Package ‘RoBMA’

August 7, 2020

Title Robust Bayesian Meta-Analyses
Version 1.0.4
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) and using Bayesian model averaging to combine them. The ensembles use Bayes Factors to test for the presence or absence of the individual components (e.g., effect vs. no effect) and model-averages parameter estimates based on posterior model probabilities (Maier, Bartoš & Wagenmakers, 2020, <doi:10.31234/osf.io/u4cns>). The user can define a wide range of non-informative or informative priors for the effect size, heterogeneity, and weight functions. 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 7.0.2
Imports runjags, bridgesampling, rjags, coda, psych, stats, graphics, extraDistr, scales, DPQ, callr, Rdpack
Suggests ggplot2, rstan, metaBMA, testthat, vdiffir, knitr, rmarkdown
RdMacros Rdpack
LinkingTo BH
VignetteBuilder knitr
NeedsCompilation yes
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]
Repository CRAN
Date/Publication 2020-08-07 07:50:07 UTC
R topics documented:

Anderson2010 ...................................................... 2
check_setup ...................................................... 3
diagnostics ....................................................... 4
is.RoBMA .......................................................... 6
plot.RoBMA ....................................................... 6
plot.RoBMA.prior .................................................. 8
print.RoBMA ..................................................... 9
print.RoBMA.prior ............................................... 10
print.summary.RoBMA ......................................... 10
prior ................................................................. 11
RoBMA .............................................................. 12
summary.RoBMA .................................................. 17
update.RoBMA .................................................... 19
weightedt ........................................................ 22

Index 25

Anderson2010 27 experimental studies from from Anderson et al. (2010) that meet the best practice criteria

Description

The dataset contains correlational coefficients, sample sizes, and labels for 27 experimental studies focusing on the effect of violent videogames on aggressive behavior. The full original data can found at https://github.com/Joe-Hilgard/Anderson-meta.

Usage

Anderson2010

Format

A data.frame with 3 columns and 23 observations.

References

**check_setup**

Prints summary of "RoBMA" ensemble implied by the specified priors

**Description**

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

**Usage**

```r
check_setup(
  priors_mu = prior(distribution = "normal", parameters = list(mean = 0, sd = 1)),
  priors_tau = prior(distribution = "invgamma", parameters = list(shape = 1, scale = 0.15)),
  priors_omega = list(prior(distribution = "two.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_odds = 1/2), prior(distribution = "two.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.05, 0.1)), prior_odds = 1/2)),
  priors_mu_null = prior(distribution = "point", parameters = list(location = 0)),
  priors_tau_null = prior(distribution = "point", parameters = list(location = 0)),
  priors_omega_null = prior(distribution = "point", parameters = list(location = 1)),
  models = FALSE,
  silent = FALSE )
```

**Arguments**

- **priors_mu** list of prior distributions for the mu parameter that will be treated as belonging to the alternative hypothesis. Defaults to prior(distribution = "normal", parameters = list(mean = 0, sd = 1)).
- **priors_tau** list of prior distributions for the tau parameter that will be treated as belonging to the alternative hypothesis. Defaults to prior(distribution = "invgamma", parameters = list(shape = 1, scale = 0.15)).
- **priors_omega** list of prior weight functions for the omega parameter that will be treated as belonging to the alternative hypothesis. Defaults to list(prior(distribution = "two.sided", parameters = list(alpha = c(1,1), steps = c(.05)), prior_odds = 1/2), prior(distribution = "two.sided", parameters = list(alpha = c(1,1,1), steps = c(.05,.10)), prior_odds = 1/2)).
- **priors_mu_null** list of prior distributions for the mu parameter that will be treated as belonging to the null hypothesis. Defaults to point distribution with location at 0 (prior(distribution = "point", parameters = list(location = 0))).
- **priors_tau_null** list of prior distributions for the tau parameter that will be treated as belonging to the null hypothesis. Defaults to point distribution with location at 0 (prior(distribution = "point", parameters = list(location = 0))).
diagnostics

priors_omega_null

list of prior weight functions for the omega parameter that will be treated as belonging to the null hypothesis. Defaults to point distribution with location at 1 (prior(distribution = "point",parameters = list(location = 0))).

models

should the models' details be printed.

silent

do not print the results.

