Package ‘multinma’

May 7, 2024

Title Bayesian Network Meta-Analysis of Individual and Aggregate Data

Version 0.7.0

Description Network meta-analysis and network meta-regression models for aggregate data, individual patient data, and mixtures of both individual and aggregate data using multilevel network meta-regression as described by Phillippo et al. (2020) <doi:10.1111/rssa.12579>. Models are estimated in a Bayesian framework using 'Stan'.

License GPL-3

Encoding UTF-8

LazyData true

Biarch true

Depends R (>= 3.4.0)

Imports methods, stats, graphics, survival, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.26.0), rstantools (>= 2.0.0), Rdpack (>= 0.7), tibble (>= 2.1.3), dplyr (>= 1.0.0), rlang, purrr, forcats, glue, randtoolbox, copula, tidyr (>= 1.0.0), stringr, Matrix, igraph, ggraph, ggplot2 (>= 3.3.0), ggdist (>= 2.1.1), truncdist, bayesplot

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

SystemRequirements GNU make

RoxygenNote 7.3.1

Suggests testthat (>= 2.1.0), vdiffr, withr, knitr, rmarkdown, parallel, R.rsp, rprojroot, loo (>= 2.0.0), logitnorm, crayon, tidygraph, pkgdown, splines2 (>= 0.5.0), flexsurv, rstpm2

RdMacros Rdpack

URL `https://dmphillippo.github.io/multinma/`, `https://github.com/dmphillippo/multinma`

BugReports `https://github.com/dmphillippo/multinma/issues`

VignetteBuilder knitr, R.rsp

NeedsCompilation yes

Author David M. Phillippo [aut, cre] (<https://orcid.org/0000-0003-2672-7841>)

Maintainer David M. Phillippo <david.phillippo@bristol.ac.uk>

Repository CRAN

Date/Publication 2024-05-07 15:40:02 UTC

R topics documented:

`multinma-package` *multinma: A Package for Network Meta-Analysis of Individual and Aggregate Data in Stan*

Description

An R package for performing network meta-analysis and network meta-regression with aggregate data, individual patient data, or mixtures of both.

Details

Network meta-analysis (NMA) combines (aggregate) data from multiple studies on multiple treatments in order to produce consistent estimates of relative treatment effects between each pair of treatments in the network (Dias et al. 2011).

Network meta-regression (NMR) extends NMA to include covariates, allowing adjustment for differences in effect-modifying variables between studies (Dias et al. 2011). NMR is typically performed using aggregate data (AgD), which lacks power and is prone to ecological bias. NMR with individual patient data (IPD) is the gold standard, if data are available.

Multilevel network meta-regression (ML-NMR) allows IPD and AgD to be incorporated together in a network meta-regression (Phillippo et al. 2020; Phillippo 2019). As in IPD NMR, an individual-level regression model is defined. AgD studies are then fitted by integrating the individual-level model over the respective covariate distributions. This correctly links the two levels of the model (instead of "plugging in" mean covariate values), avoiding aggregation bias. Population-adjusted treatment effects (Phillippo et al. 2016) can be produced for any study population in the network, or for an external target population.

Models are estimated in a Bayesian framework using Stan (Carpenter et al. 2017). Quasi-Monte Carlo numerical integration based on Sobol' sequences is used for the integration in ML-NMR models, with a Gaussian copula to account for correlations between covariates (Phillippo et al. 2020; Phillippo 2019).

Getting Started

A good place to start is with the package vignettes which walk through example analyses, see `vignette("vignette_overview")` for an overview. The series of NICE Technical Support Documents on evidence synthesis gives a detailed introduction to network meta-analysis:

Dias S, Welton NJ, Sutton AJ, Caldwell DM, Lu G, Reken S, Ades AE (2011). "NICE DSU Technical Support Documents 1-7: Evidence Synthesis for Decision Making." National Institute for Health and Care Excellence. `https://www.sheffield.ac.uk/nice-dsu`.

Multilevel network meta-regression is set out in the following methods paper:

Phillippo DM, Dias S, Ades AE, Belger M, Brnabic A, Schacht A, Saure D, Kadziola Z, Welton NJ (2020). "Multilevel Network Meta-Regression for population-adjusted treatment comparisons." *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, **183**(3), 1189–1210. doi:10.1111/rssa.12579.

Author(s)

Maintainer: David M. Phillippo `<david.phillippo@bristol.ac.uk>` (ORCID)

References

Carpenter B, Gelman A, Hoffman MD, Lee D, Goodrich B, Betancourt M, Brubaker M, Guo J, Li P, Riddell A (2017). "Stan: A Probabilistic Programming Language." *Journal of Statistical Software*, **76**(1). doi:10.18637/jss.v076.i01.

Dias S, Sutton AJ, Welton NJ, Ades AE (2011). "NICE DSU Technical Support Document 3: Heterogeneity: subgroups, meta-regression, bias and bias-adjustment." National Institute for Health and Care Excellence. `https://www.sheffield.ac.uk/nice-dsu`.

Dias S, Welton NJ, Sutton AJ, Ades AE (2011). "NICE DSU Technical Support Document 2: A generalised linear modelling framework for pair-wise and network meta-analysis of randomised controlled trials." National Institute for Health and Care Excellence. `https://www.sheffield.ac.uk/nice-dsu`.

Phillippo DM (2019). *Calibration of Treatment Effects in Network Meta-Analysis using Individual Patient Data*. Ph.D. thesis, University of Bristol. Available from `https://research-information.bris.ac.uk/`.

Phillippo DM, Ades AE, Dias S, Palmer S, Abrams KR, Welton NJ (2016). "NICE DSU Technical Support Document 18: Methods for population-adjusted indirect comparisons in submission to NICE." National Institute for Health and Care Excellence. `https://www.sheffield.ac.uk/nice-dsu`.

Phillippo DM, Dias S, Ades AE, Belger M, Brnabic A, Schacht A, Saure D, Kadziola Z, Welton NJ (2020). "Multilevel Network Meta-Regression for population-adjusted treatment comparisons." *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, **183**(3), 1189–1210. doi:10.1111/rssa.12579.

See Also

Useful links:

- `https://dmphillippo.github.io/multinma/`
- `https://github.com/dmphillippo/multinma`
- Report bugs at `https://github.com/dmphillippo/multinma/issues`

`.default` *Set default values*

Description

The `.default()` function is used internally to mark certain values as default, so that the user may be notified when default values are being used. For example, choosing a default reference treatment for a network, or using default prior distributions. The function `.is_default()` checks whether an argument/object is set to a default value. Neither of these functions are intended to be called by the user.

Usage

```
.default(x = list())

.is_default(x)
```

Arguments

`x`	An object

Value

For `.default()`, an identical object with additional attribute `.default`. For `.is_default()`, a logical value (`TRUE` or `FALSE`).

`adapt_delta` *Target average acceptance probability*

Description

The Stan control argument `adapt_delta` sets the target average acceptance probability for the No-U-Turn Sampler (NUTS) used by Stan.

Details

The default value of `adapt_delta` used by `nma()` is 0.8 for fixed effect models, and 0.95 for random effects models.

You should not need to change `adapt_delta` unless you see a warning message about divergent transitions. Increasing `adapt_delta` from the default to a value closer to 1 means that Stan will use a smaller step size, making sampling slower but more robust, and resulting in fewer divergent transitions.

For more details see the Stan documentation available from `https://mc-stan.org/users/documentation/`.

add_integration — *Add numerical integration points to aggregate data*

Description

The `add_integration()` generic creates Quasi-Monte Carlo numerical integration points using a Gaussian copula and Sobol' sequences, as described in Phillippo et al. (2020). Methods are available for networks stored in `nma_data` objects, and for data frames. The function `unnest_integration()` unnests integration points stored in a data frame, to aid plotting or other exploration.

Usage

```
add_integration(x, ...)

## Default S3 method:
add_integration(x, ...)

## S3 method for class 'data.frame'
add_integration(
  x,
  ...,
  cor = NULL,
  cor_adjust = NULL,
  n_int = 64L,
  int_args = list()
)

## S3 method for class 'nma_data'
add_integration(
  x,
  ...,
  cor = NULL,
  cor_adjust = NULL,
  n_int = 64L,
  int_args = list()
)

unnest_integration(data)
```

Arguments

`x`	An `nma_data` object, as created by the `set_*()` functions or `combine_network()`, or data frame
`...`	Distributions for covariates, see "Details"
`cor`	Correlation matrix to use for generating the integration points. By default, this takes a weighted correlation matrix from all IPD studies. Rows and columns should match the order of covariates specified in `...`.

`cor_adjust`	Adjustment to apply to the correlation matrix given by `cor` (or computed from the IPD if `cor = NULL`) to obtain the Gaussian copula correlations, either `"spearman"`, `"pearson"`, or `"none"`, see "Details". The default when `cor = NULL` is `"spearman"`, otherwise the default is `"pearson"`.
`n_int`	Number of integration points to generate, default 64. Powers of 2 are recommended, as these are expected to be particularly efficient for QMC integration.
`int_args`	A named list of arguments to pass to `sobol()`
`data`	Data frame with nested integration points, stored in list columns as `.int_<variable name>`

Details

The arguments passed to `...` specify distributions for the covariates. Argument names specify the name of the covariate, which should match a covariate name in the IPD (if IPD are present). The required marginal distribution is then specified using the function `distr()`.

The argument `cor_adjust` specifies how the correlation matrix given by `cor` (or computed from the IPD if `cor = NULL`) is adjusted to obtain the correlation matrix for the Gaussian copula, using the formulae in Xiao and Zhou (2018).

- `cor_adjust = "spearman"` should be used when the correlations `cor` have been computed using Spearman's rank correlation. Correlations between continuous covariates will be reproduced exactly by the integration points. Correlations between discrete covariates will be reproduced approximately. This is the default when `cor = NULL` and correlations are calculated from the IPD studies.
- `cor_adjust = "pearson"` should be used when the correlations `cor` have been computed using Pearson's product-moment correlation. Correlations between Normal covariates will be reproduced exactly by the integration points, all others will be reproduced approximately. Correlations between discrete covariates will be reproduced approximately (and identically to `cor_adjust = "spearman"`). This is the default when `cor` is provided by the user, since `cor()` defaults to `method = "pearson"` and Pearson correlations are most likely reported in published data. However, we recommend providing Spearman correlations (e.g. from `cor(., method = "spearman")`) and using `cor_adjust = "spearman"` where possible.
- `cor_adjust = "none"` allows the user to specify the correlation matrix for the Gaussian copula directly; no adjustment is applied.
- `cor_adjust = "legacy"` is also available, which reproduces exactly the behaviour from version 0.3.0 and earlier. This is similar to `cor_adjust = "none"`, but unadjusted Spearman correlations are used if `cor = NULL`.

When adding integration points to a network object the correlation matrix used is stored in `$int_cor`, and the copula correlation matrix and adjustment used are stored as attributes of `$int_cor`. If this correlation matrix is passed again to `add_integration()` (e.g. to reuse the correlations for an external target population) this will be detected, and the correct setting for `cor_adjust` will automatically be applied.

Value

For the `nma_data` method, an object of class nma_data. For the `data.frame` method, the input data frame is returned (as a tibble) with an added column for each covariate (prefixed with ".int_"), containing the numerical integration points nested as length-`n_int` vectors within each row. For `unnest_integration()`, a data frame with integration points unnested.

References

Phillippo DM, Dias S, Ades AE, Belger M, Brnabic A, Schacht A, Saure D, Kadziola Z, Welton NJ (2020). "Multilevel Network Meta-Regression for population-adjusted treatment comparisons." *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, **183**(3), 1189–1210. doi:10.1111/rssa.12579.

Xiao Q, Zhou S (2018). "Matching a correlation coefficient by a Gaussian copula." *Communications in Statistics - Theory and Methods*, **48**(7), 1728–1747. doi:10.1080/03610926.2018.1439962.

