps = c(0.05, 0.5)), prior_weights = 1/12), prior_weightfunction(distribution =
   "one.sided", parameters = list(alpha = c(1, 1, 1, 1), steps = c(0.025, 0.05, 0.5)),
   prior_weights = 1/12), prior_PET(distribution = "Cauchy", parameters = list(0, 1),
   truncation = list(0, Inf), prior_weights = 1/4), prior_PEESE(distribution = "Cauchy",
    parameters = list(0, 5), truncation = list(0, Inf), prior_weights = 1/4)),
 priors_effect_null = prior(distribution = "point", parameters = list(location = 0)),
 priors_heterogeneity_null = prior(distribution = "point", parameters = list(location =
    0)),
  priors_bias_null = prior_none(),
  priors_hierarchical = prior("beta", parameters = list(alpha = 1, beta = 1)),
  priors_hierarchical_null = NULL,
  prior_covariates = prior("normal", parameters = list(mean = 0, sd = 0.25)),
  prior_covariates_null = prior("spike", parameters = list(location = 0)),
  prior_factors = prior_factor("mnormal", parameters = list(mean = 0, sd = 0.25),
    contrast = "meandif"),
  prior_factors_null = prior("spike", parameters = list(location = 0)),
  models = FALSE,
  silent = FALSE,
  ...
)
```

Arguments

`formula`	a formula for the meta-regression model
`data`	a data.frame containing the data for the meta-regression. Note that the column names have to correspond to the effect sizes (`d`, `logOR`, `OR`, `r`, `z`), a measure of sampling variability (`se`, `v`, `n`, `lCI`, `uCI`, `t`), and the predictors. See `combine_data()` for a complete list of reserved names and additional information about specifying input data.

`test_predictors`	vector of predictor names to test for the presence of moderation (i.e., assigned both the null and alternative prior distributions). Defaults to `TRUE`, all predictors are tested using the default prior distributions (i.e., `prior_covariates`, `prior_covariates_null`, `prior_factors`, and `prior_factors_null`). To only estimate and adjust for the effect of predictors use `FALSE`. If `priors` is specified, any settings in `test_predictors` is overridden.
`study_names`	an optional argument with the names of the studies
`study_ids`	an optional argument specifying dependency between the studies (for using a multilevel model). Defaults to `NULL` for studies being independent.
`transformation`	transformation to be applied to the supplied effect sizes before fitting the individual models. Defaults to `"fishers_z"`. We highly recommend using `"fishers_z"` transformation since it is the only variance stabilizing measure and does not bias PET and PEESE style models. The other options are `"cohens_d"`, correlation coefficient `"r"` and `"logOR"`. Supplying `"none"` will treat the effect sizes as unstandardized and refrain from any transformations.
`prior_scale`	an effect size scale used to define priors. Defaults to `"cohens_d"`. Other options are `"fishers_z"`, correlation coefficient `"r"`, and `"logOR"`. The prior scale does not need to match the effect sizes measure - the samples from prior distributions are internally transformed to match the `transformation` of the data. The `prior_scale` corresponds to the effect size scale of default output, but can be changed within the summary function.
`standardize_predictors`	whether continuous predictors should be standardized prior to estimating the model. Defaults to `TRUE`. Continuous predictor standardization is important for applying the default prior distributions for continuous predictors. Note that the resulting output corresponds to standardized meta-regression coefficients.
`effect_direction`	the expected direction of the effect. Correctly specifying the expected direction of the effect is crucial for one-sided selection models, as they specify cut-offs using one-sided p-values. Defaults to `"positive"` (another option is `"negative"`).
`priors`	named list of prior distributions for each predictor (with names corresponding to the predictors). It allows users to specify both the null and alternative hypothesis prior distributions for each predictor by assigning the corresponding element of the named list with another named list (with `"null"` and `"alt"`). If only one prior is specified for a given parameter, it is assumed to correspond to the alternative hypotheses and the default null hypothesis is specified (i.e., `prior_covariates_null` or `prior_factors_null`). If a named list with only one named prior distribution is provided (either `"null"` or `"alt"`), only this prior distribution is used and no default distribution is filled in. Parameters without specified prior distributions are assumed to be only adjusted for using the default alternative hypothesis prior distributions (i.e., `prior_covariates` or `prior_factors`). If `priors` is specified, `test_predictors` is ignored.
`model_type`	string specifying the RoBMA ensemble. Defaults to `NULL`. The other options are `"PSMA"`, `"PP"`, and `"2w"` which override settings passed to the `priors_effect`, `priors_heterogeneity`, `priors_effect`, `priors_effect_null`, `priors_heterogeneity_null`,

	`priors_bias_null`, and `priors_effect`. See details for more information about the different model types.
`priors_effect`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the alternative hypothesis. Defaults to a standard normal distribution `prior(distribution = "normal", parameters = list(mean = 0, sd = 1))`.
`priors_heterogeneity`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the alternative hypothesis. Defaults to `prior(distribution = "invgamma", parameters = list(shape = 1, scale = .15))` that is based on heterogeneities estimates from psychology (van Erp et al. 2017).
`priors_bias`	list of prior distributions for the publication bias adjustment component that will be treated as belonging to the alternative hypothesis. Defaults to `list(prior_weightfunction(distribution = "two.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_weights = 1/12),prior_weightfunction(distribution = "two.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.05, 0.10)), prior_weights = 1/12),prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_weights = 1/12),prior_weightfu = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.025, 0.05)), prior_weights = 1/12),prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.05, 0.5)), prior_weights = 1/12),prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1, 1), steps = c(0.025, 0.05, 0.5)), prior_weights = 1/12),prior_PET(distribution = "Cauchy", parameters = list(0,1), truncation = list(0, Inf), prior_weights = 1/4),prior_PEESE(distribution = "Cauchy", parameters = list(0,5), truncation = list(0, Inf), prior_weights = 1/4))`, corresponding to the RoBMA-PSMA model introduce by Bartoš et al. (2023).
`priors_effect_null`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero, `prior(distribution = "point", parameters = list(location = 0))`.
`priors_heterogeneity_null`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero (a fixed effect meta-analytic models), `prior(distribution = "point", parameters = list(location = 0))`.
`priors_bias_null`	list of prior weight functions for the `omega` parameter that will be treated as belonging to the null hypothesis. Defaults no publication bias adjustment, `prior_none()`.
`priors_hierarchical`	list of prior distributions for the correlation of random effects (`rho`) parameter that will be treated as belonging to the alternative hypothesis. This setting allows users to fit a hierarchical (three-level) meta-analysis when `study_ids` are supplied. Note that this is an experimental feature and see News for more details. Defaults to a beta distribution `prior(distribution = "beta", parameters = list(alpha = 1, beta = 1))`.
`priors_hierarchical_null`	list of prior distributions for the correlation of random effects (`rho`) parameter that will be treated as belonging to the null hypothesis. Defaults to `NULL`.

`prior_covariates`	a prior distributions for the regression parameter of continuous covariates on the effect size under the alternative hypothesis (unless set explicitly in `priors`). Defaults to a relatively wide normal distribution `prior(distribution = "normal", parameters = list(mean = 0, sd = 0.25))`.
`prior_covariates_null`	a prior distributions for the regression parameter of continuous covariates on the effect size under the null hypothesis (unless set explicitly in `priors`). Defaults to a no effect `prior("spike", parameters = list(location = 0))`.
`prior_factors`	a prior distributions for the regression parameter of categorical covariates on the effect size under the alternative hypothesis (unless set explicitly in `priors`). Defaults to a relatively wide multivariate normal distribution specifying differences from the mean contrasts `prior_factor("mnormal", parameters = list(mean = 0, sd = 0.25), contrast = "meandif")`.
`prior_factors_null`	a prior distributions for the regression parameter of categorical covariates on the effect size under the null hypothesis (unless set explicitly in `priors`). Defaults to a no effect `prior("spike", parameters = list(location = 0))`.
`models`	should the models' details be printed.
`silent`	do not print the results.
`...`	additional arguments.

Value

`check_setup.reg` invisibly returns list of summary tables.

`check_setup.reg` invisibly returns list of summary tables.

See Also

`check_setup()` `RoBMA.reg()`

`check_setup()` `RoBMA.reg()`

`combine_data` *Combines different effect sizes into a common metric*

Description

`combine_data` combines different effect sizes into a common measure specified in `transformation`. Either a data.frame `data` with columns named corresponding to the arguments or vectors with individual values can be passed.

Usage

```
combine_data(
  d = NULL,
  r = NULL,
  z = NULL,
  logOR = NULL,
  OR = NULL,
  t = NULL,
  y = NULL,
  se = NULL,
  v = NULL,
  n = NULL,
  lCI = NULL,
  uCI = NULL,
  study_names = NULL,
  study_ids = NULL,
  weight = NULL,
  data = NULL,
  transformation = "fishers_z",
  return_all = FALSE,
  ...
)
```

Arguments

`d`	a vector of effect sizes measured as Cohen's d
`r`	a vector of effect sizes measured as correlations
`z`	a vector of effect sizes measured as Fisher's z
`logOR`	a vector of effect sizes measured as log odds ratios
`OR`	a vector of effect sizes measured as odds ratios
`t`	a vector of t/z-statistics
`y`	a vector of unspecified effect sizes (note that effect size transformations are unavailable with this type of input)
`se`	a vector of standard errors of the effect sizes
`v`	a vector of variances of the effect sizes
`n`	a vector of overall sample sizes
`lCI`	a vector of lower bounds of confidence intervals
`uCI`	a vector of upper bounds of confidence intervals
`study_names`	an optional argument with the names of the studies
`study_ids`	an optional argument specifying dependency between the studies (for using a multilevel model). Defaults to `NULL` for studies being independent.
`weight`	specifies likelihood weights of the individual estimates. Notes that this is an untested experimental feature.

`data`	a data frame with column names corresponding to the variable names used to supply data individually
`transformation`	transformation to be applied to the supplied effect sizes before fitting the individual models. Defaults to `"fishers_z"`. We highly recommend using `"fishers_z"` transformation since it is the only variance stabilizing measure and does not bias PET and PEESE style models. The other options are `"cohens_d"`, correlation coefficient `"r"` and `"logOR"`. Supplying `"none"` will treat the effect sizes as unstandardized and refrain from any transformations.
`return_all`	whether data frame containing all filled values should be returned. Defaults to `FALSE`
`...`	additional arguments.

Details

The aim of the function is to combine different, already calculated, effect size measures. In order to obtain effect size measures from raw values, e.g, mean differences, standard deviations, and sample sizes, use escalc function.

The function checks the input values and in transforming the input into a common effect size measure in the following fashion:

1. obtains missing standard errors by squaring variances
2. obtains missing standard errors from confidence intervals (after transformation to Fisher's z scale for `d` and `r`).
3. obtains missing sample sizes (or standard errors for logOR) from t-statistics and effect sizes
4. obtains missing standard errors from sample sizes and effect sizes
5. obtains missing sample sizes from standard errors and effect sizes
6. obtains missing t-statistics from sample sizes and effect sizes (or standard errors and effect sizes for logOR)
7. changes the effect sizes direction to be positive
8. transforms effect sizes into the common effect size
9. transforms standard errors into the common metric

If the `transforms` is `NULL` or an unstandardized effect size `y` is supplied, steps 4-9 are skipped.

Value

`combine_data` returns a data.frame.

See Also

`RoBMA()`, `check_setup()`, `effect_sizes()`, `standard_errors()`, and `sample_sizes()`

`contr.independent` *Independent contrast matrix*

Description

Return a matrix of independent contrasts – a level for each term.

Usage

```
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

Value

A matrix with n rows and k columns, with k = n if `contrasts = TRUE` and k = n if `contrasts = FALSE`.

References

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

Examples

```
contr.independent(c(1, 2))
contr.independent(c(1, 2, 3))
```

`contr.meandif` *Mean difference contrast matrix*

Description

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 gran 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`.

Usage

```
contr.meandif(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

Value

A matrix with n rows and k columns, with k = n - 1 if `contrasts` = `TRUE` and k = n if `contrasts` = `FALSE`.

References

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

Examples

```
contr.meandif(c(1, 2))
contr.meandif(c(1, 2, 3))
```

`contr.orthonormal` *Orthornomal contrast matrix*

Description

Return a matrix of orthornomal contrasts. Code is based on `stanova::contr.bayes` and corresponding to description by Rouder et al. (2012)

Usage

```
contr.orthonormal(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

Value

A matrix with n rows and k columns, with k = n - 1 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

```
contr.orthonormal(c(1, 2))
contr.orthonormal(c(1, 2, 3))
```

`diagnostics` *Checks a fitted RoBMA object*

Description

`diagnostics` creates visual checks of individual models convergence. Numerical overview of individual models can be obtained by `summary(object, type = "models", diagnostics = TRUE)`, or even more detailed information by `summary(object, type = "individual")`.

Usage

```
diagnostics(
  fit,
  parameter,
  type,
  plot_type = "base",
  show_models = NULL,
  lags = 30,
  title = is.null(show_models) | length(show_models) > 1,
  ...
)

diagnostics_autocorrelation(
  fit,
  parameter = NULL,
  plot_type = "base",
  show_models = NULL,
  lags = 30,
  title = is.null(show_models) | length(show_models) > 1,
  ...
)

diagnostics_trace(
  fit,
  parameter = NULL,
  plot_type = "base",
  show_models = NULL,
  title = is.null(show_models) | length(show_models) > 1,
  ...
)

diagnostics_density(
```

```
  fit,
  parameter = NULL,
  plot_type = "base",
  show_models = NULL,
  title = is.null(show_models) | length(show_models) > 1,
  ...
)
```

Arguments

`fit`	a fitted RoBMA object
`parameter`	a parameter to be plotted. Either `"mu"`, `"tau"`, `"omega"`, `"PET"`, or `"PEESE"`.
`type`	type of MCMC diagnostic to be plotted. Options are `"chains"` for the chains' trace plots, `"autocorrelation"` for autocorrelation of the chains, and `"densities"` for the overlaying densities of the individual chains. Can be abbreviated to first letters.
`plot_type`	whether to use a base plot `"base"` or ggplot2 `"ggplot"` for plotting. Defaults to `"base"`.
`show_models`	MCMC diagnostics of which models should be plotted. Defaults to `NULL` which plots MCMC diagnostics for a specified parameter for every model that is part of the ensemble.
`lags`	number of lags to be shown for `type = "autocorrelation"`. Defaults to `30`.
`title`	whether the model number should be displayed in title. Defaults to `TRUE` when more than one model is selected.
`...`	additional arguments to be passed to par if `plot_type = "base"`.

