Package 'RBesT'

August 4, 2026

Type Package

Title R Bayesian Evidence Synthesis Tools

Description Tool-set to support Bayesian evidence synthesis. This includes meta-analysis, (robust) prior derivation from historical data, operating characteristics and analysis (1 and 2 sample cases). Please refer to Weber et al. (2021) <doi:10.18637/jss.v100.i19> for details on applying this package while Neuenschwander et al. (2010) <doi:10.1177/1740774509356002> and Schmidli et al. (2014) <doi:10.1111/biom.12242> explain details on the methodology.

Version 1.11-0

Depends R (>= 3.5.0)

Imports methods, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.32.0), rstantools (>= 2.4.0), posterior (>= 1.5.0), assertthat, mvtnorm, Formula, checkmate, bayesplot (>= 1.4.0), ggplot2, dplyr, stats, utils, matrixStats, statmod, abind, rlang, jsonlite, lifecycle, Rdpack

LinkingTo BH (>= 1.72.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.32.0), StanHeaders (>= 2.32.0)

License GPL (>= 3)

LazyData true

Biarch true

NeedsCompilation yes

UseLTO true

URL https://opensource.nibr.com/RBesT/

BugReports https://github.com/Novartis/RBesT/issues

Suggests rmarkdown, knitr, MASS, testthat (>= 2.0.0), foreach, purrr, rstanarm (>= 2.17.2), scales, tools, broom, tidyr, parallel, brms, glue, ragg, withr, mirai, parallelly

VignetteBuilder knitr

SystemRequirements GNU make, pandoc (>= 1.12.3), pngquant, C++17

Encoding UTF-8

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

RdMacros Rdpack

Author Novartis Pharma AG [cph],
Sebastian Weber [aut, cre],
Beat Neuenschwander [ctb],
Heinz Schmidli [ctb],
Baldur Magnusson [ctb],
Yue Li [ctb],
Satrajit Roychoudhury [ctb],
Lukas A. Widmer [ctb] (ORCID: <https://orcid.org/0000-0003-1471-3493>),
Daniel Sabanés Bové [ctb] (ORCID:
<https://orcid.org/0000-0002-0176-9239>),
Trustees of Columbia University [cph] (R/stanmodels.R, configure,
configure.win)

Maintainer Sebastian Weber <sebastian.weber@novartis.com>

Repository CRAN

Date/Publication 2026-08-04 16:40:09 UTC

Contents

RBesT-package . . . 3
AS . . . 5
asthma . . . 6
automixfit . . . 7
BinaryExactCI . . . 8
colitis . . . 9
crohn . . . 10
decision1S . . . 11
decision1S_boundary . . . 14
decision2S . . . 16
decision2S_boundary . . . 19
draws-RBesT . . . 22
ess . . . 24
forest_plot . . . 27
gMAP . . . 29
likelihood . . . 37
lodds . . . 38
mix . . . 39
mixbeta . . . 41
mixcombine . . . 43
mixdiff . . . 44
mixfit . . . 45

mixgamma 48
mixjson 50
mixmvnorm 51
mixnorm 53
mixplot 55
mixstanvar 57
nsamples.gMAP 59
oc1S 60
oc2S 63
plot.EM 67
plot.gMAP 68
pos1S 69
pos2S 71
postmix 74
preddist 77
predict.gMAP 80
robustify 82
transplant 84

Index **85**

`RBesT-package` *R Bayesian Evidence Synthesis Tools*

Description

The RBesT tools are designed to support in the derivation of parametric informative priors, asses design characeristics and perform analyses. Supported endpoints include normal, binary and Poisson.

Details

For introductory material, please refer to the vignettes which include

- Introduction (binary)
- Introduction (normal)
- Customizing RBesT Plots
- Robust MAP, advanced usage

The main function of the package is `gMAP()`. See it's help page for a detailed description of the statistical model.

Global Options

Option	Default	Description
RBesT.MC.warmup	2000	MCMC warmup iterations
RBesT.MC.iter	6000	total MCMC iterations
RBesT.MC.chains	4	MCMC chains
RBesT.MC.thin	4	MCMC thinning
RBesT.MC.save_warmup	FALSE	MCMC warmup samples saving
RBesT.MC.control	list(adapt_delta=0.99, stepsize=0.01, max_treedepth=20)	sets control argument for Stan call
RBesT.MC.ncp	1	parametrization: 0=CP, 1=NCP, 2=Automatic
RBesT.MC.init	1	range of initial uniform $[-1, 1]$ is the default
RBesT.MC.rescale	TRUE	Automatic rescaling of raw parameters
RBesT.verbose	FALSE	requests outputs to be more verbose
RBesT.integrate_args	list(lower=-Inf, upper=Inf, rel.tol=.Machine$double.eps^0.25, abs.tol=.Machine$double.eps^0.25, subdivisions=1E3)	arguments passed to integrate for adaptive integration of densities (used when RBesT.integrate_method is "adaptive" or for non-normMix densities)
RBesT.integrate_prob_eps	1E-6	probability mass left out from tails if integratio
RBesT.integrate_method	"GQ"	integration method for mixture densities: "GQ"
RBesT.GQ_nodes	20	starting number of Gaussian quadrature nodes
RBesT.GQ_rel_tol	1E-4	relative tolerance target for GQ refinement; the
RBesT.GQ_abs_tol	1E-6	absolute tolerance floor for GQ refinement
RBesT.GQ_max_nodes	240	upper cap on the GQ node count during refiner
RBesT.GQ_node_growth	2	multiplicative growth factor for the GQ node c
RBesT.GQ_on_nonconvergence	"adaptive"	behaviour when GQ refinement reaches RBesT
RBesT.decision2S_boundary	"adaptive"	tracing scheme for the decision boundary in de

Version History

See NEWS.md file.

Author(s)

Maintainer: Sebastian Weber <sebastian.weber@novartis.com>

Authors:

- Sebastian Weber <sebastian.weber@novartis.com>

Other contributors:

- Novartis Pharma AG [copyright holder]
- Beat Neuenschwander <beat.neuenschwander@novartis.com> [contributor]
- Heinz Schmidli <heinz.schmidli@novartis.com> [contributor]

- Baldur Magnusson <baldur.magnusson@novartis.com> [contributor]
- Yue Li <yue-1.li@novartis.com> [contributor]
- Satrajit Roychoudhury <satrajit.roychoudhury@novartis.com> [contributor]
- Lukas A. Widmer <lukas_andreas.widmer@novartis.com> (ORCID) [contributor]
- Daniel Sabanés Bové <daniel.sabanes_bove@rconis.com> (ORCID) [contributor]
- Trustees of Columbia University (R/stanmodels.R, configure, configure.win) [copyright holder]

References

Stan Development Team (2020). *RStan: the R interface to Stan*. R package version 2.19.3, https://mc-stan.org.

See Also

Useful links:

- https://opensource.nibr.com/RBesT/
- Report bugs at https://github.com/Novartis/RBesT/issues

AS *Ankylosing Spondylitis.*

Description

Data set containing historical information for placebo for a phase II trial of ankylosing spondylitis patients. The primary efficacy endpoint was the percentage of patients with a 20% response according to the Assessment of SpondyloArthritis international Society criteria for improvement (ASAS20) at week 6.

Usage

```
AS
```

Format

A data frame with 8 rows and 3 variables:

study study

n study size

r number of events

References

Baeten D, others (2013). "Anti-interleukin-17A monoclonal antibody secukinumab in treatment of ankylosing spondylitis: a randomised, double-blind, placebo-controlled trial." *The Lancet*, **382**(9906), 1705–1713.

Examples

```
## Setting up dummy sampling for fast execution of example
## Please use 4 chains and 20x more warmup & iter in practice
.user_mc_options <- options(RBesT.MC.warmup=50, RBesT.MC.iter=100,
                            RBesT.MC.chains=2, RBesT.MC.thin=1)

set.seed(34563)
map_AS <- gMAP(cbind(r, n - r) ~ 1 | study,
  family = binomial,
  data = AS,
  tau.dist = "HalfNormal", tau.prior = 1,
  beta.prior = 2
)
## Recover user set sampling defaults
options(.user_mc_options)
```

---

asthma *Asthma exacerbation recurrent event data.*

Description

Data set containing historical information for placebo arms of relevant trials for the treatment of asthma. The primary outcome is the rate of asthma exacerbations, a recurrent event modelled with a negative binomial distribution. The full data set as published in Holzhauer, Wang & Schmidli (2018) summarizes ten historical placebo arms by their back-calculated log mean event rate and dispersion together with the associated standard errors on the log scale.

Usage

```
asthma
```

Format

A data frame with 10 rows and 11 variables:

study study label

NCT ClinicalTrials.gov / ISRCTN registry identifier(s)

d follow-up (exposure) duration in years

n study size

mu_hat estimated mean event rate

log_mu_hat log mean event rate

se_log_mu_hat standard error of the log mean event rate

kappa_hat estimated dispersion parameter

log_kappa_hat log dispersion parameter

se_log_kappa_hat standard error of the log dispersion parameter

phase development phase of the trial

References

Holzhauer B, Wang C, Schmidli H (2018). "Historical control information for clinical trials with a recurrent event endpoint." *Statistics in Medicine*, **37**(10), 1640–1657.

Examples

```
## Setting up dummy sampling for fast execution of example
## Please use 4 chains and 20x more warmup & iter in practice
.user_mc_options <- options(RBesT.MC.warmup=50, RBesT.MC.iter=100,
                            RBesT.MC.chains=2, RBesT.MC.thin=1)

set.seed(34563)
asthma_ph3 <- subset(asthma, phase == "phase III")
map_asthma <- gMAP(cbind(log_mu_hat, se_log_mu_hat) ~ 1 + offset(log(d)) | study,
  family = gaussian,
  data = asthma_ph3,
  tau.dist = "HalfNormal", tau.prior = 0.5,
  beta.prior = cbind(0, 2)
)
## Recover user set sampling defaults
options(.user_mc_options)
```

`automixfit` *Automatic Fitting of Mixtures of Conjugate Distributions to a Sample*

Description

Fitting a series of mixtures of conjugate distributions to a `sample`, using Expectation-Maximization (EM). The number of mixture components is specified by the vector `Nc`. First a `Nc[1]` component mixture is fitted, then a `Nc[2]` component mixture, and so on. The mixture providing the best AIC value is then selected.

Usage

```
automixfit(sample, Nc = seq(1, 4), k = 6, thresh = -Inf, verbose = FALSE, ...)
```

Arguments

`sample`	Sample to be fitted by a mixture distribution.
`Nc`	Vector of mixture components to try out (default `seq(1,4)`).
`k`	Penalty parameter for AIC calculation (default 6)
`thresh`	The procedure stops if the difference of subsequent AIC values is smaller than this threshold (default -Inf). Setting the threshold to 0 stops `automixfit` once the AIC becomes worse.
`verbose`	Enable verbose logging.
`...`	Further arguments passed to `mixfit()`, including `type`.

Details

The type argument specifies the distribution of the mixture components, and can be a normal, beta or gamma distribution.

The penalty parameter k is 2 for the standard AIC definition. *Collet (2003)* suggested to use values in the range from 2 to 6, where larger values of k penalize more complex models. To favor mixtures with fewer components a value of 6 is used as default.

Value

As result the best fitting mixture model is returned, i.e. the model with lowest AIC. All other models are saved in the attribute models.

References

Collett D (2003). *Modelling Survival Data in Medical Research*, 2nd edition. Chapman and Hall/CRC.

Examples

```
# random sample of size 1000 from a mixture of 2 beta components
bm <- mixbeta(beta1 = c(0.4, 20, 90), beta2 = c(0.6, 35, 65))
bmSamp <- rmix(bm, 1000)

# fit with EM mixture models with up to 10 components and stop if
# AIC increases
bmFit <- automixfit(bmSamp, Nc = 1:10, thresh = 0, type = "beta")
bmFit

# advanced usage: find out about all discarded models
bmFitAll <- attr(bmFit, "models")

sapply(bmFitAll, AIC, k = 6)
```

BinaryExactCI *Exact Confidence interval for Binary Proportion*

Description

This function calculates the exact confidendence interval for a response rate presented by n and r.

Usage

```
BinaryExactCI(r, n, alpha = 0.05, drop = TRUE)
```

Arguments

r	Number of success or responder
n	Sample size
alpha	confidence level
drop	Determines if drop() will be called on the result

Details

Confidence intervals are obtained by a procedure first given in Clopper and Pearson (1934). This guarantees that the confidence level is at least (1-α).

Details can be found in the publication listed below.

Value

100 (1-α)\ response rate

References

Clopper CJ, Pearson ES (1934). "The use of confidence or fiducial limits illustrated in the case of the binomial." *Biometrika*, **26**(4), 404–413.

Examples

```
BinaryExactCI(3, 20, 0.05)
```

---

colitis *Ulcerative Colitis.*

Description

Data set containing historical information for placebo arm of a phase II proof-of-concept trial for the treatment of ulcerative colitis. The primary outcome is remission at week 8 (binary).

Usage

```
colitis
```

Format

A data frame with 4 rows and 3 variables:

study study

n study size

r number of events

References

Neuenschwander B, Capkun-Niggli G, Branson M, Spiegelhalter DJ (2010). "Summarizing historical information on controls in clinical trials." *Clinical Trials*, **7**(1), 5–18. doi:10.1177/1740774509356002.

`crohn` *Crohn's disease.*

Description

Data set containing historical information for placebo arm of relevant studies for the treatment of Crohn's disease. The primary outcome is change from baseline in Crohn's Disease Activity Index (CDAI) over a duration of 6 weeks. Standard deviation of change from baseline endpoint is approximately 88.

Usage

```
crohn
```

Format

A data frame with 4 rows and 3 variables:

study study

n study size

y mean CDAI change

References

Hueber W, Sands BE, Lewitzky S, Vandemeulebroecke M, others (2012). "Secukinumab, a human anti-IL-17A monoclonal antibody, for moderate to severe Crohn's disease." *Gut*, **61**(12), 1693–1700.

Examples

```
## Setting up dummy sampling for fast execution of example
## Please use 4 chains and 20x more warmup & iter in practice
.user_mc_options <- options(RBesT.MC.warmup=50, RBesT.MC.iter=100,
                            RBesT.MC.chains=2, RBesT.MC.thin=1)

set.seed(546346)
map_crohn <- gMAP(cbind(y, y.se) ~ 1 | study,
  family = gaussian,
  data = transform(crohn, y.se = 88 / sqrt(n)),
  weights = n,
  tau.dist = "HalfNormal", tau.prior = 44,
  beta.prior = cbind(0, 88)
)
## Recover user set sampling defaults
options(.user_mc_options)
```

decision1S *Decision Function for 1 Sample Designs*

Description

The function sets up a 1 sample decision function with an arbitrary number of conditions.

Usage

```
decision1S(pc = 0.975, qc = 0, lower.tail = TRUE)

has_lower(x)

has_upper(x)

lower(x)

upper(x)

oc1Sdecision(pc = 0.975, qc = 0, lower.tail = TRUE)
```

Arguments

pc	Vector of critical cumulative probabilities.
qc	Vector of respective critical values. Must match the length of pc.
lower.tail	Logical; if TRUE (default), probabilities are $P(X \leq x)$, otherwise, $P(X > x)$. Either length 1 or same length as pc.
x	Two-sided decision function.

Details

For lower.tail being either TRUE or FALSE, the function creates a one-sided decision function which takes two arguments. The first argument is expected to be a mixture (posterior) distribution. This distribution is tested whether it fulfills all the required threshold conditions specified with the pc and qc arguments and returns 1 if all conditions are met and 0 otherwise. Hence, for lower.tail=TRUE condition i is equivalent to

$$P(\theta \leq q_{c,i}) > p_{c,i}$$

and the decision function is implemented as indicator function on the basis of the heavy-side step function $H(x)$ which is 0 for $x \leq 0$ and 1 for $x > 0$. As all conditions must be met, the final indicator function returns

$$\Pi_i H_i(P(\theta \leq q_{c,i}) - p_{c,i}).$$

For the case of a boolean vector given to `lower.tail` the direction of each decision aligns respectively, and a two-sided decision function is created.

When the second argument is set to `TRUE` a distance metric is returned component-wise per defined condition as

$$D_i = \log(P(\theta < q_{c,i})) - \log(p_{c,i}).$$

These indicator functions can be used as input for 1-sample boundary, OC or PoS calculations using `oc1S()` or `pos1S()` .

Value

The function returns a decision function (of class `decision1S_1sided` for one-sided, and of class `decision1S_2sided` for two-sided decisions) which takes two arguments. The first argument is expected to be a mixture (posterior) distribution which is tested if the specified conditions are met. The logical second argument determines if the function acts as an indicator function or if the function returns the distance from the decision boundary for each condition in log-space, i.e. the distance is 0 at the decision boundary, negative for a 0 decision and positive for a 1 decision.

For two-sided decision functions, the two components can be extracted with functions `lower()` and `upper()`. The distance as calculated by the decision function is returned as a list with components `lower` and `upper`.

Functions

- `oc1Sdecision()`: **[Deprecated]** Deprecated old function name. Please use `decision1S` instead.

References

Neuenschwander B, Rouyrre N, Hollaender H, Zuber E, Branson M (2011). "A proof of concept phase II non-inferiority criterion." *Statistics in Medicine*, **30**(13), 1618–1627.

