Package ‘RoBMA’

July 18, 2023

Title  Robust Bayesian Meta-Analyses
Version 3.1.0
Maintainer František Bartoš <f.bartos96@gmail.com>
Description A framework for estimating ensembles of meta-analytic models (assuming either presence or absence of the effect, heterogeneity, and publication bias). 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 non-informative or informative prior distributions for the effect size, heterogeneity, and publication bias components (including selection models and PET-PEESE). 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 7.2.3
SystemRequirements JAGS >= 4.3.1 (https://mcmc-jags.sourceforge.io/)
NeedsCompilation yes
Depends R (>= 4.0.0)
Imports BayesTools (>= 0.2.16), runjags, rjags, stats, graphics, mvtnorm, scales, Rdpack, rlang, ggplot2
Suggests parallel, metaBMA, metafor, weightr, lme4, fixest, metadat, testthat, vdiff, 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 2023-07-18 21:40:02 UTC

R topics documented:

RoBMA-package ........................................ 3
Anderson2010 ........................................ 4
Bem2011 .............................................. 5
BiBMA .................................................. 5
check_RoBMA .......................................... 8
check_setup ........................................... 8
check_setup.BiBMA ..................................... 11
check_setup.reg ....................................... 13
combine_data ......................................... 19
contr.independent ..................................... 21
contr.mendid .......................................... 21
corr.orthonormal ..................................... 22
diagnostics ............................................ 23
effect Sizes ............................................ 25
forest .................................................. 26
interpret ............................................... 28
is.RoBMA ............................................... 28
Kroupova2021 .......................................... 29
Lui2015 ............................................... 30
marginal_plot ......................................... 30
marginal_summary ..................................... 32
NoBMA .................................................. 33
NoBMA.reg ............................................ 36
plot.RoBMA ........................................... 40
plot_models .......................................... 42
Poulsen2006 .......................................... 43
print.marginal_summary.RoBMA ..................... 44
print.RoBMA .......................................... 45
print.summary.RoBMA .................................. 45
prior .................................................... 46
prior_factor .......................................... 47
**RoBMA-package**

prior_informed ................................................................. 49  
prior_none ................................................................. 51  
prior_PEESE ................................................................. 52  
prior_PET ................................................................. 53  
prior_weightfunction ................................................................. 54  
RoBMA ................................................................. 55  
RoBMA.reg ................................................................. 61  
RoBMA_control ................................................................. 66  
RoBMA_options ................................................................. 67  
sample_sizes ................................................................. 68  
standard_errors ................................................................. 69  
summary.RoBMA ................................................................. 71  
update.BiBMA ................................................................. 72  
update.RoBMA ................................................................. 75  
weighted_multivariate_normal ................................................................. 78  
weighted_normal ................................................................. 79

**Index**  

81

---

**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. (2022), Maier et al. (2022), 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**


**See Also**

Useful links:

- [https://fbartos.github.io/RoBMA/](https://fbartos.github.io/RoBMA/)

<table>
<thead>
<tr>
<th>Anderson2010</th>
<th>27 experimental studies from Anderson et al. (2010) that meet the best practice criteria</th>
</tr>
</thead>
</table>

**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 be found at [https://github.com/Joe-Hilgard/Anderson-meta](https://github.com/Joe-Hilgard/Anderson-meta).

**Usage**

Anderson2010

**Format**

A data.frame with 3 columns and 23 observations.

**Value**

a data.frame.

**References**

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


BiBMA

Estimate a Bayesian Model-Averaged Meta-Analysis of Binomial Data

Description

BiBMA estimate a Binomial 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.
BiBMA(  
  x1,  
  x2,  
  n1,  
  n2,  
  study_names = NULL,  
  study_ids = NULL,  
  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"),  
  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

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.
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.

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.
check_setup

Details

See RobMA() for more details.

Value

RobMA returns an object of class 'RobMA'.

See Also

RobMA(), summary.RobMA(), update.RobMA(), check_setup()

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)

Arguments

fit a fitted RobMA object.

Value

check_RobMA returns a vector of error and warning messages.

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.
check_setup

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, 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_weightfunction(distribution = "Cauchy", parameters = list(0, 1), truncation = list(0, Inf), prior_weights = 1/4),
                     prior_weightfunction(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, 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_weightfunction(distribution = "one.sided", parameters = list(alpha = c(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)), prior_weights = 1/12),
                     prior_weightfunction(distribution = "Cauchy", parameters = list(0, 1), truncation = list(0, Inf), prior_weights = 1/4),
                     prior_weightfunction(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
)
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_null, priors_heterogeneity_null, priors_effect, and priors_effect_null. 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_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.05, 0.05)), prior_weights = 1/12), prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.05, 0.05)), prior_weights = 1/12), prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.025, 0.05, 0.5)), prior_weights = 1/12), prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.025, 0.05, 0.5)), prior_weights = 1/12), prior_weightfunction(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. (2022).

