Package: metamixr (via r-universe)

October 22, 2026

Title Bayesian Meta-Analytic Mixture Modelling

Version 0.1.0

Description Fits Bayesian random-effects meta-analytic mixture models in which the true study effects are drawn from a mixture of normal components with component-specific means and between-study heterogeneity. The mixture model can be combined with step-function selection models (Vevea and Hedges, 1995, <doi:10.1007/BF02294384>) to adjust for publication bias, either with one selection model for all studies or with separate selection models for groups of studies. Models are estimated by Hamiltonian Monte Carlo using 'rstan' and returned as 'stanfit' objects, so posterior summaries, posterior predictive checks and marginal likelihoods for model comparison (via 'bridgesampling') are readily available. Also includes a data simulator and the nudging meta-analysis data of Mertens et al. (2022) <doi:10.1073/pnas.2107346118>.

License MIT + file LICENSE

URL https://github.com/MaxMaier42/metamix, https://maxmaier42.github.io/metamix/

BugReports https://github.com/MaxMaier42/metamix/issues

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Biarch true

Depends R (>= 3.5)

Imports methods, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.26.0), rstantools (>= 2.4.0), stats

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.26.0), StanHeaders (>= 2.26.0)

SystemRequirements GNU make

Suggests testthat (>= 3.0.0), bridgesampling, knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

LazyData true

Config/pak/sysreqs cmake make

Repository https://maxmaier42.r-universe.dev

Date/Publication 2026-10-09 16:58:50 UTC

RemoteUrl https://github.com/maxmaier42/metamix

RemoteRef HEAD

RemoteSha 9791f597b351539b0e8608ccdf4c56ff30f821a0

Contents

mertens_nudge 2
re_mix 4
sel_flexpb 7
sel_mix 10
sim_mix 13

Index **16**

`mertens_nudge` *Nudging Meta-Analysis Data (Mertens et al., 2022)*

Description

Effect sizes from the meta-analysis of choice-architecture ("nudge") interventions by Mertens, Herberz, Hahnel and Brosch (2022). The data set was the basis of the debate about publication bias in the nudging literature (Maier et al., 2022) and serves as the real-data example of this package. Each row is one effect size; several effect sizes may stem from the same study and several studies from the same publication.

Usage

```
mertens_nudge
```

Format

A data frame with 447 rows and 24 columns. The columns needed for model fitting are `cohens_d` and `variance_d`. The variable descriptions below are taken from the codebook of Mertens et al. (2022), available at https://osf.io/49wtf/files/dvg3w.

`publication_id` Publication ID.

`study_id` Study ID. Studies were defined based on independent samples; where an experiment compared multiple intervention conditions to the same control condition, a single study ID was assigned.

`es_id` Effect size ID.

`reference` Descriptive reference of publication (i.e., authors + year of publication).

`title` Title of publication.

`year` Year of publication.

`location` Geographical location of intervention: 0 = outside United States, 1 = inside United States.

`domain` Behavioral domain of intervention: `"health"`, `"food"`, `"environment"`, `"finance"`, `"pro-social"` or `"other"` (domains not covered by the other categories).

`intervention_category` Intervention category based on the taxonomy by Münscher et al. (2016): `"information"` (decision information), `"structure"` (decision structure) or `"assistance"` (decision assistance).

`intervention_technique` Intervention technique based on the taxonomy by Münscher et al. (2016): `"translation"` (translate information), `"visibility"` (make information visible), `"social_reference"` (provide social reference point), `"default"` (change choice default), `"effort"` (change option-related effort), `"composition"` (change range or composition of options), `"consequence"` (change option consequences), `"reminder"` (provide reminders) or `"commitment"` (facilitate commitment).

`type_experiment` Type of experiment as defined by Harrison and List (2004): `"conventional_lab"`, `"artefactual_field"`, `"framed_field"` or `"natural_field"`.

`population` Target population of intervention: 0 = children and/or adolescents, 1 = adults.

`n_study` Overall sample size of study.

`n_comparison` Sample size of control + intervention condition.

`n_control` Sample size of control condition.

`n_intervention` Sample size of intervention condition.

`binary_outcome` Scale of outcome variable: 0 = continuous, 1 = binary.