Details

The visualization functions are based on stan_plot function and its color schemes.

Value

`diagnostics` returns either `NULL` if `plot_type = "base"` or an object/list of objects (depending on the number of parameters to be plotted) of class 'ggplot2' if `plot_type = "ggplot2"`.

See Also

`RoBMA()`, `summary.RoBMA()`

Examples

```
## Not run:
# using the example data from Anderson et al. 2010 and fitting the default model
# (note that the model can take a while to fit)
fit <- RoBMA(r = Anderson2010$r, n = Anderson2010$n, study_names = Anderson2010$labels)

### ggplot2 version of all of the plots can be obtained by adding 'model_type = "ggplot"
# diagnostics function allows to visualize diagnostics of a fitted RoBMA object, for example,
```

```
# the trace plot for the mean parameter in each model model
diagnostics(fit, parameter = "mu", type = "chain")

# in order to show the trace plot only for the 11th model, add show_models parameter
diagnostics(fit, parameter = "mu", type = "chain", show_models = 11)

# furthermore, the autocorrelations
diagnostics(fit, parameter = "mu", type = "autocorrelation")

# and overlying densities for each plot can also be visualize
diagnostics(fit, parameter = "mu", type = "densities")

## End(Not run)
```

`effect_sizes` *Effect size transformations*

Description

Functions for transforming between different effect size measures.

Usage

```
d2r(d)

d2z(d)

d2logOR(d)

d2OR(d)

r2d(r)

r2z(r)

r2logOR(r)

r2OR(r)

z2r(z)

z2d(z)

z2logOR(z)

z2OR(z)
```

```
logOR2r(logOR)

logOR2z(logOR)

logOR2d(logOR)

logOR2OR(logOR)

OR2r(OR)

OR2z(OR)

OR2logOR(OR)

OR2d(OR)
```

Arguments

`d`	Cohen’s d.
`r`	correlation coefficient.
`z`	Fisher’s z.
`logOR`	log(odds ratios).
`OR`	offs ratios.

Details

All transformations are based on (Borenstein et al. 2011). In case that a direct transformation is not available, the transformations are chained to provide the effect size of interest.

References

Borenstein M, Hedges LV, Higgins JP, Rothstein HR (2011). *Introduction to meta-analysis*. John Wiley & Sons.

See Also

`standard_errors()`, `sample_sizes()`

`forest` *Forest plot for a RoBMA object*

Description

`forest` creates a forest plot for a `"RoBMA"` object.

Usage

```
forest(
  x,
  conditional = FALSE,
  plot_type = "base",
  output_scale = NULL,
  order = NULL,
  ...
)
```

Arguments

`x`	a fitted RoBMA object
`conditional`	whether conditional estimates should be plotted. Defaults to `FALSE` which plots the model-averaged estimates. Note that both `"weightfunction"` and `"PET-PEESE"` are always ignoring the other type of publication bias adjustment.
`plot_type`	whether to use a base plot `"base"` or ggplot2 `"ggplot"` for plotting. Defaults to `"base"`.
`output_scale`	transform the effect sizes and the meta-analytic effect size estimate to a different scale. Defaults to `NULL` which returns the same scale as the model was estimated on.
`order`	order of the studies. Defaults to `NULL` - ordering as supplied to the fitting function. Studies can be ordered either `"increasing"` or `"decreasing"` by effect size, or by labels `"alphabetical"`.
`...`	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

`forest` returns either `NULL` if `plot_type = "base"` or an object object of class 'ggplot2' if `plot_type = "ggplot2"`.

Examples

```
## Not run:
# using the example data from Anderson et al. 2010 and fitting the default model
# (note that the model can take a while to fit)
fit <- RoBMA(r = Anderson2010$r, n = Anderson2010$n, study_names = Anderson2010$labels)

### ggplot2 version of all of the plots can be obtained by adding 'model_type = "ggplot"
# the forest function creates a forest plot for a fitted RoBMA object, for example,
# the forest plot for the individual studies and the model-averaged effect size estimate
forest(fit)

# the conditional effect size estimate
forest(fit, conditional = TRUE)
```

```
# or transforming the effect size estimates to Fisher's z
forest(fit, output_scale = "fishers_z")

## End(Not run)
```

`interpret` *Interprets results of a RoBMA model.*

Description

`interpret` creates a brief textual summary of a fitted RoBMA object.

Usage

```
interpret(object, output_scale = NULL)
```

Arguments

`object`	a fitted RoBMA object
`output_scale`	transform the meta-analytic estimates to a different scale. Defaults to `NULL` which returns the same scale as the model was estimated on.

Value

`interpret` returns a character.

`is.RoBMA` *Reports whether x is a RoBMA object*

Description

Reports whether x is a RoBMA object

Usage

```
is.RoBMA(x)

is.RoBMA.reg(x)

is.NoBMA(x)

is.NoBMA.reg(x)

is.BiBMA(x)
```

Arguments

`x`	an object to test

Value

returns a boolean.

`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, samples sizes, type of the educational outcome, intensity of the employment, gender of the student population, study location, study design, whether the study controlled for endogenity, and whether the study controlled for motivation. The original data set including additional variables and the publication can be found at http://meta-analysis.cz/students. (Note that some standard errors and employment intensities are missing.)

Usage

```
Kroupova2021
```

Format

A data.frame with 11 columns and 881 observations.

Value

a data.frame.

References

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

`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.

Value

a data.frame.

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.

`marginal_plot` *Plots marginal estimates of a fitted RoBMA regression object*

Description

`marginal_plot` allows to visualize prior and posterior distributions of marginal estimates of a RoBMA regression model.

Usage

```
marginal_plot(
  x,
  parameter,
  conditional = FALSE,
  plot_type = "base",
  prior = FALSE,
  output_scale = NULL,
  dots_prior = NULL,
  ...
)
```

Arguments

`x`	a fitted RoBMA regression object
`parameter`	regression parameter to be plotted
`conditional`	whether conditional marginal estimates should be plotted. Defaults to `FALSE` which plots the model-averaged estimates.
`plot_type`	whether to use a base plot `"base"` or ggplot2 `"ggplot"` for plotting. Defaults to `"base"`.
`prior`	whether prior distribution should be added to figure. Defaults to `FALSE`.
`output_scale`	transform the effect sizes and the meta-analytic effect size estimate to a different scale. Defaults to `NULL` which returns the same scale as the model was estimated on.
`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.RoBMA` returns either `NULL` if `plot_type = "base"` or an object object of class 'ggplot2' if `plot_type = "ggplot2"`.

See Also

`RoBMA()`

`marginal_summary` *Summarize marginal estimates of a fitted RoBMA regression object*

Description

`marginal_summary` creates summary tables for marginal estimates of a RoBMA regression model.

Usage

```
marginal_summary(
  object,
  conditional = FALSE,
  output_scale = NULL,
  probs = c(0.025, 0.975),
  logBF = FALSE,
  BF01 = FALSE
)
```

Arguments

`object`	a fitted RoBMA regression object
`conditional`	show the conditional estimates (assuming that the alternative is true).
`output_scale`	transform the meta-analytic estimates to a different scale. Defaults to `NULL` which returns the same scale as the model was estimated on.
`probs`	quantiles of the posterior samples to be displayed. Defaults to `c(.025, .975)`
`logBF`	show log of Bayes factors. Defaults to `FALSE`.
`BF01`	show Bayes factors in support of the null hypotheses. Defaults to `FALSE`.

Value

`marginal_summary` returns a list of tables of class 'BayesTools_table'.

See Also

`RoBMA()`, `summary.RoBMA()`, `diagnostics()`, `check_RoBMA()`

NoBMA *Estimate a Bayesian Model-Averaged Meta-Analysis*

Description

`NoBMA` is a wrapper around `RoBMA()` that can be used to estimate a publication bias unadjusted Bayesian model-averaged meta-analysis. The interface allows a complete customization of the ensemble with different prior (or list of prior) distributions for each component.

Usage

```
NoBMA(
  d = NULL,
  r = NULL,
  logOR = NULL,
  OR = NULL,
  z = NULL,
  y = NULL,
  se = NULL,
  v = NULL,
  n = NULL,
  lCI = NULL,
  uCI = NULL,
  t = NULL,
  study_names = NULL,
  study_ids = NULL,
  data = NULL,
  weight = NULL,
  transformation = if (is.null(y)) "fishers_z" else "none",
  prior_scale = if (is.null(y)) "cohens_d" else "none",
  model_type = NULL,
  rescale_priors = 1,
  priors_effect = set_default_priors("effect", rescale = rescale_priors),
 priors_heterogeneity = set_default_priors("heterogeneity", rescale = rescale_priors),
  priors_effect_null = set_default_priors("effect", null = TRUE),
  priors_heterogeneity_null = set_default_priors("heterogeneity", null = TRUE),
  priors_hierarchical = set_default_priors("hierarchical"),
  priors_hierarchical_null = set_default_priors("hierarchical", null = TRUE),
  algorithm = "bridge",
  chains = 3,
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  thin = 1,
  parallel = FALSE,
  autofit = TRUE,
  autofit_control = set_autofit_control(),
```

```
  convergence_checks = set_convergence_checks(),
  save = "all",
  seed = NULL,
  silent = TRUE,
  ...
)
```

Arguments

`d`	a vector of effect sizes measured as Cohen's d
`r`	a vector of effect sizes measured as correlations
`logOR`	a vector of effect sizes measured as log odds ratios
`OR`	a vector of effect sizes measured as odds ratios
`z`	a vector of effect sizes measured as Fisher's z
`y`	a vector of unspecified effect sizes (note that effect size transformations are unavailable with this type of input)
`se`	a vector of standard errors of the effect sizes
`v`	a vector of variances of the effect sizes
`n`	a vector of overall sample sizes
`lCI`	a vector of lower bounds of confidence intervals
`uCI`	a vector of upper bounds of confidence intervals
`t`	a vector of t/z-statistics
`study_names`	an optional argument with the names of the studies
`study_ids`	an optional argument specifying dependency between the studies (for using a multilevel model). Defaults to `NULL` for studies being independent.
`data`	a data object created by the `combine_data` function. This is an alternative input entry to specifying the `d`, `r`, `y`, etc... directly. I.e., RoBMA function does not allow passing a data.frame and referencing to the columns.
`weight`	specifies likelihood weights of the individual estimates. Notes that this is an untested experimental feature.
`transformation`	transformation to be applied to the supplied effect sizes before fitting the individual models. Defaults to `"fishers_z"`. We highly recommend using `"fishers_z"` transformation since it is the only variance stabilizing measure and does not bias PET and PEESE style models. The other options are `"cohens_d"`, correlation coefficient `"r"` and `"logOR"`. Supplying `"none"` will treat the effect sizes as unstandardized and refrain from any transformations.
`prior_scale`	an effect size scale used to define priors. Defaults to `"cohens_d"`. Other options are `"fishers_z"`, correlation coefficient `"r"`, and `"logOR"`. The prior scale does not need to match the effect sizes measure - the samples from prior distributions are internally transformed to match the `transformation` of the data. The `prior_scale` corresponds to the effect size scale of default output, but can be changed within the summary function.

`model_type`	string specifying the RoBMA ensemble. Defaults to `NULL`. The other options are `"PSMA"`, `"PP"`, and `"2w"` which override settings passed to the `priors_effect`, `priors_heterogeneity`, `priors_effect`, `priors_effect_null`, `priors_heterogeneity_null`, `priors_bias_null`, and `priors_effect`. See details for more information about the different model types.
`rescale_priors`	a re-scaling factor for the prior distributions. The re-scaling factor allows to adjust the width of all default priors simultaneously. Defaults to `1`.
`priors_effect`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the alternative hypothesis. Defaults to a standard normal distribution `prior(distribution = "normal", parameters = list(mean = 0, sd = 1))`.
`priors_heterogeneity`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the alternative hypothesis. Defaults to `prior(distribution = "invgamma", parameters = list(shape = 1, scale = .15))` that is based on heterogeneities estimates from psychology (van Erp et al. 2017).
`priors_effect_null`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero, `prior(distribution = "point", parameters = list(location = 0))`.
`priors_heterogeneity_null`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero (a fixed effect meta-analytic models), `prior(distribution = "point", parameters = list(location = 0))`.
`priors_hierarchical`	list of prior distributions for the correlation of random effects (`rho`) parameter that will be treated as belonging to the alternative hypothesis. This setting allows users to fit a hierarchical (three-level) meta-analysis when `study_ids` are supplied. Note that this is an experimental feature and see News for more details. Defaults to a beta distribution `prior(distribution = "beta", parameters = list(alpha = 1, beta = 1))`.
`priors_hierarchical_null`	list of prior distributions for the correlation of random effects (`rho`) parameter that will be treated as belonging to the null hypothesis. Defaults to `NULL`.
`algorithm`	a string specifying the algorithm used for the model averaging. Defaults to `"bridge"` which results in estimating individual models using JAGS and computing the marginal likelihood using bridge sampling. An alternative is `"ss"` which uses spike and slab like parameterization to approximate the Bayesian model averaging with a single model. Note that significantly more `sample`, `burnin`, and `adapt` iterations are needed for the `"ss"` algorithm.
`chains`	a number of chains of the MCMC algorithm.
`sample`	a number of sampling iterations of the MCMC algorithm. Defaults to `5000`.
`burnin`	a number of burnin iterations of the MCMC algorithm. Defaults to `2000`.
`adapt`	a number of adaptation iterations of the MCMC algorithm. Defaults to `500`.
`thin`	a thinning of the chains of the MCMC algorithm. Defaults to `1`.