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
check_setup.BiBMA

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 be-
longing 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 sup-
plied. 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.
...

Value

check_setup.reg invisibly returns list of summary tables.
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, 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_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(0.05, 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_weightfunction(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(),
check_setup.reg

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(0.05, 0.5)), 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), 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 object created by the combine_data function. This is an alternative input entry to specifying the d, r, y, etc... directly. I.e., you cannot pass the a data.frame and reference to the columns.
test_predictors vector of predictor names that will be test (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 a 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 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.
effect_direction the expected direction of the effect. The one-sided selection sets the weights omega to 1 to significant results in the expected direction. 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 are 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_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(0.05, 0.5)), prior_weights = 1/12), prior_weightfunction(distribution = "Cauchy", parameters = list(0, 1), truncation = list(0, Inf), prior_weights = 1/4), prior_weightfunction(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. (2022).

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**

```r
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
The function `combine_data` aims 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 `escale` 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.
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.
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 \( contrasts = \text{TRUE} \) and \( k = n \) if \( contrasts = \text{FALSE} \).

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 orthonomal 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 \( contrasts = \text{TRUE} \) and \( k = n \) if \( contrasts = \text{FALSE} \).

References

**Examples**

```r
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**

```r
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(
```

```
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

```r
## 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,
```
effect_sizes

# the trace plot for the mean parameter in each 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)
forest

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    odds 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


See Also

standard_errors(), sample_sizes()

Description

forest creates a forest plot for a RoBMA object.
forest

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 /quotesingle.Var 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)
is.RoBMA  

# 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.

---


---

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

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


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_plot* 31
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(0.025, 0.975)
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 (Normal - 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,
  priors_effect = prior(distribution = "normal", parameters = list(mean = 0, sd = 1)),
  priors_heterogeneity = prior(distribution = "invgamma", parameters = list(shape = 1, scale = 0.15)),
  priors_effect_null = prior(distribution = "point", parameters = list(location = 0)),
  priors_heterogeneity_null = prior(distribution = "point", parameters = list(location = 0)),
  priors_hierarchical = prior("beta", parameters = list(alpha = 1, beta = 1)),
  priors_hierarchical_null = NULL,
  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., you cannot pass the a data.frame and reference 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 a 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 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.

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 = 0.15)) 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.

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.

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 (Normal - 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

```r
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,  
...  
)
```
priors_effect = prior(distribution = "normal", parameters = list(mean = 0, sd = 1)),
priors_heterogeneity = prior(distribution = "invgamma", parameters = list(shape = 1, scale = 0.15)),
priors_effect_null = prior(distribution = "point", parameters = list(location = 0)),
priors_heterogeneity_null = prior(distribution = "point", parameters = list(location = 0)),
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_factor("spike", parameters = list(location = 0), contrast = "meandif"),
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 object created by the combine_data function. This is an alternative input entry to specifying the d, r, y, etc... directly. I.e., you cannot pass the a data.frame and reference to the columns.
test_predictors vector of predictor names that will be test (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
a 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 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.

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_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)).

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.

See RoBMA.reg() for more details.

Value

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

See Also

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

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,
  ...
)

plot.RoBMA

Plots a fitted RoBMA object
plot.RoBMA

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 weight function), 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

```r
## 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")
```
plot_models

# 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**

```r
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 used 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 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)
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


print.marginal_summary.RoBMA

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.
print.RoBMA

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
prior

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,  # name of the prior distribution. The possible options are
  parameters,    # "point" for a point density characterized by a location parameter.
  truncation = list(lower = -Inf, upper = Inf),  # "normal" for a normal distribution characterized by a mean and sd parameters.
  prior_weights = 1  # "lognormal" for a lognormal distribution characterized by a meanlog and sdlog parameters.
)

Arguments

distribution  # "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 latter is internally converted to rate.
  "uniform" for a uniform distribution defined on a range from a to b
prior_factor

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.
prior_informed

“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.
prior_informed

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


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

```r
p2 <- prior_informed("Oral Health", parameter ="effect", type ="smd")
print(p2)
```

---

`prior_none`  
`Create a prior distribution`

**Description**

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

**Usage**

```r
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**

```r
# 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)
```
## 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

```r
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_PET

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.
prior_weightfunction

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 func-
  tion, 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 usu-
              ally 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_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 cus-

tomization 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,
  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(0.05, 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_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1),
                              steps = c(0.025), truncation = list(0, Inf),
                              prior_weights = 1/4),
                   prior_weightfunction(distribution = "Cauchy", parameters = list(0, 1),
                              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,
  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., you cannot pass the a data.frame and reference 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 a 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 scale of transformation output, but can be changed within the summary function.