Examples

```
## Plaque psoriasis ML-NMR - network setup and adding integration points
# Set up plaque psoriasis network combining IPD and AgD
library(dplyr)
pso_ipd <- filter(plaque_psoriasis_ipd,
                  studyc %in% c("UNCOVER-1", "UNCOVER-2", "UNCOVER-3"))

pso_agd <- filter(plaque_psoriasis_agd,
                  studyc == "FIXTURE")

head(pso_ipd)
head(pso_agd)

pso_ipd <- pso_ipd %>%
  mutate(# Variable transformations
    bsa = bsa / 100,
    prevsys = as.numeric(prevsys),
    psa = as.numeric(psa),
    weight = weight / 10,
    durnpso = durnpso / 10,
    # Treatment classes
    trtclass = case_when(trtn == 1 ~ "Placebo",
                         trtn %in% c(2, 3, 5, 6) ~ "IL blocker",
                         trtn == 4 ~ "TNFa blocker"),
    # Check complete cases for covariates of interest
    complete = complete.cases(durnpso, prevsys, bsa, weight, psa)
  )

pso_agd <- pso_agd %>%
  mutate(
    # Variable transformations
    bsa_mean = bsa_mean / 100,
    bsa_sd = bsa_sd / 100,
    prevsys = prevsys / 100,
    psa = psa / 100,
    weight_mean = weight_mean / 10,
    weight_sd = weight_sd / 10,
    durnpso_mean = durnpso_mean / 10,
    durnpso_sd = durnpso_sd / 10,
    # Treatment classes
    trtclass = case_when(trtn == 1 ~ "Placebo",
                         trtn %in% c(2, 3, 5, 6) ~ "IL blocker",
                         trtn == 4 ~ "TNFa blocker")
  )

# Exclude small number of individuals with missing covariates
pso_ipd <- filter(pso_ipd, complete)

pso_net <- combine_network(
  set_ipd(pso_ipd,
          study = studyc,
          trt = trtc,
          r = pasi75,
          trt_class = trtclass),
  set_agd_arm(pso_agd,
              study = studyc,
              trt = trtc,
              r = pasi75_r,
              n = pasi75_n,
              trt_class = trtclass)
)

# Print network details
pso_net

# Add integration points to the network
pso_net <- add_integration(pso_net,
  durnpso = distr(qgamma, mean = durnpso_mean, sd = durnpso_sd),
  prevsys = distr(qbern, prob = prevsys),
  bsa = distr(qlogitnorm, mean = bsa_mean, sd = bsa_sd),
  weight = distr(qgamma, mean = weight_mean, sd = weight_sd),
  psa = distr(qbern, prob = psa),
  n_int = 64)


## Adding integration points to a data frame, e.g. for prediction
# Define a data frame of covariate summaries
new_agd_int <- data.frame(
  bsa_mean = 0.6,
  bsa_sd = 0.3,
  prevsys = 0.1,
  psa = 0.2,
  weight_mean = 10,
  weight_sd = 1,
  durnpso_mean = 3,
  durnpso_sd = 1)

# Adding integration points, using the weighted average correlation matrix
# computed for the plaque psoriasis network
new_agd_int <- add_integration(new_agd_int,
  durnpso = distr(qgamma, mean = durnpso_mean, sd = durnpso_sd),
  prevsys = distr(qbern, prob = prevsys),
  bsa = distr(qlogitnorm, mean = bsa_mean, sd = bsa_sd),
  weight = distr(qgamma, mean = weight_mean, sd = weight_sd),
  psa = distr(qbern, prob = psa),
  cor = pso_net$int_cor,
  n_int = 64)

# Here, since we reused the correlation matrix pso_net$int_cor from the
# network, the correct setting of cor_adjust = "spearman" is automatically
# applied

new_agd_int
```

`as.array.stan_nma` *Convert samples into arrays, matrices, or data frames*

Description

Samples (post warm-up) from a `stan_nma` model object can be coerced into an array, matrix, or data frame.

Usage

```
## S3 method for class 'stan_nma'
as.array(x, ..., pars, include = TRUE)

## S3 method for class 'stan_nma'
as.data.frame(x, ..., pars, include = TRUE)

## S3 method for class 'stan_nma'
as_tibble(x, ..., pars, include = TRUE)

## S3 method for class 'stan_nma'
as.tibble(x, ..., pars, include = TRUE)

## S3 method for class 'stan_nma'
as.matrix(x, ..., pars, include = TRUE)
```

Arguments

`x`	A `stan_nma` object
`...`	Additional arguments passed to `as.array.stanfit()`
`pars`	Optional character vector of parameter names to include in output. If not specified, all parameters are used.
`include`	Logical, are parameters in `pars` to be included (`TRUE`, default) or excluded (`FALSE`)?

Value

The `as.array()` method produces a 3D array [Iteration, Chain, Parameter] containing posterior samples of each parameter (as class mcmc_array). This has the side effect of enabling `bayesplot` functions to seamlessly work on `stan_nma` objects.

The `as.data.frame()` method produces a data frame containing posterior samples of each parameter, combined over all chains.

The `as.matrix()` method produces a matrix containing posterior samples of each parameter, combined over all chains.

`as.igraph.nma_data` *Convert networks to graph objects*

Description

The method `as.igraph()` converts `nma_data` objects into the form used by the igraph package. The method `as_tbl_graph()` converts `nma_data` objects into the form used by the ggraph and tidygraph packages.

Usage

```
## S3 method for class 'nma_data'
as.igraph(x, ..., collapse = TRUE)

## S3 method for class 'nma_data'
as_tbl_graph(x, ...)
```

Arguments

`x`	An nma_data object to convert
`...`	Additional arguments
`collapse`	Logical, collapse edges over studies? Default `TRUE`, only one edge is produced for each comparison (by IPD or AgD study type) with a `.nstudy` attribute giving the number of studies making that comparison. If `FALSE`, repeated edges are added for each study making the comparison.

Value

An `igraph` object for `as.igraph()`, a `tbl_graph` object for `as_tbl_graph()`.

Examples

```
# Set up network of smoking cessation data
head(smoking)

smk_net <- set_agd_arm(smoking,
                       study = studyn,
                       trt = trtc,
                       r = r,
                       n = n,
                       trt_ref = "No intervention")

# Print details
smk_net

# Convert to igraph object
igraph::as.igraph(smk_net)  # Edges combined by default
igraph::as.igraph(smk_net, collapse = FALSE)  # Without combining edges

# Convert to tbl_graph object
tidygraph::as_tbl_graph(smk_net)  # Edges combined by default
tidygraph::as_tbl_graph(smk_net, collapse = FALSE)  # Without combining edges
```

as.stanfit *as.stanfit*

Description

Attempt to turn an object into a stanfit object.

Usage

```
as.stanfit(x, ...)

## S3 method for class 'stan_nma'
as.stanfit(x, ...)

## Default S3 method:
as.stanfit(x, ...)
```

Arguments

x	an object
...	additional arguments

Value

A stanfit object.

`atrial_fibrillation` *Stroke prevention in atrial fibrillation patients*

Description

Data frame containing the results of 26 trials comparing 17 treatments in 4 classes for the prevention of stroke in patients with atrial fibrillation (Cooper et al. 2009). The data are the corrected versions given by van Valkenhoef and Kuiper (2016).

Usage

```
atrial_fibrillation
```

Format

A data frame with 63 rows and 11 variables:

studyc study name

studyn numeric study ID

trtc treatment name

trtn numeric treatment code

trt_class treatment class

r number of events

n sample size

E person-years at risk

stroke proportion of individuals with prior stroke

year year of study publication

followup mean length of follow-up (years)

References

Cooper NJ, Sutton AJ, Morris D, Ades AE, Welton NJ (2009). "Addressing between-study heterogeneity and inconsistency in mixed treatment comparisons: Application to stroke prevention treatments in individuals with non-rheumatic atrial fibrillation." *Statistics in Medicine*, **28**(14), 1861–1881. doi:10.1002/sim.3594.

van Valkenhoef G, Kuiper J (2016). *gemtc: Network Meta-Analysis Using Bayesian Methods*. R package version 0.8-2, `https://CRAN.R-project.org/package=gemtc`.

`bcg_vaccine` *BCG vaccination*

Description

Data frame containing the results of 13 trials comparing BCG vaccination to no vaccination for preventing tuberculosis (TB) (Dias et al. 2011; Berkey et al. 1995). The numbers of individuals diagnosed with TB in each arm during the study follow-up period are recorded. The absolute degrees latitude at which the study was conducted are also recorded.

Usage

```
bcg_vaccine
```

Format

A data frame with 26 rows and 6 variables:

studyn numeric study ID

trtn numeric treatment code

trtc treatment name

latitude absolute degrees latitude

r number diagnosed with TB

n sample size

References

Berkey CS, Hoaglin DC, Mosteller F, Colditz GA (1995). “A random-effects regression model for meta-analysis.” *Statistics in Medicine*, **14**(4), 395–411. doi:10.1002/sim.4780140406.

Dias S, Sutton AJ, Welton NJ, Ades AE (2011). “NICE DSU Technical Support Document 3: Heterogeneity: subgroups, meta-regression, bias and bias-adjustment.” National Institute for Health and Care Excellence. `https://www.sheffield.ac.uk/nice-dsu`.

`blocker` *Beta blockers to prevent mortality after MI*

Description

Data frame containing the number of deaths in 22 trials comparing beta blockers vs. control for preventing mortality after myocardial infarction (Carlin 1992; Dias et al. 2011).

Usage

```
blocker
```

Format

A data frame with 44 rows and 5 variables:

studyn numeric study ID

trtn numeric treatment code

trtc treatment name

r total number of events

n total number of individuals

References

Carlin JB (1992). "Meta-analysis for 2 x 2 tables: A bayesian approach." *Statistics in Medicine*, **11**(2), 141–158. doi:10.1002/sim.4780110202.

Dias S, Welton NJ, Sutton AJ, Ades AE (2011). "NICE DSU Technical Support Document 2: A generalised linear modelling framework for pair-wise and network meta-analysis of randomised controlled trials." National Institute for Health and Care Excellence. `https://www.sheffield.ac.uk/nice-dsu`.

`combine_network` *Combine multiple data sources into one network*

Description

Multiple data sources created using `set_ipd()`, `set_agd_arm()`, or `set_agd_contrast()` can be combined into a single network for analysis.

Usage

```
combine_network(..., trt_ref)
```

Arguments

`...`	multiple data sources, as defined using the `set_*` functions
`trt_ref`	reference treatment for the entire network, as a string (or coerced as such) referring to the levels of the treatment factor variable

Value

An object of class nma_data

See Also

`set_ipd()`, `set_agd_arm()`, and `set_agd_contrast()` for defining different data sources.

`print.nma_data()` for the print method displaying details of the network, and `plot.nma_data()` for network plots.

Examples

```
## Parkinson's - combining contrast- and arm-based data
studies <- parkinsons$studyn
(parkinsons_arm <- parkinsons[studies %in% 1:3, ])
(parkinsons_contr <- parkinsons[studies %in% 4:7, ])

park_arm_net <- set_agd_arm(parkinsons_arm,
                            study = studyn,
                            trt = trtn,
                            y = y,
                            se = se,
                            sample_size = n)

park_contr_net <- set_agd_contrast(parkinsons_contr,
                                   study = studyn,
                                   trt = trtn,
                                   y = diff,
                                   se = se_diff,
                                   sample_size = n)

park_net <- combine_network(park_arm_net, park_contr_net)

# Print network details
park_net

# Plot network
plot(park_net, weight_edges = TRUE, weight_nodes = TRUE)

## Plaque Psoriasis - combining IPD and AgD in a network
# Set up plaque psoriasis network combining IPD and AgD
library(dplyr)
pso_ipd <- filter(plaque_psoriasis_ipd,
                  studyc %in% c("UNCOVER-1", "UNCOVER-2", "UNCOVER-3"))

pso_agd <- filter(plaque_psoriasis_agd,
                  studyc == "FIXTURE")

head(pso_ipd)
head(pso_agd)

pso_ipd <- pso_ipd %>%
  mutate(# Variable transformations
    bsa = bsa / 100,
    prevsys = as.numeric(prevsys),
    psa = as.numeric(psa),
    weight = weight / 10,
    durnpso = durnpso / 10,
    # Treatment classes
    trtclass = case_when(trtn == 1 ~ "Placebo",
                         trtn %in% c(2, 3, 5, 6) ~ "IL blocker",
                         trtn == 4 ~ "TNFa blocker"),
    # Check complete cases for covariates of interest
    complete = complete.cases(durnpso, prevsys, bsa, weight, psa)
  )

pso_agd <- pso_agd %>%
  mutate(
    # Variable transformations
    bsa_mean = bsa_mean / 100,
    bsa_sd = bsa_sd / 100,
    prevsys = prevsys / 100,
    psa = psa / 100,
    weight_mean = weight_mean / 10,
    weight_sd = weight_sd / 10,
    durnpso_mean = durnpso_mean / 10,
    durnpso_sd = durnpso_sd / 10,
    # Treatment classes
    trtclass = case_when(trtn == 1 ~ "Placebo",
                         trtn %in% c(2, 3, 5, 6) ~ "IL blocker",
                         trtn == 4 ~ "TNFa blocker")
  )

# Exclude small number of individuals with missing covariates
pso_ipd <- filter(pso_ipd, complete)

pso_net <- combine_network(
  set_ipd(pso_ipd,
          study = studyc,
          trt = trtc,
          r = pasi75,
          trt_class = trtclass),
  set_agd_arm(pso_agd,
              study = studyc,
              trt = trtc,
              r = pasi75_r,
              n = pasi75_n,
              trt_class = trtclass)
)

# Print network details
pso_net


# Plot network
plot(pso_net, weight_nodes = TRUE, weight_edges = TRUE, show_trt_class = TRUE)
```

dgent *Generalised Student's t distribution (with location and scale)*

Description

Density, distribution, and quantile function for the generalised t distribution with degrees of freedom `df`, shifted by `location` and scaled by `scale`.

Usage

```
dgent(x, df, location = 0, scale = 1)

pgent(q, df, location = 0, scale = 1)

qgent(p, df, location = 0, scale = 1)
```

Arguments

`x, q`	Vector of quantiles
`df`	Degrees of freedom, greater than zero
`location`	Location parameter
`scale`	Scale parameter, greater than zero
`p`	Vector of probabilities

Value

`dgent()` gives the density, `pgent()` gives the distribution function, `qgent()` gives the quantile function.

`diabetes` *Incidence of diabetes in trials of antihypertensive drugs*

Description

Data frame containing the number of new cases of diabetes in 22 trials of 6 antihypertensive drugs (Elliott and Meyer 2007; Dias et al. 2011). The trial duration (in years) is also recorded.

Usage

```
diabetes
```

Format

A data frame with 48 rows and 7 variables:

studyn numeric study ID

studyc study name

trtn numeric treatment code

trtc treatment name

r total number of events

n total number of individuals

time trial follow-up (years)

References

Dias S, Welton NJ, Sutton AJ, Ades AE (2011). "NICE DSU Technical Support Document 2: A generalised linear modelling framework for pair-wise and network meta-analysis of randomised controlled trials." National Institute for Health and Care Excellence. `https://www.sheffield.ac.uk/nice-dsu`.

Elliott WJ, Meyer PM (2007). "Incident diabetes in clinical trials of antihypertensive drugs: a network meta-analysis." *The Lancet*, **369**(9557), 201–207. doi:10.1016/s01406736(07)601081.