`parallel`	whether the individual models should be fitted in parallel. Defaults to `FALSE`. The implementation is not completely stable and might cause a connection error.
`autofit`	whether the model should be fitted until the convergence criteria (specified in `autofit_control`) are satisfied. Defaults to `TRUE`.
`autofit_control`	allows to pass autofit control settings with the `set_autofit_control()` function. See `?set_autofit_control` for options and default settings.
`convergence_checks`	automatic convergence checks to assess the fitted models, passed with `set_convergence_checks()` function. See `?set_convergence_checks` for options and default settings.
`save`	whether all models posterior distributions should be kept after obtaining a model-averaged result. Defaults to `"all"` which does not remove anything. Set to `"min"` to significantly reduce the size of final object, however, some model diagnostics and further manipulation with the object will not be possible.
`seed`	a seed to be set before model fitting, marginal likelihood computation, and posterior mixing for reproducibility of results. Defaults to `NULL` - no seed is set.
`silent`	whether all print messages regarding the fitting process should be suppressed. Defaults to `TRUE`. Note that `parallel = TRUE` also suppresses all messages.
`...`	additional arguments.

Details

See `RoBMA()` for more details.

Note that these default prior distributions are relatively wide and more informed prior distributions for testing for the presence of moderation should be considered.

Value

`NoBMA` returns an object of class 'RoBMA'.

See Also

`RoBMA()`, `summary.RoBMA()`, `update.RoBMA()`, `check_setup()`

`NoBMA.reg` *Estimate a Bayesian Model-Averaged Meta-Regression*

Description

`NoBMA.reg` is a wrapper around `RoBMA.reg()` that can be used to estimate a publication bias unadjusted Bayesian model-averaged meta-regression. The interface allows a complete customization of the ensemble with different prior (or list of prior) distributions for each component.

Usage

```
NoBMA.reg(
  formula,
  data,
  test_predictors = TRUE,
  study_names = NULL,
  study_ids = NULL,
  transformation = if (any(colnames(data) != "y")) "fishers_z" else "none",
  prior_scale = if (any(colnames(data) != "y")) "cohens_d" else "none",
  standardize_predictors = TRUE,
  priors = NULL,
  model_type = NULL,
  rescale_priors = 1,
  priors_effect = set_default_priors("effect", rescale = rescale_priors),
 priors_heterogeneity = set_default_priors("heterogeneity", rescale = rescale_priors),
  priors_effect_null = set_default_priors("effect", null = TRUE),
  priors_heterogeneity_null = set_default_priors("heterogeneity", null = TRUE),
  priors_hierarchical = set_default_priors("hierarchical"),
  priors_hierarchical_null = set_default_priors("hierarchical", null = TRUE),
 prior_covariates = set_default_priors("covariates", rescale = rescale_priors),
  prior_covariates_null = set_default_priors("covariates", null = TRUE),
  prior_factors = set_default_priors("factors", rescale = rescale_priors),
  prior_factors_null = set_default_priors("factors", null = TRUE),
  algorithm = "bridge",
  chains = 3,
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  thin = 1,
  parallel = FALSE,
  autofit = TRUE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  save = "all",
  seed = NULL,
  silent = TRUE,
  ...
)
```

Arguments

`formula`	a formula for the meta-regression model
`data`	a data.frame containing the data for the meta-regression. Note that the column names have to correspond to the effect sizes (`d`, `logOR`, `OR`, `r`, `z`), a measure of sampling variability (`se`, `v`, `n`, `lCI`, `uCI`, `t`), and the predictors. See `combine_data()` for a complete list of reserved names and additional information about specifying input data.

`test_predictors`	vector of predictor names to test for the presence of moderation (i.e., assigned both the null and alternative prior distributions). Defaults to `TRUE`, all predictors are tested using the default prior distributions (i.e., `prior_covariates`, `prior_covariates_null`, `prior_factors`, and `prior_factors_null`). To only estimate and adjust for the effect of predictors use `FALSE`. If `priors` is specified, any settings in `test_predictors` is overridden.
`study_names`	an optional argument with the names of the studies
`study_ids`	an optional argument specifying dependency between the studies (for using a multilevel model). Defaults to `NULL` for studies being independent.
`transformation`	transformation to be applied to the supplied effect sizes before fitting the individual models. Defaults to `"fishers_z"`. We highly recommend using `"fishers_z"` transformation since it is the only variance stabilizing measure and does not bias PET and PEESE style models. The other options are `"cohens_d"`, correlation coefficient `"r"` and `"logOR"`. Supplying `"none"` will treat the effect sizes as unstandardized and refrain from any transformations.
`prior_scale`	an effect size scale used to define priors. Defaults to `"cohens_d"`. Other options are `"fishers_z"`, correlation coefficient `"r"`, and `"logOR"`. The prior scale does not need to match the effect sizes measure - the samples from prior distributions are internally transformed to match the `transformation` of the data. The `prior_scale` corresponds to the effect size scale of default output, but can be changed within the summary function.
`standardize_predictors`	whether continuous predictors should be standardized prior to estimating the model. Defaults to `TRUE`. Continuous predictor standardization is important for applying the default prior distributions for continuous predictors. Note that the resulting output corresponds to standardized meta-regression coefficients.
`priors`	named list of prior distributions for each predictor (with names corresponding to the predictors). It allows users to specify both the null and alternative hypothesis prior distributions for each predictor by assigning the corresponding element of the named list with another named list (with `"null"` and `"alt"`). If only one prior is specified for a given parameter, it is assumed to correspond to the alternative hypotheses and the default null hypothesis is specified (i.e., `prior_covariates_null` or `prior_factors_null`). If a named list with only one named prior distribution is provided (either `"null"` or `"alt"`), only this prior distribution is used and no default distribution is filled in. Parameters without specified prior distributions are assumed to be only adjusted for using the default alternative hypothesis prior distributions (i.e., `prior_covariates` or `prior_factors`). If `priors` is specified, `test_predictors` is ignored.
`model_type`	string specifying the RoBMA ensemble. Defaults to `NULL`. The other options are `"PSMA"`, `"PP"`, and `"2w"` which override settings passed to the `priors_effect`, `priors_heterogeneity`, `priors_effect`, `priors_effect_null`, `priors_heterogeneity_null`, `priors_bias_null`, and `priors_effect`. See details for more information about the different model types.
`rescale_priors`	a re-scaling factor for the prior distributions. The re-scaling factor allows to adjust the width of all default priors simultaneously. Defaults to `1`.

`priors_effect`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the alternative hypothesis. Defaults to a standard normal distribution `prior(distribution = "normal", parameters = list(mean = 0, sd = 1))`.
`priors_heterogeneity`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the alternative hypothesis. Defaults to `prior(distribution = "invgamma", parameters = list(shape = 1, scale = .15))` that is based on heterogeneities estimates from psychology (van Erp et al. 2017).
`priors_effect_null`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero, `prior(distribution = "point", parameters = list(location = 0))`.
`priors_heterogeneity_null`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero (a fixed effect meta-analytic models), `prior(distribution = "point", parameters = list(location = 0))`.
`priors_hierarchical`	list of prior distributions for the correlation of random effects (`rho`) parameter that will be treated as belonging to the alternative hypothesis. This setting allows users to fit a hierarchical (three-level) meta-analysis when `study_ids` are supplied. Note that this is an experimental feature and see News for more details. Defaults to a beta distribution `prior(distribution = "beta", parameters = list(alpha = 1, beta = 1))`.
`priors_hierarchical_null`	list of prior distributions for the correlation of random effects (`rho`) parameter that will be treated as belonging to the null hypothesis. Defaults to `NULL`.
`prior_covariates`	a prior distributions for the regression parameter of continuous covariates on the effect size under the alternative hypothesis (unless set explicitly in `priors`). Defaults to a relatively wide normal distribution `prior(distribution = "normal", parameters = list(mean = 0, sd = 0.25))`.
`prior_covariates_null`	a prior distributions for the regression parameter of continuous covariates on the effect size under the null hypothesis (unless set explicitly in `priors`). Defaults to a no effect `prior("spike", parameters = list(location = 0))`.
`prior_factors`	a prior distributions for the regression parameter of categorical covariates on the effect size under the alternative hypothesis (unless set explicitly in `priors`). Defaults to a relatively wide multivariate normal distribution specifying differences from the mean contrasts `prior_factor("mnormal", parameters = list(mean = 0, sd = 0.25), contrast = "meandif")`.
`prior_factors_null`	a prior distributions for the regression parameter of categorical covariates on the effect size under the null hypothesis (unless set explicitly in `priors`). Defaults to a no effect `prior("spike", parameters = list(location = 0))`.
`algorithm`	a string specifying the algorithm used for the model averaging. Defaults to `"bridge"` which results in estimating individual models using JAGS and computing the marginal likelihood using bridge sampling. An alternative is `"ss"`

	which uses spike and slab like parameterization to approximate the Bayesian model averaging with a single model. Note that significantly more `sample`, `burnin`, and `adapt` iterations are needed for the `"ss"` algorithm.
`chains`	a number of chains of the MCMC algorithm.
`sample`	a number of sampling iterations of the MCMC algorithm. Defaults to `5000`.
`burnin`	a number of burnin iterations of the MCMC algorithm. Defaults to `2000`.
`adapt`	a number of adaptation iterations of the MCMC algorithm. Defaults to `500`.
`thin`	a thinning of the chains of the MCMC algorithm. Defaults to `1`.
`parallel`	whether the individual models should be fitted in parallel. Defaults to `FALSE`. The implementation is not completely stable and might cause a connection error.
`autofit`	whether the model should be fitted until the convergence criteria (specified in `autofit_control`) are satisfied. Defaults to `TRUE`.
`autofit_control`	allows to pass autofit control settings with the `set_autofit_control()` function. See `?set_autofit_control` for options and default settings.
`convergence_checks`	automatic convergence checks to assess the fitted models, passed with `set_convergence_checks()` function. See `?set_convergence_checks` for options and default settings.
`save`	whether all models posterior distributions should be kept after obtaining a model-averaged result. Defaults to `"all"` which does not remove anything. Set to `"min"` to significantly reduce the size of final object, however, some model diagnostics and further manipulation with the object will not be possible.
`seed`	a seed to be set before model fitting, marginal likelihood computation, and posterior mixing for reproducibility of results. Defaults to `NULL` - no seed is set.
`silent`	whether all print messages regarding the fitting process should be suppressed. Defaults to `TRUE`. Note that `parallel = TRUE` also suppresses all messages.
`...`	additional arguments.

Details

See `RoBMA.reg()` for more details.

Note that these default prior distributions are relatively wide and more informed prior distributions for testing for the presence of moderation should be considered.

Value

`NoBMA.reg` returns an object of class 'RoBMA'.

See Also

`RoBMA()`, `RoBMA.reg()`, `summary.RoBMA()`, `update.RoBMA()`, `check_setup()`

`plot.RoBMA` *Plots a fitted RoBMA object*

Description

`plot.RoBMA` allows to visualize different `"RoBMA"` object parameters in various ways. See `type` for the different model types.

Usage

```
## S3 method for class 'RoBMA'
plot(
  x,
  parameter = "mu",
  conditional = FALSE,
  plot_type = "base",
  prior = FALSE,
  output_scale = NULL,
  rescale_x = FALSE,
  show_data = TRUE,
  dots_prior = NULL,
  ...
)
```

Arguments

`x`	a fitted RoBMA object
`parameter`	a parameter to be plotted. Defaults to `"mu"` (for the effect size). The additional options are `"tau"` (for the heterogeneity), `"weightfunction"` (for the estimated weightfunction), or `"PET-PEESE"` (for the PET-PEESE regression).
`conditional`	whether conditional estimates should be plotted. Defaults to `FALSE` which plots the model-averaged estimates. Note that both `"weightfunction"` and `"PET-PEESE"` are always ignoring the other type of publication bias adjustment.
`plot_type`	whether to use a base plot `"base"` or ggplot2 `"ggplot"` for plotting. Defaults to `"base"`.
`prior`	whether prior distribution should be added to figure. Defaults to `FALSE`.
`output_scale`	transform the effect sizes and the meta-analytic effect size estimate to a different scale. Defaults to `NULL` which returns the same scale as the model was estimated on.
`rescale_x`	whether the x-axis of the `"weightfunction"` should be re-scaled to make the x-ticks equally spaced. Defaults to `FALSE`.
`show_data`	whether the study estimates and standard errors should be show in the `"PET-PEESE"` plot. Defaults to `TRUE`.

`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.RoBMA` returns either `NULL` if `plot_type = "base"` or an object object of class 'ggplot2' if `plot_type = "ggplot2"`.

See Also

`RoBMA()`

Examples

```
## Not run:
# using the example data from Anderson et al. 2010 and fitting the default model
# (note that the model can take a while to fit)
fit <- RoBMA(r = Anderson2010$r, n = Anderson2010$n, study_names = Anderson2010$labels)

### ggplot2 version of all of the plots can be obtained by adding 'model_type = "ggplot"
# the 'plot' function allows to visualize the results of a fitted RoBMA object, for example;
# the model-averaged effect size estimate
plot(fit, parameter = "mu")

# and show both the prior and posterior distribution
plot(fit, parameter = "mu", prior = TRUE)

# conditional plots can by obtained by specifying
plot(fit, parameter = "mu", conditional = TRUE)

# plotting function also allows to visualize the weight function
plot(fit, parameter = "weightfunction")

# re-scale the x-axis
plot(fit, parameter = "weightfunction", rescale_x = TRUE)

# or visualize the PET-PEESE regression line
plot(fit, parameter = "PET-PEESE")

## End(Not run)
```

`plot_models` *Models plot for a RoBMA object*

Description

`plot_models` plots individual models' estimates for a `"RoBMA"` object.