See Also

RoBMA(), prior()

diagnostics Checks a fitted RoBMA object

Description

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

Usage

diagnostics(
  fit,
  parameter,
  type,
  plot_type = "base",
  show_figures = if (parameter == "omega") -1 else NULL,
  show_models = NULL,
  par_transform = TRUE,
  lags = 30,
  title = is.null(show_models) | length(show_models) > 1,
  ...
)

Arguments

fit a fitted RoBMA object

parameter a parameter to be plotted. Either "mu", "tau", "theta", or "omega".

type what type of model check should 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.

plot_type whether to use a base plot "base" or ggplot2 "ggplot2" for plotting. The later requires ggplot2 package to be installed.
diagnostics

show_figures which figures should be returned in case of multiple plots are generated. Useful when parameter = "omega" when a plot for each parameter would be generated. Can be also used for parameter = "theta" to obtain only a specific subset of thetas. Set to NULL to show all parameters (default for parameter = "theta").

show_models diagnostics for which models should be produced. Defaults to NULL that shows diagnostics to all models.

par_transform whether the figures should be produced for the par_transform parameters. Defaults to TRUE.

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.

See Also

RoBMA(), summary.RoBMA()

Examples

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

### ggplot2 version of all of the plots can be obtained by adding 'model_type = "ggplot"
# diagnostics function allows to visualize diagnostics of a fitted RoBMA object, for example,
# the trace plot for the mean parameter in each model model
diagnostics(fit, parameter = "mu", type = "chain")

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

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

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

## End(Not run)
is.RoBMA  
*Reports whether x is a RoBMA object*

**Description**
Reports whether x is a RoBMA object

**Usage**
is.RoBMA(x)

**Arguments**
- **x**: an object to test

---

plot.RoBMA  
*Plots a fitted RoBMA object*

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

**Usage**
```r
## S3 method for class 'RoBMA'
plot(
x,  
parameter,  
type = "averaged",  
plot_type = "base",  
mean = TRUE,  
median = FALSE,  
CI = 0.95,  
prior = FALSE,  
order = NULL,  
digits_estimates = 2,  
show_figures = if (parameter == "omega" & (weights | any(type %in% "individual")))
  -1,
  weights = FALSE,  
rescale_x = FALSE,  
...  
)
```
Arguments

x  a fitted RoBMA object

parameter  a parameter to be plotted. Either "mu", "tau", "theta", or "omega". A bi-variate plot for model-averaged estimates of "mu" and "tau" can be obtained by c("mu","tau") if type = "averaged". In addition, a forest plot with the original estimates can be obtained by "forest" or added to the theta estimates by c("theta","forest").

type  what type of estimates should be plotted. Options are "averaged" for the model-averaged estimates, "conditional" for the conditional estimates, or "individual" for the individual models estimates. The options c("individual","conditional") can be supplied together to show only conditional individual models.

plot_type  whether to use a base plot "base" or ggplot2 "ggplot2" for plotting. The later requires ggplot2 package to be installed.

mean  whether the mean should be plotted.

median  whether the median should be plotted.

CI  width of the confidence intervals.

prior  add prior density to the plot. Only available for type = "averaged" or type = "conditional". Defaults to FALSE.

order  either (1) ordering of the studies for parameter = "theta" or parameter = "forest". Defaults to NULL - ordering as supplied to the fitting function. However, studies can be ordered either "ascending" or "descending" by effect size, or by "alphabetical" by labels. Or (2) ordering models for type = "individual". The default orders models according to their number. However, models can be ordered either "ascending" or "descending" by posterior model probability c("ascending","prob"), or marginal likelihood c("descending","marglik") by marginal likelihood.

digits_estimates  number of decimals to be displayed for parameter = "theta", parameter = "forest", and type = "individual" plot.

show_figures  which figures should be returned in the case when multiple plots are generated. Useful when parameter = "omega", type = "individual" which generates a figure for each weights cut-off. Defaults to ~1 which omits the first weight. Set to NULL to show all figures or to c(1,3) to show only the first and third one.

weights  whether the weights or weight function should be returned. Only applicable when parameter = "omega". Defaults to FALSE - the weight function is plotted.

rescale_x  whether the x-axis should be rescaled in order to make the x-ticks equally spaced. Available only for the weightfunction plot. Defaults to FALSE.