`mean_control` Mean of outcome variable in control condition.

`sd_control` Standard deviation of outcome variable in control condition.

`mean_intervention` Mean of outcome variable in intervention condition.

`sd_intervention` Standard deviation of outcome variable in intervention condition.

`cohens_d` Extracted effect size of intervention (Cohen's d).

`variance_d` Variance around extracted effect size of intervention.

`approximation` Approximation involved in effect size extraction: 0 = no, 1 = yes.

Source

Mertens, S., Herberz, M., Hahnel, U. J. J., & Brosch, T. (2022). The effectiveness of nudging: A meta-analysis of choice architecture interventions across behavioral domains. *Proceedings of the National Academy of Sciences*, 119(1), e2107346118. doi:10.1073/pnas.2107346118

References

Harrison, G. W., & List, J. A. (2004). Field experiments. *Journal of Economic Literature*, 42(4), 1009–1055.

Maier, M., Bartoš, F., Stanley, T. D., Shanks, D. R., Harris, A. J. L., & Wagenmakers, E.-J. (2022). No evidence for nudging after adjusting for publication bias. *Proceedings of the National Academy of Sciences*, 119(31), e2200300119. doi:10.1073/pnas.2200300119

Münscher, R., Vetter, M., & Scheuerle, T. (2016). A review and taxonomy of choice architecture techniques. *Journal of Behavioral Decision Making*, 29(5), 511–524. doi:10.1002/bdm.1897

Examples

```
data(mertens_nudge)
str(mertens_nudge[, c("domain", "cohens_d", "variance_d")])

# z-statistics pile up just above the two-sided significance threshold
z <- mertens_nudge$cohens_d / sqrt(mertens_nudge$variance_d)
hist(z, breaks = 60, main = "Nudging meta-analysis", xlab = "z")
abline(v = qnorm(0.975), lty = 2)
```

`re_mix` *Random-Effects Meta-Analytic Mixture Model*

Description

Fits a Bayesian random-effects meta-analysis in which the true study effects arise from a mixture of `M` normal components, each with its own mean and between-study heterogeneity (Maier, 2026). With `M = 1` the model is the ordinary random-effects meta-analysis. No publication bias adjustment is applied; see `sel_mix()` and `sel_flexpb()` for the corresponding selection models.

Usage

```
re_mix(
  y,
  sd,
  M,
  mu_sd = 1,
  tau_sd = 0.2,
  mean_prior = c("normal", "gap"),
  tau_prior = c("half_normal", "inv_gamma"),
  gap_meanlog = log(0.2),
  gap_sdlog = 0.3,
  gap_min = 0,
  tau_alpha = 2.5,
  tau_beta = 0.15,
  prior_only = FALSE,
  ...
)
```

Arguments

y	numeric vector of observed effect sizes.
sd	numeric vector of the standard errors of y (same length as y).
M	number of mixture components (a positive integer).
mu_sd	standard deviation of the normal prior on the component means (used by both mean-prior settings: as the prior on every ordered mean when mean_prior = "normal", and as the prior on the first mean when mean_prior = "gap"). Defaults to 1.
tau_sd	standard deviation of the half-normal prior on the heterogeneity SD tau (used when tau_prior = "half_normal"). Defaults to 0.2.
mean_prior	prior family for the component means. "normal" (default) places an independent normal(0, mu_sd) prior on each ordered mean. "gap" uses a lognormal prior on the positive gaps between adjacent means (controlled by gap_meanlog, gap_sdlog, gap_min), which discourages two components collapsing onto the same location.
tau_prior	prior family for the heterogeneity SD tau. "half_normal" (default) uses a half-normal(0, tau_sd) prior. "inv_gamma" places an inverse-gamma(tau_alpha, tau_beta) prior directly on tau.
gap_meanlog	log-scale location of the repulsive lognormal gap prior (only used when mean_prior = "gap"). Defaults to log(0.2).
gap_sdlog	log-scale standard deviation of the repulsive lognormal gap prior (only used when mean_prior = "gap"). Defaults to 0.3.
gap_min	hard lower floor on the gaps between adjacent means: each gap is constrained to be at least gap_min and the lognormal gap prior is truncated to [gap_min, Inf) (only used when mean_prior = "gap"). Defaults to 0.
tau_alpha	shape of the inverse-gamma prior on tau (only used when tau_prior = "inv_gamma"). Defaults to 2.5.
tau_beta	scale of the inverse-gamma prior on tau (only used when tau_prior = "inv_gamma"). Defaults to 0.15.
prior_only	if TRUE, the data are ignored and the prior distribution is sampled instead (useful for prior predictive checks).
...	further arguments passed to rstan::sampling(), e.g. chains, iter, cores, seed, refresh or control.