**effect_direction**

the expected direction of the effect. The one-sided selection sets the weights omega to 1 to significant results in the expected direction. 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.

**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), 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_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1), steps = c(0.05, 0.05)), prior_weights = 1/12), prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1), steps = c(0.05, 0.5)), 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(0.025, 0.05, 0.5)), prior_weights = 1/12), prior_weightfunction(distribution = "Cauchy", parameters = list(0, Inf), truncation = list(0, Inf), prior_weights = 1/4), prior_PET(distribution = "Cauchy", parameters = list(0, 5), truncation = list(0, Inf), prior_weights = 1/4)), corresponding to the RoBMA-PSMA model introduced by Bartoš et al. (2022).

**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.

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. (2022). The previous versions of the package (i.e., RoBMA < 2.0) used specifications proposed by Maier et al. (2022) (this specification can be easily obtained by setting model_type = "2w". The RoBMA-PP specification from Bartoš et al. (2022) can be obtained by setting model_type = "PP".)
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


See Also

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

Examples

```r
## 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
```
Estimate a Robust Bayesian Meta-Analysis Meta-Regression

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

```r
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,
)```

## End(Not run)
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_factor("spike", parameters = list(location = 0), contrast = "meandif"),
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 object created by the `combine_data` function. This is an alternative input entry to specifying the `d`, `r`, `y`, etc... directly. I.e., you cannot pass the a data.frame and reference to the columns.

**test_predictors**
vector of predictor names that will be test (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**
a 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 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.

**effect_direction**
the expected direction of the effect. The one-sided selection sets the weights omega to 1 to significant results in the expected direction. 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_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.05)), prior_weights = 1/12), prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1), steps = c(0.05, 0.5)), prior_weights = 1/12), prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.025, 0.05, 0.5)), prior_weights = 1/12), prior_weightfunction(distribution = "one.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.025, 0.05, 0.5)), prior_weights = 1/12), prior_weightfunction(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. (2022).

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)).

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 \texttt{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

\texttt{RoBMA.reg} returns an object of class `RoBMA.reg'.

References


See Also

\texttt{RoBMA()}, \texttt{summary.RoBMA()}, \texttt{update.RoBMA()}, \texttt{check_setup.reg()}

---

\textbf{RoBMA\_control}

\textit{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

```r
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
)
```

```r
set_convergence_checks(
  max_Rhat = 1.05,
  min_ESS = 500,
  max_error = NULL,
  max_SD_error = NULL,
  max_C1 << 10
)```
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

See Also
effect_sizes(), standard_errors()

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


See Also

effect_sizes(), sample_sizes()
summary.RoBMA

Summarize fitted RoBMA object

**Description**

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

**Usage**

```r
## 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
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

```r
## 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

```r
## 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,
  study_names = NULL,
  chains = NULL,
  adapt = NULL,
  burnin = NULL,
  sample = NULL,
  thin = NULL,
  autofit = NULL,
```
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
adapt
burnin
sample
thin
autofit
parallel
autofit_control
convergence_checks
save
seed
silent...

Details

See \texttt{RoBMA()} for more details.

Value

\texttt{RoBMA} returns an object of class 'RoBMA'.

See Also

\texttt{RoBMA()}, \texttt{summary.RoBMA()}, \texttt{prior()}, \texttt{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)))
weighted_multivariate_normal

Weighted multivariate normal distribution

Description

Density function for the weighted multivariate normal distribution with mean, covariance matrix \( \sigma \), critical values \( \text{crit}_x \), 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

\(.\text{dwmnorm\_fast}\) 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**