`dic` *Deviance Information Criterion (DIC)*

Description

Calculate the DIC for a model fitted using the `nma()` function.

Usage

```
dic(x, penalty = c("pD", "pV"), ...)
```

Arguments

`x`	A fitted model object, inheriting class stan_nma
`penalty`	The method for estimating the effective number of parameters, used to penalise model fit in the DIC. Either `"pD"` (the default), or `"pV"`. For survival likelihoods only `"pV"` is currently available.
`...`	Other arguments (not used)

Value

A nma_dic object.

See Also

`print.nma_dic()` for printing details, `plot.nma_dic()` for producing plots of residual deviance contributions.

Examples

```
## Smoking cessation

# Run smoking FE NMA example if not already available
if (!exists("smk_fit_FE")) example("example_smk_fe", run.donttest = TRUE)


# Run smoking RE NMA example if not already available
if (!exists("smk_fit_RE")) example("example_smk_re", run.donttest = TRUE)
```

```
# Compare DIC of FE and RE models
(smk_dic_FE <- dic(smk_fit_FE))
(smk_dic_RE <- dic(smk_fit_RE))   # substantially better fit

# Plot residual deviance contributions under RE model
plot(smk_dic_RE)

# Check for inconsistency using UME model


# Run smoking UME NMA example if not already available
if (!exists("smk_fit_RE_UME")) example("example_smk_ume", run.donttest = TRUE)


# Compare DIC
smk_dic_RE
(smk_dic_RE_UME <- dic(smk_fit_RE_UME))  # no difference in fit

# Compare residual deviance contributions
plot(smk_dic_RE, smk_dic_RE_UME, show_uncertainty = FALSE)
```

`dietary_fat` *Reduced dietary fat to prevent mortality*

Description

Data frame containing the number of deaths and person-years at risk in 10 trials comparing reduced fat diets vs. control (non-reduced fat diet) for preventing mortality (Hooper et al. 2000; Dias et al. 2011).

Usage

```
dietary_fat
```

Format

A data frame with 21 rows and 7 variables:

studyn numeric study ID

studyc study name

trtn numeric treatment code

trtc treatment name

r number of events

n number randomised

E person-years at risk

References

Dias S, Welton NJ, Sutton AJ, Ades AE (2011). “NICE DSU Technical Support Document 2: A generalised linear modelling framework for pair-wise and network meta-analysis of randomised controlled trials.” National Institute for Health and Care Excellence. `https://www.sheffield.ac.uk/nice-dsu`.

Hooper L, Summerbell CD, Higgins JPT, Thompson RL, Clements G, Capps N, Davey Smith G, Riemersma R, Ebrahim S (2000). “Reduced or modified dietary fat for preventing cardiovascular disease.” *Cochrane Database of Systematic Reviews*. ISSN 1465-1858, doi:10.1002/14651858.CD002137.

`distr` *Specify a general marginal distribution*

Description

`distr()` is used within the function `add_integration()` to specify marginal distributions for the covariates, via a corresponding inverse CDF. It is also used in `predict.stan_nma()` to specify a distribution for the baseline response (intercept) when predicting absolute outcomes.

Usage

```
distr(qfun, ...)
```

Arguments

`qfun`	an inverse CDF, either as a function name or a string
`...`	parameters of the distribution as arguments to `qfun`, these will be quoted and evaluated later in the context of the aggregate data sources

Details

The function `qfun` should have a formal argument called `p`. This restriction serves as a crude check for inverse CDFs (e.g. an error will be given if `dnorm` is used instead of `qnorm`). If a user-written CDF is supplied, it must have an argument `p` which takes a vector of probabilities.

Value

An object of class distr.

See Also

`add_integration()` where `distr()` is used to specify marginal distributions for covariates to integrate over, and `predict.stan_nma()` where `distr()` is used to specify a distribution on the baseline response.

Examples

```
## Specifying marginal distributions for integration

df <- data.frame(x1_mean = 2, x1_sd = 0.5, x2 = 0.8)

# Distribution parameters are evaluated in the context of the data frame
add_integration(df,
                x1 = distr(qnorm, mean = x1_mean, sd = x1_sd),
                x2 = distr(qbern, prob = x2),
                cor = diag(2))
```

`dlogt` *Log Student's t distribution*

Description

Density, distribution, and quantile function for the log t distribution, whose logarithm has degrees of freedom `df`, mean `location`, and standard deviation `scale`.

Usage

```
dlogt(x, df, location = 0, scale = 1)

plogt(q, df, location = 0, scale = 1)

qlogt(p, df, location = 0, scale = 1)
```

Arguments

`x, q`	Vector of quantiles
`df`	Degrees of freedom, greater than zero
`location`	Location parameter
`scale`	Scale parameter, greater than zero
`p`	Vector of probabilities

Details

If $\log(Y) \sim t_\nu(\mu, \sigma^2)$, then Y has a log t distribution with `location` μ, `scale` σ, and `df` ν.

The mean and all higher moments of the log t distribution are undefined or infinite.

If `df = 1` then the distribution is a log Cauchy distribution. As `df` tends to infinity, this approaches a log Normal distribution.

Value

`dlogt()` gives the density, `plogt()` gives the distribution function, `qlogt()` gives the quantile function.

`dmspline` *Distribution functions for M-spline baseline hazards*

Description

Density, distribution, quantile, hazard, cumulative hazard, and restricted mean survival time functions for the M-spline baseline hazards model.

Usage

```
dmspline(x, basis, scoef, rate, log = FALSE)

pmspline(q, basis, scoef, rate, lower.tail = TRUE, log.p = FALSE)

qmspline(p, basis, scoef, rate, lower.tail = TRUE, log.p = FALSE)

hmspline(x, basis, scoef, rate, log = FALSE)

Hmspline(x, basis, scoef, rate, log = FALSE)

rmst_mspline(t, basis, scoef, rate, start = 0)
```

Arguments

`x, q`	Vector of quantiles
`basis`	M-spline basis produced by `splines2::mSpline()`
`scoef`	Vector (or matrix) of spline coefficients with length (or number of columns) equal to the dimension of `basis`
`rate`	Vector of rate parameters
`log, log.p`	Logical; if `TRUE`, probabilities `p` are given as $\log(p)$
`lower.tail`	Logical; if `TRUE` (the default), probabilities are $P(X \le x)$, otherwise $P(X > x)$
`p`	Vector of probabilities
`t`	Vector of times to which the restricted mean survival time is calculated
`start`	Optional left-truncation time or times. The returned restricted mean survival will be conditioned on survival up to this time

Details

Survival models with a flexible M-spline on the baseline hazard are described by Brilleman et al. (2020). Piecewise-exponential baseline hazards are a special case where the degree of the M-spline polynomial is 0.

The d/p/h/H functions are calculated from their definitions. `qmspline()` uses numerical inversion via `flexsurv::qgeneric()`. `rmst_mspline()`uses numerical integration via `flexsurv::rmst_generic()`, except for the special case of the piecewise-exponential hazard (i.e. degree 0 M-splines) which uses the explicit formula from Royston and Parmar (2013).

Beyond the boundary knots, the hazard is assumed to be constant. (This differs from the approach in `splines2::mSpline()` that extrapolates the polynomial basis functions, which is numerically unstable and highly dependent on the data just before the boundary knots.) As with all extrapolation, care should be taken when evaluating the splines at times beyond the boundary knots (either directly through the d/p/h/H/rmst functions, or indirectly by requesting quantiles with `qmspline()` that correspond to times beyond the boundary knots). For this reason evaluating the (unrestricted) mean survival time is not generally recommended as this requires integrating over an infinite time horizon (i.e. `rmst_mspline()` with `t = Inf`).

Value

`dmspline()` gives the density, `pmspline()` gives the distribution function (CDF), `qmspline()` gives the quantile function (inverse-CDF), `hmspline()` gives the hazard function, `Hmspline()` gives the cumulative hazard function, and `rmst_mspline()` gives restricted mean survival times.

References

Brilleman SL, Elci EM, Novik JB, Wolfe R (2020). “Bayesian Survival Analysis Using the rstanarm R Package.” *arXiv*. doi:10.48550/arXiv.2002.09633, 2002.09633.

Royston P, Parmar MKB (2013). “Restricted mean survival time: an alternative to the hazard ratio for the design and analysis of randomized trials with a time-to-event outcome.” *BMC Medical Research Methodology*, **13**(1). doi:10.1186/1471228813152.

`example_ndmm` *Example newly-diagnosed multiple myeloma*

Description

Calling `example("example_ndmm")` will run a proportional hazards Weibull NMA model on the newly-diagnosed multiple myeloma data, using the code in the Examples section below. The resulting `stan_nma` object `ndmm_fit` will then be available in the global environment.

Examples

```
# Set up newly-diagnosed multiple myeloma network

head(ndmm_ipd)
head(ndmm_agd)

ndmm_net <- combine_network(
  set_ipd(ndmm_ipd,
          study, trt,
          Surv = Surv(eventtime / 12, status)),
  set_agd_surv(ndmm_agd,
               study, trt,
               Surv = Surv(eventtime / 12, status),
               covariates = ndmm_agd_covs))
```

```
# Fit Weibull (PH) model
ndmm_fit <- nma(ndmm_net,
                likelihood = "weibull",
                prior_intercept = normal(scale = 100),
                prior_trt = normal(scale = 10),
                prior_aux = half_normal(scale = 10))

ndmm_fit
```

example_pso_mlnmr *Example plaque psoriasis ML-NMR*

Description

Calling `example("example_pso_mlnmr")` will run a ML-NMR model with the plaque psoriasis IPD and AgD, using the code in the Examples section below. The resulting `stan_nma` object `pso_fit` will then be available in the global environment.

Details

Plaque psoriasis ML-NMR for use in examples.

Examples

```
# Set up plaque psoriasis network combining IPD and AgD
library(dplyr)
pso_ipd <- filter(plaque_psoriasis_ipd,
                  studyc %in% c("UNCOVER-1", "UNCOVER-2", "UNCOVER-3"))

pso_agd <- filter(plaque_psoriasis_agd,
                  studyc == "FIXTURE")

head(pso_ipd)
head(pso_agd)

pso_ipd <- pso_ipd %>%
  mutate(# Variable transformations
    bsa = bsa / 100,
    prevsys = as.numeric(prevsys),
    psa = as.numeric(psa),
    weight = weight / 10,
    durnpso = durnpso / 10,
    # Treatment classes
    trtclass = case_when(trtn == 1 ~ "Placebo",
                         trtn %in% c(2, 3, 5, 6) ~ "IL blocker",
                         trtn == 4 ~ "TNFa blocker"),
    # Check complete cases for covariates of interest
    complete = complete.cases(durnpso, prevsys, bsa, weight, psa)
  )

pso_agd <- pso_agd %>%
  mutate(
    # Variable transformations
    bsa_mean = bsa_mean / 100,
    bsa_sd = bsa_sd / 100,
    prevsys = prevsys / 100,
    psa = psa / 100,
    weight_mean = weight_mean / 10,
    weight_sd = weight_sd / 10,
    durnpso_mean = durnpso_mean / 10,
    durnpso_sd = durnpso_sd / 10,
    # Treatment classes
    trtclass = case_when(trtn == 1 ~ "Placebo",
                         trtn %in% c(2, 3, 5, 6) ~ "IL blocker",
                         trtn == 4 ~ "TNFa blocker")
  )

# Exclude small number of individuals with missing covariates
pso_ipd <- filter(pso_ipd, complete)

pso_net <- combine_network(
  set_ipd(pso_ipd,
          study = studyc,
          trt = trtc,
          r = pasi75,
          trt_class = trtclass),
  set_agd_arm(pso_agd,
              study = studyc,
              trt = trtc,
              r = pasi75_r,
              n = pasi75_n,
              trt_class = trtclass)
)

# Print network details
pso_net

# Add integration points to the network
pso_net <- add_integration(pso_net,
  durnpso = distr(qgamma, mean = durnpso_mean, sd = durnpso_sd),
  prevsys = distr(qbern, prob = prevsys),
  bsa = distr(qlogitnorm, mean = bsa_mean, sd = bsa_sd),
  weight = distr(qgamma, mean = weight_mean, sd = weight_sd),
  psa = distr(qbern, prob = psa),
  n_int = 64)


# Fitting a ML-NMR model.
# Specify a regression model to include effect modifier interactions for five
# covariates, along with main (prognostic) effects. We use a probit link and
# specify that the two-parameter Binomial approximation for the aggregate-level
# likelihood should be used. We set treatment-covariate interactions to be equal
# within each class. We narrow the possible range for random initial values with
# init_r = 0.1, since probit models in particular are often hard to initialise.
# Using the QR decomposition greatly improves sampling efficiency here, as is
# often the case for regression models.
pso_fit <- nma(pso_net,
               trt_effects = "fixed",
               link = "probit",
               likelihood = "bernoulli2",
               regression = ~(durnpso + prevsys + bsa + weight + psa)*.trt,
               class_interactions = "common",
               prior_intercept = normal(scale = 10),
               prior_trt = normal(scale = 10),
               prior_reg = normal(scale = 10),
               init_r = 0.1,
               QR = TRUE)
pso_fit
```

---

example_smk_fe *Example smoking FE NMA*

Description

Calling `example("example_smk_fe")` will run a fixed effects NMA model with the smoking cessation data, using the code in the Examples section below. The resulting `stan_nma` object `smk_fit_FE` will then be available in the global environment.

Details

Smoking FE NMA for use in examples.