... additional arguments to be passed to par if plot_type = "base". Especially useful for parameter == "theta", parameter == "forest" or type = "individual" where automatic margins might cut out parts of the labels.

See Also

RoBMA()
Examples

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

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

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

# conditional plots might by obtained by specifying
plot(fit, parameter = "mu", type = "conditional")

# plotting function also allows to visualize the weight function
# (or individual weights by adding 'weights = TRUE')
plot(fit, parameter = "omega")

# or the forest plot (the estimated study effects can be shown by setting 'parameter = "theta"')
plot(fit, parameter = "forest")

# it is also possible to compare the individual model estimates
# and order them by the posterior probability
plot(fit, parameter = "mu", type = "individual", order = "prob")
```

## End(Not run)

plot.RoBMA.prior

Plots a RoBMA.prior object

Description

Plots a RoBMA.prior object

Usage

```
## S3 method for class 'RoBMA.prior'
plot(
  x,
  plot_type = "base",
  effect_size = NULL,
  mu_transform = NULL,
  show_figures = -1,
  weights = FALSE,
  par_name = "mu",
```
samples = 1e+06,
points = 1000,
...
)

Arguments

- **x**: a RoBMA prior
- **plot_type**: whether to use a base plot "base" or ggplot2 "ggplot2" for plotting. The later requires ggplot2 package to be installed.
- **effect_size**: what type of effect size is supposed to be plotted. Only relevant if the mu parameter needs to be transformed ("r" for correlation coefficients or "OR" for odds ratios).
- **mu_transform**: whether and how should the prior distribution be transformed. If the prior distribution is constructed for effect sizes supplied as correlations, the prior for mu parameter is not defined on the correlation scale directly, but transformed into it. Only possible if the effect_size == "r", par_name = "mu". Defaults to NULL. Other options are "cohens_d" and "fishers_z".
- **show_figures**: which figures should be returned in case of multiple plots are generated. Useful when priors for the omega parameter are plotted and weights = TRUE.
- **weights**: whether the weights or weight function should be returned. Only applicable for priors on the omega parameter. Defaults to FALSE - the weight function is plotted.
- **par_name**: a name of parameter to be included in the x-axis label
- **samples**: how many samples should be drawn for the density plot (applies only for the weight functions, other prior distributions are plotted using the pdf). Defaults to 10000.
- **points**: how many points should be used for drawing the density plot. Defaults to 1000.
- **...**: additional arguments

See Also

- prior()

print.RoBMA

Prints a fitted RoBMA object

Prints a fitted RoBMA object

Usage

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

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

See Also

RoBMA()

print.RoBMA.prior Prints a RoBMA.prior object

Description

Prints a RoBMA.prior object

Usage

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

Arguments

x a RoBMA prior
...
additional arguments
silent to silently return the print message.
plot to return bquote formatted prior name for plotting.
digits_estimates number of decimals to be displayed for printed parameters.

See Also

prior()

print.summary.RoBMA Prints summary object for RoBMA method

Description

Prints summary object for RoBMA method

Usage

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


Arguments

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

See Also

RoBMA()

prior Creates a RoBMA prior

Description

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

Usage

prior(
  distribution,  
  parameters,  
  truncation = list(lower = -Inf, upper = Inf),  
  prior_odds = 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.
  "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.
  "two.sided" for a two-sided weight function characterized by a vector steps and vector alpha parameters. The alpha parameter determines an alpha parameter of Dirichlet distribution which cumulative sum is used for the weights omega.
"one.sided" for a one-sided weight function characterized by either a vector steps and vector alpha parameter, leading to a monotonic one-sided function, or by a vector steps, vector alpha1, and vector alpha2 parameters leading non-monotonic one-sided weight function. The alpha / alpha1 and alpha2 parameters determine an alpha parameter of Dirichlet distribution which cumulative sum is used for the weights omega.

"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 lower truncation point is automatically set to 0 if it is specified outside of the support of distributions defined only for positive numbers.

**prior_odds** prior odds associated with a given distribution. `RoBMA()` creates models corresponding to all combinations of prior distributions for each of the model parameters (mu, tau, omega), and sets the model priors odds to the product of its prior distributions.

**See Also**

`plot.RoBMA.prior()`, `Normal`, `Cauchy`, `LocationScaleT`, `GammaDist`, `InvGamma`.

**Examples**

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

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

# or a prior for one-sided weight function
p3 <- prior("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.RoBMA for all options)
plot(p1)
```