See Also

Other design1S: `decision1S_boundary()`, `oc1S()`, `pos1S()`

Examples

```
# see Neuenschwander et al., 2011

# example is for a time-to-event trial evaluating non-inferiority (NI)
# using a normal approximation for the log-hazard ratio

# reference scale
s <- 2
theta_ni <- 0.4
theta_a <- 0
alpha <- 0.05
beta <- 0.2
za <- qnorm(1 - alpha)
```

```
zb <- qnorm(1 - beta)
n1 <- round((s * (za + zb) / (theta_ni - theta_a))^2) # n for which design was intended
nL <- 233
c1 <- theta_ni - za * s / sqrt(n1)

# flat prior
flat_prior <- mixnorm(c(1, 0, 100), sigma = s)

# standard NI design
decA <- decision1S(1 - alpha, theta_ni, lower.tail = TRUE)

# for double criterion with indecision point (mean estimate must be
# lower than this)
theta_c <- c1

# double criterion design
# statistical significance (like NI design)
dec1 <- decision1S(1 - alpha, theta_ni, lower.tail = TRUE)
# require mean to be at least as good as theta_c
dec2 <- decision1S(0.5, theta_c, lower.tail = TRUE)
# combination
decComb <- decision1S(c(1 - alpha, 0.5), c(theta_ni, theta_c), lower.tail = TRUE)

theta_eval <- c(theta_a, theta_c, theta_ni)

# we can display the decision function definition
decComb

# and use it to decide if a given distribution fulfills all
# criterions defined
# for the prior
decComb(flat_prior)
# or for a possible outcome of the trial
# here with HR of 0.8 for 40 events
decComb(postmix(flat_prior, m = log(0.8), n = 40))

# A two-sided decision function can be useful to determine if
# certain intermediate (i.e. neither "go" nor "stop") decisions
# are to be made based on the posterior distribution.
# For example, in the above situation we might have an intermediate
# scenario where the trial is significant for non-inferiority but
# the mean estimate is in an intermediate range, say between theta_c
# theta_f:
theta_f <- 0.3
decCombIntermediate <- decision1S(
  c(1 - alpha, 0.5, 0.8),
  c(theta_ni, theta_c, theta_f),
  lower.tail = c(TRUE, FALSE, TRUE)
)
# Not fulfilled for the prior:
decCombIntermediate(flat_prior)
# But for a hypothetical trial outcome with HR 1.2 and 300 events:
decCombIntermediate(postmix(flat_prior, m = log(1.2), n = 300))
```

`decision1S_boundary` *Decision Boundary for 1 Sample Designs*

Description

Calculates the decision boundary for a 1 sample design. This is the critical value at which the decision function will change from 0 (failure) to 1 (success).

Usage

```
decision1S_boundary(prior, n, decision, ...)

## S3 method for class 'betaMix'
decision1S_boundary(prior, n, decision, ...)

## S3 method for class 'normMix'
decision1S_boundary(
  prior,
  n,
  decision,
  sigma,
  eps = 1e-06,
  family = NULL,
  offset = 0,
  ...
)

## S3 method for class 'gammaMix'
decision1S_boundary(prior, n, decision, eps = 1e-06, ...)
```

Arguments

`prior`	Prior for analysis.
`n`	Sample size for the experiment.
`decision`	One-sample decision function to use; see `decision1S`.
`...`	Optional arguments.
`sigma`	The fixed reference scale. If left unspecified, the default reference scale of the prior is assumed. When `family` is a non-Gaussian family, `sigma` must not be specified (it is determined by the family). When `family = gaussian()`, `sigma` is required and acts as the dispersion parameter.
`eps`	Support of random variables are determined as the interval covering `1-eps` probability mass. Defaults to 10^{-6}.

`family` Optional `family` object specifying a GLM family and link function (e.g.\ `binomial()`, `MASS::negative.binomial(theta)`). When provided, the sampling standard deviation varies with the parameter value via the family's variance function and link. Default is `NULL` (constant `sigma`).

`offset` Numeric scalar added to the linear predictor before evaluating the family's variance function. Relevant for Poisson/negative-binomial models with log link where `offset = log(exposure)`. Default is `0`.

Details

The specification of the 1 sample design (prior, sample size and decision function, $D(y)$), uniquely defines the decision boundary

$$y_c = \max_y\{D(y) = 1\},$$

which is the maximal value of y whenever the decision $D(y)$ function changes its value from 1 to 0 for a decision function with `lower.tail=TRUE` (otherwise the definition is $y_c = \max_y\{D(y) = 0\}$). The decision function may change at most at a single critical value as only one-sided decision functions are supported. Here, y is defined for binary and Poisson endpoints as the sufficient statistic $y = \sum_{i=1}^n y_i$ and for the normal case as the mean $\bar{y} = 1/n \sum_{i=1}^n y_i$.

The convention for the critical value y_c depends on whether a left (`lower.tail=TRUE`) or right-sided decision function (`lower.tail=FALSE`) is used. For `lower.tail=TRUE` the critical value y_c is the largest value for which the decision is 1, $D(y \leq y_c) = 1$, while for `lower.tail=FALSE` then $D(y > y_c) = 1$ holds. This is aligned with the cumulative density function definition within R (see for example `pbinom()`).

Value

Returns the critical value y_c. For two-sided decision functions a named vector with components `lower_or_equal_than` and `higher_than` is returned, containing the critical values for the lower and upper decision boundaries.

Methods (by class)

- `decision1S_boundary(betaMix)`: Applies for binomial model with a mixture beta prior. The calculations use exact expressions.
- `decision1S_boundary(normMix)`: Applies for the normal model with known standard deviation σ and a normal mixture prior for the mean. As a consequence from the assumption of a known standard deviation, the calculation discards sampling uncertainty of the second moment. The function `decision1S_boundary` has an extra argument `eps` (defaults to 10^{-6}). The critical value y_c is searched in the region of probability mass `1-eps` for y.

  When `family` is specified, the sampling standard deviation becomes a function of the parameter value θ via

  $$\sigma(\theta) = \sqrt{\phi\, V(\mu)}/|g'(\mu)|$$

  where V is the variance function, g the link function of the family, and ϕ the dispersion parameter. For the Gaussian family $\phi = \sigma^2$ (so `sigma` must be supplied); for all other families $\phi = 1$ and `sigma` must *not* be given. Specifying `family = gaussian("identity")` with `sigma` is equivalent to the standard fixed-σ path.

- `decision1S_boundary(gammaMix)`: Applies for the Poisson model with a gamma mixture prior for the rate parameter. The function `decision1S_boundary` takes an extra argument `eps` (defaults to 10^{-6}) which determines the region of probability mass `1-eps` where the boundary is searched for y.

See Also

Other design1S: `decision1S()`, `oc1S()`, `pos1S()`

Examples

```
# non-inferiority example using normal approximation of log-hazard
# ratio, see ?decision1S for all details
s <- 2
flat_prior <- mixnorm(c(1, 0, 100), sigma = s)
nL <- 233
theta_ni <- 0.4
theta_a <- 0
alpha <- 0.05
beta <- 0.2
za <- qnorm(1 - alpha)
zb <- qnorm(1 - beta)
n1 <- round((s * (za + zb) / (theta_ni - theta_a))^2)
theta_c <- theta_ni - za * s / sqrt(n1)

# double criterion design
# statistical significance (like NI design)
dec1 <- decision1S(1 - alpha, theta_ni, lower.tail = TRUE)
# require mean to be at least as good as theta_c
dec2 <- decision1S(0.5, theta_c, lower.tail = TRUE)
# combination
decComb <- decision1S(c(1 - alpha, 0.5), c(theta_ni, theta_c), lower.tail = TRUE)

# critical value of double criterion design
decision1S_boundary(flat_prior, nL, decComb)

# ... is limited by the statistical significance ...
decision1S_boundary(flat_prior, nL, dec1)

# ... or the indecision point (whatever is smaller)
decision1S_boundary(flat_prior, nL, dec2)
```

`decision2S` *Decision Function for 2 Sample Designs*

Description

The function sets up a 2 sample decision function with an arbitrary number of conditions on the difference distribution.

Usage

```
decision2S(
  pc = 0.975,
  qc = 0,
  lower.tail = TRUE,
  link = c("identity", "logit", "log")
)

oc2Sdecision(
  pc = 0.975,
  qc = 0,
  lower.tail = TRUE,
  link = c("identity", "logit", "log")
)
```

Arguments

pc	Vector of critical cumulative probabilities of the difference distribution.
qc	Vector of respective critical values of the difference distribution. Must match the length of pc.
lower.tail	Logical; if TRUE (default), probabilities are $P(X \leq x)$, otherwise, $P(X > x)$.
link	Enables application of a link function prior to evaluating the difference distribution. Can take one of the values identity (default), logit or log.

Details

This function creates a one- or two-sided decision function on the basis of the difference distribution in a 2 sample situation. To support double criterion designs, see *Neuenschwander et al., 2010*, an arbitrary number of criterions can be given. The decision function demands that the probability mass below the critical value qc of the difference $\theta_1 - \theta_2$ is at least pc. Hence, for lower.tail=TRUE condition i is equivalent to

$$P(\theta_1 - \theta_2 \leq q_{c,i}) > p_{c,i}$$

and the decision function is implemented as indicator function using the heavy-side step function $H(x)$ which is 0 for $x \leq 0$ and 1 for $x > 0$. As all conditions must be met, the final indicator function returns

$$\Pi_i H_i(P(\theta_1 - \theta_2 \leq q_{c,i}) - p_{c,i}),$$

which is 1 if all conditions are met and 0 otherwise. For lower.tail=FALSE differences must be greater than the given quantiles qc.

For the case of a boolean vector given to lower.tail the direction of each decision aligns respectively, and a two-sided decision function is created.

Note that whenever a link other than identity is requested, then the underlying densities are first transformed using the link function and then the probabilties for the differences are calculated in the transformed space. Hence, for a binary endpoint the default identity link will calculate risk

differences, the `logit` link will lead to decisions based on the differences in `logits` corresponding to a criterion based on the log-odds. The `log` link will evaluate ratios instead of absolute differences which could be useful for a binary endpoint or counting rates. The respective critical quantiles `qc` must be given on the transformed scale.

Value

The function returns a decision function, of class `decision2S_1sided` for one-sided, and of class `decision2S_2sided` for two-sided decisions.

One-sided decision functions take three arguments. The first and second argument are expected to be mixture (posterior) distributions from which the difference distribution is formed and all conditions are tested. The third argument determines if the function acts as an indicator function or if the function returns the distance from the decision boundary for each condition in log-space. That is, the distance is 0 at the decision boundary, negative for a 0 decision and positive for a 1 decision.

For two-sided decision functions, the two components can be extracted with functions `lower()` and `upper()`. The distance as calculated by the decision function is returned as a list with components `lower` and `upper`.

Functions

- `oc2Sdecision()`: **[Deprecated]** Deprecated old function name. Please use `decision2S` instead.

References

Gsponer T, Gerber F, Bornkamp B, Ohlssen D, Vandemeulebroecke M, Schmidli H (2014). "A practical guide to Bayesian group sequential designs." *Pharmaceutical Statistics*, **13**(1), 71–80. doi:10.1002/pst.1593.

See Also

Other design2S: `decision2S_boundary()`, `oc2S()`, `pos2S()`

Examples

```
# see Gsponer et al., 2010
priorT <- mixnorm(c(1, 0, 0.001), sigma = 88, param = "mn")
priorP <- mixnorm(c(1, -49, 20), sigma = 88, param = "mn")
# the success criteria is for delta which are larger than some
# threshold value which is why we set lower.tail=FALSE
successCrit <- decision2S(c(0.95, 0.5), c(0, 50), FALSE)
# the futility criterion acts in the opposite direction
futilityCrit <- decision2S(c(0.90), c(40), TRUE)
# intermediate criteria can also be defined: no futility and no success.
# version 1: not significant, but good effect size.
intermediateCrit1 <- decision2S(
  c(1 - 0.95, 0.5, 1 - 0.90),
  c(0, 50, 40),
  c(TRUE, FALSE, TRUE)
)
```

```
# version 2: significant, but too small effect size.
intermediateCrit2 <- decision2S(
  c(0.95, 1 - 0.5, 1 - 0.90),
  c(0, 50, 40),
  c(FALSE, TRUE, TRUE)
)
# version 3: not significant and small effect size.
intermediateCrit3 <- decision2S(
  c(1 - 0.95, 1 - 0.5, 1 - 0.90),
  c(0, 50, 40),
  c(TRUE, TRUE, TRUE)
)

print(successCrit)
print(futilityCrit)
print(intermediateCrit1)
print(intermediateCrit2)
print(intermediateCrit3)

# consider decision for specific outcomes
postP_interim <- postmix(priorP, n = 10, m = -50)
postT_interim <- postmix(priorT, n = 20, m = -80)
futilityCrit(postP_interim, postT_interim)
successCrit(postP_interim, postT_interim)
intermediateCrit1(postP_interim, postT_interim)
intermediateCrit2(postP_interim, postT_interim)
intermediateCrit3(postP_interim, postT_interim)

# Binary endpoint with double criterion decision on log-odds scale
# 95% certain positive difference and an odds ratio of 2 at least
decL2 <- decision2S(c(0.95, 0.5), c(0, log(2)), lower.tail = FALSE, link = "logit")
# 95% certain positive difference and an odds ratio of 3 at least
decL3 <- decision2S(c(0.95, 0.5), c(0, log(3)), lower.tail = FALSE, link = "logit")

# data scenario
post1 <- postmix(mixbeta(c(1, 1, 1)), n = 40, r = 10)
post2 <- postmix(mixbeta(c(1, 1, 1)), n = 40, r = 18)

# positive outcome and a median odds ratio of at least 2 ...
decL2(post2, post1)
# ... but not more than 3
decL3(post2, post1)
```

`decision2S_boundary` *Decision Boundary for 2 Sample Designs*

Description

The `decision2S_boundary` function defines a 2 sample design (priors, sample sizes, decision function) for the calculation of the decision boundary. A function is returned which calculates the critical

value of the first sample $y_{1,c}$ as a function of the outcome in the second sample y_2. At the decision boundary, the decision function will change between 0 (failure) and 1 (success) for the respective outcomes.

Usage

```
decision2S_boundary(prior1, prior2, n1, n2, decision, ...)

## S3 method for class 'betaMix'
decision2S_boundary(prior1, prior2, n1, n2, decision, eps, ...)

## S3 method for class 'normMix'
decision2S_boundary(
  prior1,
  prior2,
  n1,
  n2,
  decision,
  sigma1,
  sigma2,
  eps = 1e-06,
  Ngrid = 5,
  family = NULL,
  offset1 = 0,
  offset2 = offset1,
  ...
)

## S3 method for class 'gammaMix'
decision2S_boundary(prior1, prior2, n1, n2, decision, eps = 1e-06, ...)
```

Arguments

prior1	Prior for sample 1.
prior2	Prior for sample 2.
n1, n2	Sample size of the respective samples. Sample size n1 must be greater than 0 while sample size n2 must be greater or equal to 0.
decision	Two-sample decision function to use; see decision2S.
...	Optional arguments.
eps	Support of random variables are determined as the interval covering 1-eps probability mass. Defaults to 10^{-6}.
sigma1	The fixed reference scale of sample 1. If left unspecified, the default reference scale of the prior 1 is assumed.
sigma2	The fixed reference scale of sample 2. If left unspecified, the default reference scale of the prior 2 is assumed.
Ngrid	Determines density of discretization grid on which decision function is evaluated (see below for more details).

family	Optional family object specifying a GLM family and link function (e.g.\binomial(), MASS::negative.binomial(theta)). When provided, the sampling standard deviation of each sample varies with the respective parameter value via the family's variance function and link. For the Gaussian family sigma1/sigma2 act as the dispersion parameters and must be supplied; for all other families they must *not* be given (they are determined by the family). Default is NULL (constant sigma1 and sigma2).
offset1	Numeric scalar added to the linear predictor of sample 1 before evaluating the family's variance function. Relevant for Poisson/negative-binomial models with log link where offset1 = log(exposure). Default is 0.
offset2	Numeric scalar added to the linear predictor of sample 2; see offset1. Defaults to offset1.

Details

For a 2 sample design the specification of the priors, the sample sizes and the decision function, $D(y_1, y_2)$, uniquely defines the decision boundary

$$D_1(y_2) = \max_{y_1}\{D(y_1, y_2) = 1\},$$

which is the critical value of $y_{1,c}$ conditional on the value of y_2 whenever the decision $D(y_1, y_2)$ function changes its value from 0 to 1 for a decision function with lower.tail=TRUE (otherwise the definition is $D_1(y_2) = \max_{y_1}\{D(y_1, y_2) = 0\}$). The decision function may change at most at a single critical value for given y_2 as only one-sided decision functions are supported. Here, y_2 is defined for binary and Poisson endpoints as the sufficient statistic $y_2 = \sum_{i=1}^{n_2} y_{2,i}$ and for the normal case as the mean $\bar{y}_2 = 1/n_2 \sum_{i=1}^{n_2} y_{2,i}$.

Value

For one-sided decision functions, returns a function with a single argument. This function calculates in dependence of the outcome y_2 in sample 2 the critical value $y_{1,c}$ for which the defined design will change the decision from 0 to 1 (or vice versa, depending on the decision function). For two-sided decision functions, returns a list with components lower_or_equal_than and higher_than, containing the critical value functions for the lower and upper one-sided decision boundary components.

Methods (by class)

- decision2S_boundary(betaMix): Applies for binomial model with a mixture beta prior. The calculations use exact expressions. If the optional argument eps is defined, then an approximate method is used which limits the search for the decision boundary to the region of 1-eps probability mass. This is useful for designs with large sample sizes where an exact approach is very costly to calculate.
- decision2S_boundary(normMix): Applies for the normal model with known standard deviation σ and normal mixture priors for the means. As a consequence from the assumption of a known standard deviation, the calculation discards sampling uncertainty of the second moment. The function has two extra arguments (with defaults): eps (10^{-6}) and Ngrid (10). The decision boundary is searched in the region of probability mass 1-eps, respectively for y_1

and y_2. The continuous decision function is evaluated at a discrete grid, which is determined by a spacing with $\delta_2 = \sigma_2/\sqrt{N_{grid}}$. Once the decision boundary is evaluated at the discrete steps, a spline is used to inter-polate the decision boundary at intermediate points.

- `decision2S_boundary(gammaMix)`: Applies for the Poisson model with a gamma mixture prior for the rate parameter. The function `decision2S_boundary` takes an extra argument `eps` (defaults to 10^{-6}) which determines the region of probability mass `1-eps` where the boundary is searched for y_1 and y_2, respectively.