Examples

```
# Set up network of smoking cessation data
head(smoking)

smk_net <- set_agd_arm(smoking,
                       study = studyn,
                       trt = trtc,
                       r = r,
                       n = n,
                       trt_ref = "No intervention")

# Print details
smk_net
```

```
# Fitting a fixed effect model
smk_fit_FE <- nma(smk_net,
                  trt_effects = "fixed",
                  prior_intercept = normal(scale = 100),
                  prior_trt = normal(scale = 100))

smk_fit_FE
```

example_smk_nodesplit *Example smoking node-splitting*

Description

Calling `example("example_smk_nodesplit")` will run node-splitting models with the smoking cessation data, using the code in the Examples section below. The resulting `nma_nodesplit_df` object `smk_fit_RE_nodesplit` will then be available in the global environment.

Details

Smoking node-splitting for use in examples.

Examples

```
# Set up network of smoking cessation data
head(smoking)

smk_net <- set_agd_arm(smoking,
                       study = studyn,
                       trt = trtc,
                       r = r,
                       n = n,
                       trt_ref = "No intervention")

# Print details
smk_net


# Fitting all possible node-splitting models
smk_fit_RE_nodesplit <- nma(smk_net,
                            consistency = "nodesplit",
                            trt_effects = "random",
                            prior_intercept = normal(scale = 100),
                            prior_trt = normal(scale = 100),
                            prior_het = normal(scale = 5))
```

example_smk_re *Example smoking RE NMA*

Description

Calling `example("example_smk_re")` will run a random effects NMA model with the smoking cessation data, using the code in the Examples section below. The resulting `stan_nma` object `smk_fit_RE` will then be available in the global environment.

Details

Smoking RE NMA for use in examples.

Examples

```
# Set up network of smoking cessation data
head(smoking)

smk_net <- set_agd_arm(smoking,
                       study = studyn,
                       trt = trtc,
                       r = r,
                       n = n,
                       trt_ref = "No intervention")

# Print details
smk_net


# Fitting a random effects model
smk_fit_RE <- nma(smk_net,
                  trt_effects = "random",
                  prior_intercept = normal(scale = 100),
                  prior_trt = normal(scale = 100),
                  prior_het = normal(scale = 5))

smk_fit_RE
```

example_smk_ume *Example smoking UME NMA*

Description

Calling `example("example_smk_ume")` will run an unrelated mean effects (inconsistency) NMA model with the smoking cessation data, using the code in the Examples section below. The resulting `stan_nma` object `smk_fit_RE_UME` will then be available in the global environment.

Details

Smoking UME NMA for use in examples.

Examples

```
# Set up network of smoking cessation data
head(smoking)

smk_net <- set_agd_arm(smoking,
                       study = studyn,
                       trt = trtc,
                       r = r,
                       n = n,
                       trt_ref = "No intervention")

# Print details
smk_net


# Fitting an unrelated mean effects (inconsistency) model
smk_fit_RE_UME <- nma(smk_net,
                      consistency = "ume",
                      trt_effects = "random",
                      prior_intercept = normal(scale = 100),
                      prior_trt = normal(scale = 100),
                      prior_het = normal(scale = 5))

smk_fit_RE_UME
```

`geom_km` *Kaplan-Meier curves of survival data*

Description

This helper function constructs a ggplot2 geom to plot Kaplan-Meier curves from a network containing survival or time-to-event outcomes. This is useful for overlaying the "raw" survival data on the estimated survival functions created with plotted with `plot.surv_nma_summary()`, but can also be used standalone to plot Kaplan-Meier curves before fitting a model.

Usage

```
geom_km(
  network,
  ...,
  transform = c("identity", "cloglog", "log", "cumhaz"),
  curve_args = list(),
  cens_args = list()
)
```

Arguments

`network`	A `nma_data` network object containing survival outcomes
`...`	Additional arguments passed to `survival::survfit()`
`transform`	Character string giving the transformation to apply to the KM curves before plotting. The default is `"identity"` for no transformation; other options are `"cloglog"` for $\log(-\log(S))$, `"log"` for $\log(S)$, or `"cumhaz"` for the cumulative hazard $-\log(S)$.
`curve_args`	Optional list of arguments to customise the curves plotted with `ggplot2::geom_step()`
`cens_args`	Optional list of arguments to customise the censoring marks plotted with `ggplot2::geom_point()`

Value

A ggplot2 geom list that can be added to a ggplot2 plot object

Examples

```
# Set up newly-diagnosed multiple myeloma network

head(ndmm_ipd)
head(ndmm_agd)

ndmm_net <- combine_network(
  set_ipd(ndmm_ipd,
          study, trt,
          Surv = Surv(eventtime / 12, status)),
  set_agd_surv(ndmm_agd,
               study, trt,
               Surv = Surv(eventtime / 12, status),
               covariates = ndmm_agd_covs))
# Plot KM curves using ggplot2
library(ggplot2)

# We need to create an empty ggplot object to add the curves to
ggplot() + geom_km(ndmm_net)

# Adding plotting options, facets, axis labels, and a plot theme
ggplot() +
  geom_km(ndmm_net,
          curve_args = list(linewidth = 0.5),
          cens_args = list(size = 3, shape = 124)) +
  facet_wrap(vars(Study)) +
  labs(xlab = "Time", ylab = "Survival Probability") +
  theme_multinma()

# Using the transform argument to produce log-log plots (e.g. to assess the
# proportional hazards assumption)
ggplot() +
  geom_km(ndmm_net, transform = "cloglog") +
  facet_wrap(vars(Study)) +
  theme_multinma()

# Using the transform argument to produce cumulative hazard plots
ggplot() +
  geom_km(ndmm_net, transform = "cumhaz") +
  facet_wrap(vars(Study)) +
  theme_multinma()

# This function can also be used to add KM data to plots of estimated survival
# curves from a fitted model, in a similar manner

# Run newly-diagnosed multiple myeloma example if not already available
if (!exists("ndmm_fit")) example("example_ndmm", run.donttest = TRUE)

# Plot estimated survival curves, and overlay the KM data

plot(predict(ndmm_fit, type = "survival")) + geom_km(ndmm_net)
```

get_nodesplits *Direct and indirect evidence*

Description

Determine whether two treatments in a network are connected by direct and/or indirect evidence, and generate a list of comparisons with both direct and indirect evidence (i.e. potential inconsistency) for node-splitting.

Usage

```
get_nodesplits(network, include_consistency = FALSE)

has_direct(network, trt1, trt2)

has_indirect(network, trt1, trt2)
```

Arguments

`network`	An `nma_data` object, as created by the functions `set_*()` or `combine_network()`.
`include_consistency`	Logical, whether to include a row of `NA`s to indicate that a consistency model (i.e. a model with no node-splitting) should also be fitted by the `nma()` function. Default is `FALSE` when calling `get_nodesplits()` by hand, and `nma()` sets this to `TRUE` by default.
`trt1, trt2`	Treatments, each as a single integer, string, or factor

Details

The list of comparisons for node-splitting is generated following the algorithm of van Valkenhoef et al. (2016). A comparison between two treatments has the potential for inconsistency, and is thus considered for node-splitting, if the comparison has both direct evidence and *independent* indirect evidence.

The notion of independent indirect evidence is necessary when multi-arm trials are present, since by design these trials are internally consistent. A comparison between two treatments has independent indirect evidence if, after removing all studies comparing the two treatments from the network, the two treatments are still connected by a path of evidence. This is the criterion considered by the `has_indirect()` function.

Value

For `has_direct()` and `has_indirect()`, a single logical value. For `get_nodesplits()`, a data frame with two columns giving the comparisons for node-splitting.

References

van Valkenhoef G, Dias S, Ades AE, Welton NJ (2016). "Automated generation of node-splitting models for assessment of inconsistency in network meta-analysis." *Research Synthesis Methods*, **7**(1), 80–93. doi:10.1002/jrsm.1167.

Examples

```
# Parkinsons example
park_net <- set_agd_arm(parkinsons,
                        study = studyn,
                        trt = trtn,
                        y = y,
                        se = se,
                        trt_ref = 1)

# View the network plot
plot(park_net)

# The 4 vs. 5 comparison is a spur on the network
has_direct(park_net, 4, 5)
has_indirect(park_net, 4, 5)

# 1 and 5 are not directly connected
has_direct(park_net, 1, 5)
has_indirect(park_net, 1, 5)

# The 1 vs. 2 comparison does not have independent indirect evidence, since
# the 1-2-4 loop is a multi-arm study
has_indirect(park_net, 1, 2)

# Get a list of comparisons with potential inconsistency for node-splitting
get_nodesplits(park_net)

# See van Valkenhoef (2016) for a discussion of this example
```

`hta_psoriasis` *HTA Plaque Psoriasis*

Description

Data frame containing the results of 16 trials comparing 8 treatments for moderate-to-severe plaque psoriasis from an HTA report (Woolacott et al. 2006), analysed in TSD2 (Dias et al. 2011). Outcomes are success/failure to achieve 50%, 75%, or 90% reduction in symptoms on the Psoriasis Area and Severity Index (PASI) scale. Some studies report all three ordered outcomes, others only one or two. The latter are coded as missing values (see details).

Usage

```
hta_psoriasis
```

Format

A data frame with 36 rows and 9 variables:

studyn numeric study ID

studyc study name

year year of publication

trtn numeric treatment code

trtc treatment name

sample_size sample size in each arm

PASI50, PASI75, PASI90 ordered multinomial outcome counts (exclusive, see details)

Details

Outcome counts are "exclusive"; that is, for a study reporting all outcomes, the counts represent the categories 50 < PASI < 75, 75 < PASI < 90, and 90 < PASI < 100, and are named by the lower end of the interval. (As opposed to "inclusive" counts, which would represent the overlapping categories PASI > 50, PASI > 70, and PASI > 90.) The count for the fourth category (the lowest), 0 < PASI < 50, is equal to `sample_size - PASI50 - PASI75 - PASI90`.

Missing values are used where studies only report a subset of the outcomes. For a study reporting only two outcomes, say 50 and 75, the counts represent 50 < PASI < 75 and 75 < PASI < 100. For a study reporting only one outcome, say PASI 75, the count represents 75 < PASI < 100.

References

Dias S, Welton NJ, Sutton AJ, Ades AE (2011). “NICE DSU Technical Support Document 2: A generalised linear modelling framework for pair-wise and network meta-analysis of randomised controlled trials.” National Institute for Health and Care Excellence. https://www.sheffield.ac.uk/nice-dsu.

Woolacott N, Hawkins N, Mason A, Kainth A, Khadjesari Z, Bravo Vergel Y, Misso K, Light K, Chalmers R, Sculpher M, Riemsma R (2006). "Etanercept and efalizumab for the treatment of psoriasis: a systematic review." *Health Technology Assessment*, **10**(46). doi:10.3310/hta10460.

`is_network_connected` *Check network connectedness*

Description

Check whether a network is connected - whether there is a path of study evidence linking every pair of treatments in the network.

Usage

```
is_network_connected(network)
```

Arguments

`network`	An `nma_data` object, as created by the functions `set_*()` or `combine_network()`.

Details

Models will still run with disconnected networks. However, estimated relative effects between treatments across disconnected parts of the network will be entirely based on the prior distribution (typically very uncertain), as there is no information to update the prior distribution. Relative effects within each connected sub-network will be estimated as if each sub-network had been analysed separately.

Value

Logical `TRUE` or `FALSE`

Examples

```
## Smoking cessation
# Set up network of smoking cessation data
head(smoking)

smk_net <- set_agd_arm(smoking,
                       study = studyn,
                       trt = trtc,
                       r = r,
                       n = n,
                       trt_ref = "No intervention")

# Print details
smk_net

is_network_connected(smk_net)  # TRUE, network is connected
```

```
## A disconnected network
disc_net <- set_agd_arm(smoking[smoking$studyn %in% c(15, 21), ],
                        study = studyn,
                        trt = trtc,
                        r = r,
                        n = n)
is_network_connected(disc_net)  # FALSE, network is disconnected
disc_net
plot(disc_net)
```

---

loo.stan_nma — *Model comparison using the `loo` package*

Description

The `loo()` and `waic()` functions from the `loo` package may be called directly on stan_nma and stan_mlnmr objects.

Usage

```
## S3 method for class 'stan_nma'
loo(x, ...)

## S3 method for class 'stan_nma'
waic(x, ...)
```

Arguments

`x`	An object of class stan_nma or stan_mlnmr
`...`	Further arguments to `loo()` or `waic()`

---

make_knots — *Knot locations for M-spline baseline hazard models*

Description

Several different algorithms are provided to calculate knot locations for M-spline baseline hazard models. This function is called internally within the `nma()` function, but may be called directly by the user for more control.

Usage

```
make_knots(
  network,
  n_knots = 7,
 type = c("quantile", "quantile_common", "quantile_lumped", "quantile_longest", "equal",
    "equal_common")
)
```

Arguments

`network`	A network object, containing survival outcomes
`n_knots`	Non-negative integer giving the number of internal knots (default 7)
`type`	String specifying the knot location algorithm to use (see details). The default used by `nma()` is `"quantile"`, except when a regression model is specified (using `aux_regression`) in which case the default is `"quantile_common"`.

Details

The `type` argument can be used to choose between different algorithms for placing the knots:

`"quantile"` Creates separate knot locations for each study, internal knots are placed at evenly-spaced quantiles of the observed event times within each study.

`"quantile_lumped"` Creates a common set of knots for all studies, calculated as evenly-spaced quantiles of the observed event times from all studies lumped together.

`"quantile_common"` Creates a common set of knots for all studies, taking quantiles of the quantiles of the observed event times within each study. This often seems to result in a more even knot spacing than `"quantile_lumped"`, particularly when follow-up is uneven across studies, and may handle differing behaviour in the baseline hazard across studies better than `"quantile_longest"`.

`"quantile_longest"` Creates a common set of knots for all studies, using evenly-spaced quantiles of the observed event times in the longest study.

`"equal"` Creates separate knot locations for each study, at evenly-spaced times between the boundary knots in each study.