Details

See Maier (2026) for details of the model and the priors.

Value

An object of class stanfit as returned by rstan::sampling(). It contains posterior draws of

mu vector of length M: the ordered component means.

tau vector of length M: the between-study standard deviation of each component.

`theta` vector of length `M`: the mixing weights.

`mu1`, `mu_gap` the underlying parameterisation of the means (the first mean and the `M - 1` gaps between adjacent means).

`log_tau` `log(tau)`, convenient for funnel diagnostics.

`posterior_probs` `K` by `M` matrix: the posterior probability that study `i` belongs to component `m`.

`y_rep`, `sd_rep` posterior predictive draws of `K` effect sizes and the standard errors they were generated with. For each draw a component is sampled from `theta`, a study is sampled with probability proportional to its responsibility for that component, and a new effect size is generated using that study's standard error.

Use e.g. `print(fit, pars = c("mu", "tau", "theta"))`, `summary(fit)` or `rstan::extract()` to access them.

References

Maier, M. (2026). Addressing heterogeneity with Bayesian meta-analytic mixture modelling. *PsyArXiv*. doi:10.31234/osf.io/nkyqm_v2

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

Stan Development Team (2024). RStan: the R interface to Stan. https://mc-stan.org/

See Also

`sel_mix()` and `sel_flexpb()` for the same mixture model combined with step-function selection models; `sim_mix()` to simulate data from the model.

Other model fitting functions: `sel_flexpb()`, `sel_mix()`

Examples

```
# Simulate 40 studies from a two-component mixture without publication bias
set.seed(1)
dat <- sim_mix(K = 40, M = 2, mu = c(0, 0.6), tau = c(0.1, 0.1),
               steps = c(0.95, 0.975), weights = c(1, 1, 1), one_sided = TRUE)

# A single short chain keeps the example fast; use e.g. chains = 4 and
# iter = 2000 (the rstan defaults) for a real analysis.
fit <- re_mix(dat$y, dat$sds, M = 2, chains = 1, iter = 500, refresh = 0)
print(fit, pars = c("mu", "tau", "theta"))

# Posterior probability that each study belongs to the second component
resp <- rstan::extract(fit, pars = "posterior_probs")$posterior_probs
head(colMeans(resp)[, 2])

# Sample from the prior only (no data enter the model)
prior <- re_mix(dat$y, dat$sds, M = 2, prior_only = TRUE,
                chains = 1, iter = 500, refresh = 0)
print(prior, pars = c("mu", "tau"))
```

sel_flexpb *Meta-Analytic Mixture Model with Group-Specific Selection Models*

Description

Fits the meta-analytic mixture model with step-function selection of `sel_mix()`, but estimates a separate selection model (vector of selection weights `omega`) for each group of studies, e.g. for several meta-analyses that are analysed jointly, or for published versus preregistered studies. The mixture components (means, heterogeneity and mixing weights) are shared across groups. Groups that are known to be free of publication bias can be declared via `unselected`, in which case their selection weights are fixed at 1.

Usage

```
sel_flexpb(
  y,
  sd,
  grp,
  M,
  steps = c(0.9, 0.95),
  one_sided = TRUE,
  mu_sd = 1,
  tau_sd = 0.2,
  mean_prior = c("normal", "gap"),
  tau_prior = c("half_normal", "inv_gamma"),
  gap_meanlog = log(0.2),
  gap_sdlog = 0.3,
  gap_min = 0,
  tau_alpha = 2.5,
  tau_beta = 0.15,
  unselected = NULL,
  prior_only = FALSE,
  ...
)
```

Arguments

`y`	numeric vector of observed effect sizes.
`sd`	numeric vector of the standard errors of `y` (same length as `y`).
`grp`	integer vector of the same length as `y` giving the group (e.g. meta-analysis or study type) of each study, coded `1, ..., G`.
`M`	number of mixture components (a positive integer).
`steps`	cutoffs of the p-value intervals of the selection model, given as cumulative probabilities of the standard normal distribution (i.e. `qnorm(steps)` are the cutoffs on the z-statistic; `steps = c(0.95, 0.975)` corresponds to one-sided p-values of .05 and .025, or two-sided p-values of .10 and .05 in the expected direction).