See Also

Other design2S: `decision2S()`, `oc2S()`, `pos2S()`

Examples

```
# see ?decision2S for details of example
priorT <- mixnorm(c(1, 0, 0.001), sigma = 88, param = "mn")
priorP <- mixnorm(c(1, -49, 20), sigma = 88, param = "mn")
# the success criteria is for delta which are larger than some
# threshold value which is why we set lower.tail=FALSE
successCrit <- decision2S(c(0.95, 0.5), c(0, 50), FALSE)
# the futility criterion acts in the opposite direction
futilityCrit <- decision2S(c(0.90), c(40), TRUE)

# success criterion boundary
successBoundary <- decision2S_boundary(priorP, priorT, 10, 20, successCrit)

# futility criterion boundary
futilityBoundary <- decision2S_boundary(priorP, priorT, 10, 20, futilityCrit)

curve(successBoundary(x), -25:25 - 49, xlab = "y2", ylab = "critical y1")
curve(futilityBoundary(x), lty = 2, add = TRUE)

# hence, for mean in sample 2 of 10, the critical value for y1 is
y1c <- futilityBoundary(-10)

# around the critical value the decision for futility changes
futilityCrit(postmix(priorP, m = y1c + 1E-3, n = 10), postmix(priorT, m = -10, n = 20))
futilityCrit(postmix(priorP, m = y1c - 1E-3, n = 10), postmix(priorT, m = -10, n = 20))
```

`draws-RBesT` *Transform* `gMAP` *to* `draws` *objects*

Description

Transform a `gMAP` object to a format supported by the **posterior** package.

Usage

```
## S3 method for class 'gMAP'
as_draws(x, variable = NULL, regex = FALSE, inc_warmup = FALSE, ...)

## S3 method for class 'gMAP'
as_draws_matrix(x, variable = NULL, regex = FALSE, inc_warmup = FALSE, ...)

## S3 method for class 'gMAP'
as_draws_array(x, variable = NULL, regex = FALSE, inc_warmup = FALSE, ...)

## S3 method for class 'gMAP'
as_draws_df(x, variable = NULL, regex = FALSE, inc_warmup = FALSE, ...)

## S3 method for class 'gMAP'
as_draws_list(x, variable = NULL, regex = FALSE, inc_warmup = FALSE, ...)

## S3 method for class 'gMAP'
as_draws_rvars(x, variable = NULL, regex = FALSE, inc_warmup = FALSE, ...)
```

Arguments

x	A gMAP object.
variable	A character vector providing the variables to extract. By default, all variables are extracted.
regex	Logical; Should variable be treated as a (vector of) regular expressions? Any variable in x matching at least one of the regular expressions will be selected. Defaults to FALSE.
inc_warmup	Should warmup draws be included? Defaults to FALSE.
...	Arguments passed to individual methods (if applicable).

Details

To subset iterations, chains, or draws, use the posterior::subset_draws() method after transforming the input object to a draws object.

See Also

posterior::draws() posterior::subset_draws()

Examples

```
## Setting up dummy sampling for fast execution of example
## Please use 4 chains and 20x more warmup & iter in practice
.user_mc_options <- options(RBesT.MC.warmup=50, RBesT.MC.iter=100,
                            RBesT.MC.chains=2, RBesT.MC.thin=1)


set.seed(34563)
```

```
map_AS <- gMAP(cbind(r, n - r) ~ 1 | study,
  family = binomial,
  data = AS,
  tau.dist = "HalfNormal", tau.prior = 1,
  beta.prior = 2
)

post_AS <- as_draws(map_AS)

## Recover user set sampling defaults
options(.user_mc_options)
```

---

ess *Effective Sample Size for a Conjugate Prior*

Description

Calculates the Effective Sample Size (ESS) for a mixture prior. The ESS indicates how many experimental units the prior is roughly equivalent to.

Usage

```
ess(mix, method = c("elir", "moment", "morita"), ...)

## S3 method for class 'betaMix'
ess(mix, method = c("elir", "moment", "morita"), ..., s = 100)

## S3 method for class 'gammaMix'
ess(mix, method = c("elir", "moment", "morita"), ..., s = 100, eps = 1e-04)

## S3 method for class 'normMix'
ess(
  mix,
  method = c("elir", "moment", "morita"),
  ...,
  family = gaussian,
  sigma,
  s = 100
)
```

Arguments

`mix`	Prior (mixture of conjugate distributions).
`method`	Selects the used method. Can be either `elir` (default), `moment` or `morita`.
`...`	Optional arguments applicable to specific methods.
`s`	For `morita` method large constant to ensure that the prior scaled by this value is vague (default 100); see Morita et al. (2008) for details.

eps	Probability mass left out from the numerical integration of the expected information for the Poisson-Gamma case of Morita method (defaults to 1E-4).
family	defines data likelihood and link function (binomial, gaussian, or poisson).
sigma	reference scale.

Details

The ESS is calculated using either the expected local information ratio (elir) *Neuenschwander et al. (2020)*, the moments approach or the method by *Morita et al. (2008)*.

The elir approach measures effective sample size in terms of the average curvature of the prior in relation to the Fisher information. Informally this corresponds to the average peakiness of the prior in relation to the information content of a single observation. The elir approach is the only ESS which fulfills predictive consistency. The predictive consistency of the ESS requires that the ESS of a prior is consistent when considering an averaged posterior ESS of additional data distributed according to the predictive distribution of the prior. The expectation of the posterior ESS is taken wrt to the prior predictive distribution and the averaged posterior ESS corresponds to the sum of the prior ESS and the number of forward simulated data items. The elir approach results in ESS estimates which are neither conservative nor liberal whereas the moments method yields conservative and the morita method liberal results. See the example section for a demonstration of predictive consistency.

For the moments method the mean and standard deviation of the mixture are calculated and then approximated by the conjugate distribution with the same mean and standard deviation. For conjugate distributions, the ESS is well defined. See the examples for a step-wise calculation in the beta mixture case.

The Morita method used here evaluates the mixture prior at the mode instead of the mean as proposed originally by Morita. The method may lead to very optimistic ESS values, especially if the mixture contains many components. The calculation of the Morita approach here follows the approach presented in Neuenschwander B. et all (2019) which avoids the need for a minimization and does not restrict the ESS to be an integer.

The arguments sigma and family are specific for normal mixture densities. These specify the sampling standard deviation for a gaussian family (the default) while also allowing to consider the ESS of standard one-parameter exponential families, i.e. binomial or poisson. The function supports non-gaussian families with unit dispersion only.

Value

Returns the ESS of the prior as floating point number.

Methods (by class)

- ess(betaMix): ESS for beta mixtures.
- ess(gammaMix): ESS for gamma mixtures.
- ess(normMix): ESS for normal mixtures.

Supported Conjugate Prior-Likelihood Pairs

Prior/Posterior	Likelihood	Predictive	Summaries
Beta	Binomial	Beta-Binomial	`n`, `r`
Normal	Normal (*fixed* σ)	Normal	`n`, `m`, `se`
Gamma	Poisson	Gamma-Poisson	`n`, `m`
Gamma	Exponential	Gamma-Exp (*not supported*)	`n`, `m`

References

Morita S, Thall PF, Mueller P (2008). "Determining the effective sample size of a parametric prior." *Biometrics*, **64**(2), 595–602.

Neuenschwander B, Weber S, Schmidli H, O'Hagan A (2020). "Predictively consistent prior effective sample sizes." *Biometrics*, **76**(2), 578–587. doi:10.1111/biom.13252.

Examples

```
# Conjugate Beta example
a <- 5
b <- 15
prior <- mixbeta(c(1, a, b))

ess(prior)
(a + b)

# Beta mixture example
bmix <- mixbeta(rob = c(0.2, 1, 1), inf = c(0.8, 10, 2))

ess(bmix, "elir")

ess(bmix, "moment")
# moments method is equivalent to
# first calculate moments
bmix_sum <- summary(bmix)
# then calculate a and b of a matching beta
ab_matched <- ms2beta(bmix_sum["mean"], bmix_sum["sd"])
# finally take the sum of a and b which are equivalent
# to number of responders/non-responders respectivley
round(sum(ab_matched))

ess(bmix, method = "morita")

# One may also calculate the ESS on the logit scale, which
# gives slightly different results due to the parameter
# transformation, e.g.:
prior_logit <- mixnorm(c(1, log(5 / 15), sqrt(1 / 5 + 1 / 15)))
ess(prior_logit, family = binomial)

bmix_logit <- mixnorm(
  rob = c(0.2, 0, 2),
  inf = c(0.8, log(10 / 2), sqrt(1 / 10 + 1 / 2))
```

```
)
ess(bmix_logit, family = binomial)

# Predictive consistency of elir
n_forward <- 1E1
bmixPred <- preddist(bmix, n = n_forward)
pred_samp <- rmix(bmixPred, 1E2)
# use more samples here for greater accuracy, e.g.
# pred_samp <- rmix(bmixPred, 1E3)
pred_ess <- sapply(pred_samp, function(r) {
  ess(postmix(bmix, r = r, n = n_forward), "elir")
})
ess(bmix, "elir")
mean(pred_ess) - n_forward

# Normal mixture example
nmix <- mixnorm(rob = c(0.5, 0, 2), inf = c(0.5, 3, 4), sigma = 10)

ess(nmix, "elir")

ess(nmix, "moment")

# the reference scale determines the ESS
sigma(nmix) <- 20
ess(nmix)

# we may also interpret normal mixtures as densities assigned to
# parameters of a logit transformed response rate of a binomial
nmix_logit <- mixnorm(c(1, logit(1 / 4), 2 / sqrt(10)))
ess(nmix_logit, family = binomial)

# Gamma mixture example
gmix <- mixgamma(rob = c(0.3, 20, 4), inf = c(0.7, 50, 10))

ess(gmix) ## interpreted as appropriate for a Poisson likelihood (default)

likelihood(gmix) <- "exp"
ess(gmix) ## interpreted as appropriate for an exponential likelihood
```

`forest_plot` *Forest Plot*

Description

Creates a forest plot for `gMAP()` analysis objects.

Usage

```
forest_plot(
  x,
```

```
  prob = 0.95,
  est = c("both", "MAP", "Mean", "none"),
  model = c("stratified", "both", "meta"),
  point_est = c("median", "mean"),
  size = 1.25,
  linewidth = size,
  alpha = 0.5
)
```

Arguments

x	gMAP() object.
prob	confidence interval width and probability mass of credible intervals.
est	can be set to one of both (default), MAP, Mean or none. Controls which model estimates are to be included.
model	controls which estimates are displayed per study. Either stratified (default), both or meta.
point_est	shown point estimate. Either median (default) or mean.
size	controls point size.
linewidth	controls linewidth; set by default to same value as size.
alpha	transparency of reference line. Setting alpha=0 suppresses the reference line.

Details

The function creates a forest plot suitable for gMAP() analyses. Note that the Meta-Analytic-Predictive prior is included by default in the plot as opposed to only showing the estimated model mean. See the examples below to obtain standard forest plots.

Also note that the plot internally flips the x and y-axis. Therefore, if you want to manipulate the x-axis, you have to give commands affecting the y-axis (see examples).

Value

The function returns a **ggplot2** plot object.

Customizing ggplot2 plots

The returned plot is a **ggplot2** object. Please refer to the "Customizing Plots" vignette which is part of **RBesT** documentation for an introduction. For simple modifications (change labels, add reference lines, ...) consider the commands found in bayesplot-helpers. For more advanced customizations please use the **ggplot2** package directly. A description of the most common tasks can be found in the R Cookbook and a full reference of available commands can be found at the ggplot2 documentation site.

See Also

gMAP()

Examples

```
# we consider the example AS MAP analysis
example(AS)

# default forest plot for a gMAP analysis
forest_plot(map_AS)

# standard forest plot (only stratified estimate and Mean)
forest_plot(map_AS, est = c("Mean"), model = "stratified")

# to further customize these plots, first load bayesplot and ggplot2
library(bayesplot)
library(ggplot2)

# to make plots with red colors, big fonts for presentations, suppress
# the x axis label and add another title (with a subtitle)
color_scheme_set("red")
theme_set(theme_default(base_size = 16))
forest_plot(map_AS, size = 2) +
  yaxis_title(FALSE) +
  ggtitle("Ankylosing Spondylitis Forest Plot",
    subtitle = "Control Group Response Rate"
  )

# the defaults are set with
color_scheme_set("blue")
theme_set(theme_default(base_size = 12))
```

---

gMAP *Meta-Analytic-Predictive Analysis for Generalized Linear Models*

Description

Meta-Analytic-Predictive (MAP) analysis for generalized linear models suitable for normal, binary, or Poisson data. Model specification and overall syntax follows mainly `stats::glm()` conventions.

Usage

```
gMAP(
  formula,
  family = gaussian,
  data,
  weights,
  offset,
  tau.strata,
 tau.dist = c("HalfNormal", "TruncNormal", "Uniform", "Gamma", "InvGamma", "LogNormal",
    "TruncCauchy", "Exp", "Fixed"),
  tau.prior,
```

```
  tau.strata.pred = 1,
  beta.prior,
  prior_PD = FALSE,
  REdist = c("normal", "t"),
  t.df = 5,
  contrasts = NULL,
  iter = getOption("RBesT.MC.iter", 6000),
  warmup = getOption("RBesT.MC.warmup", 2000),
  thin = getOption("RBesT.MC.thin", 4),
  init = getOption("RBesT.MC.init", 1),
  chains = getOption("RBesT.MC.chains", 4),
  cores = getOption("mc.cores", 1L)
)

## S3 method for class 'gMAP'
print(x, digits = 3, probs = c(0.025, 0.5, 0.975), ...)

## S3 method for class 'gMAP'
fitted(object, type = c("response", "link"), probs = c(0.025, 0.5, 0.975), ...)

## S3 method for class 'gMAP'
coef(object, probs = c(0.025, 0.5, 0.975), ...)

## S3 method for class 'gMAP'
as.matrix(x, ...)

## S3 method for class 'gMAP'
summary(
  object,
  type = c("response", "link"),
  probs = c(0.025, 0.5, 0.975),
  ...
)
```

Arguments

formula	the model formula describing the linear predictor and encoding the grouping; see details
family	defines data likelihood and link function (binomial, gaussian, or poisson)
data	optional data frame containing the variables of the model. If not found in data, the variables are taken from environment(formula).
weights	optional weight vector; see details below.
offset	offset term in statistical model used for Poisson data
tau.strata	sets the exchangability stratum per study. That is, it is expected that each study belongs to a single stratum. Default is to assign all studies to stratum 1. See section differential heterogeniety below.

tau.dist	type of prior distribution for tau; supported priors are HalfNormal (default), TruncNormal, Uniform, Gamma, InvGamma, LogNormal, TruncCauchy, Exp and Fixed.
tau.prior	parameters of prior distribution for tau; see section prior specification below.
tau.strata.pred	the index for the prediction stratum; default is 1.
beta.prior	mean and standard deviation for normal priors of regression coefficients, see section prior specification below.
prior_PD	logical to indicate if the prior predictive distribution should be sampled (no conditioning on the data). Defaults to FALSE.
REdist	type of random effects distribution. Normal (default) or t.
t.df	degrees of freedom if random-effects distribution is t.
contrasts	an optional list; See contrasts.arg from stats::model.matrix.default().
iter	number of iterations (including warmup).
warmup	number of warmup iterations.
thin	period of saving samples.
init	positive number to specify uniform range on unconstrained space for random initialization. See stan.
chains	number of Markov chains.
cores	number of cores for parallel sampling of chains.
x, object	gMAP analysis object created by gMAP function
digits	number of displayed significant digits.
probs	defines quantiles to be reported.
...	optional arguments are ignored
type	sets reported scale (response (default) or link).

Details

Setting chains = 0 runs the model setup without Stan sampling and returns a gMAP skeleton without posterior draws. This mode is intended for advanced workflows such as fixture construction where draws are injected later.

The meta-analytic-predictive (MAP) approach derives a prior from historical data using a hierarchical model. The statistical model is formulated as a generalized linear mixed model for binary, normal (with fixed σ) and Poisson endpoints:

$$y_{ih}|\theta_{ih} \sim f(y_{ih}|\theta_{ih})$$

Here, $i = 1, \ldots, N$ is the index for observations, and $h = 1, \ldots, H$ is the index for the grouping (usually studies). The model assumes the linear predictor for a transformed mean as

$$g(\theta_{ih}; x_{ih}, \beta) = x_{ih}\,\beta + \epsilon_h$$

with x_{ih} being the row vector of k covariates for observation i. The variance component is assumed by default normal

$$\epsilon_h \sim N(0, \tau^2), \qquad h = 1, \ldots, H$$

Lastly, the Bayesian implementation assumes independent normal priors for the k regression coefficients and a prior for the between-group standard deviation τ (see `taud.dist` for available distributions).

The MAP prior will then be derived from the above model as the conditional distribution of $\theta_\star$ given the available data and the vector of covariates $x_\star$ defining the overall intercept

$$\theta_\star | x_\star, y.$$

A simple and common case arises for one observation (summary statistic) per trial. For a normal endpoint, the model then simplifies to the standard normal-normal hierarchical model. In the above notation, $i = h = 1, \ldots, H$ and

$$\begin{aligned} y_h | \theta_h &\sim N(\theta_h, s_h^2) \\ \theta_h &= \mu + \epsilon_h \\ \epsilon_h &\sim N(0, \tau^2), \end{aligned}$$

where the more common μ is used for the only (intercept) parameter β_1. Since there are no covariates, the MAP prior is simply $Pr(\theta_\star | y_1, \ldots, y_H)$.

The hierarchical model is a compromise between the two extreme cases of full pooling ($\tau = 0$, full borrowing, no discounting) and no pooling ($\tau = \infty$, no borrowing, stratification). The information content of the historical data grows with H (number of historical data items) indefinitely for full pooling whereas no information is gained in a stratified analysis. For a fixed τ, the maximum effective sample size of the MAP prior is n_∞ ($H \to \infty$), which for a normal endpoint with fixed σ is

$$n_\infty = \left(\frac{\tau^2}{\sigma^2} \right)^{-1},$$

(*Neuenschwander et al., 2010*). Hence, the ratio τ/σ limits the amount of information a MAP prior is equivalent to. This allows for a classification of τ values in relation to σ, which is crucial to define a prior P_τ. The following classification is useful in a clinical trial setting:

Heterogeneity	τ/σ	n_∞
small	0.0625	256
moderate	0.125	64
substantial	0.25	16
large	0.5	4
very large	1.0	1

The above formula for n_∞ assumes a known τ. This is unrealistic as the between-trial heterogeneity parameter is often not well estimable, in particular if the number of trials is small (H small). The

above table helps to specify a prior distribution for τ appropriate for the given context which defines the crucial parameter σ. For binary and Poisson endpoints, normal approximations can be used to determine σ. See examples below for concrete cases.