---

**Description**

`RoBMA` is used to estimate a Robust Bayesian Meta-Analysis. Either t-statistics (t) and sample sizes of the original studies (n or n1 and n2), or effect sizes (d) and standard errors (se) can be used to estimate the model.
Usage

```
RoBMA(
    t = NULL,
    d = NULL,
    r = NULL,
    y = NULL,
    OR = NULL,
    se = NULL,
    n = NULL,
    n1 = NULL,
    n2 = NULL,
    lCI = NULL,
    uCI = NULL,
    test_type = "two.sample",
    study_names = NULL,
    mu_transform = if (!is.null(r)) "cohens_d" else if (!is.null(OR)) "log_OR" else NULL,
    effect_direction = "positive",
    priors_mu = prior(distribution = "normal", parameters = list(mean = 0, sd = 1)),
    priors_tau = prior(distribution = "invgamma", parameters = list(shape = 1, scale = 0.15)),
    priors_omega = list(prior(distribution = "two.sided", parameters = list(alpha = c(1, 1), steps = c(0.05)), prior_odds = 1/2),
        prior(distribution = "two.sided", parameters = list(alpha = c(1, 1, 1), steps = c(0.05, 0.1)), prior_odds = 1/2)),
    priors_mu_null = prior(distribution = "point", parameters = list(location = 0)),
    priors_tau_null = prior(distribution = "point", parameters = list(location = 0)),
    priors_omega_null = prior(distribution = "point", parameters = list(location = 1)),
    chains = 3,
    iter = 10000,
    burnin = 5000,
    thin = 1,
    control = NULL,
    save = "all",
    seed = NULL
)
```

Arguments

t        a vector of t-statistics.
d        a vector of effect sizes measured as Cohen’s d.
r        a vector of effect sizes measured as correlations.
y        a vector of unspecified effect sizes.
OR       a vector of odds ratios.
se       a vector of standard errors of the effect sizes.
n        a vector of overall sample sizes.
n1       a vector of sample sizes for the first group.
n2       a vector of sample sizes for the second group.
lCI  a vector of lower bounds of confidence intervals.

uCI  a vector of upper bounds of confidence intervals.

test_type  a type of test used in the original studies. Options are "two.sample" (default) and "one.sample". Only available if $d$ is supplied.

study_names  an optional argument with the names of the studies.

mu_transform  transformation to be applied to the supplied effect sizes before fitting the individual models. Only available if correlations or odds ratios are supplied as input. Defaults to "cohens_d" for correlations (another option is "fishers_z") and "log.OR" for odds ratios (another option is "cohens_d"). Note that priors are specified on the transformed scale and estimates are transformed back (apart from tau).

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_mu  list of prior distributions for the mu parameter that will be treated as belonging to the alternative hypothesis. Defaults to prior(distribution = "normal", parameters = list(mean = 0, sd = 1)).

priors_tau  list of prior distributions for the tau parameter that will be treated as belonging to the alternative hypothesis. Defaults to prior(distribution = "invgamma", parameters = list(shape = 1, scale = .15)).

priors_omega  list of prior weight functions for the omega parameter that will be treated as belonging to the alternative hypothesis. Defaults to list(prior(distribution = "two.sided", parameters = list(alpha = c(1,1), steps = c(.05)), prior_odds = 1/2), prior(distribution = "two.sided", parameters = list(alpha = c(1,1,1), steps = c(.05,.10)), prior_odds = 1/2)).