Usage

```
plot_models(
  x,
  parameter = "mu",
  conditional = FALSE,
  output_scale = NULL,
  plot_type = "base",
  order = "decreasing",
  order_by = "model",
  ...
)
```

Arguments

`x`	a fitted RoBMA object
`parameter`	a parameter to be plotted. Defaults to `"mu"` (for the effect size). The additional option is `"tau"` (for the heterogeneity).
`conditional`	whether conditional estimates should be plotted. Defaults to `FALSE` which plots the model-averaged estimates. Note that both `"weightfunction"` and `"PET-PEESE"` are always ignoring the other type of publication bias adjustment.
`output_scale`	transform the effect sizes and the meta-analytic effect size estimate to a different scale. Defaults to `NULL` which returns the same scale as the model was estimated on.
`plot_type`	whether to use a base plot `"base"` or ggplot2 `"ggplot"` for plotting. Defaults to `"base"`.
`order`	how the models should be ordered. Defaults to `"decreasing"` which orders them in decreasing order in accordance to `order_by` argument. The alternative is `"increasing"`.
`order_by`	what feature should be use to order the models. Defaults to `"model"` which orders the models according to their number. The alternatives are `"estimate"` (for the effect size estimates), `"probability"` (for the posterior model probability), and `"BF"` (for the inclusion Bayes factor).
`...`	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_models` returns either `NULL` if `plot_type = "base"` or an object object of class 'ggplot2' if `plot_type = "ggplot2"`.

Examples

```
## Not run:
# using the example data from Anderson et al. 2010 and fitting the default model
# (note that the model can take a while to fit)
fit <- RoBMA(r = Anderson2010$r, n = Anderson2010$n, study_names = Anderson2010$labels)

### ggplot2 version of all of the plots can be obtained by adding 'model_type = "ggplot"
# the plot_models function creates a plot for of the individual models' estimates, for example,
# the effect size estimates from the individual models can be obtained with
plot_models(fit)

# and effect size estimates from only the conditional models
plot_models(fit, conditional = TRUE)

## 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.

Value

a data.frame.

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.

`print.marginal_summary.RoBMA` *Prints marginal_summary object for RoBMA method*

Description

Prints marginal_summary object for RoBMA method

Usage

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

Arguments

`x`	a summary of a RoBMA object
`...`	additional arguments

Value

`print.marginal_summary.RoBMA` invisibly returns the print statement.

See Also

`RoBMA()`

`print.RoBMA` *Prints a fitted RoBMA object*

Description

Prints a fitted RoBMA object

Usage

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

Arguments

`x`	a fitted RoBMA object.
`...`	additional arguments.

Value

`print.RoBMA` invisibly returns the print statement.

See Also

`RoBMA()`

`print.summary.RoBMA` *Prints summary object for RoBMA method*

Description

Prints summary object for RoBMA method

Usage

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

Arguments

`x`	a summary of a RoBMA object
`...`	additional arguments

Value

`print.summary.RoBMA` invisibly returns the print statement.

See Also

`RoBMA()`

`prior` *Creates a prior distribution*

Description

`prior` creates a prior distribution. The prior can be visualized by the `plot` function.

Usage

```
prior(
  distribution,
  parameters,
  truncation = list(lower = -Inf, upper = Inf),
  prior_weights = 1
)
```

Arguments

`distribution`	name of the prior distribution. The possible options are `"point"` for a point density characterized by a `location` parameter. `"normal"` for a normal distribution characterized by a `mean` and `sd` parameters. `"lognormal"` for a lognormal distribution characterized by a `meanlog` and `sdlog` parameters. `"cauchy"` for a Cauchy distribution characterized by a `location` and `scale` parameters. Internally converted into a generalized t-distribution with `df = 1`. `"t"` for a generalized t-distribution characterized by a `location`, `scale`, and `df` parameters. `"gamma"` for a gamma distribution characterized by either `shape` and `rate`, or `shape` and `scale` parameters. The later is internally converted to the `shape` and `rate` parametrization `"invgamma"` for an inverse-gamma distribution characterized by a `shape` and `scale` parameters. The JAGS part uses a 1/gamma distribution with a shape and rate parameter. `"beta"` for a beta distribution characterized by an `alpha` and `beta` parameters. `"exp"` for an exponential distribution characterized by either `rate` or `scale` parameter. The later is internally converted to `rate`. `"uniform"` for a uniform distribution defined on a range from `a` to `b`
`parameters`	list of appropriate parameters for a given `distribution`.
`truncation`	list with two elements, `lower` and `upper`, that define the lower and upper truncation of the distribution. Defaults to `list(lower = -Inf, upper = Inf)`. The truncation is automatically set to the bounds of the support.
`prior_weights`	prior odds associated with a given distribution. The value is passed into the model fitting function, which creates models corresponding to all combinations of prior distributions for each of the model parameters and sets the model priors odds to the product of its prior distributions.

Value

`prior` and `prior_none` return an object of class 'prior'. A named list containing the distribution name, parameters, and prior weights.

See Also

`plot.prior()`, Normal, Lognormal, Cauchy, Beta, Exponential, LocationScaleT, InvGamma.

Examples

```
# create a standard normal prior distribution
p1 <- prior(distribution = "normal", parameters = list(mean = 1, sd = 1))

# create a half-normal standard normal prior distribution
p2 <- prior(distribution = "normal", parameters = list(mean = 1, sd = 1),
truncation = list(lower = 0, upper = Inf))

# the prior distribution can be visualized using the plot function
# (see ?plot.prior for all options)
plot(p1)
```

`prior_factor` *Creates a prior distribution for factors*

Description

`prior_factor` creates a prior distribution for fitting models with factor predictors. (Note that results across different operating systems might vary due to differences in JAGS numerical precision.)

Usage

```
prior_factor(
  distribution,
  parameters,
  truncation = list(lower = -Inf, upper = Inf),
  prior_weights = 1,
  contrast = "meandif"
)
```

Arguments

`distribution`	name of the prior distribution. The possible options are `"point"` for a point density characterized by a `location` parameter. `"normal"` for a normal distribution characterized by a `mean` and `sd` parameters. `"lognormal"` for a lognormal distribution characterized by a `meanlog` and `sdlog` parameters.

	`"cauchy"` for a Cauchy distribution characterized by a `location` and `scale` parameters. Internally converted into a generalized t-distribution with `df = 1`. `"t"` for a generalized t-distribution characterized by a `location`, `scale`, and `df` parameters. `"gamma"` for a gamma distribution characterized by either `shape` and `rate`, or `shape` and `scale` parameters. The later is internally converted to the `shape` and `rate` parametrization `"invgamma"` for an inverse-gamma distribution characterized by a `shape` and `scale` parameters. The JAGS part uses a 1/gamma distribution with a shape and rate parameter. `"beta"` for a beta distribution characterized by an `alpha` and `beta` parameters. `"exp"` for an exponential distribution characterized by either `rate` or `scale` parameter. The later is internally converted to `rate`. `"uniform"` for a uniform distribution defined on a range from `a` to `b`
`parameters`	list of appropriate parameters for a given `distribution`.
`truncation`	list with two elements, `lower` and `upper`, that define the lower and upper truncation of the distribution. Defaults to `list(lower = -Inf, upper = Inf)`. The truncation is automatically set to the bounds of the support.
`prior_weights`	prior odds associated with a given distribution. The value is passed into the model fitting function, which creates models corresponding to all combinations of prior distributions for each of the model parameters and sets the model priors odds to the product of its prior distributions.
`contrast`	type of contrast for the prior distribution. The possible options are `"meandif"` for contrast centered around the grand mean with equal marginal distributions, making the prior distribution exchangeable across factor levels. In contrast to `"orthonormal"`, the marginal distributions are identical regardless of the number of factor levels and the specified prior distribution corresponds to the difference from grand mean for each factor level. Only supports `distribution = "mnormal"` and `distribution = "mt"` which generates the corresponding multivariate normal/t distributions. `"orthonormal"` for contrast centered around the grand mean with equal marginal distributions, making the prior distribution exchangeable across factor levels. Only supports `distribution = "mnormal"` and `distribution = "mt"` which generates the corresponding multivariate normal/t distributions. `"treatment"` for contrasts using the first level as a comparison group and setting equal prior distribution on differences between the individual factor levels and the comparison level. `"independent"` for contrasts specifying dependent prior distribution for each factor level (note that this leads to an overparameterized model if the intercept is included).

Value

return an object of class 'prior'.

See Also

`prior()`

Examples

```
# create an orthonormal prior distribution
p1 <- prior_factor(distribution = "mnormal", contrast = "orthonormal",
                   parameters = list(mean = 0, sd = 1))
```

`prior_informed` *Creates an informed prior distribution based on research*

Description

`prior_informed` creates an informed prior distribution based on past research. The prior can be visualized by the `plot` function.

Usage

```
prior_informed(name, parameter = NULL, type = "smd")
```

Arguments

`name`	name of the prior distribution. There are many options based on prior psychological or medical research. For psychology, the possible options are `"van Erp"` for an informed prior distribution for the heterogeneity parameter tau of meta-analytic effect size estimates based on standardized mean differences (van Erp et al. 2017), `"Oosterwijk"` for an informed prior distribution for the effect sizes expected in social psychology based on prior elicitation with dr. Oosterwijk (Gronau et al. 2017). For medicine, the possible options are based on Bartoš et al. (2021) and Bartoš et al. (2023) who developed empirical prior distributions for the effect size and heterogeneity parameters of the continuous outcomes (standardized mean differences), dichotomous outcomes (logOR, logRR, and risk differences), and time to event outcomes (logHR) based on the Cochrane database of systematic reviews. Use `"Cochrane"` for a prior distribution based on the whole database or call `print(prior_informed_medicine_names)` to inspect the names of all 46 subfields and set the appropriate `parameter` and `type`.
`parameter`	parameter name describing what prior distribution is supposed to be produced in cases where the `name` corresponds to multiple prior distributions. Relevant only for the empirical medical prior distributions.
`type`	prior type describing what prior distribution is supposed to be produced in cases where the `name` and `parameter` correspond to multiple prior distributions. Relevant only for the empirical medical prior distributions with the following options

`"smd"` for standardized mean differences
`"logOR"` for log odds ratios
`"logRR"` for log risk ratios
`"RD"` for risk differences
`"logHR"` for hazard ratios

Details

Further details can be found in van Erp et al. (2017), Gronau et al. (2017), and Bartoš et al. (2021).

Value

`prior_informed` returns 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.

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.

See Also

`prior()`, prior_informed_medicine_names

Examples

```
# prior distribution representing expected effect sizes in social psychology
# based on prior elicitation with dr. Oosterwijk
p1 <- prior_informed("Oosterwijk")

# the prior distribution can be visualized using the plot function
# (see ?plot.prior for all options)
plot(p1)


# empirical prior distribution for the standardized mean differences from the oral health
# medical subfield based on meta-analytic effect size estimates from the
# Cochrane database of systematic reviews
p2 <- prior_informed("Oral Health", parameter ="effect", type ="smd")
print(p2)
```

`prior_none` *Creates a prior distribution*

Description

`prior` creates a prior distribution. The prior can be visualized by the `plot` function.

Usage

```
prior_none(prior_weights = 1)
```

Arguments

`prior_weights`	prior odds associated with a given distribution. The value is passed into the model fitting function, which creates models corresponding to all combinations of prior distributions for each of the model parameters and sets the model priors odds to the product of its prior distributions.

Value

`prior` and `prior_none` return an object of class 'prior'. A named list containing the distribution name, parameters, and prior weights.

See Also

`plot.prior()`, Normal, Lognormal, Cauchy, Beta, Exponential, LocationScaleT, InvGamma.

Examples

```
# create a standard normal prior distribution
p1 <- prior(distribution = "normal", parameters = list(mean = 1, sd = 1))

# create a half-normal standard normal prior distribution
p2 <- prior(distribution = "normal", parameters = list(mean = 1, sd = 1),
truncation = list(lower = 0, upper = Inf))

# the prior distribution can be visualized using the plot function
# (see ?plot.prior for all options)
plot(p1)
```

`prior_PEESE` *Creates a prior distribution for PET or PEESE models*

Description

`prior` creates a prior distribution for fitting a PET or PEESE style models in RoBMA. The prior distribution can be visualized by the `plot` function.

Usage

```
prior_PEESE(
  distribution,
  parameters,
  truncation = list(lower = 0, upper = Inf),
  prior_weights = 1
)
```

Arguments

`distribution`	name of the prior distribution. The possible options are `"point"` for a point density characterized by a `location` parameter. `"normal"` for a normal distribution characterized by a `mean` and `sd` parameters. `"lognormal"` for a lognormal distribution characterized by a `meanlog` and `sdlog` parameters. `"cauchy"` for a Cauchy distribution characterized by a `location` and `scale` parameters. Internally converted into a generalized t-distribution with `df = 1`. `"t"` for a generalized t-distribution characterized by a `location`, `scale`, and `df` parameters. `"gamma"` for a gamma distribution characterized by either `shape` and `rate`, or `shape` and `scale` parameters. The later is internally converted to the `shape` and `rate` parametrization `"invgamma"` for an inverse-gamma distribution characterized by a `shape` and `scale` parameters. The JAGS part uses a 1/gamma distribution with a shape and rate parameter. `"beta"` for a beta distribution characterized by an `alpha` and `beta` parameters. `"exp"` for an exponential distribution characterized by either `rate` or `scale` parameter. The later is internally converted to `rate`. `"uniform"` for a uniform distribution defined on a range from `a` to `b`
`parameters`	list of appropriate parameters for a given `distribution`.
`truncation`	list with two elements, `lower` and `upper`, that define the lower and upper truncation of the distribution. Defaults to `list(lower = -Inf, upper = Inf)`. The truncation is automatically set to the bounds of the support.

`prior_weights`	prior odds associated with a given distribution. The value is passed into the model fitting function, which creates models corresponding to all combinations of prior distributions for each of the model parameters and sets the model priors odds to the product of its prior distributions.

Value

`prior_PET` and `prior_PEESE` return an object of class 'prior'.

See Also

`plot.prior()`, `prior()`

Examples