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

```r
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
)
```

```r
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
)
```

```r
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 ≤ x], otherwise, P[X ≥ 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

* datasets
  Anderson2010, 4
  Bem2011, 5
  Kroupova2021, 29
  Lui2015, 30
  Poulsen2006, 43

* package
  RoBMA-package, 3
  _PACKAGE (RoBMA-package), 3

Anderson2010, 4
autorun.jags, 60
BayesTools, 60
Bem2011, 5
Beta, 47, 51
BiBMA, 5
BiBMA(), 13, 74, 75
bridge_sampler, 60

Cauchy, 47, 51
check_RoBMA, 8
check_RoBMA(), 32, 72
check_setup, 8
check_setup(), 8, 13, 18, 21, 36, 40, 60, 75, 77
check_setup.BiBMA, 11
check_setup.reg, 13
check_setup.reg(), 11, 66
check_setup.RoBMA.reg
  (check_setup.reg), 13
combine_data, 19
corr.independent, 21
corr.meandif, 21
corr.orthonormal, 22
d2logOR (effect_sizes), 25
d2OR (effect_sizes), 25
d2r (effect_sizes), 25
d2z (effect_sizes), 25
diagnostics, 23
diagnostics(), 32, 60, 72
diagnostics_autocorrelation
  (diagnostics), 23
diagnostics_density (diagnostics), 23
diagnostics_trace (diagnostics), 23
difftime, 67
dwnorm (weighted_normal), 79
effect_sizes, 25
effect_sizes(), 21, 69, 70
escalc, 20
Exponential, 47, 51
forest, 26
interpret, 28
InvGamma, 47, 51
is.BiBMA (is.RoBMA), 28
is.NoBMA (is.RoBMA), 28
is.RoBMA, 28
Kroupova2021, 29
LocationScaleT, 47, 51
Lognormal, 47, 51
logOR2d (effect_sizes), 25
logOR2OR (effect_sizes), 25
logOR2r (effect_sizes), 25
logOR2z (effect_sizes), 25
Lui2015, 30
marginal_plot, 30
marginal_summary, 32
n_d (sample_sizes), 68
n_r (sample_sizes), 68
n_z (sample_sizes), 68
NoBMA, 33
NoBMA.reg, 36
Normal, 47, 51, 78, 80
OR2d (effect_sizes), 25
OR2logOR (effect_sizes), 25
OR2r (effect_sizes), 25
OR2z (effect_sizes), 25
par, 24
plot.prior(), 47, 51, 53–55
plot.RoBMA, 40
plot.RoBMA(), 60
plot.models, 42
Poulsen2006, 43
print.marginal.summary.RoBMA, 44
print.RoBMA, 45
print.RoBMA(), 60
print.summary.RoBMA, 45
prior, 46
prior(), 49, 50, 53, 54, 60, 75, 77
prior_factor, 47
prior_informed, 49
prior_informed_medicine_names, 50
prior_none, 51
prior_PEESE, 52
prior_PEESE(), 60
prior_PET, 53
prior_PET(), 60
prior_weightfunction, 54
prior_weightfunction(), 60
qwnorm (weighted_normal), 79
r2d (effect_sizes), 25
r2logOR (effect_sizes), 25
r2OR (effect_sizes), 25
r2z (effect_sizes), 25
RoBMA, 55, 67
RoBMA(), 8, 11, 21, 24, 31–33, 36, 40, 41, 45, 46, 60, 66, 72, 77
RoBMA-package, 3
RoBMA.get_option (RoBMA_options), 67
RoBMA.options (RoBMA_options), 67
RoBMA.package (RoBMA-package), 3
RoBMA.reg, 61
RoBMA.reg(), 18, 36, 40
RoBMA.control, 66
RoBMA.options, 67
RoBMA.package (RoBMA-package), 3
rwnorm (weighted_normal), 79
sample_sizes, 68
sample_sizes(), 21, 26, 70
se_d (sample_sizes), 68
se_d2se_logOR (standard_errors), 69
se_d2se_r (standard_errors), 69
se_d2se_z (standard_errors), 69
se_logOR2se_d (standard_errors), 69
se_logOR2se_r (standard_errors), 69
se_logOR2se_z (standard_errors), 69
se_r (sample_sizes), 68
se_r2se_d (standard_errors), 69
se_r2se_logOR (standard_errors), 69
se_r2se_z (standard_errors), 69
se_z (sample_sizes), 68
se_z2se_d (standard_errors), 69
se_z2se_logOR (standard_errors), 69
se_z2se_r (standard_errors), 69
set_autofit_control (RoBMA_control), 66
set_autofit_control(), 7, 35, 39, 59, 65, 73, 74, 76, 77
set_autofit_control (RoBMA_control), 66
set_convergence_checks (RoBMA_control), 66
set_convergence_checks(), 7, 36, 39, 59, 65, 74, 77
stan_plot, 24
standard_errors, 69
summary.RoBMA, 71
summary.RoBMA(), 8, 24, 32, 36, 40, 60, 66, 75, 77
update.BiBMA, 72
update.RoBMA, 67, 75
update.RoBMA(), 8, 36, 40, 60, 66
weighted_multivariate_normal, 78
weighted_normal, 78, 79
z2d (effect_sizes), 25
z2logOR (effect_sizes), 25
z2OR (effect_sizes), 25
z2r (effect_sizes), 25