One or two strictly increasing values in (0, 1) are supported. Defaults to `c(0.9, 0.95)`.

`one_sided` if `TRUE` (default) selection only operates on p-values in the expected direction (z-statistic `y/sd`); to model selection favouring negative effects, negate `y` before fitting. If `FALSE` selection is two-sided (`abs(y/sd)`), in which case all `steps` must exceed 0.5.

`mu_sd` standard deviation of the normal prior on the component means (used by both mean-prior settings: as the prior on every ordered mean when `mean_prior = "normal"`, and as the prior on the first mean when `mean_prior = "gap"`). Defaults to `1`.

`tau_sd` standard deviation of the half-normal prior on the heterogeneity SD `tau` (used when `tau_prior = "half_normal"`). Defaults to `0.2`.

`mean_prior` prior family for the component means. `"normal"` (default) places an independent normal(0, `mu_sd`) prior on each ordered mean. `"gap"` uses a lognormal prior on the positive gaps between adjacent means (controlled by `gap_meanlog`, `gap_sdlog`, `gap_min`), which discourages two components collapsing onto the same location.

`tau_prior` prior family for the heterogeneity SD `tau`. `"half_normal"` (default) uses a half-normal(0, `tau_sd`) prior. `"inv_gamma"` places an inverse-gamma(`tau_alpha`, `tau_beta`) prior directly on `tau`.

`gap_meanlog` log-scale location of the repulsive lognormal gap prior (only used when `mean_prior = "gap"`). Defaults to `log(0.2)`.

`gap_sdlog` log-scale standard deviation of the repulsive lognormal gap prior (only used when `mean_prior = "gap"`). Defaults to `0.3`.

`gap_min` hard lower floor on the gaps between adjacent means: each gap is constrained to be at least `gap_min` and the lognormal gap prior is truncated to `[gap_min, Inf)` (only used when `mean_prior = "gap"`). Defaults to `0`.

`tau_alpha` shape of the inverse-gamma prior on `tau` (only used when `tau_prior = "inv_gamma"`). Defaults to `2.5`.

`tau_beta` scale of the inverse-gamma prior on `tau` (only used when `tau_prior = "inv_gamma"`). Defaults to `0.15`.

`unselected` integer vector of group indices assumed NOT subject to publication bias: their selection weights `omega` are fixed to 1, so the likelihood for their studies reduces to the no-selection model, and the selection model is estimated only for the remaining groups (e.g. `unselected = 1` if group 1 is an unselected registry sample). Defaults to `NULL` (selection estimated for every group).

`prior_only` if `TRUE`, the data are ignored and the prior distribution is sampled instead (useful for prior predictive checks).

`...` further arguments passed to `rstan::sampling()`, e.g. `chains`, `iter`, `cores`, `seed`, `refresh` or `control`.

Details

See Maier (2026) for details of the model and the priors.

Value

An object of class `stanfit` as returned by `rstan::sampling()`. It contains posterior draws of

`mu` vector of length `M`: the ordered component means.

`tau` vector of length `M`: the between-study standard deviation of each component.

`theta` vector of length `M`: the mixing weights among the published studies.

`theta_preselection` vector of length `M`: the mixing weights corrected for selection, i.e. the estimated proportions of the components among all *conducted* studies rather than among the published ones.

`omega` `G` by `length(steps) + 1` matrix: one row of relative publication probabilities of the p-value intervals per group, non-decreasing with the last element fixed at 1. Rows of groups listed in `unselected` are constant at 1.

`omega_raw` the underlying simplex of each group, `omega[g, ] = cumsum(omega_raw[g, ])`.