```
# create a half-Cauchy prior distribution
# (PET and PEESE specific functions automatically set lower truncation at 0)
p1 <- prior_PET(distribution = "Cauchy", parameters = list(location = 0, scale = 1))

plot(p1)
```

`prior_PET` *Creates a prior distribution for PET or PEESE models*

Description

`prior` creates a prior distribution for fitting a PET or PEESE style models in RoBMA. The prior distribution can be visualized by the `plot` function.

Usage

```
prior_PET(
  distribution,
  parameters,
  truncation = list(lower = 0, upper = Inf),
  prior_weights = 1
)
```

Arguments

`distribution` name of the prior distribution. The possible options are

- `"point"` for a point density characterized by a `location` parameter.
- `"normal"` for a normal distribution characterized by a `mean` and `sd` parameters.
- `"lognormal"` for a lognormal distribution characterized by a `meanlog` and `sdlog` parameters.
- `"cauchy"` for a Cauchy distribution characterized by a `location` and `scale` parameters. Internally converted into a generalized t-distribution with `df = 1`.

	`"t"` for a generalized t-distribution characterized by a `location`, `scale`, and `df` parameters. `"gamma"` for a gamma distribution characterized by either `shape` and `rate`, or `shape` and `scale` parameters. The later is internally converted to the `shape` and `rate` parametrization `"invgamma"` for an inverse-gamma distribution characterized by a `shape` and `scale` parameters. The JAGS part uses a 1/gamma distribution with a shape and rate parameter. `"beta"` for a beta distribution characterized by an `alpha` and `beta` parameters. `"exp"` for an exponential distribution characterized by either `rate` or `scale` parameter. The later is internally converted to `rate`. `"uniform"` for a uniform distribution defined on a range from `a` to `b`
`parameters`	list of appropriate parameters for a given `distribution`.
`truncation`	list with two elements, `lower` and `upper`, that define the lower and upper truncation of the distribution. Defaults to `list(lower = -Inf, upper = Inf)`. The truncation is automatically set to the bounds of the support.
`prior_weights`	prior odds associated with a given distribution. The value is passed into the model fitting function, which creates models corresponding to all combinations of prior distributions for each of the model parameters and sets the model priors odds to the product of its prior distributions.

Value

`prior_PET` and `prior_PEESE` return an object of class 'prior'.

See Also

`plot.prior()`, `prior()`

Examples

```
# create a half-Cauchy prior distribution
# (PET and PEESE specific functions automatically set lower truncation at 0)
p1 <- prior_PET(distribution = "Cauchy", parameters = list(location = 0, scale = 1))

plot(p1)
```

`prior_weightfunction` *Creates a prior distribution for a weight function*

Description

`prior_weightfunction` creates a prior distribution for fitting a RoBMA selection model. The prior can be visualized by the `plot` function.

Usage

```
prior_weightfunction(distribution, parameters, prior_weights = 1)
```

Arguments

`distribution`	name of the prior distribution. The possible options are `"two.sided"` for a two-sided weight function characterized by a vector `steps` and vector `alpha` parameters. The `alpha` parameter determines an alpha parameter of Dirichlet distribution which cumulative sum is used for the weights omega. `"one.sided"` for a one-sided weight function characterized by either a vector `steps` and vector `alpha` parameter, leading to a monotonic one-sided function, or by a vector `steps`, vector `alpha1`, and vector `alpha2` parameters leading non-monotonic one-sided weight function. The `alpha` / `alpha1` and `alpha2` parameters determine an alpha parameter of Dirichlet distribution which cumulative sum is used for the weights omega.
`parameters`	list of appropriate parameters for a given `distribution`.
`prior_weights`	prior odds associated with a given distribution. The model fitting function usually creates models corresponding to all combinations of prior distributions for each of the model parameters, and sets the model priors odds to the product of its prior distributions.

Details

Constrained cases of weight functions can be specified by adding ".fixed" after the distribution name, i.e., `"two.sided.fixed"` and `"one.sided.fixed"`. In these cases, the functions are specified using `steps` and `omega` parameters, where the `omega` parameter is a vector of weights that corresponds to the relative publication probability (i.e., no parameters are estimated).

Value

`prior_weightfunction` returns an object of class 'prior'.

See Also

`plot.prior()`

Examples

```
p1 <- prior_weightfunction("one-sided", parameters = list(steps = c(.05, .10), alpha = c(1, 1, 1)))

# the prior distribution can be visualized using the plot function
# (see ?plot.prior for all options)
plot(p1)
```

RoBMA *Estimate a Robust Bayesian Meta-Analysis*

Description

`RoBMA` is used to estimate a robust Bayesian meta-analysis. The interface allows a complete customization of the ensemble with different prior (or list of prior) distributions for each component.

Usage

```
RoBMA(
  d = NULL,
  r = NULL,
  logOR = NULL,
  OR = NULL,
  z = NULL,
  y = NULL,
  se = NULL,
  v = NULL,
  n = NULL,
  lCI = NULL,
  uCI = NULL,
  t = NULL,
  study_names = NULL,
  study_ids = NULL,
  data = NULL,
  weight = NULL,
  transformation = if (is.null(y)) "fishers_z" else "none",
  prior_scale = if (is.null(y)) "cohens_d" else "none",
  effect_direction = "positive",
  model_type = NULL,
  rescale_priors = 1,
  priors_effect = set_default_priors("effect", rescale = rescale_priors),
 priors_heterogeneity = set_default_priors("heterogeneity", rescale = rescale_priors),
  priors_bias = set_default_priors("bias", rescale = rescale_priors),
  priors_effect_null = set_default_priors("effect", null = TRUE),
  priors_heterogeneity_null = set_default_priors("heterogeneity", null = TRUE),
  priors_bias_null = set_default_priors("bias", null = TRUE),
  priors_hierarchical = set_default_priors("hierarchical"),
  priors_hierarchical_null = set_default_priors("hierarchical", null = TRUE),
  algorithm = "bridge",
  chains = 3,
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  thin = 1,
  parallel = FALSE,
```

```
  autofit = TRUE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  save = "all",
  seed = NULL,
  silent = TRUE,
  ...
)
```

Arguments

`d`	a vector of effect sizes measured as Cohen's d
`r`	a vector of effect sizes measured as correlations
`logOR`	a vector of effect sizes measured as log odds ratios
`OR`	a vector of effect sizes measured as odds ratios
`z`	a vector of effect sizes measured as Fisher's z
`y`	a vector of unspecified effect sizes (note that effect size transformations are unavailable with this type of input)
`se`	a vector of standard errors of the effect sizes
`v`	a vector of variances of the effect sizes
`n`	a vector of overall sample sizes
`lCI`	a vector of lower bounds of confidence intervals
`uCI`	a vector of upper bounds of confidence intervals
`t`	a vector of t/z-statistics
`study_names`	an optional argument with the names of the studies
`study_ids`	an optional argument specifying dependency between the studies (for using a multilevel model). Defaults to `NULL` for studies being independent.
`data`	a data object created by the `combine_data` function. This is an alternative input entry to specifying the `d`, `r`, `y`, etc... directly. I.e., RoBMA function does not allow passing a data.frame and referencing to the columns.
`weight`	specifies likelihood weights of the individual estimates. Notes that this is an untested experimental feature.
`transformation`	transformation to be applied to the supplied effect sizes before fitting the individual models. Defaults to `"fishers_z"`. We highly recommend using `"fishers_z"` transformation since it is the only variance stabilizing measure and does not bias PET and PEESE style models. The other options are `"cohens_d"`, correlation coefficient `"r"` and `"logOR"`. Supplying `"none"` will treat the effect sizes as unstandardized and refrain from any transformations.
`prior_scale`	an effect size scale used to define priors. Defaults to `"cohens_d"`. Other options are `"fishers_z"`, correlation coefficient `"r"`, and `"logOR"`. The prior scale does not need to match the effect sizes measure - the samples from prior distributions are internally transformed to match the `transformation` of the data. The `prior_scale` corresponds to the effect size scale of default output, but can be changed within the summary function.

`effect_direction`	the expected direction of the effect. Correctly specifying the expected direction of the effect is crucial for one-sided selection models, as they specify cut-offs using one-sided p-values. Defaults to `"positive"` (another option is `"negative"`).
`model_type`	string specifying the RoBMA ensemble. Defaults to `NULL`. The other options are `"PSMA"`, `"PP"`, and `"2w"` which override settings passed to the `priors_effect`, `priors_heterogeneity`, `priors_effect`, `priors_effect_null`, `priors_heterogeneity_null`, `priors_bias_null`, and `priors_effect`. See details for more information about the different model types.
`rescale_priors`	a re-scaling factor for the prior distributions. The re-scaling factor allows to adjust the width of all default priors simultaneously. Defaults to `1`.
`priors_effect`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the alternative hypothesis. Defaults to a standard normal distribution `prior(distribution = "normal", parameters = list(mean = 0, sd = 1))`.
`priors_heterogeneity`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the alternative hypothesis. Defaults to `prior(distribution = "invgamma", parameters = list(shape = 1, scale = .15))` that is based on heterogeneities estimates from psychology (van Erp et al. 2017).
`priors_bias`	list of prior distributions for the publication bias adjustment component that will be treated as belonging to the alternative hypothesis. Defaults to `list(prior_weightfunction(distribution = "two.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_weights = 1/12),prior_weightfunction(distribution = "two.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.05, 0.10)), prior_weights = 1/12),prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_weights = 1/12),prior_weightfu = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.025, 0.05)), prior_weights = 1/12),prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.05, 0.5)), prior_weights = 1/12),prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1, 1), steps = c(0.025, 0.05, 0.5)), prior_weights = 1/12),prior_PET(distribution = "Cauchy", parameters = list(0,1), truncation = list(0, Inf), prior_weights = 1/4),prior_PEESE(distribution = "Cauchy", parameters = list(0,5), truncation = list(0, Inf), prior_weights = 1/4))`, corresponding to the RoBMA-PSMA model introduce by Bartoš et al. (2023).
`priors_effect_null`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero, `prior(distribution = "point", parameters = list(location = 0))`.
`priors_heterogeneity_null`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero (a fixed effect meta-analytic models), `prior(distribution = "point", parameters = list(location = 0))`.
`priors_bias_null`	list of prior weight functions for the `omega` parameter that will be treated as belonging to the null hypothesis. Defaults no publication bias adjustment, `prior_none()`.

`priors_hierarchical`	list of prior distributions for the correlation of random effects (`rho`) parameter that will be treated as belonging to the alternative hypothesis. This setting allows users to fit a hierarchical (three-level) meta-analysis when `study_ids` are supplied. Note that this is an experimental feature and see News for more details. Defaults to a beta distribution `prior(distribution = "beta", parameters = list(alpha = 1, beta = 1))`.
`priors_hierarchical_null`	list of prior distributions for the correlation of random effects (`rho`) parameter that will be treated as belonging to the null hypothesis. Defaults to `NULL`.
`algorithm`	a string specifying the algorithm used for the model averaging. Defaults to `"bridge"` which results in estimating individual models using JAGS and computing the marginal likelihood using bridge sampling. An alternative is `"ss"` which uses spike and slab like parameterization to approximate the Bayesian model averaging with a single model. Note that significantly more `sample`, `burnin`, and `adapt` iterations are needed for the `"ss"` algorithm.
`chains`	a number of chains of the MCMC algorithm.
`sample`	a number of sampling iterations of the MCMC algorithm. Defaults to `5000`.
`burnin`	a number of burnin iterations of the MCMC algorithm. Defaults to `2000`.
`adapt`	a number of adaptation iterations of the MCMC algorithm. Defaults to `500`.
`thin`	a thinning of the chains of the MCMC algorithm. Defaults to `1`.
`parallel`	whether the individual models should be fitted in parallel. Defaults to `FALSE`. The implementation is not completely stable and might cause a connection error.
`autofit`	whether the model should be fitted until the convergence criteria (specified in `autofit_control`) are satisfied. Defaults to `TRUE`.
`autofit_control`	allows to pass autofit control settings with the `set_autofit_control()` function. See `?set_autofit_control` for options and default settings.
`convergence_checks`	automatic convergence checks to assess the fitted models, passed with `set_convergence_checks()` function. See `?set_convergence_checks` for options and default settings.
`save`	whether all models posterior distributions should be kept after obtaining a model-averaged result. Defaults to `"all"` which does not remove anything. Set to `"min"` to significantly reduce the size of final object, however, some model diagnostics and further manipulation with the object will not be possible.
`seed`	a seed to be set before model fitting, marginal likelihood computation, and posterior mixing for reproducibility of results. Defaults to `NULL` - no seed is set.
`silent`	whether all print messages regarding the fitting process should be suppressed. Defaults to `TRUE`. Note that `parallel = TRUE` also suppresses all messages.
`...`	additional arguments.

Details

The default settings of the RoBMA 2.0 package corresponds to the RoBMA-PSMA ensemble proposed by Bartoš et al. (2023). The previous versions of the package (i.e., RoBMA < 2.0) used specifications proposed by Maier et al. (2023) (this specification can be easily obtained by setting `model_type = "2w"`. The RoBMA-PP specification from Bartoš et al. (2023) can be obtained by setting `model_type = "PP"`. The complete list of default prior distributions is described at `set_default_priors()`. Note that inclusion of the PET and PEESE style publication bias adjustments models might pick up on small-study effects. To remove true heterogeneity due to study design, sub-populations, treatments etc. potentially causing small-study effects, use meta-regression via the `RoBMA.reg()` function, or remove the PET and PEESE style models from the publication bias adjustment component of the ensemble.

The `vignette("CustomEnsembles", package = "RoBMA")` and `vignette("ReproducingBMA", package = "RoBMA")` vignettes describe how to use `RoBMA()` to fit custom meta-analytic ensembles (see `prior()`, `prior_weightfunction()`, `prior_PET()`, and `prior_PEESE()` for more information about prior distributions).

The RoBMA function first generates models from a combination of the provided priors for each of the model parameters. Then, the individual models are fitted using autorun.jags function. A marginal likelihood is computed using bridge_sampler function. The individual models are then combined into an ensemble using the posterior model probabilities using BayesTools package.

Generic `summary.RoBMA()`, `print.RoBMA()`, and `plot.RoBMA()` functions are provided to facilitate manipulation with the ensemble. A visual check of the individual model diagnostics can be obtained using the `diagnostics()` function. The fitted model can be further updated or modified by `update.RoBMA()` function.

Value

`RoBMA` returns an object of class 'RoBMA'.

References

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.

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.

See Also

`summary.RoBMA()`, `update.RoBMA()`, `check_setup()`

Examples