`"equal_common"` Creates a common set of knots for all studies, at evenly-spaced times between the earliest entry time and last event/censoring time in the network.

Boundary knots are calculated as follows:

- For separate knot locations in each study, boundary knots are placed at the earliest entry time and last event/censoring time in each study.
- For a common set of knots across all studies, boundary knots are placed at the earliest entry time and last event/censoring time across all studies.

Models with regression on the spline coefficients (i.e. with `aux_regression` specified) require a common set of knots across all studies.

Provided that a sufficient number of knots are used, model fit should be largely unaffected by the knot locations. However, sampling difficulties can sometimes occur if knot placement is poor, for example if a knot is placed just before the last follow-up time in a study.

Value

A named list of vectors giving the knot locations in each study.

Examples

```
# Set up newly-diagnosed multiple myeloma network

head(ndmm_ipd)
head(ndmm_agd)

ndmm_net <- combine_network(
  set_ipd(ndmm_ipd,
          study, trt,
          Surv = Surv(eventtime / 12, status)),
  set_agd_surv(ndmm_agd,
               study, trt,
               Surv = Surv(eventtime / 12, status),
               covariates = ndmm_agd_covs))

# The default knot locations
make_knots(ndmm_net, type = "quantile")

# Increasing the number of knots
make_knots(ndmm_net, n_knots = 10)

# Comparing alternative knot positioning algorithms
# Visualise these with a quick function
plot_knots <- function(network, knots) {
  ggplot2::ggplot() +
    geom_km(network) +
    ggplot2::geom_vline(ggplot2::aes(xintercept = .data$knot),
                data = tidyr::pivot_longer(as.data.frame(knots), cols = dplyr::everything(),
                                                names_to = "Study", values_to = "knot"),
                        linetype = 2, colour = "grey60") +
    ggplot2::facet_wrap(~Study) +
    theme_multinma()
}

plot_knots(ndmm_net, make_knots(ndmm_net, type = "quantile"))
plot_knots(ndmm_net, make_knots(ndmm_net, type = "quantile_common"))
plot_knots(ndmm_net, make_knots(ndmm_net, type = "quantile_lumped"))
plot_knots(ndmm_net, make_knots(ndmm_net, type = "quantile_longest"))
plot_knots(ndmm_net, make_knots(ndmm_net, type = "equal"))
plot_knots(ndmm_net, make_knots(ndmm_net, type = "equal_common"))
```

`marginal_effects` *Marginal treatment effects*

Description

Generate population-average marginal treatment effects. These are formed from population-average absolute predictions, so this function is a wrapper around `predict.stan_nma()`.

Usage

```
marginal_effects(
  object,
  ...,
  mtype = c("difference", "ratio", "link"),
  all_contrasts = FALSE,
  trt_ref = NULL,
  probs = c(0.025, 0.25, 0.5, 0.75, 0.975),
  predictive_distribution = FALSE,
  summary = TRUE
)
```

Arguments

`object`	A `stan_nma` object created by `nma()`.
`...`	Arguments passed to `predict.stan_nma()`, for example to specify the covariate distribution and baseline risk for a target population, e.g. `newdata`, `baseline`, and related arguments. For survival outcomes, `type` can also be specified to determine the quantity from which to form a marginal effect. For example, `type = "hazard"` with `mtype = "ratio"` produces marginal hazard ratios, `type = "median"` with `mtype = "difference"` produces marginal median survival time differences, and so on.
`mtype`	The type of marginal effect to construct from the average absolute effects, either `"difference"` (the default) for a difference of absolute effects such as a risk difference, `"ratio"` for a ratio of absolute effects such as a risk ratio, or `"link"` for a difference on the scale of the link function used in fitting the model such as a marginal log odds ratio.
`all_contrasts`	Logical, generate estimates for all contrasts (`TRUE`), or just the "basic" contrasts against the network reference treatment (`FALSE`)? Default `FALSE`.
`trt_ref`	Reference treatment to construct relative effects against, if `all_contrasts = FALSE`. By default, relative effects will be against the network reference treatment. Coerced to character string.
`probs`	Numeric vector of quantiles of interest to present in computed summary, default `c(0.025, 0.25, 0.5, 0.75, 0.975)`
`predictive_distribution`	Logical, when a random effects model has been fitted, should the predictive distribution for marginal effects in a new study be returned? Default `FALSE`.
`summary`	Logical, calculate posterior summaries? Default `TRUE`.

Value

A nma_summary object if `summary = TRUE`, otherwise a list containing a 3D MCMC array of samples and (for regression models) a data frame of study information.

Examples

```
## Smoking cessation

# Run smoking RE NMA example if not already available
if (!exists("smk_fit_RE")) example("example_smk_re", run.donttest = TRUE)


# Marginal risk difference in each study population in the network
marginal_effects(smk_fit_RE, mtype = "difference")

# Since there are no covariates in the model, the marginal and conditional
# (log) odds ratios here coincide
marginal_effects(smk_fit_RE, mtype = "link")
relative_effects(smk_fit_RE)

# Marginal risk differences in a population with 67 observed events out of
# 566 individuals on No Intervention, corresponding to a Beta(67, 566 - 67)
# distribution on the baseline probability of response
(smk_rd_RE <- marginal_effects(smk_fit_RE,
                               baseline = distr(qbeta, 67, 566 - 67),
                               baseline_type = "response",
                               mtype = "difference"))
plot(smk_rd_RE)


## Plaque psoriasis ML-NMR

# Run plaque psoriasis ML-NMR example if not already available
if (!exists("pso_fit")) example("example_pso_mlnmr", run.donttest = TRUE)


# Population-average marginal probit differences in each study in the network
(pso_marg <- marginal_effects(pso_fit, mtype = "link"))
plot(pso_marg, ref_line = c(0, 1))

# Population-average marginal probit differences in a new target population,
# with means and SDs or proportions given by
new_agd_int <- data.frame(
  bsa_mean = 0.6,
  bsa_sd = 0.3,
  prevsys = 0.1,
  psa = 0.2,
  weight_mean = 10,
  weight_sd = 1,
  durnpso_mean = 3,
  durnpso_sd = 1
)

# We need to add integration points to this data frame of new data
# We use the weighted mean correlation matrix computed from the IPD studies
new_agd_int <- add_integration(new_agd_int,
                           durnpso = distr(qgamma, mean = durnpso_mean, sd = durnpso_sd),
                              prevsys = distr(qbern, prob = prevsys),
                              bsa = distr(qlogitnorm, mean = bsa_mean, sd = bsa_sd),
                            weight = distr(qgamma, mean = weight_mean, sd = weight_sd),
                              psa = distr(qbern, prob = psa),
                              cor = pso_net$int_cor,
                              n_int = 64)

# Population-average marginal probit differences of achieving PASI 75 in this
# target population, given a Normal(-1.75, 0.08^2) distribution on the
# baseline probit-probability of response on Placebo (at the reference levels
# of the covariates), are given by
(pso_marg_new <- marginal_effects(pso_fit,
                                  mtype = "link",
                                  newdata = new_agd_int,
                                  baseline = distr(qnorm, -1.75, 0.08)))
plot(pso_marg_new)


## Progression free survival with newly-diagnosed multiple myeloma

# Run newly-diagnosed multiple myeloma example if not already available
if (!exists("ndmm_fit")) example("example_ndmm", run.donttest = TRUE)


# We can produce a range of marginal effects from models with survival
# outcomes, specified with the mtype and type arguments. For example:

# Marginal survival probability difference at 5 years, all contrasts
marginal_effects(ndmm_fit, type = "survival", mtype = "difference",
                 times = 5, all_contrasts = TRUE)

# Marginal difference in RMST up to 5 years
marginal_effects(ndmm_fit, type = "rmst", mtype = "difference", times = 5)

# Marginal median survival time ratios
marginal_effects(ndmm_fit, type = "median", mtype = "ratio")

# Marginal log hazard ratios
# With no covariates in the model, these are constant over time and study
# populations, and are equal to the log hazard ratios from relative_effects()
plot(marginal_effects(ndmm_fit, type = "hazard", mtype = "link"),
     # The hazard is infinite at t=0 in some studies, giving undefined logHRs at t=0
     na.rm = TRUE)

# The NDMM vignette demonstrates the production of time-varying marginal
# hazard ratios from a ML-NMR model that includes covariates, see
# `vignette("example_ndmm")`

# Marginal survival difference over time
plot(marginal_effects(ndmm_fit, type = "survival", mtype = "difference"))
```

mcmc_array-class *Working with 3D MCMC arrays*

Description

3D MCMC arrays (Iterations, Chains, Parameters) are produced by `as.array()` methods applied to `stan_nma` or `nma_summary` objects.

Usage

```
## S3 method for class 'mcmc_array'
summary(object, ..., probs = c(0.025, 0.25, 0.5, 0.75, 0.975))

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

## S3 method for class 'mcmc_array'
plot(x, ...)

## S3 method for class 'mcmc_array'
names(x)

## S3 replacement method for class 'mcmc_array'
names(x) <- value
```

Arguments

`...`	Further arguments passed to other methods
`probs`	Numeric vector of quantiles of interest
`x, object`	A 3D MCMC array of class `mcmc_array`
`value`	Character vector of replacement parameter names

Value

The `summary()` method returns a nma_summary object, the `print()` method returns `x` invisibly. The `names()` method returns a character vector of parameter names, and `names()<-` returns the object with updated parameter names. The `plot()` method is a shortcut for `plot(summary(x), ...)`, passing all arguments on to `plot.nma_summary()`.

Examples

```
## Smoking cessation

# Run smoking RE NMA example if not already available
if (!exists("smk_fit_RE")) example("example_smk_re", run.donttest = TRUE)
```

```
# Working with arrays of posterior draws (as mcmc_array objects) is
# convenient when transforming parameters

# Transforming log odds ratios to odds ratios
LOR_array <- as.array(relative_effects(smk_fit_RE))
OR_array <- exp(LOR_array)

# mcmc_array objects can be summarised to produce a nma_summary object
smk_OR_RE <- summary(OR_array)

# This can then be printed or plotted
smk_OR_RE
plot(smk_OR_RE, ref_line = 1)

# Transforming heterogeneity SD to variance
tau_array <- as.array(smk_fit_RE, pars = "tau")
tausq_array <- tau_array^2

# Correct parameter names
names(tausq_array) <- "tausq"

# Summarise
summary(tausq_array)
```

multi *Multinomial outcome data*

Description

This function aids the specification of multinomial outcome data when setting up a network with `set_agd_arm()` or `set_ipd()`. It takes a set of columns (or, more generally, numeric vectors of the same length) of outcome counts in each category, and binds these together to produce a matrix.

Usage

```
multi(..., inclusive = FALSE, type = c("ordered", "competing"))
```

Arguments

`...`	Two or more numeric columns (or vectors) of category counts. Argument names (optional) will be used to label the categories.
`inclusive`	Logical, are ordered category counts inclusive (`TRUE`) or exclusive (`FALSE`)? Default `FALSE`. Only used when `type = "ordered"`. See details.
`type`	String, indicating whether categories are `"ordered"` or `"competing"`. Currently only ordered categorical outcomes are supported by the modelling functions in this package.

Details

When specifying ordered categorical counts, these can either be given as *exclusive* counts (`inclusive = FALSE`, the default) where individuals are only counted in the highest category they achieve, or *inclusive* counts (`inclusive = TRUE`) where individuals are counted in every category up to and including the highest category achieved. (Competing outcomes, by nature, are always specified as exclusive counts.)

`NA` values can be used to indicate categories/cutpoints that were not measured.

Value

A matrix of (exclusive) category counts

Examples

```
# These two data sets specify the same ordered categorical data for outcomes
# r0 < r1 < r2, but the first uses the "inclusive" format and the second the
# "exclusive" format.
df_inclusive <- tibble::tribble(~r0, ~r1, ~r2,
                                1, 1, 1,
                                5, 4, 1,
                                5, 2, 2,
                                10, 5, 0,
                                5, 5, 0,
                                7, NA, 6,   # Achieved r2 or not (no r1)
                                10, 4, NA)  # Achieved r1 or not (no r2)

df_exclusive <- tibble::tribble(~r0, ~r1, ~r2,
                                0, 0, 1,
                                1, 3, 1,
                                3, 0, 2,
                                5, 5, 0,
                                0, 5, 0,
                                1, NA, 6,   # Achieved r2 or not (no r1)
                                6, 4, NA)   # Achieved r1 or not (no r2)

(r_inclusive <- with(df_inclusive, multi(r0, r1, r2, inclusive = TRUE)))
(r_exclusive <- with(df_exclusive, multi(r0, r1, r2, inclusive = FALSE)))

# Counts are always stored in exclusive format
stopifnot(isTRUE(all.equal(r_inclusive, r_exclusive)))


## HTA Plaque Psoriasis
library(dplyr)

# Ordered outcomes here are given as "exclusive" counts
head(hta_psoriasis)

# Calculate lowest category count (failure to achieve PASI 50)
pso_dat <- hta_psoriasis %>%
  mutate(`PASI<50` = sample_size - rowSums(cbind(PASI50, PASI75, PASI90), na.rm = TRUE))
```

```
# Set up network
pso_net <- set_agd_arm(pso_dat,
                       study = paste(studyc, year),
                       trt = trtc,
                       r = multi(`PASI<50`, PASI50, PASI75, PASI90,
                                 inclusive = FALSE,
                                 type = "ordered"))

pso_net
```

---

ndmm_ipd *Newly diagnosed multiple myeloma*

Description

Three data frames, ndmm_ipd, ndmm_agd, and ndmm_agd_covs containing (simulated) individual patient data (IPD) from three studies and aggregate data (AgD) from two studies on newly diagnosed multiple myeloma. The outcome of interest is progression-free survival after autologous stem cell transplant. The IPD studies in ndmm_ipd provide event/censoring times and covariate values for each individual. The AgD studies provide reconstructed event/censoring times from digitized Kaplan-Meier curves in ndmm_agd and covariate summaries in ndmm_agd_covs, obtained from published trial reports. The data are constructed to resemble those used by Leahy and Walsh (2019).