`avg_omega` vector of length `M`: the average selection weight of the studies attributed to each component, i.e. how strongly each component is affected by selection.

`mu1`, `mu_gap` the underlying parameterisation of the means (the first mean and the `M - 1` gaps between adjacent means).

`log_tau` `log(tau)`, convenient for funnel diagnostics.

`posterior_probs` `K` by `M` matrix: the posterior probability that study `i` belongs to component `m`.

`y_rep`, `sd_rep` posterior predictive draws of `K` effect sizes and the standard errors they were generated with.

Use e.g. `print(fit, pars = c("mu", "tau", "omega"))`, `summary(fit)` or `rstan::extract()` to access them.

References

Maier, M. (2026). Addressing heterogeneity with Bayesian meta-analytic mixture modelling. *PsyArXiv*. doi:10.31234/osf.io/nkyqm_v2

Vevea, J. L., & Hedges, L. V. (1995). A general linear model for estimating effect size in the presence of publication bias. *Psychometrika*, 60(3), 419–435. doi:10.1007/BF02294384

See Also

`sel_mix()` for a single selection model, `re_mix()` for no selection, `sim_mix()` to simulate data.

Other model fitting functions: `re_mix()`, `sel_mix()`

Examples

```
# Two sets of studies on the same effect: a published literature subject to
# selection (group 1) and preregistered replications that are not (group 2)
set.seed(1)
lit <- sim_mix(K = 15, M = 1, mu = 0.3, tau = 0.1, steps = c(0.95, 0.975),
               weights = c(0.2, 0.5, 1), one_sided = TRUE)
rep <- sim_mix(K = 15, M = 1, mu = 0.3, tau = 0.1, steps = c(0.95, 0.975),
               weights = c(1, 1, 1), one_sided = TRUE)
```

```
y   <- c(lit$y, rep$y)
sd  <- c(lit$sds, rep$sds)
grp <- rep(1:2, each = 15)

# A single short chain keeps the example fast; use e.g. chains = 4 and
# iter = 2000 (the rstan defaults) for a real analysis. Group 2 is known to
# be unselected, so a selection model is estimated for group 1 only.
fit <- sel_flexpb(y, sd, grp, M = 1, steps = c(0.95, 0.975), unselected = 2,
                  chains = 1, iter = 400, refresh = 0)
print(fit, pars = c("mu", "tau", "omega"))
```

---

sel_mix *Meta-Analytic Mixture Model with a Step-Function Selection Model*

Description

Fits the random-effects meta-analytic mixture model of `re_mix()` combined with a step-function (weight-function) selection model to adjust for publication bias (Maier, 2026). One common selection model is estimated for all studies. Use `sel_flexpb()` to estimate separate selection models for groups of studies.

Usage

```
sel_mix(
  y,
  sd,
  M,
  steps = c(0.9, 0.95),
  one_sided = TRUE,
  mu_sd = 1,
  tau_sd = 0.2,
  mean_prior = c("normal", "gap"),
  tau_prior = c("half_normal", "inv_gamma"),
  gap_meanlog = log(0.2),
  gap_sdlog = 0.3,
  gap_min = 0,
  tau_alpha = 2.5,
  tau_beta = 0.15,
  prior_only = FALSE,
  ...
)
```

Arguments

y	numeric vector of observed effect sizes.
sd	numeric vector of the standard errors of y (same length as y).