```
## Not run:
# using the example data from Bem 2011 and fitting the default (RoBMA-PSMA) model
fit <- RoBMA(d = Bem2011$d, se = Bem2011$se, study_names = Bem2011$study)

# in order to speed up the process, we can turn the parallelization on
```

```
fit <- RoBMA(d = Bem2011$d, se = Bem2011$se, study_names = Bem2011$study, parallel = TRUE)

# we can get a quick overview of the model coefficients just by printing the model
fit

# a more detailed overview using the summary function (see '?summary.RoBMA' for all options)
summary(fit)

# the model-averaged effect size estimate can be visualized using the plot function
# (see ?plot.RoBMA for all options)
plot(fit, parameter = "mu")

# forest plot can be obtained with the forest function (see ?forest for all options)
forest(fit)

# plot of the individual model estimates can be obtained with the plot_models function
#  (see ?plot_models for all options)
plot_models(fit)

# diagnostics for the individual parameters in individual models can be obtained using diagnostics
# function (see 'diagnostics' for all options)
diagnostics(fit, parameter = "mu", type = "chains")

# the RoBMA-PP can be fitted with addition of the 'model_type' argument
fit_PP <- RoBMA(d = Bem2011$d, se = Bem2011$se, study_names = Bem2011$study, model_type = "PP")

# as well as the original version of RoBMA (with two weightfunctions)
fit_original <- RoBMA(d = Bem2011$d, se = Bem2011$se, study_names = Bem2011$study,
                      model_type = "2w")

# or different prior distribution for the effect size (e.g., a half-normal distribution)
# (see 'vignette("CustomEnsembles")' for a detailed guide on specifying a custom model ensemble)
fit <- RoBMA(d = Bem2011$d, se = Bem2011$se, study_names = Bem2011$study,
             priors_effect = prior("normal", parameters = list(0, 1),
                                   truncation = list(0, Inf)))

## End(Not run)
```

`RoBMA.reg` *Estimate a Robust Bayesian Meta-Analysis Meta-Regression*

Description

`RoBMA` is used to estimate a robust Bayesian meta-regression. The interface allows a complete customization of the ensemble with different prior (or list of prior) distributions for each component.

Usage

```
RoBMA.reg(
```

```
  formula,
  data,
  test_predictors = TRUE,
  study_names = NULL,
  study_ids = NULL,
  transformation = if (any(colnames(data) != "y")) "fishers_z" else "none",
  prior_scale = if (any(colnames(data) != "y")) "cohens_d" else "none",
  standardize_predictors = TRUE,
  effect_direction = "positive",
  priors = NULL,
  model_type = NULL,
  rescale_priors = 1,
  priors_effect = set_default_priors("effect", rescale = rescale_priors),
 priors_heterogeneity = set_default_priors("heterogeneity", rescale = rescale_priors),
  priors_bias = set_default_priors("bias", rescale = rescale_priors),
  priors_effect_null = set_default_priors("effect", null = TRUE),
  priors_heterogeneity_null = set_default_priors("heterogeneity", null = TRUE),
  priors_bias_null = set_default_priors("bias", null = TRUE),
  priors_hierarchical = set_default_priors("hierarchical"),
  priors_hierarchical_null = set_default_priors("hierarchical", null = TRUE),
 prior_covariates = set_default_priors("covariates", rescale = rescale_priors),
  prior_covariates_null = set_default_priors("covariates", null = TRUE),
  prior_factors = set_default_priors("factors", rescale = rescale_priors),
  prior_factors_null = set_default_priors("factors", null = TRUE),
  algorithm = "bridge",
  chains = 3,
  sample = 5000,
  burnin = 2000,
  adapt = 500,
  thin = 1,
  parallel = FALSE,
  autofit = TRUE,
  autofit_control = set_autofit_control(),
  convergence_checks = set_convergence_checks(),
  save = "all",
  seed = NULL,
  silent = TRUE,
  ...
)
```

Arguments

`formula`	a formula for the meta-regression model
`data`	a data.frame containing the data for the meta-regression. Note that the column names have to correspond to the effect sizes (`d`, `logOR`, `OR`, `r`, `z`), a measure of sampling variability (`se`, `v`, `n`, `lCI`, `uCI`, `t`), and the predictors. See `combine_data()` for a complete list of reserved names and additional information about specifying input data.

`test_predictors`	vector of predictor names to test for the presence of moderation (i.e., assigned both the null and alternative prior distributions). Defaults to `TRUE`, all predictors are tested using the default prior distributions (i.e., `prior_covariates`, `prior_covariates_null`, `prior_factors`, and `prior_factors_null`). To only estimate and adjust for the effect of predictors use `FALSE`. If `priors` is specified, any settings in `test_predictors` is overridden.
`study_names`	an optional argument with the names of the studies
`study_ids`	an optional argument specifying dependency between the studies (for using a multilevel model). Defaults to `NULL` for studies being independent.
`transformation`	transformation to be applied to the supplied effect sizes before fitting the individual models. Defaults to `"fishers_z"`. We highly recommend using `"fishers_z"` transformation since it is the only variance stabilizing measure and does not bias PET and PEESE style models. The other options are `"cohens_d"`, correlation coefficient `"r"` and `"logOR"`. Supplying `"none"` will treat the effect sizes as unstandardized and refrain from any transformations.
`prior_scale`	an effect size scale used to define priors. Defaults to `"cohens_d"`. Other options are `"fishers_z"`, correlation coefficient `"r"`, and `"logOR"`. The prior scale does not need to match the effect sizes measure - the samples from prior distributions are internally transformed to match the `transformation` of the data. The `prior_scale` corresponds to the effect size scale of default output, but can be changed within the summary function.
`standardize_predictors`	whether continuous predictors should be standardized prior to estimating the model. Defaults to `TRUE`. Continuous predictor standardization is important for applying the default prior distributions for continuous predictors. Note that the resulting output corresponds to standardized meta-regression coefficients.
`effect_direction`	the expected direction of the effect. Correctly specifying the expected direction of the effect is crucial for one-sided selection models, as they specify cut-offs using one-sided p-values. Defaults to `"positive"` (another option is `"negative"`).
`priors`	named list of prior distributions for each predictor (with names corresponding to the predictors). It allows users to specify both the null and alternative hypothesis prior distributions for each predictor by assigning the corresponding element of the named list with another named list (with `"null"` and `"alt"`). If only one prior is specified for a given parameter, it is assumed to correspond to the alternative hypotheses and the default null hypothesis is specified (i.e., `prior_covariates_null` or `prior_factors_null`). If a named list with only one named prior distribution is provided (either `"null"` or `"alt"`), only this prior distribution is used and no default distribution is filled in. Parameters without specified prior distributions are assumed to be only adjusted for using the default alternative hypothesis prior distributions (i.e., `prior_covariates` or `prior_factors`). If `priors` is specified, `test_predictors` is ignored.
`model_type`	string specifying the RoBMA ensemble. Defaults to `NULL`. The other options are `"PSMA"`, `"PP"`, and `"2w"` which override settings passed to the `priors_effect`, `priors_heterogeneity`, `priors_effect`, `priors_effect_null`, `priors_heterogeneity_null`,

	`priors_bias_null`, and `priors_effect`. See details for more information about the different model types.
`rescale_priors`	a re-scaling factor for the prior distributions. The re-scaling factor allows to adjust the width of all default priors simultaneously. Defaults to `1`.
`priors_effect`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the alternative hypothesis. Defaults to a standard normal distribution `prior(distribution = "normal", parameters = list(mean = 0, sd = 1))`.
`priors_heterogeneity`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the alternative hypothesis. Defaults to `prior(distribution = "invgamma", parameters = list(shape = 1, scale = .15))` that is based on heterogeneities estimates from psychology (van Erp et al. 2017).
`priors_bias`	list of prior distributions for the publication bias adjustment component that will be treated as belonging to the alternative hypothesis. Defaults to `list(prior_weightfunction(distribution = "two.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_weights = 1/12),prior_weightfunction(distribution = "two.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.05, 0.10)), prior_weights = 1/12),prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_weights = 1/12),prior_weightfu = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.025, 0.05)), prior_weights = 1/12),prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.05, 0.5)), prior_weights = 1/12),prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1, 1), steps = c(0.025, 0.05, 0.5)), prior_weights = 1/12),prior_PET(distribution = "Cauchy", parameters = list(0,1), truncation = list(0, Inf), prior_weights = 1/4),prior_PEESE(distribution = "Cauchy", parameters = list(0,5), truncation = list(0, Inf), prior_weights = 1/4))`, corresponding to the RoBMA-PSMA model introduce by Bartoš et al. (2023).
`priors_effect_null`	list of prior distributions for the effect size (`mu`) parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero, `prior(distribution = "point", parameters = list(location = 0))`.
`priors_heterogeneity_null`	list of prior distributions for the heterogeneity `tau` parameter that will be treated as belonging to the null hypothesis. Defaults to a point null hypotheses at zero (a fixed effect meta-analytic models), `prior(distribution = "point", parameters = list(location = 0))`.
`priors_bias_null`	list of prior weight functions for the `omega` parameter that will be treated as belonging to the null hypothesis. Defaults no publication bias adjustment, `prior_none()`.
`priors_hierarchical`	list of prior distributions for the correlation of random effects (`rho`) parameter that will be treated as belonging to the alternative hypothesis. This setting allows users to fit a hierarchical (three-level) meta-analysis when `study_ids` are supplied. Note that this is an experimental feature and see News for more details. Defaults to a beta distribution `prior(distribution = "beta", parameters = list(alpha = 1, beta = 1))`.

`priors_hierarchical_null`	list of prior distributions for the correlation of random effects (`rho`) parameter that will be treated as belonging to the null hypothesis. Defaults to `NULL`.
`prior_covariates`	a prior distributions for the regression parameter of continuous covariates on the effect size under the alternative hypothesis (unless set explicitly in `priors`). Defaults to a relatively wide normal distribution `prior(distribution = "normal", parameters = list(mean = 0, sd = 0.25))`.
`prior_covariates_null`	a prior distributions for the regression parameter of continuous covariates on the effect size under the null hypothesis (unless set explicitly in `priors`). Defaults to a no effect `prior("spike", parameters = list(location = 0))`.
`prior_factors`	a prior distributions for the regression parameter of categorical covariates on the effect size under the alternative hypothesis (unless set explicitly in `priors`). Defaults to a relatively wide multivariate normal distribution specifying differences from the mean contrasts `prior_factor("mnormal", parameters = list(mean = 0, sd = 0.25), contrast = "meandif")`.
`prior_factors_null`	a prior distributions for the regression parameter of categorical covariates on the effect size under the null hypothesis (unless set explicitly in `priors`). Defaults to a no effect `prior("spike", parameters = list(location = 0))`.
`algorithm`	a string specifying the algorithm used for the model averaging. Defaults to `"bridge"` which results in estimating individual models using JAGS and computing the marginal likelihood using bridge sampling. An alternative is `"ss"` which uses spike and slab like parameterization to approximate the Bayesian model averaging with a single model. Note that significantly more `sample`, `burnin`, and `adapt` iterations are needed for the `"ss"` algorithm.
`chains`	a number of chains of the MCMC algorithm.
`sample`	a number of sampling iterations of the MCMC algorithm. Defaults to `5000`.
`burnin`	a number of burnin iterations of the MCMC algorithm. Defaults to `2000`.
`adapt`	a number of adaptation iterations of the MCMC algorithm. Defaults to `500`.
`thin`	a thinning of the chains of the MCMC algorithm. Defaults to `1`.
`parallel`	whether the individual models should be fitted in parallel. Defaults to `FALSE`. The implementation is not completely stable and might cause a connection error.
`autofit`	whether the model should be fitted until the convergence criteria (specified in `autofit_control`) are satisfied. Defaults to `TRUE`.
`autofit_control`	allows to pass autofit control settings with the `set_autofit_control()` function. See `?set_autofit_control` for options and default settings.
`convergence_checks`	automatic convergence checks to assess the fitted models, passed with `set_convergence_checks()` function. See `?set_convergence_checks` for options and default settings.
`save`	whether all models posterior distributions should be kept after obtaining a model-averaged result. Defaults to `"all"` which does not remove anything. Set to `"min"` to significantly reduce the size of final object, however, some model diagnostics and further manipulation with the object will not be possible.

`seed`	a seed to be set before model fitting, marginal likelihood computation, and posterior mixing for reproducibility of results. Defaults to `NULL` - no seed is set.
`silent`	whether all print messages regarding the fitting process should be suppressed. Defaults to `TRUE`. Note that `parallel = TRUE` also suppresses all messages.
`...`	additional arguments.

Details

The `vignette("/MetaRegression", package = "RoBMA")` vignette describes how to use `RoBMA.reg()` function to fit Bayesian meta-regression ensembles. See Bartoš et al. (2023) for more details about the methodology and `RoBMA()` for more details about the function options. By default, the function standardizes continuous predictors. As such, the output should be interpreted as standardized meta-regression coefficients.

The RoBMA.reg function first generates models from a combination of the provided priors for each of the model parameters. Then, the individual models are fitted using autorun.jags function. A marginal likelihood is computed using bridge_sampler function. The individual models are then combined into an ensemble using the posterior model probabilities using BayesTools package.

Generic `summary.RoBMA()`, `print.RoBMA()`, and `plot.RoBMA()` functions are provided to facilitate manipulation with the ensemble. A visual check of the individual model diagnostics can be obtained using the `diagnostics()` function. The fitted model can be further updated or modified by `update.RoBMA()` function. Estimated marginal means can be computed by `marginal_summary()` function and visualized by the `marginal_plot()` function.

Value

`RoBMA.reg` returns an object of class 'RoBMA.reg'.

References

Bartoš F, Maier M, Stanley TD, Wagenmakers E (2023). “Robust Bayesian meta-regression: Model-averaged moderation analysis in the presence of publication bias.” doi:10.31234/osf.io/98xb5, Preprint available at https://doi.org/10.31234/osf.io/98xb5.

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.

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.

See Also

`RoBMA()` `summary.RoBMA()`, `update.RoBMA()`, `check_setup.reg()`

Examples

```
## Not run:
# using the example data from Andrews et al. (2021) and reproducing the example from
# Bartos et al. (2024) with measure and age covariate.