Usage

```
ndmm_ipd

ndmm_agd

ndmm_agd_covs
```

Format

The individual patient data are contained in a data frame ndmm_ipd with 1325 rows, one per individual, and 10 variables:

study, studyf study name

trt, trtf treatment name

eventtime event/censoring time

status censoring indicator (0 = censored, 1 = event)

age age (years)

iss_stage3 ISS stage 3 (0 = no, 1 = yes)

response_cr_vgpr complete or very good partial response (0 = no, 1 = yes)

male male sex (0 = no, 1 = yes)

The reconstructed Kaplan-Meier data for the aggregate studies are contained in a data frame `ndmm_agd` with 2819 rows and 6 variables:

study, studyf study name

trt, trtf treatment name

eventtime event/censoring time

status censoring indicator (0 = censored, 1 = event)

The covariate summaries extracted from published reportes for the aggregate studies are contained in a data frame `ndmm_agd_covs` with 4 rows, one per study arm, and 15 columns:

study, studyf study name

trt, trtf treatment name

sample_size sample size in each arm

age_min, age_iqr_l, age_median, age_iqr_h, age_max, age_mean, age_sd summary statistics for age (years)

iss_stage3 proportion of participants with ISS stage 3

response_cr_vgpr proportion of participants with complete or very good partial response

male proportion of male participants

References

Leahy J, Walsh C (2019). "Assessing the impact of a matching-adjusted indirect comparison in a Bayesian network meta-analysis." *Research Synthesis Methods*, **10**(4), 546–568. doi:10.1002/jrsm.1372.

`nma` *Network meta-analysis models*

Description

The `nma` function fits network meta-analysis and (multilevel) network meta-regression models in Stan.

Usage

```
nma(
  network,
  consistency = c("consistency", "ume", "nodesplit"),
  trt_effects = c("fixed", "random"),
  regression = NULL,
  class_interactions = c("common", "exchangeable", "independent"),
  likelihood = NULL,
  link = NULL,
  ...,
  nodesplit = get_nodesplits(network, include_consistency = TRUE),
  prior_intercept = .default(normal(scale = 100)),
  prior_trt = .default(normal(scale = 10)),
  prior_het = .default(half_normal(scale = 5)),
  prior_het_type = c("sd", "var", "prec"),
  prior_reg = .default(normal(scale = 10)),
  prior_aux = .default(),
  prior_aux_reg = .default(),
  aux_by = NULL,
  aux_regression = NULL,
  QR = FALSE,
  center = TRUE,
  adapt_delta = NULL,
  int_thin = 0,
  int_check = TRUE,
  mspline_degree = 3,
  n_knots = 7,
  knots = NULL,
  mspline_basis = NULL
)
```

Arguments

`network`	An `nma_data` object, as created by the functions `set_*()`, `combine_network()`, or `add_integration()`
`consistency`	Character string specifying the type of (in)consistency model to fit, either `"consistency"`, `"ume"`, or `"nodesplit"`
`trt_effects`	Character string specifying either `"fixed"` or `"random"` effects
`regression`	A one-sided model formula, specifying the prognostic and effect-modifying terms for a regression model. Any references to treatment should use the `.trt` special variable, for example specifying effect modifier interactions as `variable:.trt` (see details).
`class_interactions`	Character string specifying whether effect modifier interactions are specified as `"common"`, `"exchangeable"`, or `"independent"`.
`likelihood`	Character string specifying a likelihood, if unspecified will be inferred from the data (see details)
`link`	Character string specifying a link function, if unspecified will default to the canonical link (see details)
`...`	Further arguments passed to `sampling()`, such as `iter`, `chains`, `cores`, etc.
`nodesplit`	For `consistency = "nodesplit"`, the comparison(s) to split in the node-splitting model(s). Either a length 2 vector giving the treatments in a single comparison, or a 2 column data frame listing multiple treatment comparisons to split in turn. By default, all possible comparisons will be chosen (see `get_nodesplits()`).
`prior_intercept`	Specification of prior distribution for the intercept

`prior_trt`	Specification of prior distribution for the treatment effects
`prior_het`	Specification of prior distribution for the heterogeneity (if `trt_effects = "random"`)
`prior_het_type`	Character string specifying whether the prior distribution `prior_het` is placed on the heterogeneity standard deviation τ (`"sd"`, the default), variance τ^2 (`"var"`), or precision $1/\tau^2$ (`"prec"`).
`prior_reg`	Specification of prior distribution for the regression coefficients (if `regression` formula specified)
`prior_aux`	Specification of prior distribution for the auxiliary parameter, if applicable (see details). For `likelihood = "gengamma"` this should be a list of prior distributions with elements `sigma` and `k`.
`prior_aux_reg`	Specification of prior distribution for the auxiliary regression parameters, if `aux_regression` is specified (see details).
`aux_by`	Vector of variable names listing the variables to stratify the auxiliary parameters by. Currently only used for survival models, see details. Cannot be used with `aux_regression`.
`aux_regression`	A one-sided model formula giving a regression model for the auxiliary parameters. Currently only used for survival models, see details. Cannot be used with `aux_by`.
`QR`	Logical scalar (default `FALSE`), whether to apply a QR decomposition to the model design matrix
`center`	Logical scalar (default `TRUE`), whether to center the (numeric) regression terms about the overall means
`adapt_delta`	See adapt_delta for details
`int_thin`	A single integer value, the thinning factor for returning cumulative estimates of integration error. Saving cumulative estimates is disabled by `int_thin = 0`, which is the default.
`int_check`	Logical, check sufficient accuracy of numerical integration by fitting half of the chains with `n_int/2`? When `TRUE`, `Rhat` and `n_eff` diagnostic warnings will be given if numerical integration has not sufficiently converged (suggesting increasing `n_int` in `add_integration()`). Default `TRUE`, but disabled (`FALSE`) when `int_thin > 0`.
`mspline_degree`	Non-negative integer giving the degree of the M-spline polynomial for `likelihood = "mspline"`. Piecewise exponential hazards (`likelihood = "pexp"`) are a special case with `mspline_degree = 0`.
`n_knots`	For `mspline` and `pexp` likelihoods, a non-negative integer giving the number of internal knots for partitioning the baseline hazard into intervals. The knot locations within each study will be determined by the corresponding quantiles of the observed event times, plus boundary knots at the earliest entry time (0 with no delayed entry) and the maximum event/censoring time. For example, with `n_knots = 3`, the internal knot locations will be at the 25%, 50%, and 75% quantiles of the observed event times. The default is `n_knots = 7`; overfitting is avoided by shrinking towards a constant hazard with a random walk prior (see details). If `aux_regression` is specified then a single set of knot locations will be calculated across all studies in the network. See `make_knots()` for more details on the knot positioning algorithms. Ignored when `knots` is specified.

`knots`	For `mspline` and `pexp` likelihoods, a named list of numeric vectors of knot locations (including boundary knots) for each of the studies in the network. Currently, each vector must have the same length (i.e. each study must use the same number of knots). Alternatively, a single numeric vector of knot locations can be provided which will be shared across all studies in the network. If unspecified (the default), the knots will be chosen based on `n_knots` as described above. If `aux_regression` is specified then `knots` should be a single numeric vector of knot locations which will be shared across all studies in the network. `make_knots()` can be used to help specify `knots` directly, or to investigate knot placement prior to model fitting.
`mspline_basis`	Instead of specifying `mspline_degree` and `n_knots` or `knots`, a named list of M-spline bases (one for each study) can be provided with `mspline_basis` which will be used directly. In this case, all other M-spline options will be ignored.

Details

When specifying a model formula in the `regression` argument, the usual formula syntax is available (as interpreted by `model.matrix()`). The only additional requirement here is that the special variable `.trt` should be used to refer to treatment. For example, effect modifier interactions should be specified as `variable:.trt`. Prognostic (main) effects and interactions can be included together compactly as `variable*.trt`, which expands to `variable + variable:.trt` (plus `.trt`, which is already in the NMA model).

For the advanced user, the additional specials `.study` and `.trtclass` are also available, and refer to studies and (if specified) treatment classes respectively. When node-splitting models are fitted (`consistency = "nodesplit"`) the special `.omega` is available, indicating the arms to which the node-splitting inconsistency factor is added.

See `?priors` for details on prior specification. Default prior distributions are available, but may not be appropriate for the particular setting and will raise a warning if used. No attempt is made to tailor these defaults to the data provided. Please consider appropriate prior distributions for the particular setting, accounting for the scales of outcomes and covariates, etc. The function `plot_prior_posterior()` may be useful in examining the influence of the chosen prior distributions on the posterior distributions, and the `summary()` method for `nma_prior` objects prints prior intervals.

Value

`nma()` returns a stan_nma object, except when `consistency = "nodesplit"` when a nma_nodesplit or nma_nodesplit_df object is returned. `nma.fit()` returns a stanfit object.

Likelihoods and link functions

Currently, the following likelihoods and link functions are supported for each data type:

Data type	**Likelihood**
Binary	`bernoulli`, `bernoulli2`
Count	`binomial`, `binomial2`
Rate	`poisson`
Continuous	`normal`

Ordered	`ordered`
Survival	`exponential`, `weibull`, `gompertz`, `exponential-aft`, `weibull-aft`, `lognormal`, `loglogistic`, `gamma`, `geng`

The `bernoulli2` and `binomial2` likelihoods correspond to a two-parameter Binomial likelihood for arm-based AgD, which more closely matches the underlying Poisson Binomial distribution for the summarised aggregate outcomes in a ML-NMR model than the typical (one parameter) Binomial distribution (see Phillippo et al. 2020).

When a `cloglog` link is used, including an offset for log follow-up time (i.e. `regression = ~offset(log(time))`) results in a model on the log hazard (see Dias et al. 2011).

For survival data, all accelerated failure time models (`exponential-aft`, `weibull-aft`, `lognormal`, `loglogistic`, `gamma`, `gengamma`) are parameterised so that the treatment effects and any regression parameters are log Survival Time Ratios (i.e. a coefficient of $\log(2)$ means that the treatment or covariate is associated with a doubling of expected survival time). These can be converted to log Acceleration Factors using the relation $\log(\mathrm{AF}) = -\log(\mathrm{STR})$ (or equivalently $\mathrm{AF} = 1/\mathrm{STR}$).

Further details on each likelihood and link function are given by Dias et al. (2011).

Auxiliary parameters

Auxiliary parameters are only present in the following models.

Normal likelihood with IPD:

When a Normal likelihood is fitted to IPD, the auxiliary parameters are the arm-level standard deviations σ_{jk} on treatment k in study j.

Ordered multinomial likelihood:

When fitting a model to M ordered outcomes, the auxiliary parameters are the latent cutoffs between each category, $c_0 < c_1 < \cdots < c_M$. Only c_2 to c_{M-1} are estimated; we fix $c_0 = -\infty$, $c_1 = 0$, and $c_M = \infty$. When specifying priors for these latent cutoffs, we choose to specify priors on the *differences* $c_{m+1} - c_m$. Stan automatically truncates any priors so that the ordering constraints are satisfied.

Survival (time-to-event) likelihoods:

All survival likelihoods except the `exponential` and `exponential-aft` likelihoods have auxiliary parameters. These are typically study-specific shape parameters $\gamma_j > 0$, except for the `lognormal` likelihood where the auxiliary parameters are study-specific standard deviations on the log scale $\sigma_j > 0$.

The `gengamma` likelihood has two sets of auxiliary parameters, study-specific scale parameters $\sigma_j > 0$ and shape parameters k_j, following the parameterisation of Lawless (1980), which permits a range of behaviours for the baseline hazard including increasing, decreasing, bathtub and arc-shaped hazards. This parameterisation is related to that discussed by Cox et al. (2007) and implemented in the `flexsurv` package with $Q = k^{-0.5}$. The parameterisation used here effectively bounds the shape parameter k away from numerical instabilities as $k \to \infty$ (i.e. away from $Q \to 0$, the log-Normal distribution) via the prior distribution. Implicitly, this parameterisation is restricted to $Q > 0$ and so certain survival distributions like the inverse-Gamma and inverse-Weibull are not part of the parameter space; however, $Q > 0$ still encompasses the other survival distributions implemented in this package.

For the `mspline` and `pexp` likelihoods, the auxiliary parameters are the spline coefficients for each study. These form a unit simplex (i.e. lie between 0 and 1, and sum to 1), and are given a random walk prior distribution. `prior_aux` specifies the hyperprior on the random walk standard deviation σ which controls the level of smoothing of the baseline hazard, with $\sigma = 0$ corresponding to a constant baseline hazard.

The auxiliary parameters can be stratified by additional factors through the `aux_by` argument. For example, to allow the shape of the baseline hazard to vary between treatment arms as well as studies, use `aux_by = c(".study", ".trt")`. (Technically, `.study` is always included in the stratification even if omitted from `aux_by`, but we choose here to make the stratification explicit.) This is a common way of relaxing the proportional hazards assumption. The default is equivalent to `aux_by = ".study"` which stratifies the auxiliary parameters by study, as described above.