The design matrix X is defined by the formula for the linear predictor and is always of the form `response ~ predictor | grouping`, which follows `stats::glm()` conventions. The syntax has been extended to include a specification of the grouping (for example study) factor of the data with a horizontal bar, `|`. The bar separates the optionally specified grouping level, i.e. in the binary endpoint case `cbind(r, n-r) ~ 1 | study`. By default it is assumed that each row corresponds to an individual group (for which an individual parameter is estimated). Specifics for the different endpoints are:

normal `family=gaussian` assumes an identity link function. The `response` should be given as matrix with two columns with the first column being the observed mean value y_{ih} and the second column the standard error se_{ih} (of the mean). Additionally, it is recommended to specify with the `weight` argument the number of units which contributed to the (mean) measurement y_{ih}. This information is used to estimate σ.

binary `family=binomial` assumes a logit link function. The `response` must be given as two-column matrix with number of responders r (first column) and non-responders $n - r$ (second column).

Poisson `family=poisson` assumes a log link function. The `response` is a vector of counts. The total exposure times can be specified by an `offset`, which will be linearly added to the linear predictor. The `offset` can be given as part of the formula, `y ~ 1 + offset(log(exposure))` or as the `offset` argument to `gMAP`. Note that the exposure unit must be given as log-offset.

Value

The function returns a S3 object of type `gMAP`. See the methods section below for applicable functions to query the object.

Methods (by generic)

- `print(gMAP)`: displays a summary of the gMAP analysis.
- `fitted(gMAP)`: returns the quantiles of the posterior shrinkage estimates for each data item used during the analysis of the given `gMAP` object.
- `coef(gMAP)`: returns the quantiles of the predictive distribution. User can choose with `type` if the result is on the response or the link scale.
- `as.matrix(gMAP)`: **[Deprecated]** extracts the posterior sample of the model. Use `posterior::as_draws_matrix()` instead.
- `summary(gMAP)`: returns the summaries of a gMAP. analysis. Output is a `gMAPsummary` object, which is a list containing

  `tau` posterior summary of the heterogeneity standard deviation

  `beta` posterior summary of the regression coefficients

  `theta.pred` summary of the predictive distribution (given in dependence on the `type` argument either on `response` or `link` scale)

  `theta` posterior summary of the mean estimate (also depends on the `type` argument)

Differential Discounting

The above model assumes the same between-group standard deviation τ, which implies that the data are equally relevant. This assumption can be relaxed to more than one τ. That is,

$$\epsilon_h \sim N(0, \tau^2_{s(h)})$$

where $s(h)$ assignes group h to one of S between-group heterogeneity strata.

For example, in a situation with two randomized and four observational studies, one may want to assume τ_1 (for trials 1 and 2) and τ_2 (for trials 3-6) for the between-trial standard deviations of the control means. More heterogeneity (less relevance) for the observational studies can then be expressed by appropriate priors for τ_1 and τ_2. In this case, $S = 2$ and the strata assignments (see `tau.strata` argument) would be $s(1) = s(2) = 1, s(3) = \ldots = s(6) = 2$.

Prior Specification

The prior distribution for the regression coefficients β is normal.

- If a single number is given, then this is used as the standard deviation and the default mean of 0 is used.
- If a vector is given, it must be of the same length as number of covariates defined and is used as standard deviation.
- If a matrix with a single row is given, its first row will be used as mean and the second row will be used as standard deviation for all regression coefficients.
- Lastly, a two-column matrix (mean and standard deviation columns) with as many columns as regression coefficients can be given.

It is recommended to always specify a `beta.prior`. Per default a mean of 0 is set. The standard deviation is set to 2 for the binary case, to 100 * `sd(y)` for the normal case and to `sd(log(y + 0.5 + offset))` for the Poisson case.

For the between-trial heterogeniety τ prior, a dispersion parameter must always be given for each exchangeability stratum. For the different `tau.prior` distributions, two parameters are needed out of which one is set to a default value if applicable:

Prior	a	b	default
`HalfNormal`	$\mu = 0$	σ	
`TruncNormal`	μ	σ	$\mu = 0$
`Uniform`	a	b	a = 0
`Gamma`	α	β	
`InvGamma`	α	β	
`LogNormal`	$\mu_{\log}$	$\sigma_{\log}$	
`TruncCauchy`	μ	σ	$\mu = 0$
`Exp`	β	0	
`Fixed`	a	0	

For a prior distribution with a default location parameter, a vector of length equal to the number of exchangability strata can be given. Otherwise, a two-column matrix with as many rows as exchangability strata must be given, except for a single τ stratum, for which a vector of length two defines the parameters a and b.

Random seed

The MAP analysis is performed using Markov-Chain-Monte-Carlo (MCMC) in `rstan::rstan()`. MCMC is a stochastic algorithm. To obtain exactly reproducible results you must use the `base::set.seed()` function before calling `gMAP`. See `RBesT()` overview page for global options on setting further MCMC simulation parameters.

References

Neuenschwander B, Capkun-Niggli G, Branson M, Spiegelhalter DJ (2010). "Summarizing historical information on controls in clinical trials." *Clinical Trials*, **7**(1), 5–18. doi:10.1177/1740774509356002.

Schmidli H, Gsteiger S, Roychoudhury S, O'Hagan A, Spiegelhalter D, Neuenschwander B (2014). "Robust meta-analytic-predictive priors in clinical trials with historical control information." *Biometrics*, **70**(4), 1023–1032. doi:10.1111/biom.12242.

Weber S, Li Y, Seaman JW, Kakizume T, Schmidli H (2021). "Applying Meta-Analytic-Predictive Priors with the R Bayesian Evidence Synthesis Tools." *Journal of Statistical Software*, **100**(19), 1–32. doi:10.18637/jss.v100.i19.

See Also

`plot.gMAP()`, `forest_plot()`, `automixfit()`, `predict.gMAP()`

Examples

```
## Setting up dummy sampling for fast execution of example
## Please use 4 chains and 20x more warmup & iter in practice
.user_mc_options <- options(RBesT.MC.warmup=50, RBesT.MC.iter=100,
                            RBesT.MC.chains=2, RBesT.MC.thin=1)

# Binary data example 1

# Mean response rate is ~0.25. For binary endpoints
# a conservative choice for tau is a HalfNormal(0,1) as long as
# the mean response rate is in the range of 0.2 to 0.8. For
# very small or large rates consider the n_infinity approach
# illustrated below.
# for exact reproducible results, the seed must be set
set.seed(34563)
map_AS <- gMAP(cbind(r, n - r) ~ 1 | study,
  family = binomial,
  data = AS,
  tau.dist = "HalfNormal", tau.prior = 1,
  beta.prior = 2
)
print(map_AS)

# obtain numerical summaries
map_sum <- summary(map_AS)
print(map_sum)
names(map_sum)
# [1] "tau"        "beta"       "theta.pred" "theta"
```

```
map_sum$theta.pred


# graphical model checks (returns list of ggplot2 plots)
map_checks <- plot(map_AS)
# forest plot with shrinkage estimates
map_checks$forest_model
# density of MAP prior on response scale
map_checks$densityThetaStar
# density of MAP prior on link scale
map_checks$densityThetaStarLink


# obtain shrinkage estimates
fitted(map_AS)

# regression coefficients
coef(map_AS)

# finally fit MAP prior with parametric mixture
map_mix <- mixfit(map_AS, Nc = 2)
plot(map_mix)$mix


# optionally select number of components automatically via AIC
map_automix <- automixfit(map_AS)
plot(map_automix)$mix


# Normal example 2, see normal vignette

# Prior considerations

# The general principle to derive a prior for tau can be based on the
# n_infinity concept as discussed in Neuenschwander et al., 2010.
# This assumes a normal approximation which applies for the colitis
# data set as:
p_bar <- mean(with(colitis, r / n))
s <- round(1 / sqrt(p_bar * (1 - p_bar)), 1)
# s is the approximate sampling standard deviation and a
# conservative prior is tau ~ HalfNormal(0,s/2)
tau_prior_sd <- s / 2

# Evaluate HalfNormal prior for tau
tau_cat <- c(
  pooling = 0,
  small = 0.0625,
  moderate = 0.125,
  substantial = 0.25,
  large = 0.5,
  veryLarge = 1,
  stratified = Inf
)
```

```
# Interval probabilites (basically saying we are assuming
# heterogeniety to be smaller than very large)
diff(2 * pnorm(tau_cat * s, 0, tau_prior_sd))
# Cumulative probabilities as 1-F
1 - 2 * (pnorm(tau_cat * s, 0, tau_prior_sd) - 0.5)

## Recover user set sampling defaults
options(.user_mc_options)
```

`likelihood` *Read and Set Likelihood to the Corresponding Conjugate Prior*

Description

Read and set the likelihood distribution corresponding to the conjugate prior distribution.

Usage

```
likelihood(mix)

likelihood(mix) <- value
```

Arguments

`mix`	Prior mixture distribution.
`value`	New likelihood. **Should** only be changed for Gamma priors as these are supported with either Poisson (`value="poisson"`) or Exponential (`value="exp"`) likelihoods.

Details

If the prior and posterior distributions are in the same family, then the prior distribution is called a conjugate prior for the likelihood function.

Supported Conjugate Prior-Likelihood Pairs

Prior/Posterior	Likelihood	Predictive	Summaries
Beta	Binomial	Beta-Binomial	`n`, `r`
Normal	Normal (*fixed* σ)	Normal	`n`, `m`, `se`
Gamma	Poisson	Gamma-Poisson	`n`, `m`
Gamma	Exponential	Gamma-Exp (*not supported*)	`n`, `m`

Examples

```
# Gamma mixture
gmix <- mixgamma(c(0.3, 20, 4), c(0.7, 50, 10))

# read out conjugate partner
likelihood(gmix)

ess(gmix)

# set conjugate partner
likelihood(gmix) <- "exp"

# ... which changes the interpretation of the mixture
ess(gmix)
```

---

lodds *Logit (log-odds) and inverse-logit function.*

Description

Calculates the logit (log-odds) and inverse-logit.

Usage

```
logit(mu)

inv_logit(eta)
```

Arguments

mu	A numeric object with probabilies, with values in the in the range $[0, 1]$. Missing values (NAs) are allowed.
eta	A numeric object with log-odds values, with values in the range $[-\infty, \infty]$. Missing values (NAs) are allowed.

Details

Values of mu equal to 0 or 1 will return $-\infty$ or ∞ respectively.

Value

A numeric object of the same type as mu and eta containing the logits or inverse logit of the input values. The logit and inverse transformation equates to

$$\text{logit}(\mu) = \log(\mu/(1-\mu))$$
$$\text{logit}^{-1}(\eta) = \exp(\eta)/(1+\exp(\eta)).$$

Examples

```
logit(0.2)
inv_logit(-1.386)
```

`mix` *Mixture Distributions*

Description

Density, cumulative distribution function, quantile function and random number generation for supported mixture distributions. (d/p/q/r)mix are generic and work with any mixture supported by BesT (see table below).

Usage

```
dmix(mix, x, log = FALSE)

pmix(mix, q, lower.tail = TRUE, log.p = FALSE)

qmix(mix, p, lower.tail = TRUE, log.p = FALSE)

rmix(mix, n)

## S3 method for class 'mix'
mix[[..., rescale = FALSE]]
```

Arguments

`mix`	mixture distribution object
`x`, `q`	vector of quantiles
`log`, `log.p`	logical; if `TRUE` (not default), probabilities p are given as $\log(p)$
`lower.tail`	logical; if `TRUE` (default), probabilities are $P[X \leq x]$ 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
`...`	components to subset given mixture.
`rescale`	logical; if `TRUE`, mixture weights will be rescaled to sum to 1

Details

A mixture distribution is defined as a linear superposition of K densities of the same distributional class. The mixture distributions supported have the form

$$f(x, \mathbf{w}, \mathbf{a}, \mathbf{b}) = \sum_{k=1}^{K} w_k \, f_k(x, a_k, b_k).$$

The w_k are the mixing coefficients which must sum to 1. Moreover, each density f is assumed to be parametrized by two parameters such that each component k is defined by a triplet, (w_k, a_k, b_k).

Individual mixture components can be extracted using the `[[` operator, see examples below.

The supported densities are normal, beta and gamma which can be instantiated with `mixnorm()`, `mixbeta()`, or `mixgamma()`, respectively. In addition, the respective predictive distributions are supported. These can be obtained by calling `preddist()` which returns appropriate normal, beta-binomial or Poisson-gamma mixtures.

For convenience a `summary` function is defined for all mixtures. It returns the mean, standard deviation and the requested quantiles which can be specified with the argument `probs`.

Value

`dmix` gives the weighted sum of the densities of each component.

`pmix` calculates the distribution function by evaluating the weighted sum of each components distribution function.

`qmix` returns the quantile for the given `p` by using that the distribution function is monotonous and hence a gradient based minimization scheme can be used to find the matching quantile `q`.

`rmix` generates a random sample of size `n` by first sampling a latent component indicator in the range $1..K$ for each draw and then the function samples from each component a random draw using the respective sampling function. The `rnorm` function returns the random draws as numerical vector with an additional attribute `ind` which gives the sampled component indicator.

Supported Conjugate Prior-Likelihood Pairs

Prior/Posterior	Likelihood	Predictive	Summaries
Beta	Binomial	Beta-Binomial	`n`, `r`
Normal	Normal (*fixed* σ)	Normal	`n`, `m`, `se`
Gamma	Poisson	Gamma-Poisson	`n`, `m`
Gamma	Exponential	Gamma-Exp (*not supported*)	`n`, `m`

See Also

`plot.mix()`

Other mixdist: `mixbeta()`, `mixcombine()`, `mixgamma()`, `mixjson`, `mixmvnorm()`, `mixnorm()`, `mixplot`

Examples

```
## a beta mixture
bm <- mixbeta(weak = c(0.2, 2, 10), inf = c(0.4, 10, 100), inf2 = c(0.4, 30, 80))

## extract the two most informative components
bm[[c(2, 3)]]
## rescaling needed in order to plot
plot(bm[[c(2, 3), rescale = TRUE]])

summary(bm)
```

---

mixbeta *Beta Mixture Density*

Description

The Beta mixture density and auxilary functions.

Usage

```
mixbeta(..., param = c("ab", "ms", "mn"))

ms2beta(m, s, drop = TRUE)

mn2beta(m, n, drop = TRUE)

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

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

## S3 method for class 'betaMix'
summary(object, probs = c(0.025, 0.5, 0.975), ...)

## S3 method for class 'betaBinomialMix'
summary(object, probs = c(0.025, 0.5, 0.975), ...)
```

Arguments

...	List of mixture components.
param	Determines how the parameters in the list are interpreted. See details.
m	Vector of means of beta mixture components.
s	Vector of standard deviations of beta mixture components.
drop	Delete the dimensions of an array which have only one level.

n	Vector of number of observations.
x	The mixture to print
object	Beta mixture object.
probs	Quantiles reported by the summary function.

Details

Each entry in the ... argument list is expected to be a triplet of numbers which defines the weight w_k, first and second parameter of the mixture component k. A triplet can optionally be named which will be used appropriately.

The first and second parameter can be given in different parametrizations which is set by the param option:

ab Natural parametrization of Beta density (a=shape1 and b=shape2). Default.

ms Mean and standard deviation, $m = a/(a+b)$ and $s = \sqrt{\frac{m(1-m)}{1+n}}$, where $n = a + b$ is the number of observations. Note that s must be less than $\sqrt{m(1-m)}$.

mn Mean and number of observations, $n = a + b$.

Value

mixbeta returns a beta mixture with the specified mixture components. ms2beta and mn2beta return the equivalent natural a and b parametrization given parameters m, s, or n.

See Also

Other mixdist: mix, mixcombine(), mixgamma(), mixjson, mixmvnorm(), mixnorm(), mixplot

Examples

```
## a beta mixture
bm <- mixbeta(rob = c(0.2, 2, 10), inf = c(0.4, 10, 100), inf2 = c(0.4, 30, 80))

# mean/standard deviation parametrization
bm2 <- mixbeta(rob = c(0.2, 0.3, 0.2), inf = c(0.8, 0.4, 0.01), param = "ms")

# mean/observations parametrization
bm3 <- mixbeta(rob = c(0.2, 0.3, 5), inf = c(0.8, 0.4, 30), param = "mn")

# even mixed is possible
bm4 <- mixbeta(rob = c(0.2, mn2beta(0.3, 5)), inf = c(0.8, ms2beta(0.4, 0.1)))

# print methods are defined
bm4
print(bm4)
```

mixcombine *Combine Mixture Distributions*

Description

Combining mixture distributions of the same class to form a new mixture.

Usage

```
mixcombine(..., weight, rescale = TRUE)
```

Arguments

...	arbitrary number of mixtures with same distributional class. Each component with values of mixture weight and model parameters.
weight	relative weight for each component in new mixture distribution. The vector must be of the same length as input mixtures components. The default value gives equal weight to each component.
rescale	boolean value indicates if the weights are rescaled to sum to 1.

Details

Combines mixtures of the same class of random variable to form a new mixture distribution.

Value

A R-object with the new mixture distribution.

See Also

robustify()

Other mixdist: mix, mixbeta(), mixgamma(), mixjson, mixmvnorm(), mixnorm(), mixplot

Examples

```
# beta with two informative components
bm <- mixbeta(inf = c(0.5, 10, 100), inf2 = c(0.5, 30, 80))

# robustified with mixcombine, i.e. a 10% uninformative part added
unif <- mixbeta(rob = c(1, 1, 1))
mixcombine(bm, unif, weight = c(9, 1))
```

`mixdiff` *Difference of mixture distributions*

Description

Density, cumulative distribution function, quantile function and random number generation for the difference of two mixture distributions.

Usage

```
dmixdiff(mix1, mix2, x)

pmixdiff(mix1, mix2, q, lower.tail = TRUE)

qmixdiff(mix1, mix2, p, lower.tail = TRUE)

rmixdiff(mix1, mix2, n)
```

Arguments

`mix1`	first mixture density
`mix2`	second mixture density
`x`	vector of values for which density values are computed
`q`	vector of quantiles for which cumulative probabilities are computed
`lower.tail`	logical; if `TRUE` (default), probabilities are $P[X <= x]$, otherwise $P[X > x]$.
`p`	vector of cumulative probabilities for which quantiles are computed
`n`	size of random sample

Details

If $x_1 \sim f_1(x_1)$ and $x_2 \sim f_2(x_2)$, the density of the difference $d \equiv x_1 - x_2$ is given by

$$f_d(d) = \int f_1(u)\, f_2(u-d)\, du.$$

The cumulative distribution function equates to

$$F_d(d) = \int f_1(u)\, (1 - F_2(u-d))\, du.$$

Both integrals are performed over the full support of the densities and use the numerical integration function `integrate()`.