priors_mu_null  list of prior distributions for the mu parameter that will be treated as belonging to the null hypothesis. Defaults to point distribution with location at 0 (prior(distribution = "point", parameters = list(location = 0))).

priors_omega_null  list of prior weight functions for the omega parameter that will be treated as belonging to the null hypothesis. Defaults to point distribution with location at 1 (prior(distribution = "point", parameters = list(location = 0))).

chains  a number of chains of the MCMC algorithm.

iter  a number of sampling iterations of the MCMC algorithm. Defaults to 10000, with a minimum of 4000.

burnin  a number of burnin iterations of the MCMC algorithm. Defaults to 5000.

thin  a thinning of the chains of the MCMC algorithm. Defaults to 1.

control  a list of additional arguments for the MCMC algorithm. Possible options are:
**autofit** Whether the models should be refitted until convergence. Defaults to FALSE.

**max_error** The target MCMC error for the autofit function. The argument is passed to `raftery.diag` as `'r'`. Defaults to .01.

**max_time** A string specifying the maximum fitting time in case of autofit. Defaults to Inf. Can be specified as a number and a unit (Acceptable units include 'seconds', 'minutes', 'hours', 'days', 'weeks', or the first letter(s) of each), i.e. "1hr".

**adapt** A number of iterations used for MCMC adaptation. Defaults to 1000.

**bridge_max_iter** Maximum number of iterations for the `bridge_sampler` function. Defaults to 10000.

**allow_max_error** Maximum allowed MCMC error for a model to be taken into consideration. The model will be removed from the ensemble if it fails to achieve the set MCMC error. Defaults to NULL - no model will be removed based on MCMC error.

**allow_max_rhat** Maximum allowed Rhat for a model to be taken into consideration. Model will be removed from the ensemble if it fails to achieve the set Rhat. Defaults to NULL - no model will be removed based on Rhat.

**allow_min_ESS** Minimum allowed ESS for a model to be taken into consideration. Model will be removed from the ensemble if it fails to achieve the set ESS. Defaults to NULL - no model will be removed based on ESS.

**allow_inc_theta** Whether the diagnostics for theta should be included into model removal decision. Defaults to NULL - only 'mu', 'tau', and 'omega' estimates will be taken into account.

**balance_prob** Whether the prior probability of the removed model should be redistributed to other models with the same type if possible (crossing of effect / heterogeneity / publication bias). Defaults to TRUE.

**silent** Whether all fitting messages should be suppressed. Defaults to FALSE. Ideal for getting rid of the "full precision may not have been achieved in pntfinal" warning that cannot be suppressed in any other way.

**boost** Whether the likelihood functions implemented using the boost C++ library should be used as the first option. The higher precision of boost allows to estimate models in difficult cases. Defaults to FALSE. The R distributions are used as default and boost is used only if they fail. Warning: the estimation using boost takes about twice as long.

**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 `check()` will not be available.

**seed** a seed to be set before model fitting, marginal likelihood computation, and posterior mixing for exact results reproducibility. Defaults to NULL - no seed is set.

**Details**

The default settings with either t-statistics / Cohen’s d effect sizes and sample sizes / standard errors correspond to the ensemble proposed by (Maier et al. 2020). The vignette("CustomEnsembles")
and vignette("ReproducingBMA") vignettes describe how to use `RoBMA()` to fit custom meta-analytic ensembles (see `prior()` for more information about prior distributions). To get help with the error and warning messages, see vignette("WarningsAndErrors").

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.