A regression model may be specified on the auxiliary parameters using `aux_regression`. This is useful if we wish to model departures from non-proportionality, rather than allowing the baseline hazards to be completely independent using `aux_by`. This is necessary if absolute predictions (e.g. survival curves) are required in a population for unobserved combinations of covariates; for example, if `aux_by = .trt` then absolute predictions may only be produced for the observed treatment arms in each study population, whereas if `aux_regression = ~.trt` then absolute predictions can be produced for all treatments in any population. For `mspline` and `pexp` likelihoods, the regression coefficients are smoothed over time using a random walk prior to avoid overfitting: `prior_aux_reg` specifies the hyperprior for the random walk standard deviation. For other parametric likelihoods, `prior_aux_reg` specifies the prior for the auxiliary regression coefficients.

References

Cox C, Chu H, Schneider MF, Muñoz A (2007). "Parametric survival analysis and taxonomy of hazard functions for the generalized gamma distribution." *Statistics in Medicine*, **26**(23), 4352–4374. doi:10.1002/sim.2836.

Dias S, Welton NJ, Sutton AJ, Ades AE (2011). "NICE DSU Technical Support Document 2: A generalised linear modelling framework for pair-wise and network meta-analysis of randomised controlled trials." National Institute for Health and Care Excellence. `https://www.sheffield.ac.uk/nice-dsu`.

Lawless JF (1980). "Inference in the Generalized Gamma and Log Gamma Distributions." *Technometrics*, **22**(3), 409–419. doi:10.1080/00401706.1980.10486173.

Phillippo DM, Dias S, Ades AE, Belger M, Brnabic A, Schacht A, Saure D, Kadziola Z, Welton NJ (2020). "Multilevel Network Meta-Regression for population-adjusted treatment comparisons." *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, **183**(3), 1189–1210. doi:10.1111/rssa.12579.

Examples

```
## Smoking cessation NMA
# Set up network of smoking cessation data
head(smoking)

smk_net <- set_agd_arm(smoking,
                       study = studyn,
                       trt = trtc,
                       r = r,
                       n = n,
                       trt_ref = "No intervention")

# Print details
smk_net


# Fitting a fixed effect model
smk_fit_FE <- nma(smk_net,
                  trt_effects = "fixed",
                  prior_intercept = normal(scale = 100),
                  prior_trt = normal(scale = 100))

smk_fit_FE



# Fitting a random effects model
smk_fit_RE <- nma(smk_net,
                  trt_effects = "random",
                  prior_intercept = normal(scale = 100),
                  prior_trt = normal(scale = 100),
                  prior_het = normal(scale = 5))

smk_fit_RE



# Fitting an unrelated mean effects (inconsistency) model
smk_fit_RE_UME <- nma(smk_net,
                      consistency = "ume",
                      trt_effects = "random",
                      prior_intercept = normal(scale = 100),
                      prior_trt = normal(scale = 100),
                      prior_het = normal(scale = 5))

smk_fit_RE_UME



# Fitting all possible node-splitting models
smk_fit_RE_nodesplit <- nma(smk_net,
                            consistency = "nodesplit",
                            trt_effects = "random",
                            prior_intercept = normal(scale = 100),
                            prior_trt = normal(scale = 100),
                            prior_het = normal(scale = 5))



# Summarise the node-splitting results
```

```
summary(smk_fit_RE_nodesplit)


## Plaque psoriasis ML-NMR
# Set up plaque psoriasis network combining IPD and AgD
library(dplyr)
pso_ipd <- filter(plaque_psoriasis_ipd,
                  studyc %in% c("UNCOVER-1", "UNCOVER-2", "UNCOVER-3"))

pso_agd <- filter(plaque_psoriasis_agd,
                  studyc == "FIXTURE")

head(pso_ipd)
head(pso_agd)

pso_ipd <- pso_ipd %>%
  mutate(# Variable transformations
    bsa = bsa / 100,
    prevsys = as.numeric(prevsys),
    psa = as.numeric(psa),
    weight = weight / 10,
    durnpso = durnpso / 10,
    # Treatment classes
    trtclass = case_when(trtn == 1 ~ "Placebo",
                         trtn %in% c(2, 3, 5, 6) ~ "IL blocker",
                         trtn == 4 ~ "TNFa blocker"),
    # Check complete cases for covariates of interest
    complete = complete.cases(durnpso, prevsys, bsa, weight, psa)
  )

pso_agd <- pso_agd %>%
  mutate(
    # Variable transformations
    bsa_mean = bsa_mean / 100,
    bsa_sd = bsa_sd / 100,
    prevsys = prevsys / 100,
    psa = psa / 100,
    weight_mean = weight_mean / 10,
    weight_sd = weight_sd / 10,
    durnpso_mean = durnpso_mean / 10,
    durnpso_sd = durnpso_sd / 10,
    # Treatment classes
    trtclass = case_when(trtn == 1 ~ "Placebo",
                         trtn %in% c(2, 3, 5, 6) ~ "IL blocker",
                         trtn == 4 ~ "TNFa blocker")
  )

# Exclude small number of individuals with missing covariates
pso_ipd <- filter(pso_ipd, complete)

pso_net <- combine_network(
  set_ipd(pso_ipd,
          study = studyc,
          trt = trtc,
          r = pasi75,
          trt_class = trtclass),
  set_agd_arm(pso_agd,
              study = studyc,
              trt = trtc,
              r = pasi75_r,
              n = pasi75_n,
              trt_class = trtclass)
)

# Print network details
pso_net

# Add integration points to the network
pso_net <- add_integration(pso_net,
  durnpso = distr(qgamma, mean = durnpso_mean, sd = durnpso_sd),
  prevsys = distr(qbern, prob = prevsys),
  bsa = distr(qlogitnorm, mean = bsa_mean, sd = bsa_sd),
  weight = distr(qgamma, mean = weight_mean, sd = weight_sd),
  psa = distr(qbern, prob = psa),
  n_int = 64)


# Fitting a ML-NMR model.
# Specify a regression model to include effect modifier interactions for five
# covariates, along with main (prognostic) effects. We use a probit link and
# specify that the two-parameter Binomial approximation for the aggregate-level
# likelihood should be used. We set treatment-covariate interactions to be equal
# within each class. We narrow the possible range for random initial values with
# init_r = 0.1, since probit models in particular are often hard to initialise.
# Using the QR decomposition greatly improves sampling efficiency here, as is
# often the case for regression models.
pso_fit <- nma(pso_net,
               trt_effects = "fixed",
               link = "probit",
               likelihood = "bernoulli2",
               regression = ~(durnpso + prevsys + bsa + weight + psa)*.trt,
               class_interactions = "common",
               prior_intercept = normal(scale = 10),
               prior_trt = normal(scale = 10),
               prior_reg = normal(scale = 10),
               init_r = 0.1,
               QR = TRUE)
pso_fit


## Newly-diagnosed multiple myeloma NMA
# Set up newly-diagnosed multiple myeloma network

head(ndmm_ipd)
head(ndmm_agd)
```

```
ndmm_net <- combine_network(
  set_ipd(ndmm_ipd,
          study, trt,
          Surv = Surv(eventtime / 12, status)),
  set_agd_surv(ndmm_agd,
               study, trt,
               Surv = Surv(eventtime / 12, status),
               covariates = ndmm_agd_covs))

# Fit Weibull (PH) model
ndmm_fit <- nma(ndmm_net,
                likelihood = "weibull",
                prior_intercept = normal(scale = 100),
                prior_trt = normal(scale = 10),
                prior_aux = half_normal(scale = 10))

ndmm_fit
```

---

nma_data-class   *The nma_data class*

Description

The nma_data class contains the data for a NMA in a standard format, created using the functions set_ipd(), set_agd_arm(), set_agd_contrast(), set_agd_surv(), or combine_network(). The sub-class mlnmr_data is created by the function add_integration(), and further contains numerical integration points for the aggregate data.

Details

Objects of class nma_data have the following components:

agd_arm data from studies with aggregate data (arm format)

agd_contrast data from studies with aggregate data (contrast format)

ipd data from studies with individual patient data

treatments treatment coding factor for entire network

classes treatment class coding factor (same length as treatments for entire network)

studies study coding factor for entire network

outcome outcome type for each data source, named list

The agd_arm, agd_contrast, and ipd components are tibbles with the following columns:

.study study (as factor)

.trt treatment (as factor)

.trtclass treatment class (as factor), if specified

.y continuous outcome

`.se` standard error (continuous)

`.r` event count (discrete)

`.n` event count denominator (discrete, `agd_arm` only)

`.E` time at risk (discrete)

`.Surv` survival outcome of type `Surv` (time-to-event), nested by study arm

`.sample_size` sample size (`agd_*` only)

`...` other columns (typically covariates) from the original data frame

Objects of class `mlnmr_data` additionally have components:

`n_int` number of numerical integration points

`int_names` names of covariates with numerical integration points

`int_cor` correlation matrix for covariates used to generate numerical integration points

The `agd_arm` and `agd_contrast` tibbles have additional list columns with prefix `.int_`, one for each covariate, which contain the numerical integration points nested as length-`n_int` vectors within each row.

See Also

`print.nma_data()` for the print method displaying details of the network, and `plot.nma_data()` for network plots.

`nma_dic-class` *The nma_dic class*

Description

The `nma_dic` class contains details of the Deviance Information Criterion (DIC), produced using the `dic()` function.

Details

Objects of class `nma_dic` have the following components:

`dic` The DIC value

`pd`, `pv` The effective number of parameters

`resdev` The total residual deviance

`pointwise` A list of data frames containing the pointwise contributions for the IPD and AgD.

`resdev_array` A 3D MCMC array [Iterations, Chains, Parameters] of posterior residual deviance samples.

See Also

`dic()`, `print.nma_dic()`, `plot.nma_dic()`.

nma_nodesplit-class *The nma_nodesplit class*

Description

The `nma_nodesplit` and `nma_nodesplit_df` classes contains the results from running a node-splitting model with the function `nma()`.

Details

Objects of class `nma_nodesplit` inherit from the stan_nma class, and contain the results of fitting a single node-split model. They have one additional component, `nodesplit`, which gives the comparison that was node-split as a length 2 vector.

Objects of class `nma_nodesplit_df` are tibble data frames with one row for each node-split comparison and columns:

`trt1`, `trt2` Treatments forming the comparison

`model` A list column containing the results of each model as a `nma_nodesplit` object

Optionally, there will be an additional row for the consistency model if this was fitted (e.g. by `get_nodesplits(., include_consistency = TRUE)`) with `trt1` and `trt2` both `NA`.

nma_prior-class *The nma_prior class*

Description

The `nma_prior` class is used to specify prior distributions.

Details

Objects of class `nma_prior` have the following components:

`dist` Distribution name

`fun` Name of constructor function, as string (e.g. `"normal"`)

`...` Parameters of the distribution

The distribution parameters, specified as named components in `...`, match those in the constructor functions (see priors).

`nma_summary-class` *The* `nma_summary` *class*

Description

The `nma_summary` class contains posterior summary statistics of model parameters or other quantities of interest, and the draws used to obtain these statistics.

Details

Objects of class `nma_summary` have the following components:

summary A data frame containing the computed summary statistics. Column `.trt` indicates the corresponding treatment, or columns `.trta` and `.trtb` indicate the corresponding contrast (`.trtb` vs. `.trta`). If a regression model was fitted with effect modifier interactions with treatment, these summaries will be study-specific. In this case, the corresponding study population is indicated in the `.study` column. If a multinomial model was fitted, the `.category` column indicates the corresponding category.

sims A 3D array [Iteration, Chain, Parameter] of MCMC simulations

studies (Optional) A data frame containing study information, printed along with the corresponding summary statistics if `summary` contains a `.study` column. Should have a matching `.study` column.

The following attributes may also be set:

xlab Label for x axis in plots, usually either `"Treatment"` or `"Contrast"`.

ylab Label for y axis in plots, usually used for the scale e.g. `"log Odds Ratio"`.

The subclass `nma_rank_probs` is used by the function `posterior_rank_probs()`, and contains posterior rank probabilities. This subclass does not have a `sims` component, as the rank probabilities are themselves posterior summaries of the ranks (i.e. they do not have a posterior distribution). The posterior ranks from which the rank probabilities are calculated may be obtained from `posterior_ranks()`.

`nodesplit_summary-class` *The* `nodesplit_summary` *class*

Description

The `nodesplit_summary` class contains posterior summary statistics for node-splitting models, as a result of calling `summary()` on a `nma_nodesplit` or `nma_nodesplit_df` object.

Details

Objects of class `nodesplit_summary` are tibble data frames, with one row for each node-split comparison and columns:

`trt1`, `trt2` Treatments forming the comparison

`summary` A list column containing nma_summary objects with the posterior summaries and draws for each of the node-splitting parameters

`p_value` Bayesian p-value for inconsistency

`dic` A list column containing nma_dic objects, giving the model fit statistics

The parameters included in `summary` are:

`d_net` Network estimate from the corresponding consistency model, if available

`d_dir` Direct estimate from the node-splitting model

`d_ind` Indirect estimate from the node-splitting model

`omega` Inconsistency factor $\omega = d_{\mathrm{dir}} - d_{\mathrm{ind}}$

`tau` Heterogeneity standard deviation from the node-splitting model, if a random effects model was fitted

`tau_consistency` Heterogeneity standard deviation from the corresponding consistency model, if available and if a random effects model was fitted

---

`pairs.stan_nma` *Matrix of plots for a* `stan_nma` *object*

Description

A `pairs()` method for `stan_nma` objects, which calls `bayesplot::mcmc_pairs()` on the underlying `stanfit` object.

Usage

```
## S3 method for class 'stan_nma'
pairs(x, ..., pars, include = TRUE)
```

Arguments

`x`	An object of class `stan_nma`
`...`	Other arguments passed to `bayesplot::mcmc_pairs()`
`pars`	Optional character vector of parameter names to include in output. If not specified, all parameters are used.
`include`	Logical, are parameters in `pars` to be included (`TRUE`, default) or excluded (`FALSE`)?