Value

Respective density, quantile, cumulative density or random numbers.

Examples

```
# 1. Difference between two beta distributions, i.e. Pr( mix1 - mix2 > 0)
mix1 <- mixbeta(c(1, 11, 4))
mix2 <- mixbeta(c(1, 8, 7))
pmixdiff(mix1, mix2, 0, FALSE)

# Interval probability, i.e. Pr( 0.3 > mix1 - mix2 > 0)
pmixdiff(mix1, mix2, 0.3) - pmixdiff(mix1, mix2, 0)

# 2. two distributions, one of them a mixture
m1 <- mixbeta(c(1, 30, 50))
m2 <- mixbeta(c(0.75, 20, 50), c(0.25, 1, 1))

# random sample of difference
set.seed(23434)
rM <- rmixdiff(m1, m2, 1E4)

# histogram of random numbers and exact density
hist(rM, prob = TRUE, new = TRUE, nclass = 40)
curve(dmixdiff(m1, m2, x), add = TRUE, n = 51)

# threshold probabilities for difference, at 0 and 0.2
pmixdiff(m1, m2, 0)
mean(rM < 0)
pmixdiff(m1, m2, 0.2)
mean(rM < 0.2)

# median of difference
mdn <- qmixdiff(m1, m2, 0.5)
mean(rM < mdn)

# 95%-interval
qmixdiff(m1, m2, c(0.025, 0.975))
quantile(rM, c(0.025, 0.975))
```

---

mixfit *Fit of Mixture Densities to Samples*

Description

Expectation-Maximization (EM) based fitting of parametric mixture densities to numerical samples. This provides a convenient approach to approximate MCMC samples with a parametric mixture distribution.

Usage

```
mixfit(sample, type = c("norm", "beta", "gamma", "mvnorm"), thin, ...)
```

```
## Default S3 method:
mixfit(sample, type = c("norm", "beta", "gamma", "mvnorm"), thin, ...)

## S3 method for class 'gMAP'
mixfit(sample, type, thin, ...)

## S3 method for class 'gMAPpred'
mixfit(sample, type, thin, ...)

## S3 method for class 'array'
mixfit(sample, type, thin, ...)
```

Arguments

sample	Sample to be fitted.
type	Mixture density to use. Can be either norm, beta or gamma.
thin	Thinning applied to the sample. See description for default behavior.
...	Parameters passed to the low-level EM fitting functions. Parameter Nc is mandatory.

Details

Parameters of EM fitting functions

Nc Number of mixture components. Required parameter.

mix_init Initial mixture density. If missing (default) then a k-nearest-neighbor algorithm is used to find an initial mixture density.

Ninit Number of data points used for initialization. Defaults to 50.

verbose If set to TRUE the function will inform about fitting process

maxIter Maximal number of iterations. Defaults to 500.

tol Defines a convergence criteria as an upper bound for the change in the log-likelihood, i.e. once the derivative (with respect to iterations) of the log-likelihood falls below tol, the function declares convergence and stops.

eps Must be a triplet of numbers which set the desired accuracy of the inferred parameters per mixture component. See below for a description of the parameters used during EM. EM is stopped once a running mean of the absolute difference between the last successive Neps estimates is below the given eps for all parameters. Defaults to 5E-3 for each parameter.

Neps Number of iterations used for the running mean of parameter estimates to test for convergence. Defaults to 5.

constrain_gt1 Logical value controlling if the Beta EM constrains all parameters a & b to be greater than 1. By default constraints are turned on (new since 1.6-0).

By default the EM convergence is declared when the desired accuracy of the parameters has been reached over the last Neps estimates. If tol and Neps is specified, then whatever criterion is met first will stop the EM.

The parameters per component k used internally during fitting are for the different EM procedures:

normal $logit(w_k), \mu_k, \log(\sigma_k)$

beta $logit(w_k), \log(a_k), \log(b_k)$

constrained beta $logit(w_k), \log(a_k - 1), \log(b_k - 1)$

gamma $logit(w_k), \log(\alpha_k), \log(\beta_k)$

Note: Whenever no `mix_init` argument is given, the EM fitting routines assume that the data vector is given in random order. If in the unlikely event that the EM gets caught in a local extremum, then random reordering of the data vector may alleviate the issue.

Value

A mixture object according the requested `type` is returned. The object has additional information attached, i.e. the log-likelihood can be queried and diagnostic plots can be generated. See links below.

Methods (by class)

- `mixfit(default)`: Performs an EM fit for the given sample. Thinning is applied only if thin is specified.
- `mixfit(gMAP)`: Fits the default predictive distribution from a gMAP analysis. Automatically obtains the predictive distribution of the intercept only case on the response scale mixture from the `gMAP()` object. For the binomial case a beta mixture, for the gaussian case a normal mixture and for the Poisson case a gamma mixture will be used. In the gaussian case, the resulting normal mixture will set the reference scale to the estimated sigma in `gMAP()` call.
- `mixfit(gMAPpred)`: Fits a mixture density for each prediction from the `gMAP()` prediction.
- `mixfit(array)`: Fits a mixture density for an MCMC sample. It is recommended to provide a thinning argument which roughly yields independent draws (i.e. use `acf()` to identify a thinning lag with small auto-correlation). The input array is expected to have 3 dimensions which are nested as iterations, chains, and draws.

References

Dempster AP, Laird NM, Rubin DB (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society, Series B*, **39**(1), 1–38.

See Also

Other EM: `plot.EM()`

Examples

```
bmix <- mixbeta(rob = c(0.2, 1, 1), inf = c(0.8, 10, 2))

bsamp <- rmix(bmix, 1000)

bfit <- mixfit(bsamp, type = "beta", Nc = 2)

# diagnostic plots can easily by generated from the EM fit with
bfit.check <- plot(bfit)
```

```
names(bfit.check)

# check convergence of parameters
bfit.check$mix
bfit.check$mixdens
bfit.check$mixecdf

# obtain the log-likelihood
logLik(bfit)

# or AIC
AIC(bfit)
```

---

mixgamma *The Gamma Mixture Distribution*

Description

The gamma mixture density and auxiliary functions.

Usage

```
mixgamma(..., param = c("ab", "ms", "mn"), likelihood = c("poisson", "exp"))

ms2gamma(m, s, drop = TRUE)

mn2gamma(m, n, likelihood = c("poisson", "exp"), drop = TRUE)

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

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

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

## S3 method for class 'gammaMix'
summary(object, probs = c(0.025, 0.5, 0.975), ...)

## S3 method for class 'gammaPoissonMix'
summary(object, probs = c(0.025, 0.5, 0.975), ...)
```

Arguments

...	List of mixture components.

param	Determines how the parameters in the list are interpreted. See details.
likelihood	Defines with what likelihood the Gamma density is used (Poisson or Exp). Defaults to poisson.
m	Vector of means of the Gamma mixture components
s	Vector of standard deviations of the gamma mixture components,
drop	Delete the dimensions of an array which have only one level.
n	Vector of sample sizes of the Gamma mixture components.
x	The mixture to print
object	Gamma mixture object.
probs	Quantiles reported by the summary function.

Details

Each entry in the ... argument list is expected to be a triplet of numbers which defines the weight w_k, first and second parameter of the mixture component k. A triplet can optionally be named which will be used appropriately.

The first and second parameter can be given in different parametrizations which is set by the param option:

ab Natural parametrization of Gamma density (a=shape and b=rate). Default.

ms Mean and standard deviation, $m = a/b$ and $s = \sqrt{a}/b$.

mn Mean and number of observations. Translation to natural parameter depends on the likelihood argument. For a Poisson likelihood $n = b$ (and $a = m \cdot n$), for an Exp likelihood $n = a$ (and $b = n/m$).

Value

mixgamma returns a gamma mixture with the specified mixture components. ms2gamma and mn2gamma return the equivalent natural a and b parametrization given parameters m, s, or n.

See Also

Other mixdist: mix, mixbeta(), mixcombine(), mixjson, mixmvnorm(), mixnorm(), mixplot

Examples

```
# Gamma mixture with robust and informative component
gmix <- mixgamma(rob = c(0.3, 20, 4), inf = c(0.7, 50, 10))

# objects can be printed
gmix
# or explicitly
print(gmix)

# summaries are defined
summary(gmix)
```

```
# sub-components may be extracted
# by component number
gmix[[2]]
# or component name
gmix[["inf"]]

# alternative mean and standard deviation parametrization
gmsMix <- mixgamma(rob = c(0.5, 8, 0.5), inf = c(0.5, 9, 2), param = "ms")

# or mean and number of observations parametrization
gmnMix <- mixgamma(rob = c(0.2, 2, 1), inf = c(0.8, 2, 5), param = "mn")

# and mixed parametrizations are also possible
gfmix <- mixgamma(rob1 = c(0.15, mn2gamma(2, 1)),
                  rob2 = c(0.15, ms2gamma(2, 5)),
                  inf = c(0.7, 50, 10))
```

`mixjson` *Write and Read a Mixture Object with JSON*

Description

[Experimental]

These functions write and read a mixture object in the JSON format.

Usage

```
write_mix_json(mix, con, ...)

read_mix_json(con, ..., rescale = TRUE)
```

Arguments

`mix`	A mixture object to be saved to JSON.
`con`	A connection specifying where the JSON will be written to or read.
`...`	Additional arguments passed to the `jsonlite::toJSON()` and `jsonlite::fromJSON()` function for writing and reading, respectively.
`rescale`	A logical value indicating whether to rescale the mixture weights so that they sum to 1. Defaults to `TRUE`.

Details

The mixture objects are written or read from the connection `con`, which can be a character string specifying a file path or a connection object as detailed in `base::connections()`.

When writing mixture objects as JSON it is strongly recommended to explicitly set the number of digits (argument `digits`) to be used for the numerical representation in order to control the accuracy of the JSON representation of the mixture object. If the mixture object inherits from the `"EM"` class

(as is the case when the mixture is created using the `mixfit()` function), then the mixture object will be cast to a simple mixture object such that diagnostics from the `"EM"` fitting procedure are dropped from the object. For easier readability the user is encouraged to set the argument `pretty=TRUE`, which is passed to the `jsonlite::toJSON()` function and makes the output more human readable.

Note that when reading in mixture objects, then these are not necessarily equal to the mixtures passed to the `write_mix_json` function. This is a consequence of the limited precision of the textual representation as defined by the `digits` argument.

Value

The `write_mix_json` function does not return a value while the `read_mix_json` returns the mixture object stored in the connection specified.

See Also

Other mixdist: `mix`, `mixbeta()`, `mixcombine()`, `mixgamma()`, `mixmvnorm()`, `mixnorm()`, `mixplot`

Examples

```
## Not run:
nm <- mixnorm(rob = c(0.2, 0, 2), inf = c(0.8, 2, 2), sigma = 5)

write_mix_json(nm, "normal_mixture.json", pretty=TRUE, digits=1)

mix <- read_mix_json("normal_mixture.json")

## End(Not run)
```

---

mixmvnorm *Multivariate Normal Mixture Density*

Description

The multivariate normal mixture density and auxiliary functions.

Usage

```
mixmvnorm(..., sigma, param = c("ms", "mn", "msr"))

msr2mvnorm(m, s, r, unlist = TRUE)

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

## S3 method for class 'mvnormMix'
summary(object, ...)

## S3 method for class 'mvnormMix'
sigma(object, ...)
```

Arguments

...	List of mixture components.
sigma	Reference covariance.
param	Determines how the parameters in the list are interpreted. See details.
m	Mean vector.
s	Standard deviation vector.
r	Vector of correlations in column-major format of the lower triangle of the correlation matrix.
unlist	Logical. Controls whether the result is a flattened vector (TRUE) or a list with mean m and covariance s (FALSE). Defaults to TRUE.
object, x	Multivariate normal mixture object.

Details

Each entry in the ... argument list is a numeric vector defining one component of the mixture multivariate normal distribution. The first entry of the component defining vector is the weight of the mixture component followed by the vector of means in each dimension and finally a specification of the covariance matrix, which depends on the chosen parametrization. The covariance matrix is expected to be given as numeric vector in a column-major format, which is standard conversion applied to matrices by the vector concatenation function base::c(). Please refer to the examples section below.

Each component defining vector can be specified in different ways as determined by the param option:

ms Mean vector and covariance matrix s. Default.

mn Mean vector and number of observations. n determines the covariance for each component via the relation Σ/n with Σ being the known reference covariance.

msr Mean vector, standard deviations and correlations in column-major format (corresponds to order when printing multi-variate normal mixtures).

The reference covariance Σ is the known covariance in the normal-normal model (observation covariance). The function sigma can be used to query the reference covariance and may also be used to assign a new reference covariance, see examples below. In case sigma is not specified, the user has to supply sigma as argument to functions which require a reference covariance.

Value

Returns a multivariate normal mixture with the specified mixture components.

See Also

Other mixdist: mix, mixbeta(), mixcombine(), mixgamma(), mixjson, mixnorm(), mixplot

Examples

```
# default mean & covariance parametrization
S <- diag(c(1, 2)) %*% matrix(c(1, 0.5, 0.5, 1), 2, 2) %*% diag(c(1, 2))
mvnm1 <- mixmvnorm(
  rob = c(0.2, c(0, 0), diag(c(2, 2)^2)),
  inf = c(0.8, c(0.5, 1), S / 4), sigma = S
)

print(mvnm1)
summary(mvnm1)

set.seed(657846)
mixSamp1 <- rmix(mvnm1, 500)
colMeans(mixSamp1)

# alternative mean, sd and correlation parametrization
mvnm1_alt <- mixmvnorm(
  rob = c(0.2, c(0, 0), c(2, 2), 0.0),
  inf = c(0.8, c(0.5, 1), c(1, 2) / 2, 0.5),
  sigma = msr2mvnorm(s = c(1, 2), r = 0.5, unlist = FALSE)$s,
  param = "msr"
)

print(mvnm1_alt)
```

---

mixnorm *Normal Mixture Density*

Description

The normal mixture density and auxiliary functions.

Usage

```
mixnorm(..., sigma, param = c("ms", "mn"))

mn2norm(m, n, sigma, drop = TRUE)

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

## S3 method for class 'normMix'
summary(object, probs = c(0.025, 0.5, 0.975), ...)

## S3 method for class 'normMix'
sigma(object, ...)

sigma(object) <- value
```

Arguments

...	List of mixture components.
sigma	Reference scale.
param	Determines how the parameters in the list are interpreted. See details.
m	Vector of means
n	Vector of sample sizes.
drop	Delete the dimensions of an array which have only one level.
x	The mixture to print
object	Normal mixture object.
probs	Quantiles reported by the summary function.
value	New value of the reference scale sigma.

Details

Each entry in the ... argument list is expected to be a triplet of numbers which defines the weight w_k, first and second parameter of the mixture component k. A triplet can optionally be named which will be used appropriately.

The first and second parameter can be given in different parametrizations which is set by the param option:

ms Mean and standard deviation. Default.

mn Mean and number of observations. n determines s via the relation $s = \sigma/\sqrt{n}$ with σ being the fixed reference scale.

The reference scale σ is the fixed standard deviation in the one-parameter normal-normal model (observation standard deviation). The function sigma can be used to query the reference scale and may also be used to assign a new reference scale, see examples below. In case the sigma is not specified, the user has to supply sigma as argument to functions which require a reference scale.

Value

Returns a normal mixture with the specified mixture components. mn2norm returns the mean and standard deviation given a mean and sample size parametrization.

Functions

- sigma(object) <- value: Allows to assign a new reference scale sigma.

See Also

Other mixdist: mix, mixbeta(), mixcombine(), mixgamma(), mixjson, mixmvnorm(), mixplot

Examples

```
nm <- mixnorm(rob = c(0.2, 0, 2), inf = c(0.8, 2, 2), sigma = 5)

print(nm)
summary(nm)
plot(nm)

set.seed(57845)
mixSamp <- rmix(nm, 500)
plot(nm, samp = mixSamp)

# support defined by quantiles
qmix(nm, c(0.01, 0.99))

# density function
dmix(nm, seq(-5, 5, by = 2))

# distribution function
pmix(nm, seq(-5, 5, by = 2))

# the reference scale can be changed (it determines the ESS)
ess(nm)

sigma(nm) <- 10
ess(nm)
```

---

mixplot *Plot mixture distributions*

Description

Plotting for mixture distributions

Usage

```
## S3 method for class 'mix'
plot(
  x,
  prob = 0.99,
  fun = dmix,
  log = FALSE,
  comp = TRUE,
  linewidth = 1.25,
  size = deprecated(),
  ...
)

## S3 method for class 'mvnormMix'
plot(x, prob = 0.99, fun = dmix, log = FALSE, comp = TRUE, size = 1.25, ...)
```

Arguments

x	mixture distribution
prob	defining lower and upper percentile of x-axis. Defaults to the 99\% central probability mass.
fun	function to plot which can be any of dmix, qmix or pmix.
log	log argument passed to the function specified in fun.
comp	for the density function this can be set to TRUE which will display colour-coded each mixture component of the density in addition to the density.
linewidth	controls line sizes of the plotted function.
size	**[Deprecated]** size has been deprecated by ggplot2 for line sizes and has been renamed to linewidth.
...	extra arguments passed on to the plotted function.

Details

Plot function for mixture distribution objects. It shows the density/quantile/cumulative distribution (corresponds to d/q/pmix function) for some specific central probability mass defined by prob. By default the x-axis is chosen to show 99\% of the probability density mass.

Value

A ggplot2::ggplot() object is returned.

Customizing ggplot2 plots

The returned plot is a **ggplot2** object. Please refer to the "Customizing Plots" vignette which is part of **RBesT** documentation for an introduction. For simple modifications (change labels, add reference lines, ...) consider the commands found in bayesplot-helpers. For more advanced customizations please use the **ggplot2** package directly. A description of the most common tasks can be found in the R Cookbook and a full reference of available commands can be found at the ggplot2 documentation site.