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()`, `prior()`, `check_setup()`

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)

# in order to speed up the process, we can reduce the default number of chains, iteration,
# and disable the autofit functionality (see ?RoBMA for all possible settings)
fit_faster <- RoBMA(r = Anderson2010$r, n = Anderson2010$n, study_names = Anderson2010$labels,
chains = 2, iter = 5000, control = list(autofit = FALSE))

# RoBMA function allows to use different prior specifications
# for example, change the prior for tau to be half normal and specify one-sided selection only
# on significant p-values (see '?prior' for all options regarding prior distributions)
fit1 <- RoBMA(r = Anderson2010$r, n = Anderson2010$n, study_names = Anderson2010$labels,
priors_tau = prior("normal",
parameters = list(mean = 0, sd = 1),
truncation = list(lower = 0, upper = Inf)),
priors_omega = prior("one-sided",
parameters = list(cuts = c(.05), alpha = c(1, 1))))

# the priors for the null models can be modified or even omitted in a similar manner,
# allowing to test different (non-null-null) hypotheses
fit2 <- RoBMA(r = Anderson2010$r, n = Anderson2010$n, study_names = Anderson2010$labels,
priors_mu_null = prior("normal",
parameters = list(mean = 0, sd = 1),
truncation = list(lower = 0, upper = Inf)))
```
parameters = list(mean = 0, sd = .1),
truncation = list(lower = -0.1, upper = 0.1)))

# an already fitted RoBMA model can be further updated or modified by using the update function
# for example, the prior model probabilities can be changed after the fitting by
# (but see ?update.RoBMA' for other posibilities including refitting or adding more models)
fit3 <- update(fit2, prior_odds = c(10,1,1,1,1,1,1,1,1,1,1,1))

# 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)

# results of the models can be visualized using the plot function (see ?plot.RoBMA for all options)
# for example, the model-averaged mean estimate
plot(fit, parameter = "mu")

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

## End(Not run)

summary.RoBMA

---

**Summary**

**RoBMA**

A fitted RoBMA model.

**Usage**

```r
summary.RoBMA(object, type = if (diagnostics) "models" else "ensemble", conditional = FALSE, diagnostics = FALSE, include_theta = FALSE, probs = c(0.025, 0.975), logBF = FALSE, BF01 = FALSE, digits_estimates = 3, digits_BF = 3, ...)
```

**Examples**

```r
params = list(mean = 0, sd = .1), truncation = list(lower = -0.1, upper = 0.1))
fit3 <- update(fit2, prior_odds = c(10,1,1,1,1,1,1,1,1,1,1,1))
summary(fit)
```

---

**Description**

*summary.RoBMA* creates a numerical summary of the RoBMA object.

**Usage**

```r
summary.RoBMA(object, type = if (diagnostics) "models" else "ensemble", conditional = FALSE, diagnostics = FALSE, include_theta = FALSE, probs = c(0.025, 0.975), logBF = FALSE, BF01 = FALSE, digits_estimates = 3, digits_BF = 3, ...)
```
Arguments

- **object**: a fitted RoBMA object.
- **type**: whether to show the overall RoBMA results ("ensemble"), an overview of the individual models ("models"), or detailed summary for the individual models ("individual").
- **conditional**: show the conditional estimates (assuming that the alternative is true). Defaults to FALSE. Only available for type == "conditional".
- **diagnostics**: show the maximum R-hat and minimum ESS for the main parameters in each of the models. Only available for type = "ensemble".
- **include_theta**: whether the estimated random effects should be included either in the summaries.
- **probs**: quantiles of the posterior samples to be displayed. Defaults to c(.025, .50, .975)
- **logBF**: show log of the BFs. Defaults to FALSE.
- **BF01**: show BF in support of the null hypotheses. Defaults to FALSE.
- **digits_estimates**: a number of decimals for rounding the estimates. Defaults to 3.
- **digits_BF**: a number of decimals for rounding the BFs. Defaults to 3.
- **...**: additional arguments

Value

- summary of a RoBMA object

Note

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

See Also

- `RoBMA()`
- `diagnostics()`

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)

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

# note that the summary function contains additional arguments
# that allow to obtain a specific output, i.e, the conditional estimates
# (assuming that the non-null models are true) can be obtained
summary(fit, conditional = TRUE)

# overview of the models and their prior and posterior probability, marginal likelihood,
```
update.RoBMA

# and inclusion Bayes factor:
summary(fit, type = "models")

# and the model diagnostics overview, containing maximum R-hat and minimum ESS across parameters
# but see '?diagnostics' for diagnostics plots for individual model parameters
summary(fit, type = "models", diagnostics = TRUE)