M	number of mixture components (a positive integer).
steps	cutoffs of the p-value intervals of the selection model, given as cumulative probabilities of the standard normal distribution (i.e. `qnorm(steps)` are the cutoffs on the z-statistic; `steps = c(0.95, 0.975)` corresponds to one-sided p-values of .05 and .025, or two-sided p-values of .10 and .05 in the expected direction). One or two strictly increasing values in (0, 1) are supported. Defaults to `c(0.9, 0.95)`.
one_sided	if `TRUE` (default) selection only operates on p-values in the expected direction (z-statistic `y/sd`); to model selection favouring negative effects, negate `y` before fitting. If `FALSE` selection is two-sided (`abs(y/sd)`), in which case all `steps` must exceed 0.5.
mu_sd	standard deviation of the normal prior on the component means (used by both mean-prior settings: as the prior on every ordered mean when `mean_prior = "normal"`, and as the prior on the first mean when `mean_prior = "gap"`). Defaults to `1`.
tau_sd	standard deviation of the half-normal prior on the heterogeneity SD `tau` (used when `tau_prior = "half_normal"`). Defaults to `0.2`.
mean_prior	prior family for the component means. `"normal"` (default) places an independent normal(0, `mu_sd`) prior on each ordered mean. `"gap"` uses a lognormal prior on the positive gaps between adjacent means (controlled by `gap_meanlog`, `gap_sdlog`, `gap_min`), which discourages two components collapsing onto the same location.
tau_prior	prior family for the heterogeneity SD `tau`. `"half_normal"` (default) uses a half-normal(0, `tau_sd`) prior. `"inv_gamma"` places an inverse-gamma(`tau_alpha`, `tau_beta`) prior directly on `tau`.
gap_meanlog	log-scale location of the repulsive lognormal gap prior (only used when `mean_prior = "gap"`). Defaults to `log(0.2)`.
gap_sdlog	log-scale standard deviation of the repulsive lognormal gap prior (only used when `mean_prior = "gap"`). Defaults to `0.3`.
gap_min	hard lower floor on the gaps between adjacent means: each gap is constrained to be at least `gap_min` and the lognormal gap prior is truncated to `[gap_min, Inf)` (only used when `mean_prior = "gap"`). Defaults to `0`.
tau_alpha	shape of the inverse-gamma prior on `tau` (only used when `tau_prior = "inv_gamma"`). Defaults to `2.5`.
tau_beta	scale of the inverse-gamma prior on `tau` (only used when `tau_prior = "inv_gamma"`). Defaults to `0.15`.
prior_only	if `TRUE`, the data are ignored and the prior distribution is sampled instead (useful for prior predictive checks).
...	further arguments passed to `rstan::sampling()`, e.g. `chains`, `iter`, `cores`, `seed`, `refresh` or `control`.

Details

See Maier (2026) for details of the model and the priors.

Value

An object of class `stanfit` as returned by `rstan::sampling()`. It contains posterior draws of

`mu` vector of length `M`: the ordered component means.

`tau` vector of length `M`: the between-study standard deviation of each component.

`theta` vector of length `M`: the mixing weights among the published studies.

`theta_preselection` vector of length `M`: the mixing weights corrected for selection, i.e. the estimated proportions of the components among all *conducted* studies rather than among the published ones.

`omega` vector of length `length(steps) + 1`: the relative publication probabilities of the p-value intervals, non-decreasing with the last element fixed at 1.

`omega_raw` the underlying simplex, `omega = cumsum(omega_raw)`.

`avg_omega` vector of length `M`: the average selection weight of the studies attributed to each component, i.e. how strongly each component is affected by selection.

`mu1`, `mu_gap` the underlying parameterisation of the means (the first mean and the `M - 1` gaps between adjacent means).

`log_tau` `log(tau)`, convenient for funnel diagnostics.

`posterior_probs` `K` by `M` matrix: the posterior probability that study `i` belongs to component `m`.

`y_rep`, `sd_rep` posterior predictive draws of `K` effect sizes and the standard errors they were generated with.

Use e.g. `print(fit, pars = c("mu", "tau", "omega"))`, `summary(fit)` or `rstan::extract()` to access them.

References

Maier, M. (2026). Addressing heterogeneity with Bayesian meta-analytic mixture modelling. *PsyArXiv*. doi:10.31234/osf.io/nkyqm_v2

Vevea, J. L., & Hedges, L. V. (1995). A general linear model for estimating effect size in the presence of publication bias. *Psychometrika*, 60(3), 419–435. doi:10.1007/BF02294384

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

See Also

`re_mix()` for the model without selection, `sel_flexpb()` for group-specific selection models, `sim_mix()` to simulate data.