See Also

Other mixdist: mix, mixbeta(), mixcombine(), mixgamma(), mixjson, mixmvnorm(), mixnorm()

Examples

```
# beta with two informative components
bm <- mixbeta(inf = c(0.5, 10, 100), inf2 = c(0.5, 30, 80))
plot(bm)
plot(bm, fun = pmix)

# for customizations of the plot we need to load ggplot2 first
library(ggplot2)

# show a histogram along with the density
plot(bm) + geom_histogram(
```

```
  data = data.frame(x = rmix(bm, 1000)),
  aes(y = ..density..), bins = 50, alpha = 0.4
)


# note: we can also use bayesplot for histogram plots with a density ...
library(bayesplot)
mh <- mcmc_hist(data.frame(x = rmix(bm, 1000)), freq = FALSE) +
  overlay_function(fun = dmix, args = list(mix = bm))
# ...and even add each component
for (k in 1:ncol(bm)) {
  mh <- mh + overlay_function(fun = dmix, args = list(mix = bm[[k]]), linetype = I(2))
}
print(mh)


# normal mixture
nm <- mixnorm(rob = c(0.2, 0, 2), inf = c(0.8, 6, 2), sigma = 5)
plot(nm)
plot(nm, fun = qmix)

# obtain ggplot2 object and change title
pl <- plot(nm)
pl + ggtitle("Normal 2-Component Mixture")
```

`mixstanvar` *Mixture distributions as* `brms` *priors*

Description

Adapter function converting mixture distributions for use with `brms::brm()` models via the `brms::stanvar()` facility.

Usage

```
mixstanvar(..., verbose = FALSE)
```

Arguments

`...`	List of mixtures to convert.
`verbose`	Enables printing of the mixture priors when chains start to sample. Defaults to `FALSE`.

Details

To declare mixture priors in a `brms::brm()` model requires two steps: First, the mixture densities need to be converted by the adapter function `mixstanvar` into a `stanvars` object which is passed to the `stanvars` argument of the `brms::brm()` function. Doing so extends the Stan code and

data generated by brms::brm() such that the mixture densities can be used as priors within the brms::brm() model. The second step is to assign parameters of the brms::brm() model to the mixture density as prior using the brms::set_prior() command of brms.

The adapter function translates the mixture distributions as defined in R to the respective mixture distribution in Stan. Within Stan the mixture distributions are named in accordance to the R functions used to create the respective mixture distributions. That is, a mixture density of normals created by mixnorm() is referred to as mixnorm_lpdf in Stan such that one can refer to the density as mixnorm within the brms::set_prior() functions (the suffix _lpdf is automatically added by brms::brm()). The arguments to these mixture distributions depend on the specific distribution type as follows:

Density	Arguments
mixbeta(w, a, b)	w weights, a shapes, b shapes
mixgamma(w, a, b)	w weights, a shapes, b inverse scales
mixnorm(w, m, s)	w weights, m means, s standard deviations
mixmvnorm(w, m, sigma_L)	w weights, m means, sigma_L cholesky factors of covariances

These arguments to the mixture densities refer to the different density parameters and are automatically extracted from the mixtures when converted. Important here are the argument names as these must be used to declare the mixture prior. For each mixture to convert as part of the ... argument to mixstanvar a label is determined using the name given in the list. In case no name is given, then the name of the R object itself is used. To declare a prior for a parameter the mixture distribution must be used with it's arguments following the convention label_argument. Please refer to the examples section for an illustration.

Note: Models created by brms::brm() do use by default a data-dependent centering of covariates leading to a shift of the overall intercept. This is commonly not desirable in applications of this functionality. It is therefore strongly recommended to pass the option center=FALSE as argument to the brms formula created with the brms::bf() function as demonstrated with the example below.

Value

stanvars object to be used as argument to the stanvars argument of a brms::brm() model.

Examples

```
## Not run:
# The mixstanvar adapter requires the optional packages brms and glue
stopifnot(require("brms"), require("glue"))

# Assume we prefer a logistic regression MCMC analysis rather than a
# beta-binomial analysis for the responder endpoint of the ankylosing
# spondylitis (AS) example. Reasons to prefer a regression analysis is
# to allow for baseline covariate adjustments, for example.
map_AS_beta <- mixbeta(c(0.62, 19.2, 57.8), c(0.38, 3.5, 9.4))

# First we need to convert the beta mixture to a respective mixture on
# the log odds scale and approximate it with a normal mixture density.
map_AS_samp <- rmix(map_AS_beta, 1E4)
```

```
map_AS <- mixfit(logit(map_AS_samp), type = "norm", Nc = 2)

# Trial results for placebo and secukinumab.
trial <- data.frame(
  n = c(6, 24),
  r = c(1, 15),
  arm = factor(c("placebo", "secukinumab"))
)

# Define brms model such that the overall intercept corresponds to the
# placebo response rate on the logit scale. NOTE: The use of
# center=FALSE is required here as detailed in the note above.
model <- bf(r | trials(n) ~ 1 + arm, family = binomial, center = FALSE)
# to obtain detailed information on the declared model parameters use
# get_prior(model, data=trial)
# declare model prior with reference to mixture normal map prior...
model_prior <- prior(mixnorm(map_w, map_m, map_s), coef = Intercept) +
  prior(normal(0, 2), class = b)

# ... which must be made available to brms using the mixstanvar adapter.
# Note that the map_AS prior is labeled "map" as referred to in the
# previous prior declaration.
analysis <- brm(model,
  data = trial, prior = model_prior,
  stanvars = mixstanvar(map = map_AS),
  seed = 365634, refresh = 0
)

# Let's compare the logistic regression estimate for the probability
# of a positive treatment effect (secukinumab response rate exceeding
# the response rate of placebo) to the direct beta-binomial analysis:
hypothesis(analysis, "armsecukinumab > 0")

post_secukinumab <- postmix(mixbeta(c(1, 0.5, 1)), r = 15, n = 24)
post_placebo <- postmix(map_AS_beta, r = 1, n = 6)
pmixdiff(post_secukinumab, post_placebo, 0, lower.tail = FALSE)
# The posterior probability for a positive treatment effect
# is very close to unity in both cases.

## End(Not run)
```

`nsamples.gMAP` *Return the number of posterior samples*

Description

Return the number of posterior samples

Usage

```
## S3 method for class 'gMAP'
nsamples(object, ...)
```

Arguments

object	fitted model object
...	not used in this function

Examples

```
## Setting up dummy sampling for fast execution of example
## Please use 4 chains and 20x more warmup & iter in practice
.user_mc_options <- options(RBesT.MC.warmup=50, RBesT.MC.iter=100,
                            RBesT.MC.chains=2, RBesT.MC.thin=1)


set.seed(34563)
map_AS <- gMAP(cbind(r, n - r) ~ 1 | study,
  family = binomial,
  data = AS,
  tau.dist = "HalfNormal", tau.prior = 1,
  beta.prior = 2
)

nsamples(map_AS)

## Recover user set sampling defaults
options(.user_mc_options)
```

---

oc1S *Operating Characteristics for 1 Sample Design*

Description

The oc1S function defines a 1 sample design (prior, sample size, decision function) for the calculation of the frequency at which the decision is evaluated to 1 conditional on assuming known parameters. A function is returned which performs the actual operating characteristics calculations.

Usage

```
oc1S(prior, n, decision, ...)

## S3 method for class 'betaMix'
oc1S(prior, n, decision, ...)

## S3 method for class 'normMix'
```

```
oc1S(prior, n, decision, sigma, eps = 1e-06, family = NULL, offset = 0, ...)

## S3 method for class 'gammaMix'
oc1S(prior, n, decision, eps = 1e-06, ...)
```

Arguments

prior	Prior for analysis.
n	Sample size for the experiment.
decision	One-sample decision function to use; see decision1S.
...	Optional arguments.
sigma	The fixed reference scale. If left unspecified, the default reference scale of the prior is assumed. When family is a non-Gaussian family, sigma must not be specified (it is determined by the family). When family = gaussian(), sigma is required and acts as the dispersion parameter.
eps	Support of random variables are determined as the interval covering 1-eps probability mass. Defaults to 10^{-6}.
family	Optional family object specifying a GLM family and link function (e.g.\ binomial(), MASS::negative.binomial(theta)). When provided, the sampling standard deviation varies with the parameter value via the family's variance function and link. Default is NULL (constant sigma).
offset	Numeric scalar added to the linear predictor before evaluating the family's variance function. Relevant for Poisson/negative-binomial models with log link where offset = log(exposure). Default is 0.

Details

The oc1S function defines a 1 sample design and returns a function which calculates its operating characteristics. This is the frequency with which the decision function is evaluated to 1 under the assumption of a given true distribution of the data defined by a known parameter θ. The 1 sample design is defined by the prior, the sample size and the decision function, $D(y)$. These uniquely define the decision boundary, see decision1S_boundary().

When calling the oc1S function, then internally the critical value y_c (using decision1S_boundary()) is calculated and a function is returns which can be used to calculated the desired frequency which is evaluated as

$$F(y_c|\theta).$$

Value

Returns a function with one argument theta which calculates the frequency at which the decision function is evaluated to 1 for the defined 1 sample design. Note that the returned function takes vectors arguments.

Methods (by class)

- `oc1S(betaMix)`: Applies for binomial model with a mixture beta prior. The calculations use exact expressions.
- `oc1S(normMix)`: Applies for the normal model with known standard deviation σ and a normal mixture prior for the mean. As a consequence from the assumption of a known standard deviation, the calculation discards sampling uncertainty of the second moment. The function `oc1S` has an extra argument `eps` (defaults to 10^{-6}). The critical value y_c is searched in the region of probability mass `1-eps` for y.

  When `family` is specified, the sampling standard deviation becomes a function of the parameter value θ via

  $$\sigma(\theta) = \sqrt{\phi\, V(\mu)}/|g'(\mu)|$$

  where V is the variance function, g the link function of the family, and ϕ the dispersion parameter. For the Gaussian family $\phi = \sigma^2$ (so `sigma` must be supplied); for all other families $\phi = 1$ and `sigma` must *not* be given. Specifying `family = gaussian("identity")` with `sigma` is equivalent to the standard fixed-σ path.
- `oc1S(gammaMix)`: Applies for the Poisson model with a gamma mixture prior for the rate parameter. The function `oc1S` takes an extra argument `eps` (defaults to 10^{-6}) which determines the region of probability mass `1-eps` where the boundary is searched for y.

See Also

Other design1S: `decision1S()`, `decision1S_boundary()`, `pos1S()`

Examples

```
# non-inferiority example using normal approximation of log-hazard
# ratio, see ?decision1S for all details
s <- 2
flat_prior <- mixnorm(c(1, 0, 100), sigma = s)
nL <- 233
theta_ni <- 0.4
theta_a <- 0
alpha <- 0.05
beta <- 0.2
za <- qnorm(1 - alpha)
zb <- qnorm(1 - beta)
n1 <- round((s * (za + zb) / (theta_ni - theta_a))^2)
theta_c <- theta_ni - za * s / sqrt(n1)

# standard NI design
decA <- decision1S(1 - alpha, theta_ni, lower.tail = TRUE)

# double criterion design
# statistical significance (like NI design)
dec1 <- decision1S(1 - alpha, theta_ni, lower.tail = TRUE)
# require mean to be at least as good as theta_c
dec2 <- decision1S(0.5, theta_c, lower.tail = TRUE)
# combination
decComb <- decision1S(c(1 - alpha, 0.5), c(theta_ni, theta_c), lower.tail = TRUE)
```

```
theta_eval <- c(theta_a, theta_c, theta_ni)

# evaluate different designs at two sample sizes
designA_n1 <- oc1S(flat_prior, n1, decA)
designA_nL <- oc1S(flat_prior, nL, decA)
designC_n1 <- oc1S(flat_prior, n1, decComb)
designC_nL <- oc1S(flat_prior, nL, decComb)

# evaluate designs at the key log-HR of positive treatment (HR<1),
# the indecision point and the NI margin

designA_n1(theta_eval)
designA_nL(theta_eval)
designC_n1(theta_eval)
designC_nL(theta_eval)

# to understand further the dual criterion design it is useful to
# evaluate the criterions separately:
# statistical significance criterion to warrant NI...
designC1_nL <- oc1S(flat_prior, nL, dec1)
# ... or the clinically determined indifference point
designC2_nL <- oc1S(flat_prior, nL, dec2)

designC1_nL(theta_eval)
designC2_nL(theta_eval)

# see also ?decision1S_boundary to see which of the two criterions
# will drive the decision

# it can also be of interest to evaluate intermediate decisions
# where the trial is significant for non-inferiority but
# the mean estimate is in an intermediate range, say between theta_c
# and theta_f:
theta_f <- 0.3
decCombIntermediate <- decision1S(
  c(1 - alpha, 0.5, 0.8),
  c(theta_ni, theta_c, theta_f),
  lower.tail = c(TRUE, FALSE, TRUE)
)

# evaluate at two sample sizes again:
designIntermediate_n1 <- oc1S(flat_prior, n1, decCombIntermediate)
designIntermediate_nL <- oc1S(flat_prior, nL, decCombIntermediate)

designIntermediate_n1(theta_eval)
designIntermediate_nL(theta_eval)
```

---

oc2S *Operating Characteristics for 2 Sample Design*

Description

The oc2S function defines a 2 sample design (priors, sample sizes & decision function) for the calculation of operating characeristics. A function is returned which calculates the calculates the frequency at which the decision function is evaluated to 1 when assuming known parameters.

Usage

```
oc2S(prior1, prior2, n1, n2, decision, ...)

## S3 method for class 'betaMix'
oc2S(prior1, prior2, n1, n2, decision, eps, ...)

## S3 method for class 'normMix'
oc2S(
  prior1,
  prior2,
  n1,
  n2,
  decision,
  sigma1,
  sigma2,
  eps = 1e-06,
  Ngrid = 5,
  family = NULL,
  offset1 = 0,
  offset2 = offset1,
  ...
)

## S3 method for class 'gammaMix'
oc2S(prior1, prior2, n1, n2, decision, eps = 1e-06, ...)
```

Arguments

prior1	Prior for sample 1.
prior2	Prior for sample 2.
n1, n2	Sample size of the respective samples. Sample size n1 must be greater than 0 while sample size n2 must be greater or equal to 0.
decision	Two-sample decision function to use; see decision2S.
...	Optional arguments.
eps	Support of random variables are determined as the interval covering 1-eps probability mass. Defaults to 10^{-6}.
sigma1	The fixed reference scale of sample 1. If left unspecified, the default reference scale of the prior 1 is assumed.
sigma2	The fixed reference scale of sample 2. If left unspecified, the default reference scale of the prior 2 is assumed.

Ngrid	Determines density of discretization grid on which decision function is evaluated (see below for more details).
family	Optional family object specifying a GLM family and link function (e.g.\ binomial(), MASS::negative.binomial(theta)). When provided, the sampling standard deviation of each sample varies with the respective parameter value via the family's variance function and link. For the Gaussian family sigma1/sigma2 act as the dispersion parameters and must be supplied; for all other families they must *not* be given (they are determined by the family). Default is NULL (constant sigma1 and sigma2).
offset1	Numeric scalar added to the linear predictor of sample 1 before evaluating the family's variance function. Relevant for Poisson/negative-binomial models with log link where offset1 = log(exposure). Default is 0.
offset2	Numeric scalar added to the linear predictor of sample 2; see offset1. Defaults to offset1.

Details

The oc2S function defines a 2 sample design and returns a function which calculates its operating characteristics. This is the frequency with which the decision function is evaluated to 1 under the assumption of a given true distribution of the data defined by the known parameter θ_1 and θ_2. The 2 sample design is defined by the priors, the sample sizes and the decision function, $D(y_1, y_2)$. These uniquely define the decision boundary , see decision2S_boundary().

Calling the oc2S function calculates the decision boundary $D_1(y_2)$ (see decision2S_boundary()) and returns a function which can be used to calculate the desired frequency which is evaluated as

$$\int f_2(y_2|\theta_2)F_1(D_1(y_2)|\theta_1)dy_2.$$

See below for examples and specifics for the supported mixture priors.

Value

Returns a function which when called with two arguments theta1 and theta2 will return the frequencies at which the decision function is evaluated to 1 whenever the data is distributed according to the known parameter values in each sample. Note that the returned function takes vector arguments.

Methods (by class)

- oc2S(betaMix): Applies for binomial model with a mixture beta prior. The calculations use exact expressions. If the optional argument eps is defined, then an approximate method is used which limits the search for the decision boundary to the region of 1-eps probability mass. This is useful for designs with large sample sizes where an exact approach is very costly to calculate.
- oc2S(normMix): Applies for the normal model with known standard deviation σ and normal mixture priors for the means. As a consequence from the assumption of a known standard deviation, the calculation discards sampling uncertainty of the second moment. The function has two extra arguments (with defaults): eps (10^{-6}) and Ngrid (10). The decision boundary is

searched in the region of probability mass `1-eps`, respectively for y_1 and y_2. The continuous decision function is evaluated at a discrete grid, which is determined by a spacing with $\delta_2 = \sigma_2/\sqrt{N_{grid}}$. Once the decision boundary is evaluated at the discrete steps, a spline is used to inter-polate the decision boundary at intermediate points.

- `oc2S(gammaMix)`: Applies for the Poisson model with a gamma mixture prior for the rate parameter. The function `oc2S` takes an extra argument `eps` (defaults to 10^{-6}) which determines the region of probability mass `1-eps` where the boundary is searched for y_1 and y_2, respectively.

References

Schmidli H, Gsteiger S, Roychoudhury S, O'Hagan A, Spiegelhalter D, Neuenschwander B (2014). "Robust meta-analytic-predictive priors in clinical trials with historical control information." *Biometrics*, **70**(4), 1023–1032. doi:10.1111/biom.12242.

See Also

Other design2S: `decision2S()`, `decision2S_boundary()`, `pos2S()`

Examples

```
# example from Schmidli et al., 2014
dec <- decision2S(0.975, 0, lower.tail = FALSE)

prior_inf <- mixbeta(c(1, 4, 16))
prior_rob <- robustify(prior_inf, weight = 0.2, mean = 0.5)
prior_uni <- mixbeta(c(1, 1, 1))

N <- 40
N_ctl <- N - 20

# compare designs with different priors
design_uni <- oc2S(prior_uni, prior_uni, N, N_ctl, dec)
design_inf <- oc2S(prior_uni, prior_inf, N, N_ctl, dec)
design_rob <- oc2S(prior_uni, prior_rob, N, N_ctl, dec)

# type I error
curve(design_inf(x, x), 0, 1)
curve(design_uni(x, x), lty = 2, add = TRUE)
curve(design_rob(x, x), lty = 3, add = TRUE)

# power
curve(design_inf(0.2 + x, 0.2), 0, 0.5)
curve(design_uni(0.2 + x, 0.2), lty = 2, add = TRUE)
curve(design_rob(0.2 + x, 0.2), lty = 3, add = TRUE)
```

plot.EM *Diagnostic plots for EM fits*

Description

Produce diagnostic plots of EM fits returned from `mixfit()`.