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

## End(Not run)

---

**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_odds`), see the vignette("CustomEnsembles") vignette,
2. change the prior odds of fitted models by specifying a vector `prior_odds` 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,
  prior_mu = NULL,
  prior_tau = NULL,
  prior_omega = NULL,
  prior_odds = NULL,
  prior_mu_null = NULL,
  prior_tau_null = NULL,
  prior_omega_null = NULL,
  study_names = NULL,
  control = NULL,
  chains = NULL,
  iter = NULL,
  burnin = NULL,
)```

---
thin = NULL,
...
)

Arguments

object a fitted RoBMA object.
refit_failed whether failed models should be refitted. Relevant only if new priors or prior_odds are not supplied. Defaults to TRUE.
prior_mu a prior distribution for the mu parameter that will be treated as belonging to the alternative hypothesis.
prior_tau a prior distribution for the tau parameter that will be treated as belonging to the alternative hypothesis.
prior_omega a prior weight function for the omega parameter that will be treated as belonging to the alternative hypothesis.
prior_odds either a single value specifying prior model odds of a newly specified model using priors argument, or a vector of the same length as already fitted models to update their prior odds.
prior_mu_null list of prior distribution for the mu parameter that will be treated as belonging to the null hypothesis. Defaults to point distribution with location at 0 (prior(distribution = "point",parameters = list(location = 0))).
prior_tau_null a prior distribution for the tau parameter that will be treated as belonging to the null hypothesis.
prior_omega_null a prior weight function for the omega parameter that will be treated as belonging to the null hypothesis.
study_names an optional argument with the names of the studies.
control a list of additional arguments for the MCMC algorithm. Possible options are:
  autofit Whether the models should be refitted until convergence. Defaults to FALSE
  max_error The target MCMC error for the autofit function. The argument is passed to raftery.diag as 'r'. Defaults to .01.
  max_time A string specifying the maximum fitting time in case of autofit. Defaults to Inf. Can be specified as a number and a unit (Acceptable units include 'seconds', 'minutes', 'hours', 'days', 'weeks', or the first letter(s) of each), i.e. "1hr".
  adapt A number of iterations used for MCMC adaptation. Defaults to 1000.
  bridge_max_iter Maximum number of iterations for the bridge_sampler function. Defaults to 10000
  allow_max_error Maximum allowed MCMC error for a model to be taken into consideration. The model will be removed from the ensemble if it fails to achieve the set MCMC error. Defaults to NULL - no model will be removed based on MCMC error.
allow_max_rhat  Maximum allowed Rhat for a model to be taken into consideration. Model will be removed from the ensemble if it fails to achieve the set Rhat. Defaults to NULL - no model will be removed based on Rhat.

allow_min_ESS  Minimum allowed ESS for a model to be taken into consideration. Model will be removed from the ensemble if it fails to achieve the set ESS. Defaults to NULL - no model will be removed based on ESS.

allow_inc_theta  Whether the diagnostics for theta should be included into model removal decision. Defaults to NULL - only 'mu', 'tau', and 'omega' estimates will be taken into account.

balance_prob  Whether the prior probability of the removed model should be redistributed to other models with the same type if possible (crossing of effect / heterogeneity / publication bias). Defaults to TRUE.

silent  Whether all fitting messages should be suppressed. Defaults to FALSE. Ideal for getting rid of the "full precision may not have been achieved in pntfinal" warning that cannot be suppressed in any other way.

boost  Whether the likelihood functions implemented using the boost C++ library should be used as the first option. The higher precision of boost allows to estimate models in difficult cases. Defaults to FALSE. The R distributions are used as default and boost is used only if they fail. Warning: the estimation using boost takes about twice as long.

chains  a number of chains of the MCMC algorithm.

iter  a number of sampling iterations of the MCMC algorithm. Defaults to 10000, with a minimum of 4000.

burnin  a number of burnin iterations of the MCMC algorithm. Defaults to 5000.

thin  a thinning of the chains of the MCMC algorithm. Defaults to 1.