 # note the the Andrews2021 data.frame columns identify the effect size "r" and
 # the standard error "se" of the effect size that are used to estimate the model
 fit_RoBMA <- RoBMA.reg(~ measure + age, data = Andrews2021, parallel = TRUE, seed = 1)

 # summarize the results
 summary(fit_RoBMA, output_scale = "r")

 # compute effect size estimates for each group
 marginal_summary(fit_RoBMA, output_scale = "r")

 # visualize the effect size estimates for each group
 marginal_plot(fit_RoBMA, parameter = "measure", output_scale = "r", lwd = 2)

## End(Not run)
```

`RoBMA_control` *Control MCMC fitting process*

Description

Controls settings for the autofit process of the MCMC JAGS sampler (specifies termination criteria), and values for the 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
)

set_convergence_checks(
  max_Rhat = 1.05,
  min_ESS = 500,
  max_error = NULL,
  max_SD_error = NULL,
  remove_failed = FALSE,
  balance_probability = TRUE
)
```

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"`, `"weeks"`, `"years"`). 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`.
`remove_failed`	whether models not satisfying the convergence checks should be removed from the inference. Defaults to `FALSE` - only a warning is raised.
`balance_probability`	whether prior model probability should be balanced across the combinations of models with the same H0/H1 for effect / heterogeneity / bias in the case of non-convergence. Defaults to `TRUE`.

Value

`set_autofit_control` returns a list of autofit control settings and `set_convergence_checks` returns a list of convergence checks settings.

See Also

RoBMA, update.RoBMA

`RoBMA_options` *Options for the RoBMA package*

Description

A placeholder object and functions for the RoBMA package. (adapted from the runjags R package).

Usage

```
RoBMA.options(...)

RoBMA.get_option(name)
```

Arguments

`...`	named option(s) to change - for a list of available options, see details below.
`name`	the name of the option to get the current value of - for a list of available options, see details below.

Value

The current value of all available RoBMA options (after applying any changes specified) is returned invisibly as a named list.

`sample_sizes` *Sample sizes to standard errors calculations*

Description

Functions for transforming between standard errors and sample sizes (assuming equal sample sizes per group).

Usage

```
se_d(d, n)

n_d(d, se)

se_r(r, n)

n_r(r, se)

se_z(n)

n_z(se)
```

Arguments

`d`	Cohen's d
`n`	sample size of the corresponding effect size
`se`	standard error of the corresponding effect size
`r`	correlation coefficient

Details

Calculations for Cohen's d, Fisher's z, and log(OR) are based on (Borenstein et al. 2011). Calculations for correlation coefficient were modified to make the standard error corresponding to the computed on Fisher's z scale under the same sample size (in order to make all other transformations consistent). In case that a direct transformation is not available, the transformations are chained to provide the effect size of interest.

Note that sample size and standard error calculation for log(OR) is not available. The standard error is highly dependent on the odds within the groups and sample sizes for individual events are required. Theoretically, the sample size could be obtained by transforming the effect size and standard error to a different measure and obtaining the sample size using corresponding function, however, it leads to a very poor approximation and it is not recommended.

References

Borenstein M, Hedges LV, Higgins JP, Rothstein HR (2011). *Introduction to meta-analysis*. John Wiley & Sons.

See Also

`effect_sizes()`, `standard_errors()`

`set_default_binomial_priors` *Set default prior distributions for binomial meta-analytic models*

Description

Set default prior distributions for BiBMA models.

Usage

```
set_default_binomial_priors(parameter, null = FALSE, rescale = 1)
```

Arguments

`parameter`	a character string specifying the parameter for which the prior distribution should be set. Available options are "effect", "heterogeneity", "baseline", "covariates", "factors".
`null`	a logical indicating whether the prior distribution should be set for the null hypothesis. Defaults to `FALSE`.
`rescale`	a numeric value specifying the re-scaling factor for the default prior distributions. Defaults to 1. Allows convenient re-scaling of prior distributions simultaneously.

Details

The default prior are based on the binary outcome meta-analyses in the Cochrane Database of Systematic Reviews outlined in Bartoš et al. (2023).

Specifically, the prior distributions are:

For the alternative hypothesis:

- **Effect:** T distribution with mean 0, scale 0.58, and 4 degrees of freedom.

- **Heterogeneity:** Inverse gamma distribution with shape 1.77 and scale 0.55.
- **Baseline:** No prior distribution.
- **Standardized continuous covariates:** Normal distribution with mean 0 and standard deviation 0.29.
- **Factors (via by-level differences from the grand mean):** Normal distribution with mean 0 and standard deviation 0.29.

For the null hypothesis:

- **Effect:** Point distribution at 0.
- **Heterogeneity:** Point distribution at 0.
- **Baseline:** Independent uniform distributions.
- **Standardized continuous covariates:** Point distribution at 0.
- **Factors (via by-level differences from the grand mean):** Point distribution at 0.

The rescaling factor adjusts the width of the effect, heterogeneity, covariates, factor, and PEESE-style model prior distributions. PET-style and weight function prior distributions are scale-invariant.

Value

A prior distribution object or a list of prior distribution objects.

Examples

```
set_default_binomial_priors("effect")
set_default_binomial_priors("heterogeneity")
set_default_binomial_priors("baseline")
```

`set_default_priors` *Set default prior distributions*

Description

Set default prior distributions for RoBMA models.

Usage

```
set_default_priors(parameter, null = FALSE, rescale = 1)
```

Arguments

`parameter`	a character string specifying the parameter for which the prior distribution should be set. Available options are "effect", "heterogeneity", "bias", "hierarchical", "covariates", "factors".
`null`	a logical indicating whether the prior distribution should be set for the null hypothesis. Defaults to `FALSE`.
`rescale`	a numeric value specifying the re-scaling factor for the default prior distributions. Defaults to 1. Allows convenient re-scaling of prior distributions simultaneously.

Details

The default prior distributions corresponds to the specification of RoBMA-PSMA and RoBMA-regression outlined in Bartoš et al. (2023) and Bartoš et al. (2023).

Specifically, the prior distributions are:

For the alternative hypothesis:

- **Effect:** Normal distribution with mean 0 and standard deviation 1.
- **Heterogeneity:** Inverse gamma distribution with shape 1 and scale 0.15.
- **Bias:** A list of 8 prior distributions defining the publication bias adjustments:
  - Two-sided: Weight function with steps 0.05.
  - Two-sided: Weight function with steps 0.05 and 0.1.
  - One-sided: Weight function with steps 0.05.
  - One-sided: Weight function with steps 0.025 and 0.05.
  - One-sided: Weight function with steps 0.05 and 0.5.
  - One-sided: Weight function with steps 0.025, 0.05, and 0.5.
  - PET-type model with regression coefficient: Cauchy distribution with location 0 and scale 1.
  - PEESE-type model with regression coefficient: Cauchy distribution with location 0 and scale 5.

  All weight functions use a unit cumulative Dirichlet prior distribution on relative prior probabilities.
- **Standardized continuous covariates:** Normal distribution with mean 0 and standard deviation 0.25.
- **Factors (via by-level differences from the grand mean):** Normal distribution with mean 0 and standard deviation 0.25.

For the null hypothesis:

- **Effect:** Point distribution at 0.
- **Heterogeneity:** Point distribution at 0.
- **Bias:** No prior distribution.
- **Standardized continuous covariates:** Point distribution at 0.
- **Factors (via by-level differences from the grand mean):** Point distribution at 0.

The rescaling factor adjusts the width of the effect, heterogeneity, covariates, factor, and PEESE-style model prior distributions. PET-style and weight function prior distributions are scale-invariant.

Value

A prior distribution object or a list of prior distribution objects.

Examples

```
set_default_priors("effect")
set_default_priors("heterogeneity")
set_default_priors("bias")
```

`standard_errors` *Standard errors transformations*

Description

Functions for transforming between standard errors of different effect size measures.

Usage

```
se_d2se_logOR(se_d, logOR)

se_d2se_r(se_d, d)

se_r2se_d(se_r, r)

se_logOR2se_d(se_logOR, logOR)

se_d2se_z(se_d, d)

se_r2se_z(se_r, r)

se_r2se_logOR(se_r, r)

se_logOR2se_r(se_logOR, logOR)

se_logOR2se_z(se_logOR, logOR)

se_z2se_d(se_z, z)

se_z2se_r(se_z, z)

se_z2se_logOR(se_z, z)
```

Arguments

`se_d`	standard error of Cohen's d
`logOR`	log(odds ratios)

`d`	Cohen's d
`se_r`	standard error of correlation coefficient
`r`	correlation coefficient
`se_logOR`	standard error of log(odds ratios)
`se_z`	standard error of Fisher's z
`z`	Fisher's z

Details

Transformations for Cohen's d, Fisher's z, and log(OR) are based on (Borenstein et al. 2011). Calculations for correlation coefficient were modified to make the standard error corresponding to the computed on Fisher's z scale under the same sample size (in order to make all other transformations consistent). In case that a direct transformation is not available, the transformations are chained to provide the effect size of interest.

It is important to keep in mind that the transformations are only approximations to the true values. From our experience, `se_d2se_z` works well for values of se(Cohen's d) < 0.5. Do not forget that the effect sizes are standardized and variance of Cohen's d = 1. Therefore, a standard error of study cannot be larger unless the participants provided negative information (of course, the variance is dependent on the effect size as well, and, can therefore be larger).

When setting prior distributions, do NOT attempt to transform a standard normal distribution on Cohen's d (mean = 0, sd = 1) to a normal distribution on Fisher's z with mean 0 and sd = `se_d2se_z(0, 1)`. The approximation does NOT work well in this range of values. Instead, approximate the sd of distribution on Fisher's z using samples in this way: `sd(d2z(rnorm(10000, 0, 1)))` or, specify the distribution on Cohen's d directly.

References

Borenstein M, Hedges LV, Higgins JP, Rothstein HR (2011). *Introduction to meta-analysis*. John Wiley & Sons.

See Also

`effect_sizes()`, `sample_sizes()`

`summary.RoBMA` *Summarize fitted RoBMA object*

Description

`summary.RoBMA` creates summary tables for a RoBMA object.

Usage

```
## S3 method for class 'RoBMA'
summary(
  object,
  type = "ensemble",
  conditional = FALSE,
  output_scale = NULL,
  probs = c(0.025, 0.975),
  logBF = FALSE,
  BF01 = FALSE,
  short_name = FALSE,
  remove_spike_0 = FALSE,
  ...
)
```

Arguments

`object`	a fitted RoBMA object
`type`	whether to show the overall RoBMA results (`"ensemble"`), an overview of the individual models (`"models"`), an overview of the individual models MCMC diagnostics (`"diagnostics"`), or a detailed summary of the individual models (`"individual"`). Can be abbreviated to first letters.
`conditional`	show the conditional estimates (assuming that the alternative is true). Defaults to `FALSE`. Only available for `type == "ensemble"`.
`output_scale`	transform the meta-analytic estimates to a different scale. Defaults to `NULL` which returns the same scale as the model was estimated on.
`probs`	quantiles of the posterior samples to be displayed. Defaults to `c(.025, .975)`
`logBF`	show log of Bayes factors. Defaults to `FALSE`.
`BF01`	show Bayes factors in support of the null hypotheses. Defaults to `FALSE`.
`short_name`	whether priors names should be shortened to the first (couple) of letters. Defaults to `FALSE`.
`remove_spike_0`	whether spike prior distributions with location at zero should be omitted from the summary. Defaults to `FALSE`.
`...`	additional arguments

Value

`summary.RoBMA` returns a list of tables of class 'BayesTools_table'.

Note

See `diagnostics()` for visual convergence checks of the individual models.

See Also

`RoBMA()`, `diagnostics()`, `check_RoBMA()`

Examples

```
## Not run:
# using the example data from Anderson et al. 2010 and fitting the default model
# (note that the model can take a while to fit)
fit <- RoBMA(r = Anderson2010$r, n = Anderson2010$n, study_names = Anderson2010$labels)

# summary can provide many details about the model
summary(fit)

# estimates from the conditional models can be obtained with
summary(fit, conditional = TRUE)

# overview of the models and their prior and posterior probability, marginal likelihood,
# and inclusion Bayes factor can be obtained with
summary(fit, type = "models")

# diagnostics overview, containing the maximum R-hat, minimum ESS, maximum MCMC error, and
# maximum MCMC error / sd across parameters for each individual model can be obtained with
summary(fit, type = "diagnostics")

# summary of individual models and their parameters can be further obtained by
summary(fit, type = "individual")

## End(Not run)
```

`summary_heterogeneity` *Summarizes heterogeneity of a RoBMA model*

Description

Computes the prediction interval, the absolute heterogeneity (tau, tau^2), and relative measures of heterogeneity (I^2, H^2) for a fitted RoBMA object.

Usage

```
summary_heterogeneity(
  object,
  type = "ensemble",
  conditional = FALSE,
  output_scale = NULL,
  probs = c(0.025, 0.975),
  short_name = FALSE,
  remove_spike_0 = FALSE
)
```

Arguments

`object`	a fitted RoBMA object
`type`	whether to show the overall RoBMA results (`"ensemble"`) or a detailed summary of the individual models (`"individual"`). Can be abbreviated to first letters.
`conditional`	show the conditional estimates (assuming that the alternative is true). Defaults to `FALSE`. Only available for `type == "ensemble"`.
`output_scale`	transform the meta-analytic estimates to a different scale. Defaults to `NULL` which returns the same scale as the model was estimated on.
`probs`	quantiles of the posterior samples to be displayed. Defaults to `c(.025, .975)`
`short_name`	whether priors names should be shortened to the first (couple) of letters. Defaults to `FALSE`.
`remove_spike_0`	whether spike prior distributions with location at zero should be omitted from the summary. Defaults to `FALSE`.