Usage

```
## S3 method for class 'EM'
plot(
  x,
  size = 1.25,
  linewidth = size,
  link = c("identity", "logit", "log"),
  ...
)
```

Arguments

x	EM fit
size	controls point size.
linewidth	controls linewidth; set by default to same value as size.
link	Choice of an applied link function. Can take one of the values identity (default), logit or log.
...	Ignored.
	Overlays the fitted mixture density with a histogram and a density plot of the raw sample fitted. Applying a link function can be beneficial, for example a logit (log) link for beta (gamma) mixtures obtained from a Binomial (Poisson) gMAP() analysis.

Value

A list of ggplot2::ggplot() plots for diagnostics of the EM run. Detailed EM diagnostic plots are included only if the global option RBesT.verbose is set to TRUE. These include plots of the parameters of each component vs the iteration. The plot of the mixture density with a histogram and a density of the fitted sample is always returned.

Customizing ggplot2 plots

The returned plot is a **ggplot2** object. Please refer to the "Customizing Plots" vignette which is part of **RBesT** documentation for an introduction. For simple modifications (change labels, add reference lines, ...) consider the commands found in bayesplot-helpers. For more advanced customizations please use the **ggplot2** package directly. A description of the most common tasks can be found in the R Cookbook and a full reference of available commands can be found at the ggplot2 documentation site.

See Also

Other EM: `mixfit()`

Examples

```
bmix <- mixbeta(rob = c(0.2, 1, 1), inf = c(0.8, 10, 2))
bsamp <- rmix(bmix, 1000)
bfit <- mixfit(bsamp, type = "beta", Nc = 2)
pl <- plot(bfit)

print(pl$mixdens)
print(pl$mix)


# a number of additional plots are generated in verbose mode
.user_option <- options(RBesT.verbose = TRUE)
pl_all <- plot(bfit)

# recover previous user options
options(.user_option)

names(pl_all)
# [1] "mixdist" "a"   "b"   "w"   "m"   "N"   "Lm"  "lN"  "Lw"  "lli" "mixdens" "mixecdf" "mix"
```

`plot.gMAP` *Diagnostic plots for gMAP analyses*

Description

Diagnostic plots for gMAP analyses

Usage

```
## S3 method for class 'gMAP'
plot(x, size = NULL, linewidth = NULL, ...)
```

Arguments

`x`	`gMAP()` object
`size`	Controls size of forest plot.
`linewidth`	Controls line sizes of traceplots.
`...`	Ignored.

Details

Creates MCMC diagnostics and a forest plot (including model estimates) for a `gMAP()` analysis. For a customized forest plot, please use the dedicated function `forest_plot()`.

Value

The function returns a list of `ggplot2::ggplot()` objects.

Customizing ggplot2 plots

The returned plot is a **ggplot2** object. Please refer to the "Customizing Plots" vignette which is part of **RBesT** documentation for an introduction. For simple modifications (change labels, add reference lines, ...) consider the commands found in `bayesplot-helpers`. For more advanced customizations please use the **ggplot2** package directly. A description of the most common tasks can be found in the R Cookbook and a full reference of available commands can be found at the ggplot2 documentation site.

---

pos1S *Probability of Success for a 1 Sample Design*

Description

The `pos1S` function defines a 1 sample design (prior, sample size, decision function) for the calculation of the frequency at which the decision is evaluated to 1 when assuming a distribution for the parameter. A function is returned which performs the actual operating characteristics calculations.

Usage

```
pos1S(prior, n, decision, ...)

## S3 method for class 'betaMix'
pos1S(prior, n, decision, ...)

## S3 method for class 'normMix'
pos1S(prior, n, decision, sigma, eps = 1e-06, family = NULL, offset = 0, ...)

## S3 method for class 'gammaMix'
pos1S(prior, n, decision, eps = 1e-06, ...)
```

Arguments

`prior`	Prior for analysis.
`n`	Sample size for the experiment.
`decision`	One-sample decision function to use; see `decision1S`.
`...`	Optional arguments.
`sigma`	The fixed reference scale. If left unspecified, the default reference scale of the prior is assumed. When `family` is a non-Gaussian family, `sigma` must not be specified (it is determined by the family). When `family = gaussian()`, `sigma` is required and acts as the dispersion parameter.
`eps`	Support of random variables are determined as the interval covering `1-eps` probability mass. Defaults to 10^{-6}.

`family` Optional `family` object specifying a GLM family and link function (e.g.\ `binomial()`, `MASS::negative.binomial(theta)`). When provided, the sampling standard deviation varies with the parameter value via the family's variance function and link. Default is `NULL` (constant `sigma`).

`offset` Numeric scalar added to the linear predictor before evaluating the family's variance function. Relevant for Poisson/negative-binomial models with log link where `offset = log(exposure)`. Default is `0`.

Details

The `pos1S` function defines a 1 sample design and returns a function which calculates its probability of success. The probability of success is the frequency with which the decision function is evaluated to 1 under the assumption of a given true distribution of the data implied by a distribution of the parameter θ.

Calling the `pos1S` function calculates the critical value y_c and returns a function which can be used to evaluate the PoS for different predictive distributions and is evaluated as

$$\int F(y_c|\theta)p(\theta)d\theta,$$

where F is the distribution function of the sampling distribution and $p(\theta)$ specifies the assumed true distribution of the parameter θ. The distribution $p(\theta)$ is a mixture distribution and given as the `mix` argument to the function.

Value

Returns a function that takes as single argument `mix`, which is the mixture distribution of the control parameter. Calling this function with a mixture distribution then calculates the PoS.

Methods (by class)

- `pos1S(betaMix)`: Applies for binomial model with a mixture beta prior. The calculations use exact expressions.
- `pos1S(normMix)`: Applies for the normal model with known standard deviation σ and a normal mixture prior for the mean. As a consequence from the assumption of a known standard deviation, the calculation discards sampling uncertainty of the second moment. The function `pos1S` has an extra argument `eps` (defaults to 10^{-6}). The critical value y_c is searched in the region of probability mass `1-eps` for y.

  When `family` is specified, the sampling standard deviation becomes a function of the parameter value θ via

  $$\sigma(\theta) = \sqrt{\phi\, V(\mu)}/|g'(\mu)|$$

  where V is the variance function, g the link function of the family, and ϕ the dispersion parameter. For the Gaussian family $\phi = \sigma^2$ (so `sigma` must be supplied); for all other families $\phi = 1$ and `sigma` must *not* be given. Specifying `family = gaussian("identity")` with `sigma` is equivalent to the standard fixed-σ path.
- `pos1S(gammaMix)`: Applies for the Poisson model with a gamma mixture prior for the rate parameter. The function `pos1S` takes an extra argument `eps` (defaults to 10^{-6}) which determines the region of probability mass `1-eps` where the boundary is searched for y.

See Also

Other design1S: decision1S(), decision1S_boundary(), oc1S()

Examples

```
# non-inferiority example using normal approximation of log-hazard
# ratio, see ?decision1S for all details
s <- 2
flat_prior <- mixnorm(c(1, 0, 100), sigma = s)
nL <- 233
theta_ni <- 0.4
theta_a <- 0
alpha <- 0.05
beta <- 0.2
za <- qnorm(1 - alpha)
zb <- qnorm(1 - beta)
n1 <- round((s * (za + zb) / (theta_ni - theta_a))^2)
theta_c <- theta_ni - za * s / sqrt(n1)

# assume we would like to conduct at an interim analysis
# of PoS after having observed 20 events with a HR of 0.8.
# We first need the posterior at the interim ...
post_ia <- postmix(flat_prior, m = log(0.8), n = 20)

# dual criterion
decComb <- decision1S(c(1 - alpha, 0.5), c(theta_ni, theta_c), lower.tail = TRUE)

# ... and we would like to know the PoS for a successful
# trial at the end when observing 10 more events
pos_ia <- pos1S(post_ia, 10, decComb)

# our knowledge at the interim is just the posterior at
# interim such that the PoS is
pos_ia(post_ia)
```

---

pos2S *Probability of Success for 2 Sample Design*

Description

The pos2S function defines a 2 sample design (priors, sample sizes & decision function) for the calculation of the probability of success. A function is returned which calculates the calculates the frequency at which the decision function is evaluated to 1 when parameters are distributed according to the given distributions.

Usage

```
pos2S(prior1, prior2, n1, n2, decision, ...)

## S3 method for class 'betaMix'
pos2S(prior1, prior2, n1, n2, decision, eps, ...)

## S3 method for class 'normMix'
pos2S(
  prior1,
  prior2,
  n1,
  n2,
  decision,
  sigma1,
  sigma2,
  eps = 1e-06,
  Ngrid = 5,
  family = NULL,
  offset1 = 0,
  offset2 = offset1,
  ...
)

## S3 method for class 'gammaMix'
pos2S(prior1, prior2, n1, n2, decision, eps = 1e-06, ...)
```

Arguments

prior1	Prior for sample 1.
prior2	Prior for sample 2.
n1, n2	Sample size of the respective samples. Sample size n1 must be greater than 0 while sample size n2 must be greater or equal to 0.
decision	Two-sample decision function to use; see decision2S.
...	Optional arguments.
eps	Support of random variables are determined as the interval covering 1-eps probability mass. Defaults to 10^{-6}.
sigma1	The fixed reference scale of sample 1. If left unspecified, the default reference scale of the prior 1 is assumed.
sigma2	The fixed reference scale of sample 2. If left unspecified, the default reference scale of the prior 2 is assumed.
Ngrid	Determines density of discretization grid on which decision function is evaluated (see below for more details).
family	Optional family object specifying a GLM family and link function (e.g.\ binomial(), MASS::negative.binomial(theta)). When provided, the sampling standard deviation of each sample varies with the respective parameter value via the family's variance function and link. For the Gaussian family sigma1/sigma2 act

as the dispersion parameters and must be supplied; for all other families they must *not* be given (they are determined by the family). Default is NULL (constant sigma1 and sigma2).

offset1 Numeric scalar added to the linear predictor of sample 1 before evaluating the family's variance function. Relevant for Poisson/negative-binomial models with log link where offset1 = log(exposure). Default is 0.

offset2 Numeric scalar added to the linear predictor of sample 2; see offset1. Defaults to offset1.

Details

The pos2S function defines a 2 sample design and returns a function which calculates its probability of success. The probability of success is the frequency with which the decision function is evaluated to 1 under the assumption of a given true distribution of the data implied by a distirbution of the parameters θ_1 and θ_2.

The calculation is analogous to the operating characeristics oc2S() with the difference that instead of assuming known (point-wise) true parameter values a distribution is specified for each parameter.

Calling the pos2S function calculates the decision boundary $D_1(y_2)$ and returns a function which can be used to evaluate the PoS for different predictive distributions. It is evaluated as

$$\int\int\int f_2(y_2|\theta_2)\, p(\theta_2)\, F_1(D_1(y_2)|\theta_1)\, p(\theta_1)\, dy_2 d\theta_2 d\theta_1.$$

where F is the distribution function of the sampling distribution and $p(\theta_1)$ and $p(\theta_2)$ specifies the assumed true distribution of the parameters θ_1 and θ_2, respectively. Each distribution $p(\theta_1)$ and $p(\theta_2)$ is a mixture distribution and given as the mix1 and mix2 argument to the function.

For example, in the binary case an integration of the predictive distribution, the BetaBinomial, instead of the binomial distribution will be performed over the data space wherever the decision function is evaluated to 1. All other aspects of the calculation are as for the 2-sample operating characteristics, see oc2S().

Value

Returns a function which when called with two arguments mix1 and mix2 will return the frequencies at which the decision function is evaluated to 1. Each argument is expected to be a mixture distribution representing the assumed true distribution of the parameter in each group.

Methods (by class)

- pos2S(betaMix): Applies for binomial model with a mixture beta prior. The calculations use exact expressions. If the optional argument eps is defined, then an approximate method is used which limits the search for the decision boundary to the region of 1-eps probability mass. This is useful for designs with large sample sizes where an exact approach is very costly to calculate.
- pos2S(normMix): Applies for the normal model with known standard deviation σ and normal mixture priors for the means. As a consequence from the assumption of a known standard deviation, the calculation discards sampling uncertainty of the second moment. The function has two extra arguments (with defaults): eps (10^{-6}) and Ngrid (10). The decision boundary is

searched in the region of probability mass `1-eps`, respectively for y_1 and y_2. The continuous decision function is evaluated at a discrete grid, which is determined by a spacing with $\delta_2 = \sigma_2/\sqrt{N_{grid}}$. Once the decision boundary is evaluated at the discrete steps, a spline is used to inter-polate the decision boundary at intermediate points.

- `pos2S(gammaMix)`: Applies for the Poisson model with a gamma mixture prior for the rate parameter. The function `pos2S` takes an extra argument `eps` (defaults to 10^{-6}) which determines the region of probability mass `1-eps` where the boundary is searched for y_1 and y_2, respectively.

See Also

Other design2S: `decision2S()`, `decision2S_boundary()`, `oc2S()`

Examples

```
# see ?decision2S for details of example
priorT <- mixnorm(c(1, 0, 0.001), sigma = 88, param = "mn")
priorP <- mixnorm(c(1, -49, 20), sigma = 88, param = "mn")
# the success criteria is for delta which are larger than some
# threshold value which is why we set lower.tail=FALSE
successCrit <- decision2S(c(0.95, 0.5), c(0, 50), FALSE)

# example interim outcome
postP_interim <- postmix(priorP, n = 10, m = -50)
postT_interim <- postmix(priorT, n = 20, m = -80)

# assume that mean -50 / -80 were observed at the interim for
# placebo control(n=10) / active treatment(n=20) which gives
# the posteriors
postP_interim
postT_interim

# then the PoS to succeed after another 20/30 patients is
pos_final <- pos2S(postP_interim, postT_interim, 20, 30, successCrit)

pos_final(postP_interim, postT_interim)
```

`postmix` *Conjugate Posterior Analysis*

Description

Calculates the posterior distribution for data `data` given a prior `priormix`, where the prior is a mixture of conjugate distributions. The posterior is then also a mixture of conjugate distributions.

Usage

```
postmix(priormix, data, ...)

## S3 method for class 'betaMix'
postmix(priormix, data, n, r, ...)

## S3 method for class 'normMix'
postmix(priormix, data, n, m, se, ...)

## S3 method for class 'gammaMix'
postmix(priormix, data, n, m, ...)
```

Arguments

priormix	prior (mixture of conjugate distributions).
data	individual data. If the individual data is not given, then summary data has to be provided (see below).
...	includes arguments which depend on the specific case, see description below.
n	sample size.
r	Number of successes.
m	Sample mean.
se	Sample standard error.

Details

A conjugate prior-likelihood pair has the convenient property that the posterior is in the same distributional class as the prior. This property also applies to mixtures of conjugate priors. Let

$$p(\theta; \mathbf{w}, \mathbf{a}, \mathbf{b})$$

denote a conjugate mixture prior density for data

$$y|\theta \sim f(y|\theta),$$

where $f(y|\theta)$ is the likelihood. Then the posterior is again a mixture with each component k equal to the respective posterior of the kth prior component and updated weights w'_k,

$$p(\theta; \mathbf{w}', \mathbf{a}', \mathbf{b}'|y) = \sum_{k=1}^{K} w'_k \, p_k(\theta; a'_k, b'_k|y).$$

The weight w'_k for kth component is determined by the marginal likelihood of the new data y under the kth prior distribution which is given by the predictive distribution of the kth component,

$$w'_k \propto w_k \int p_k(\theta; a_k, b_k) \, f(y|\theta) \, d\theta \equiv w^*_k.$$

The final weight w'_k is then given by appropriate normalization, $w'_k = w^*_k / \sum_{k=1}^{K} w^*_k$. In other words, the weight of component k is proportional to the likelihood that data y is generated from the respective component, i.e. the marginal probability; for details, see for example *Schmidli et al., 2015*.

Note: The prior weights w_k are fixed, but the posterior weights $w'_k \neq w_k$ still change due to the changing normalization.

The data y can either be given as individual data or as summary data (sufficient statistics). See below for details for the implemented conjugate mixture prior densities.

Methods (by class)

- `postmix(betaMix)`: Calculates the posterior beta mixture distribution. The individual data vector is expected to be a vector of 0 and 1, i.e. a series of Bernoulli experiments. Alternatively, the sufficient statistics `n` and `r` can be given, i.e. number of trials and successes, respectively.
- `postmix(normMix)`: Calculates the posterior normal mixture distribution with the sampling likelihood being a normal with fixed standard deviation. Either an individual data vector `data` can be given or the sufficient statistics which are the standard error `se` and sample mean `m`. If the sample size `n` is used instead of the sample standard error, then the reference scale of the prior is used to calculate the standard error. Should standard error `se` and sample size `n` be given, then the reference scale of the prior is updated; however it is recommended to use the command `sigma()` to set the reference standard deviation.
- `postmix(gammaMix)`: Calculates the posterior gamma mixture distribution for Poisson and exponential likelihoods. Only the Poisson case is supported in this version.

Supported Conjugate Prior-Likelihood Pairs

Prior/Posterior	Likelihood	Predictive	Summaries
Beta	Binomial	Beta-Binomial	`n`, `r`
Normal	Normal (*fixed* σ)	Normal	`n`, `m`, `se`
Gamma	Poisson	Gamma-Poisson	`n`, `m`
Gamma	Exponential	Gamma-Exp (*not supported*)	`n`, `m`

References

Schmidli H, Gsteiger S, Roychoudhury S, O'Hagan A, Spiegelhalter D, Neuenschwander B (2014). "Robust meta-analytic-predictive priors in clinical trials with historical control information." *Biometrics*, **70**(4), 1023–1032. doi:10.1111/biom.12242.