Other model fitting functions: `re_mix()`, `sel_flexpb()`

Examples

```
# 30 studies with a true mean of 0.3 under one-sided selection: studies that
# are not significant in the expected direction (two-sided p > .10) are
# published with probability .2, marginally significant ones with .5
set.seed(1)
dat <- sim_mix(K = 30, M = 1, mu = 0.3, tau = 0.1, steps = c(0.95, 0.975),
               weights = c(0.2, 0.5, 1), one_sided = TRUE)

# A single short chain keeps the example fast; use e.g. chains = 4 and
# iter = 2000 (the rstan defaults) for a real analysis.
fit <- sel_mix(dat$y, dat$sds, M = 1, steps = c(0.95, 0.975),
               chains = 1, iter = 400, refresh = 0)
print(fit, pars = c("mu", "tau", "omega"))
```

---

sim_mix — *Simulate Meta-Analytic Data from a Mixture Model with Publication Bias*

Description

Simulates K published studies from the random-effects mixture model of re_mix() combined with the step-function selection model of sel_mix(). Studies are generated until K of them have passed the selection step, so the returned data set has exactly K rows.

Usage

```
sim_mix(
  K,
  M,
  mu,
  tau,
  steps,
  weights,
  one_sided,
  thetas = c(rep(1/M, M)),
  N_low = 25,
  N_high = 500,
  N_shape = 2,
  N_scale = 58
)
```

Arguments

K	number of (published) studies to return.
M	number of mixture components.
mu	numeric vector of length M: the mean true effect of each component.

tau	numeric vector of length M: the between-study standard deviation of the true effects within each component.
steps	cutoffs of the p-value intervals, in the same format as in sel_mix() (cumulative probabilities of the standard normal distribution; e.g. c(0.95, 0.975) for one-sided p-values of .05 and .025, i.e. two-sided .10 and .05 in the expected direction). Strictly increasing values in (0, 1).
weights	publication probabilities of the p-value intervals, a vector of length length(steps) + 1 with values in [0, 1]. The first element applies to the least significant interval, e.g. c(0.2, 0.5, 1).
one_sided	if TRUE selection depends on the one-sided p-value (z-statistic y/sd), if FALSE on the two-sided p-value (abs(y/sd)).
thetas	mixing weights of the components; defaults to equal weights.
N_low	lower bound on the total sample size of a primary study.
N_high	upper bound on the total sample size of a primary study.
N_shape	shape (size) of the negative binomial distribution used to generate sample sizes (see Maier et al., 2023).
N_scale	scale of the negative binomial distribution used to generate sample sizes; the success probability is 1 / (N_scale + 1) (see Maier et al., 2023).

Details

See Maier (2026) for details of the model and the simulation design.

Value

A data frame with K rows and the columns y (observed effect size) and sds (its standard error), ready to be passed to re_mix(), sel_mix() or sel_flexpb().

References

Maier, M. (2026). Addressing heterogeneity with Bayesian meta-analytic mixture modelling. *PsyArXiv*. doi:10.31234/osf.io/nkyqm_v2

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

See Also

re_mix(), sel_mix(), sel_flexpb()

Examples

```
set.seed(1)
# Two components (null effect and medium effect) without publication bias
dat <- sim_mix(K = 100, M = 2, mu = c(0, 0.5), tau = c(0.05, 0.1),
               steps = c(0.95, 0.975), weights = c(1, 1, 1), one_sided = TRUE)
head(dat)
```

```
# Strong one-sided selection: studies that are not significant in the
# expected direction are published with probability .1 only
dat_pb <- sim_mix(K = 100, M = 1, mu = 0, tau = 0.1, steps = c(0.95, 0.975),
                  weights = c(0.1, 0.5, 1), one_sided = TRUE)
mean(dat_pb$y / dat_pb$sds > qnorm(0.975))
```

Index

∗ **datasets**
    mertens_nudge, 2
∗ **model fitting functions**
    re_mix, 4
    sel_flexpb, 7
    sel_mix, 10

mertens_nudge, 2

re_mix, 4, *9*, *12*
re_mix(), *9*, *10*, *12–14*
rstan::extract(), *6*, *9*, *12*
rstan::sampling(), *5*, *8*, *9*, *11*, *12*

sel_flexpb, *6*, 7, *12*
sel_flexpb(), *4*, *6*, *10*, *12*, *14*
sel_mix, *6*, *9*, 10
sel_mix(), *4*, *6*, *7*, *9*, *13*, *14*
sim_mix, 13
sim_mix(), *6*, *9*, *12*
stanfit, *5*, *9*, *12*