Value

`summary.RoBMA` returns a list of tables of class 'BayesTools_table'.

`update.BiBMA` *Updates a fitted BiBMA object*

Description

`update.BiBMA` can be used to

1. add an additional model to an existing `"BiBMA"` object by specifying either a null or alternative prior for each parameter and the prior odds of the model (`prior_weights`), see the `vignette("CustomEnsembles")` vignette,
2. change the prior odds of fitted models by specifying a vector `prior_weights` of the same length as the fitted models,
3. refitting models that failed to converge with updated settings of control parameters,
4. or changing the convergence criteria and recalculating the ensemble results by specifying new `control` argument and setting `refit_failed == FALSE`.

Usage

```
## S3 method for class 'BiBMA'
update(
  object,
  refit_failed = TRUE,
  extend_all = FALSE,
  prior_effect = NULL,
  prior_heterogeneity = NULL,
```

```
    prior_baseline = NULL,
    prior_weights = NULL,
    prior_effect_null = NULL,
    prior_heterogeneity_null = NULL,
    prior_baseline_null = NULL,
    study_names = NULL,
    chains = NULL,
    adapt = NULL,
    burnin = NULL,
    sample = NULL,
    thin = NULL,
    autofit = NULL,
    parallel = NULL,
    autofit_control = NULL,
    convergence_checks = NULL,
    save = "all",
    seed = NULL,
    silent = TRUE,
    ...
)
```

Arguments

`object`	a fitted BiBMA object
`refit_failed`	whether failed models should be refitted. Relevant only if new priors or `prior_weights` are not supplied. Defaults to `TRUE`.
`extend_all`	extend sampling in all fitted models based on `"sample_extend"` argument in `set_autofit_control()` function. Defaults to `FALSE`.
`prior_effect`	prior distribution for the effect size (`mu`) parameter that will be treated as belonging to the alternative hypothesis. Defaults to `NULL`.
`prior_heterogeneity`	prior distribution for the heterogeneity `tau` parameter that will be treated as belonging to the alternative hypothesis. Defaults to `NULL`.
`prior_baseline`	prior distribution for the intercepts (`pi`) of each study that will be treated as belonging to the alternative hypothesis. Defaults to `NULL`.
`prior_weights`	either a single value specifying prior model weight of a newly specified model using priors argument, or a vector of the same length as already fitted models to update their prior weights.
`prior_effect_null`	prior distribution for the effect size (`mu`) parameter that will be treated as belonging to the null hypothesis. Defaults to `NULL`.
`prior_heterogeneity_null`	prior distribution for the heterogeneity `tau` parameter that will be treated as belonging to the null hypothesis. Defaults to `NULL`.
`prior_baseline_null`	prior distribution for the intercepts (`pi`) of each study that will be treated as belonging to the null hypothesis. Defaults to `NULL`.

`study_names`	an optional argument with the names of the studies
`chains`	a number of chains of the MCMC algorithm.
`adapt`	a number of adaptation iterations of the MCMC algorithm. Defaults to `500`.
`burnin`	a number of burnin iterations of the MCMC algorithm. Defaults to `2000`.
`sample`	a number of sampling iterations of the MCMC algorithm. Defaults to `5000`.
`thin`	a thinning of the chains of the MCMC algorithm. Defaults to `1`.
`autofit`	whether the model should be fitted until the convergence criteria (specified in `autofit_control`) are satisfied. Defaults to `TRUE`.
`parallel`	whether the individual models should be fitted in parallel. Defaults to `FALSE`. The implementation is not completely stable and might cause a connection error.
`autofit_control`	allows to pass autofit control settings with the `set_autofit_control()` function. See `?set_autofit_control` for options and default settings.
`convergence_checks`	automatic convergence checks to assess the fitted models, passed with `set_convergence_checks()` function. See `?set_convergence_checks` for options and default settings.
`save`	whether all models posterior distributions should be kept after obtaining a model-averaged result. Defaults to `"all"` which does not remove anything. Set to `"min"` to significantly reduce the size of final object, however, some model diagnostics and further manipulation with the object will not be possible.
`seed`	a seed to be set before model fitting, marginal likelihood computation, and posterior mixing for reproducibility of results. Defaults to `NULL` - no seed is set.
`silent`	whether all print messages regarding the fitting process should be suppressed. Defaults to `TRUE`. Note that `parallel = TRUE` also suppresses all messages.
`...`	additional arguments.

Details

See `BiBMA()` for more details.

Value

`BiBMA` returns an object of class 'BiBMA'.

See Also

`BiBMA()`, `summary.RoBMA()`, `prior()`, `check_setup()`

`update.RoBMA` *Updates a fitted RoBMA object*

Description

`update.RoBMA` can be used to

1. add an additional model to an existing `"RoBMA"` object by specifying either a null or alternative prior for each parameter and the prior odds of the model (`prior_weights`), see the `vignette("CustomEnsembles")` vignette,
2. change the prior odds of fitted models by specifying a vector `prior_weights` of the same length as the fitted models,
3. refitting models that failed to converge with updated settings of control parameters,
4. or changing the convergence criteria and recalculating the ensemble results by specifying new `control` argument and setting `refit_failed` == `FALSE`.

Usage

```
## S3 method for class 'RoBMA'
update(
  object,
  refit_failed = TRUE,
  extend_all = FALSE,
  prior_effect = NULL,
  prior_heterogeneity = NULL,
  prior_bias = NULL,
  prior_hierarchical = NULL,
  prior_weights = NULL,
  prior_effect_null = NULL,
  prior_heterogeneity_null = NULL,
  prior_bias_null = NULL,
  prior_hierarchical_null = NULL,
  study_names = NULL,
  chains = NULL,
  adapt = NULL,
  burnin = NULL,
  sample = NULL,
  thin = NULL,
  autofit = NULL,
  parallel = NULL,
  autofit_control = NULL,
  convergence_checks = NULL,
  save = "all",
  seed = NULL,
  silent = TRUE,
  ...
)
```

Arguments

`object`	a fitted RoBMA object
`refit_failed`	whether failed models should be refitted. Relevant only if new priors or `prior_weights` are not supplied. Defaults to `TRUE`.
`extend_all`	extend sampling in all fitted models based on `"sample_extend"` argument in `set_autofit_control()` function. Defaults to `FALSE`.
`prior_effect`	prior distribution for the effect size (`mu`) parameter that will be treated as belonging to the alternative hypothesis. Defaults to `NULL`.
`prior_heterogeneity`	prior distribution for the heterogeneity `tau` parameter that will be treated as belonging to the alternative hypothesis. Defaults to `NULL`.
`prior_bias`	prior distribution for the publication bias adjustment component that will be treated as belonging to the alternative hypothesis. Defaults to `NULL`.
`prior_hierarchical`	prior distribution for the correlation of random effects (`rho`) parameter that will be treated as belonging to the alternative hypothesis. This setting allows users to fit a hierarchical (three-level) meta-analysis when `study_ids` are supplied. Note that this is an experimental feature and see News for more details. Defaults to a beta distribution `prior(distribution = "beta", parameters = list(alpha = 1, beta = 1))`.
`prior_weights`	either a single value specifying prior model weight of a newly specified model using priors argument, or a vector of the same length as already fitted models to update their prior weights.
`prior_effect_null`	prior distribution for the effect size (`mu`) parameter that will be treated as belonging to the null hypothesis. Defaults to `NULL`.
`prior_heterogeneity_null`	prior distribution for the heterogeneity `tau` parameter that will be treated as belonging to the null hypothesis. Defaults to `NULL`.
`prior_bias_null`	prior distribution for the publication bias adjustment component that will be treated as belonging to the null hypothesis. Defaults to `NULL`.
`prior_hierarchical_null`	prior distribution for the correlation of random effects (`rho`) parameter that will be treated as belonging to the null hypothesis. Defaults to `NULL`.
`study_names`	an optional argument with the names of the studies
`chains`	a number of chains of the MCMC algorithm.
`adapt`	a number of adaptation iterations of the MCMC algorithm. Defaults to `500`.
`burnin`	a number of burnin iterations of the MCMC algorithm. Defaults to `2000`.
`sample`	a number of sampling iterations of the MCMC algorithm. Defaults to `5000`.
`thin`	a thinning of the chains of the MCMC algorithm. Defaults to `1`.
`autofit`	whether the model should be fitted until the convergence criteria (specified in `autofit_control`) are satisfied. Defaults to `TRUE`.

`parallel`	whether the individual models should be fitted in parallel. Defaults to `FALSE`. The implementation is not completely stable and might cause a connection error.
`autofit_control`	allows to pass autofit control settings with the `set_autofit_control()` function. See `?set_autofit_control` for options and default settings.
`convergence_checks`	automatic convergence checks to assess the fitted models, passed with `set_convergence_checks()` function. See `?set_convergence_checks` for options and default settings.
`save`	whether all models posterior distributions should be kept after obtaining a model-averaged result. Defaults to `"all"` which does not remove anything. Set to `"min"` to significantly reduce the size of final object, however, some model diagnostics and further manipulation with the object will not be possible.
`seed`	a seed to be set before model fitting, marginal likelihood computation, and posterior mixing for reproducibility of results. Defaults to `NULL` - no seed is set.
`silent`	whether all print messages regarding the fitting process should be suppressed. Defaults to `TRUE`. Note that `parallel = TRUE` also suppresses all messages.
`...`	additional arguments.

Details

See `RoBMA()` for more details.

Value

`RoBMA` returns an object of class 'RoBMA'.

See Also

`RoBMA()`, `summary.RoBMA()`, `prior()`, `check_setup()`

Examples

```
## Not run:
# using the example data from Bem 2011 and fitting the default (RoBMA-PSMA) model
fit <- RoBMA(d = Bem2011$d, se = Bem2011$se, study_names = Bem2011$study)

# the update function allows us to change the prior model weights of each model
fit1 <- update(fit, prior_weights = c(0, rep(1, 35)))

# add an additional model with different priors specification
# (see '?prior' for more information)
fit2 <- update(fit,
               priors_effect_null = prior("point", parameters = list(location = 0)),
               priors_heterogeneity = prior("normal",
                                   parameters = list(mean = 0, sd = 1),
                                   truncation = list(lower = 0, upper = Inf)),
               priors_bias = prior_weightfunction("one-sided",
                                     parameters = list(cuts = c(.05, .10, .20),
                                                       alpha = c(1, 1, 1, 1))))
```

```
# update the models with an increased number of sample iterations
fit3 <- update(fit, autofit_control = set_autofit_control(sample_extend = 1000), extend_all = TRUE)

## End(Not run)
```

`weighted_multivariate_normal`

Weighted multivariate normal distribution

Description

Density function for the weighted multivariate normal distribution with `mean`, covariance matrix `sigma`, critical values `crit_x`, and weights `omega`.

Arguments

`x`	quantiles.
`p`	vector of probabilities.
`mean`	mean
`sigma`	covariance matrix.
`crit_x`	vector of critical values defining steps.
`omega`	vector of weights defining the probability of observing a t-statistics between each of the two steps.
`type`	type of weight function (defaults to `"two.sided"`).
`log`, `log.p`	logical; if `TRUE`, probabilities `p` are given as `log(p)`.

Value

`.dwmnorm_fast` returns a density of the multivariate weighted normal distribution.

See Also

Normal, weighted_normal

`weighted_normal` *Weighted normal distribution*

Description

Density, distribution function, quantile function and random generation for the weighted normal distribution with `mean`, standard deviation `sd`, steps `steps` (or critical values) `crit_x`), and weights `omega`.

Usage

```
dwnorm(
  x,
  mean,
  sd,
  steps = if (!is.null(crit_x)) NULL,
  omega,
  crit_x = if (!is.null(steps)) NULL,
  type = "two.sided",
  log = FALSE
)

pwnorm(
  q,
  mean,
  sd,
  steps = if (!is.null(crit_x)) NULL,
  omega,
  crit_x = if (!is.null(steps)) NULL,
  type = "two.sided",
  lower.tail = TRUE,
  log.p = FALSE
)

qwnorm(
  p,
  mean,
  sd,
  steps = if (!is.null(crit_x)) NULL,
  omega,
  crit_x = if (!is.null(steps)) NULL,
  type = "two.sided",
  lower.tail = TRUE,
  log.p = FALSE
)

rwnorm(
```

```
  n,
  mean,
  sd,
  steps = if (!is.null(crit_x)) NULL,
  omega,
  crit_x = if (!is.null(steps)) NULL,
  type = "two.sided"
)
```

Arguments

`x, q`	vector of quantiles.
`mean`	mean
`sd`	standard deviation.
`steps`	vector of steps for the weight function.
`omega`	vector of weights defining the probability of observing a t-statistics between each of the two steps.
`crit_x`	vector of critical values defining steps (if `steps` are not supplied).
`type`	type of weight function (defaults to `"two.sided"`).
`log, log.p`	logical; if `TRUE`, probabilities `p` are given as `log(p)`.
`lower.tail`	logical; if `TRUE` (default), probabilities are $P[X \le x]$, otherwise, $P[X \ge x]$.
`p`	vector of probabilities.
`n`	number of observations. If length(n) > 1, the length is taken to be the number required.

Details

The `mean`, `sd`, `steps`, `omega` can be supplied as a vectors (`mean`, `sd`) or matrices (`steps`, `omega`) with length / number of rows equal to `x`/`q`/ `p`. Otherwise, they are recycled to the length of the result.

Value

`dwnorm` gives the density, `dwnorm` gives the distribution function, `qwnorm` gives the quantile function, and `rwnorm` generates random deviates.

See Also

Normal

Index