Examples

```
# binary example with individual data (1=event,0=no event), uniform prior
prior.unif <- mixbeta(c(1, 1, 1))
data.indiv <- c(1, 0, 1, 1, 0, 1)
posterior.indiv <- postmix(prior.unif, data.indiv)
print(posterior.indiv)
```

```
# or with summary data (number of events and number of patients)
r <- sum(data.indiv)
n <- length(data.indiv)
posterior.sum <- postmix(prior.unif, n = n, r = r)
print(posterior.sum)

# binary example with robust informative prior and conflicting data
prior.rob <- mixbeta(c(0.5, 4, 10), c(0.5, 1, 1))
posterior.rob <- postmix(prior.rob, n = 20, r = 18)
print(posterior.rob)

# normal example with individual data
sigma <- 88
prior.mean <- -49
prior.se <- sigma / sqrt(20)
prior <- mixnorm(c(1, prior.mean, prior.se), sigma = sigma)
data.indiv <- c(-46, -227, 41, -65, -103, -22, 7, -169, -69, 90)
posterior.indiv <- postmix(prior, data.indiv)
# or with summary data (mean and number of patients)
mn <- mean(data.indiv)
n <- length(data.indiv)
posterior.sum <- postmix(prior, m = mn, n = n)
print(posterior.sum)
```

`preddist` *Predictive Distributions for Mixture Distributions*

Description

Predictive distribution for mixture of conjugate distributions (beta, normal, gamma).

Usage

```
preddist(mix, ...)

## S3 method for class 'betaMix'
preddist(mix, n = 1, ...)

## S3 method for class 'normMix'
preddist(mix, n = 1, sigma, ...)

## S3 method for class 'gammaMix'
preddist(mix, n = 1, ...)

## S3 method for class 'mvnormMix'
preddist(mix, ...)
```

Arguments

`mix`	mixture distribution
`...`	includes arguments which depend on the specific prior-likelihood pair, see description below.
`n`	predictive sample size, set by default to 1
`sigma`	The fixed reference scale of a normal mixture. If left unspecified, the default reference scale of the mixture is assumed.

Details

Given a mixture density (either a posterior or a prior)

$$p(\theta, \mathbf{w}, \mathbf{a}, \mathbf{b})$$

and a data likelihood of

$$y|\theta \sim f(y|\theta),$$

the predictive distribution of a one-dimensional summary y_n of n future observations is distributed as

$$y_n \sim \int p(\theta, \mathbf{w}, \mathbf{a}, \mathbf{b})\, f(y_n|\theta)\, d\theta.$$

This distribution is the marginal distribution of the data under the mixture density. For binary and Poisson data $y_n = \sum_{i=1}^{n} y_i$ is the sum over future events. For normal data, it is the mean$\bar{y}_n = 1/n \sum_{i=1}^{n} y_i$.

Value

The function returns for a normal, beta or gamma mixture the matching predictive distribution for y_n.

Methods (by class)

- `preddist(betaMix)`: Obtain the matching predictive distribution for a beta distribution, the BetaBinomial.
- `preddist(normMix)`: Obtain the matching predictive distribution for a Normal with constant standard deviation. Note that the reference scale of the returned Normal mixture is meaningless as the individual components are updated appropriatley.
- `preddist(gammaMix)`: Obtain the matching predictive distribution for a Gamma. Only Poisson likelihoods are supported.
- `preddist(mvnormMix)`: Multivariate normal mixtures predictive densities are not (yet) supported.

Supported Conjugate Prior-Likelihood Pairs

Prior/Posterior	Likelihood	Predictive	Summaries
Beta	Binomial	Beta-Binomial	`n`, `r`
Normal	Normal (*fixed* σ)	Normal	`n`, `m`, `se`
Gamma	Poisson	Gamma-Poisson	`n`, `m`
Gamma	Exponential	Gamma-Exp (*not supported*)	`n`, `m`

Examples

```
# Example 1: predictive distribution from uniform prior.
bm <- mixbeta(c(1, 1, 1))
bmPred <- preddist(bm, n = 10)
# predictive proabilities and cumulative predictive probabilities
x <- 0:10
d <- dmix(bmPred, x)
names(d) <- x
barplot(d)
cd <- pmix(bmPred, x)
names(cd) <- x
barplot(cd)
# median
mdn <- qmix(bmPred, 0.5)
mdn

# Example 2: 2-comp Beta mixture

bm <- mixbeta(inf = c(0.8, 15, 50), rob = c(0.2, 1, 1))
plot(bm)
bmPred <- preddist(bm, n = 10)
plot(bmPred)
mdn <- qmix(bmPred, 0.5)
mdn
d <- dmix(bmPred, x = 0:10)

n.sim <- 100000
r <- rmix(bmPred, n.sim)
d
table(r) / n.sim


# Example 3: 3-comp Normal mixture

m3 <- mixnorm(c(0.50, -0.2, 0.1), c(0.25, 0, 0.2), c(0.25, 0, 0.5), sigma = 10)
print(m3)
summary(m3)
plot(m3)
predm3 <- preddist(m3, n = 2)
plot(predm3)
print(predm3)
```

```
summary(predm3)
```

`predict.gMAP` *Predictions from gMAP analyses*

Description

Produces a sample of the predictive distribution.

Usage

```
## S3 method for class 'gMAP'
predict(
  object,
  newdata,
  type = c("response", "link"),
  probs = c(0.025, 0.5, 0.975),
  na.action = na.pass,
  thin,
  ...
)

## S3 method for class 'gMAPpred'
print(x, digits = 3, ...)

## S3 method for class 'gMAPpred'
summary(object, ...)

## S3 method for class 'gMAPpred'
as.matrix(x, ...)
```

Arguments

`newdata`	data.frame which must contain the same columns as input into the gMAP analysis. If left out, then a posterior prediction for the fitted data entries from the gMAP object is performed (shrinkage estimates).
`type`	sets reported scale (`response` (default) or `link`).
`probs`	defines quantiles to be reported.
`na.action`	how to handle missings.
`thin`	thinning applied is derived from the `gMAP` object.
`...`	ignored.
`x, object`	gMAP analysis object for which predictions are performed
`digits`	number of displayed significant digits.

Details

Predictions are made using the τ prediction stratum of the gMAP object. For details on the syntax, please refer to `predict.glm()` and the example below.

See Also

`gMAP()`, `predict.glm()`

Examples

```
## Setting up dummy sampling for fast execution of example
## Please use 4 chains and 20x more warmup & iter in practice
.user_mc_options <- options(RBesT.MC.warmup=50, RBesT.MC.iter=100,
                            RBesT.MC.chains=2, RBesT.MC.thin=1)

# create a fake data set with a covariate
trans_cov <- transform(
  transplant,
  country = cut(1:11, c(0, 5, 8, Inf), c("CH", "US", "DE"))
)
set.seed(34246)
map <- gMAP(
  cbind(r, n - r) ~ 1 + country | study,
  data = trans_cov,
  tau.dist = "HalfNormal",
  tau.prior = 1,
  # Note on priors: we make the overall intercept weakly-informative
  # and the regression coefficients must have tighter sd as these are
  # deviations in the default contrast parametrization
  beta.prior = rbind(c(0, 2), c(0, 1), c(0, 1)),
  family = binomial,
  ## ensure fast example runtime
  thin = 1,
  chains = 1
)

# posterior predictive distribution for each input data item (shrinkage estimates)
pred_cov <- predict(map)
pred_cov

# extract sample as matrix
samp <- as.matrix(pred_cov)

# predictive distribution for each input data item (if the input studies were new ones)
pred_cov_pred <- predict(map, trans_cov)
pred_cov_pred


# a summary function returns the results as matrix
summary(pred_cov)

# obtain a prediction for new data with specific covariates
```

```
pred_new <- predict(map, data.frame(country = "CH", study = 12))
pred_new
## Recover user set sampling defaults
options(.user_mc_options)
```

`robustify` *Robustify Mixture Priors*

Description

Add a non-informative component to a mixture prior.

Usage

```
robustify(priormix, weight, mean, n = 1, ...)

## S3 method for class 'betaMix'
robustify(priormix, weight, mean, n = 1, ...)

## S3 method for class 'gammaMix'
robustify(priormix, weight, mean, n = 1, ...)

## S3 method for class 'normMix'
robustify(priormix, weight, mean, n = 1, ..., sigma)
```

Arguments

`priormix`	orior (mixture of conjugate distributions).
`weight`	weight given to the non-informative component (0 < `weight` < 1).
`mean`	mean of the non-informative component. It is recommended to set this parameter explicitly.
`n`	number of observations the non-informative prior corresponds to, defaults to 1.
`...`	optional arguments are ignored.
`sigma`	Sampling standard deviation for the case of Normal mixtures.

Details

It is recommended to robustify informative priors derived with `gMAP()` using unit-information priors . This protects against prior-data conflict, see for example *Schmidli et al., 2015*.

The procedure can be used with beta, gamma and normal mixture priors. A unit-information prior (see *Kass and Wasserman, 1995*) corresponds to a prior which represents the observation of n=1 at the null hypothesis. As the null is problem dependent we *strongly recommend* to make use of the `mean` argument accordingly. See below for the definition of the default mean.

The weights of the mixture priors are rescaled to `(1-weight)` while the non-informative prior is assigned the `weight` given.

Value

New mixture with an extra non-informative component named `robust`.

Methods (by class)

- `robustify(betaMix)`: The default `mean` is set to 1/2 which represents no difference between the occurrence rates for one of the two outcomes. As the uniform `Beta(1,1)` is more appropriate in practical applications, `RBesT` uses `n+1` as the sample size such that the default robust prior is the uniform instead of the `Beta(1/2,1/2)` which strictly defined would be the unit information prior in this case.
- `robustify(gammaMix)`: The default `mean` is set to the mean of the prior mixture. It is strongly recommended to explicitly set the mean to the location of the null hypothesis.
- `robustify(normMix)`: The default `mean` is set to the mean of the prior mixture. It is strongly recommended to explicitly set the mean to the location of the null hypothesis, which is very often equal to 0. It is also recommended to explicitly set the sampling standard deviation using the `sigma` argument.

References

Schmidli H, Gsteiger S, Roychoudhury S, O'Hagan A, Spiegelhalter D, Neuenschwander B (2014). "Robust meta-analytic-predictive priors in clinical trials with historical control information." *Biometrics*, **70**(4), 1023–1032. doi:10.1111/biom.12242.

Kass RE, Wasserman L A Reference Bayesian Test for Nested Hypotheses and its Relationship to the Schwarz Criterion *J Amer Statist Assoc* 1995; 90(431):928-934.

See Also

`mixcombine()`

Examples

```
bmix <- mixbeta(inf1 = c(0.2, 8, 3), inf2 = c(0.8, 10, 2))
plot(bmix)
rbmix <- robustify(bmix, weight = 0.1, mean = 0.5)
rbmix
plot(rbmix)

gmnMix <- mixgamma(inf1 = c(0.2, 2, 3), inf2 = c(0.8, 2, 5), param = "mn")
plot(gmnMix)
rgmnMix <- robustify(gmnMix, weight = 0.1, mean = 2)
rgmnMix
plot(rgmnMix)

nm <- mixnorm(inf1 = c(0.2, 0.5, 0.7), inf2 = c(0.8, 2, 1), sigma = 2)
plot(nm)
rnMix <- robustify(nm, weight = 0.1, mean = 0, sigma = 2)
rnMix
plot(rnMix)
```

`transplant` *Transplant.*

Description

Data set containing historical information for standard treatment for a phase IV trial in de novo transplant patients. The primary outcome is treament failure (binary).

Usage

```
transplant
```

Format

A data frame with 4 rows and 3 variables:

study study

n study size

r number of events

References

Neuenschwander B, Capkun-Niggli G, Branson M, Spiegelhalter DJ (2010). "Summarizing historical information on controls in clinical trials." *Clinical Trials*, **7**(1), 5–18. doi:10.1177/1740774509356002.

Index

∗ **EM**
  mixfit, 45
  plot.EM, 67
∗ **datasets**
  AS, 5
  asthma, 6
  colitis, 9
  crohn, 10
  transplant, 84
∗ **design1S**
  decision1S, 11
  decision1S_boundary, 14
  oc1S, 60
  pos1S, 69
∗ **design2S**
  decision2S, 16
  decision2S_boundary, 19
  oc2S, 63
  pos2S, 71
∗ **mixdist**
  mix, 39
  mixbeta, 41
  mixcombine, 43
  mixgamma, 48
  mixjson, 50
  mixmvnorm, 51
  mixnorm, 53
  mixplot, 55
[[.mix *(mix)*, 39

acf(), *47*
AS, 5
as.matrix.gMAP *(gMAP)*, 29
as.matrix.gMAPpred *(predict.gMAP)*, 80
as_draws *(draws-RBesT)*, 22
as_draws_array *(draws-RBesT)*, 22
as_draws_df *(draws-RBesT)*, 22
as_draws_list *(draws-RBesT)*, 22
as_draws_matrix *(draws-RBesT)*, 22
as_draws_rvars *(draws-RBesT)*, 22
asthma, 6
automixfit, 7
automixfit(), *35*

base::c(), *52*
base::connections(), *50*
base::set.seed(), *35*
BinaryExactCI, 8
brms::bf(), *58*
brms::brm(), *57, 58*
brms::set_prior(), *58*
brms::stanvar(), *57*

coef.gMAP *(gMAP)*, 29
colitis, 9
crohn, 10

decision1S, 11, *14*, *61*, *69*
decision1S(), *16*, *62*, *71*
decision1S_boundary, 14
decision1S_boundary(), *12*, *61, 62*, *71*
decision2S, 16, *20*, *64*, *72*
decision2S(), *22*, *66*, *74*
decision2S_boundary, 19
decision2S_boundary(), *4*, *18*, *65, 66*, *74*
dmix *(mix)*, 39
dmixdiff *(mixdiff)*, 44
draws-RBesT, 22
drop(), *9*

ess, 24

family, *15*, *21*, *61*, *65*, *70*, *72*
fitted.gMAP *(gMAP)*, 29
forest_plot, 27
forest_plot(), *35*, *68*

ggplot2::ggplot(), *56*, *67*, *69*
gMAP, 29
gMAP(), *3*, *27, 28*, *47*, *67, 68*, *81, 82*

has_lower *(decision1S)*, 11
has_upper *(decision1S)*, 11

integrate(), *44*
inv_logit *(lodds)*, 38

jsonlite::fromJSON(), *50*
jsonlite::toJSON(), *50, 51*

likelihood, 37
likelihood<- *(likelihood)*, 37
lodds, 38
logit *(lodds)*, 38
lower *(decision1S)*, 11
lower(), *12*, *18*

mix, 39, *42, 43*, *49*, *51, 52*, *54*, *56*
mixbeta, 41
mixbeta(), *40*, *43*, *49*, *51, 52*, *54*, *56*
mixcombine, 43
mixcombine(), *40*, *42*, *49*, *51, 52*, *54*, *56*, *83*
mixdiff, 44
mixfit, 45
mixfit(), *7*, *51*, *67, 68*
mixgamma, 48
mixgamma(), *40*, *42, 43*, *51, 52*, *54*, *56*
mixjson, *40*, *42, 43*, *49*, 50, *52*, *54*, *56*
mixmvnorm, 51
mixmvnorm(), *40*, *42, 43*, *49*, *51*, *54*, *56*
mixnorm, 53
mixnorm(), *40*, *42, 43*, *49*, *51, 52*, *56*, *58*
mixplot, *40*, *42, 43*, *49*, *51, 52*, *54*, 55
mixstanvar, 57
mn2beta *(mixbeta)*, 41
mn2gamma *(mixgamma)*, 48
mn2norm *(mixnorm)*, 53
ms2beta *(mixbeta)*, 41
ms2gamma *(mixgamma)*, 48
msr2mvnorm *(mixmvnorm)*, 51

nsamples *(nsamples.gMAP)*, 59
nsamples.gMAP, 59

oc1S, 60
oc1S(), *12*, *16*, *71*
oc1Sdecision *(decision1S)*, 11
oc2S, 63
oc2S(), *18*, *22*, *73, 74*
oc2Sdecision *(decision2S)*, 16

pbinom(), *15*
plot.EM, 67
plot.EM(), *47*
plot.gMAP, 68
plot.gMAP(), *35*
plot.mix *(mixplot)*, 55
plot.mix(), *40*
plot.mvnormMix *(mixplot)*, 55
pmix *(mix)*, 39
pmixdiff *(mixdiff)*, 44
pos1S, 69
pos1S(), *12*, *16*, *62*
pos2S, 71
pos2S(), *18*, *22*, *66*
posterior::as_draws_matrix(), *33*
posterior::draws(), *23*
posterior::subset_draws(), *23*
postmix, 74
preddist, 77
preddist(), *40*
predict.glm(), *81*
predict.gMAP, 80
predict.gMAP(), *35*
print.betaBinomialMix *(mixbeta)*, 41
print.betaMix *(mixbeta)*, 41
print.gammaExpMix *(mixgamma)*, 48
print.gammaMix *(mixgamma)*, 48
print.gammaPoissonMix *(mixgamma)*, 48
print.gMAP *(gMAP)*, 29
print.gMAPpred *(predict.gMAP)*, 80
print.mvnormMix *(mixmvnorm)*, 51
print.normMix *(mixnorm)*, 53

qmix *(mix)*, 39
qmixdiff *(mixdiff)*, 44

RBesT *(RBesT-package)*, 3
RBesT(), *35*
RBesT-package, 3
read_mix_json *(mixjson)*, 50
rmix *(mix)*, 39
rmixdiff *(mixdiff)*, 44
robustify, 82
robustify(), *43*
rstan::rstan(), *35*

sigma *(mixnorm)*, 53
sigma(), *76*
sigma.mvnormMix *(mixmvnorm)*, 51

sigma<- *(mixnorm)*, 53
stan, *31*
stats::glm(), *29*, *33*
stats::model.matrix.default(), *31*
summary.betaBinomialMix *(mixbeta)*, 41
summary.betaMix *(mixbeta)*, 41
summary.gammaMix *(mixgamma)*, 48
summary.gammaPoissonMix *(mixgamma)*, 48
summary.gMAP *(gMAP)*, 29
summary.gMAPpred *(predict.gMAP)*, 80
summary.mvnormMix *(mixmvnorm)*, 51
summary.normMix *(mixnorm)*, 53

transplant, 84

upper *(decision1S)*, 11
upper(), *12*, *18*

write_mix_json *(mixjson)*, 50