Details

See RoBMA() for more details.

Value

RoBMA returns an object of class "RoBMA".

See Also

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

Examples

## Not run:
# using the example data from 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)

# the update function allows us to change the prior model probability of each model
fit1 <- update(fit, prior_odds = c(10,1,1,1,1,1,1,1,1,1,1))
# add an additional model with different priors specification (see 'prior' for more information)
fit2 <- update(fit,
  priors_mu_null = prior("point", parameters = list(location = 0)),
  priors_tau = prior("normal",
    parameters = list(mean = 0, sd = 1),
    truncation = list(lower = 0, upper = Inf)),
  priors_omega = prior("one-sided",
    parameters = list(cuts = c(.05), alpha = c(1, 1))))

# change the model convergence criteria to mark models with ESS lower than 2000 as non-covergent
fit3 <- update(fit, control = list(allow_min_ESS = 2000))

# and refit them failed models with increased number of burnin iterations
fit4 <- update(fit3, burnin = 10000)

## End(Not run)

---

**weightedt**

*Weighted t distribution*

**Description**

Density, distribution function, quantile function and random generation for the weighted t distribution with df degrees of freedom, non-centrality parameter ncp, steps steps (or critical t-values crit_t), and weights omega.

**Usage**

```r
dwt(
  x, 
  df, 
  ncp, 
  steps = if (!is.null(crit_t)) NULL, 
  omega, 
  crit_t = if (!is.null(steps)) NULL, 
  type = "two.sided", 
  log = FALSE
)
pwt(
  q, 
  df, 
  ncp, 
  steps = if (!is.null(crit_t)) NULL, 
  omega, 
```
crit_t = if (!is.null(steps)) NULL, type = "two.sided",
lower.tail = TRUE, log.p = FALSE)
qwt(
p, df, ncp,
steps = if (!is.null(crit_t)) NULL, omega,
crit_t = if (!is.null(steps)) NULL, type = "two.sided",
lower.tail = TRUE, log.p = FALSE)
rwt(
n, df, ncp,
steps = if (!is.null(crit_t)) NULL, omega,
crit_t = if (!is.null(steps)) NULL, type = "two.sided"
)

Arguments

x, q vector of quantiles.
df degrees of freedom (> 0, maybe non-integer). df = Inf is allowed.
ncp non-centrality parameter delta.
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_t vector of t-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 df, ncp, steps, omega can be supplied as a vectors (df, ncp) or matrices (steps, omega) with length / number of rows equal to x/q / p. Otherwise, they are recycled to the length of the result.

The functions quickly lose precision in the tails since they depend on sums of distribution functions of t distribution <stats::pt>. In cases where the density of t distribution cannot be computed by <stats::dt>, the implementation switches to <DPQ::dnt>.

See Also

Normal, dnt
Index

* datasets
  Anderson2010, 2
  autorun.jags, 16
  bquote, 10
  bridge_sampler, 15, 16, 20

Cauchy, 12
check(), 15
check_setup, 3
check_setup(), 16, 21
class, 16, 21

diagnostics, 4
diagnostics(), 16, 18
dnt, 24
DPQ::dnt, 24
dwt (weightedt), 22

GammaDist, 12

InvGamma, 12
is.RoBMA, 6

LocationScaleT, 12

Normal, 12, 24

par, 5, 7
plot.RoBMA, 6
plot.RoBMA(), 16
plot.RoBMA.prior, 8
plot.RoBMA.prior(), 12
print.RoBMA, 9
print.RoBMA(), 16
print.RoBMA.prior, 10
print.summary.RoBMA, 10
prior, 11
prior(), 4, 9, 10, 16, 21

pwt (weightedt), 22
qwt (weightedt), 22
raftery.diag, 15, 20
RoBMA, 12
RoBMA(), 4, 5, 7, 10–12, 16, 18, 21
rwt (weightedt), 22

stan_plot, 5
stats::dt, 24
stats::pt, 24
summary.RoBMA, 17
summary.RoBMA(), 5, 16, 21

update.RoBMA, 19
update.RoBMA(), 16

weightedt, 22