Value

A grid of ggplot objects produced by `bayesplot::mcmc_pairs()`.

Examples

```
## Not run:
## Parkinson's mean off time reduction
park_net <- set_agd_arm(parkinsons,
                        study = studyn,
                        trt = trtn,
                        y = y,
                        se = se,
                        sample_size = n)

# Fitting a RE model
park_fit_RE <- nma(park_net,
                   trt_effects = "random",
                   prior_intercept = normal(scale = 100),
                   prior_trt = normal(scale = 100),
                   prior_het = half_normal(scale = 5))

# We see a small number of divergent transition errors
# These do not go away entirely when adapt_delta is increased

# Try to diagnose with a pairs plot
pairs(park_fit_RE, pars = c("mu[4]", "d[3]", "delta[4: 3]", "tau"))

# Transforming tau onto log scale
pairs(park_fit_RE, pars = c("mu[4]", "d[3]", "delta[4: 3]", "tau"),
      transformations = list(tau = "log"))

# The divergent transitions occur in the upper tail of the heterogeneity
# standard deviation. In this case, with only a small number of studies, there
# is not very much information to estimate the heterogeneity standard deviation
# and the prior distribution may be too heavy-tailed. We could consider a more
# informative prior distribution for the heterogeneity variance to aid
# estimation.

## End(Not run)
```

`parkinsons` *Mean off-time reduction in Parkison's disease*

Description

Data frame containing the mean off-time reduction in patients given dopamine agonists as adjunct therapy in Parkinson's disease, from 7 trials comparing four active drugs and placebo (Dias et al. 2011).

Usage

```
parkinsons
```

Format

A data frame with 15 rows and 7 variables:

studyn numeric study ID

trtn numeric treatment code (placebo = 1)

y mean off-time reduction

se standard error

n sample size

diff mean difference vs. treatment in reference arm

se_diff standard error of mean difference, see details

Details

This dataset may be analysed using either an arm-based likelihood using `y` and `se`, or a contrast-based likelihood using `diff` and `se_diff` (or a combination of the two across different studies).

The contrast-based data is formatted as described in `set_agd_contrast()`. That is, for the chosen reference arm in each study, the mean difference `diff` is set to `NA`, and `se_diff` is set to the standard error `se` of the outcome on the reference arm.

References

Dias S, Welton NJ, Sutton AJ, Ades AE (2011). "NICE DSU Technical Support Document 2: A generalised linear modelling framework for pair-wise and network meta-analysis of randomised controlled trials." National Institute for Health and Care Excellence. https://www.sheffield.ac.uk/nice-dsu.

`plaque_psoriasis_ipd` *Plaque psoriasis data*

Description

Two data frames, `plaque_psoriasis_ipd` and `plaque_psoriasis_agd`, containing (simulated) individual patient data from four studies and aggregate data from five studies (Phillippo 2019). Outcomes are binary success/failure to achieve 75%, 90%, or 100% reduction in symptoms on the Psoriasis Area and Severity Index (PASI) scale.

Usage

```
plaque_psoriasis_ipd

plaque_psoriasis_agd
```

Format

The individual patient data are contained in a data frame `plaque_psoriasis_ipd` with 4118 rows, one per individual, and 16 variables:

studyc study name

trtc_long treatment name (long format)

trtc treatment name

trtn numeric treatment code

pasi75 binary PASI 75 outcome

pasi90 binary PASI 90 outcome

pasi100 binary PASI 100 outcome

age age (years)

bmi body mass index (BMI)

pasi_w0 PASI score at week 0

male male sex (TRUE or FALSE)

bsa body surface area (percent)

weight weight (kilograms)

durnpso duration of psoriasis (years)

prevsys previous systemic treatment (TRUE or FALSE)

psa psoriatic arthritis (TRUE or FALSE)

The aggregate data are contained in a data frame `plaque_psoriasis_agd` with 15 rows, one per study arm, and 26 variables:

studyc study name

trtc_long treatment name (long format)

trtc treatment name

trtn numeric treatment code

pasi75_r, pasi75_n PASI 75 outcome count and denominator

pasi90_r, pasi90_n PASI 75 outcome count and denominator

pasi100_r, pasi100_n PASI 75 outcome count and denominator

sample_size_w0 sample size at week zero

age_mean, age_sd mean and standard deviation of age (years)

bmi_mean, bmi_sd mean and standard deviation of BMI

pasi_w0_mean, pasi_w0_sd mean and standard deviation of PASI score at week 0

male percentage of males

bsa_mean, bsa_sd mean and standard deviation of body surface area (percent)

weight_mean, weight_sd mean and standard deviation of weight (kilograms)

durnpso_mean, durnpso_sd mean and standard deviation of duration of psoriasis (years)

prevsys percentage of individuals with previous systemic treatment

psa percentage of individuals with psoriatic arthritis

An object of class `data.frame` with 15 rows and 26 columns.

References

Phillippo DM (2019). *Calibration of Treatment Effects in Network Meta-Analysis using Individual Patient Data*. Ph.D. thesis, University of Bristol. Available from https://research-information.bris.ac.uk/.

`plot.nma_data` *Network plots*

Description

Create a network plot from a `nma_data` network object.

Usage

```
## S3 method for class 'nma_data'
plot(
  x,
  ...,
  layout,
  circular,
  weight_edges = TRUE,
  weight_nodes = FALSE,
  show_trt_class = FALSE,
  nudge = 0
)
```

Arguments

`x`	A nma_data object to plot
`...`	Additional arguments passed to `ggraph()` and on to the layout function
`layout`	The type of layout to create. Any layout accepted by `ggraph()` may be used, including all of the layout functions provided by igraph.
`circular`	Whether to use a circular representation. See `ggraph()`.
`weight_edges`	Weight edges by the number of studies? Default is `TRUE`.
`weight_nodes`	Weight nodes by the total sample size? Default is `FALSE`.
`show_trt_class`	Colour treatment nodes by class, if `trt_class` is set? Default is `FALSE`.
`nudge`	Numeric value to nudge the treatment labels away from the nodes when `weight_nodes = TRUE`. Default is `0` (no adjustment to label position). A small value like `0.1` is usually sufficient.

Details

The default is equivalent to `layout = "linear"` and `circular = TRUE`, which places the treatment nodes on a circle in the order defined by the treatment factor variable. An alternative layout which may give good results for simple networks is `"sugiyama"`, which attempts to minimise the number of edge crossings.

`weight_nodes = TRUE` requires that sample sizes have been specified for any aggregate data in the network, using the `sample_size` option of `set_agd_*()`.

Value

A `ggplot` object, as produced by `ggraph()`.

Examples

```
## Stroke prevention in atrial fibrillation
# Setting up the network
af_net <- set_agd_arm(atrial_fibrillation,
                      study = studyc,
                      trt = abbreviate(trtc, minlength = 3),
                      r = r,
                      n = n,
                      trt_class = trt_class)
af_net

# Basic plot
plot(af_net)

# Turn off weighting edges by number of studies
plot(af_net, weight_edges = FALSE)

# Turn on weighting nodes by sample size
plot(af_net, weight_nodes = TRUE)

# Colour treatment nodes by class
plot(af_net, weight_nodes = TRUE, show_trt_class = TRUE)

# Nudge the treatment labels away from the nodes
plot(af_net, weight_nodes = TRUE, show_trt_class = TRUE, nudge = 0.1)

# Output may be customised using standard ggplot commands
# For example, to display the legends below the plot:
plot(af_net, weight_nodes = TRUE, show_trt_class = TRUE) +
  ggplot2::theme(legend.position = "bottom",
                 legend.box = "vertical",
                 legend.margin = ggplot2::margin(0, 0, 0, 0),
                 legend.spacing = ggplot2::unit(0.5, "lines"))

# Choosing a different ggraph layout, hiding some legends
plot(af_net, weight_nodes = TRUE, show_trt_class = TRUE,
     layout = "star") +
  ggplot2::guides(edge_width = "none", size = "none")
```

plot.nma_dic *Plots of model fit diagnostics*

Description

The `plot()` method for nma_dic objects produced by `dic()` produces several useful diagnostic plots for checking model fit and model comparison. Further detail on these plots and their interpretation is given by Dias et al. (2011).

Usage

```
## S3 method for class 'nma_dic'
plot(
  x,
  y,
  ...,
  show_uncertainty = TRUE,
  stat = "pointinterval",
  orientation = c("vertical", "horizontal", "x", "y")
)
```

Arguments

`x`	A `nma_dic` object
`y`	(Optional) A second `nma_dic` object, to produce "dev-dev" plots for model comparison.
`...`	Additional arguments passed on to other methods
`show_uncertainty`	Logical, show uncertainty with a `ggdist` plot stat? Default `TRUE`.
`stat`	Character string specifying the `ggdist` plot stat to use if `show_uncertainty = TRUE`, default `"pointinterval"`. If `y` is provided, currently only `"pointinterval"` is supported.
`orientation`	Whether the `ggdist` geom is drawn horizontally (`"horizontal"`) or vertically (`"vertical"`). Only used for residual deviance plots, default `"vertical"`.

Details

When a single `nma_dic` object is given, a plot of the residual deviance contribution for each data point is produced. For a good fitting model, each data point is expected to have a residual deviance of 1; larger values indicate data points that are fit poorly by the model.

When two `nma_dic` objects are given, a "dev-dev" plot comparing the residual deviance contributions under each model is produced. Data points with residual deviance contributions lying on the line of equality are fit equally well under either model. Data points lying below the line of equality indicate better fit under the second model (`y`); conversely, data points lying above the line of equality indicate better fit under the first model (x). A common use case is to compare a standard consistency model (fitted using nma() with consistency = "consistency") with an unrelated mean effects (UME) inconsistency model (fitted using nma() with consistency = "ume"), to check for potential inconsistency.

See Dias et al. (2011) for further details.

Value

A ggplot object.

References

Dias S, Welton NJ, Sutton AJ, Ades AE (2011). "NICE DSU Technical Support Document 2: A generalised linear modelling framework for pair-wise and network meta-analysis of randomised controlled trials." National Institute for Health and Care Excellence. https://www.sheffield.ac.uk/nice-dsu.

Examples

```
## Smoking cessation

# Run smoking FE NMA example if not already available
if (!exists("smk_fit_FE")) example("example_smk_fe", run.donttest = TRUE)


# Run smoking RE NMA example if not already available
if (!exists("smk_fit_RE")) example("example_smk_re", run.donttest = TRUE)


# Compare DIC of FE and RE models
(smk_dic_FE <- dic(smk_fit_FE))
(smk_dic_RE <- dic(smk_fit_RE))   # substantially better fit

# Plot residual deviance contributions under RE model
plot(smk_dic_RE)

# Further customisation is possible using ggplot commands
# For example, highlighting data points with residual deviance above a certain threshold
plot(smk_dic_RE) +
  ggplot2::aes(colour = ggplot2::after_stat(ifelse(y > 1.5, "darkorange", "black"))) +
  ggplot2::scale_colour_identity()

# Or by posterior probability, for example here a central probability of 0.6
# corresponds to a lower tail probability of (1 - 0.6)/2 = 0.2
plot(smk_dic_RE, .width = c(0.6, 0.95)) +
  ggplot2::aes(colour = ggplot2::after_stat(ifelse(ymin > 1, "darkorange", "black"))) +
  ggplot2::scale_colour_identity()

# Check for inconsistency using UME model


# Run smoking UME NMA example if not already available
```

```
if (!exists("smk_fit_RE_UME")) example("example_smk_ume", run.donttest = TRUE)


# Compare DIC
smk_dic_RE
(smk_dic_RE_UME <- dic(smk_fit_RE_UME))  # no difference in fit

# Compare residual deviance contributions with a "dev-dev" plot
plot(smk_dic_RE, smk_dic_RE_UME)

# By default the dev-dev plot can be a little cluttered
# Hiding the credible intervals
plot(smk_dic_RE, smk_dic_RE_UME, show_uncertainty = FALSE)

# Changing transparency
plot(smk_dic_RE, smk_dic_RE_UME, point_alpha = 0.5, interval_alpha = 0.1)
```

`plot.nma_summary` *Plots of summary results*

Description

The `plot` method for `nma_summary` objects is used to produce plots of parameter estimates (when called on a `stan_nma` object or its summary), relative effects (when called on the output of `relative_effects()`), absolute predictions (when called on the output of `predict.stan_nma()`), posterior ranks and rank probabilities (when called on the output of `posterior_ranks()` or `posterior_rank_probs()`).

Usage

```
## S3 method for class 'nma_summary'
plot(
  x,
  ...,
  stat = "pointinterval",
  orientation = c("horizontal", "vertical", "y", "x"),
  ref_line = NA_real_
)

## S3 method for class 'nma_parameter_summary'
plot(
  x,
  ...,
  stat = "pointinterval",
  orientation = c("horizontal", "vertical", "y", "x"),
  ref_line = NA_real_
)
```

```
## S3 method for class 'nma_rank_probs'
plot(x, ...)

## S3 method for class 'surv_nma_summary'
plot(x, ..., stat = "lineribbon")
```

Arguments

`x`	A `nma_summary` object
`...`	Additional arguments passed on to the underlying `ggdist` plot stat, see Details
`stat`	Character string specifying the `ggdist` plot stat to use, default `"pointinterval"`, except when plotting estimated survival/hazard/cumulative hazard curves from survival models where the default is `"lineribbon"`
`orientation`	Whether the `ggdist` geom is drawn horizontally (`"horizontal"`) or vertically (`"vertical"`), default `"horizontal"`
`ref_line`	Numeric vector of positions for reference lines, by default no reference lines are drawn

Details

Plotting is handled by ggplot2 and the stats and geoms provided in the ggdist package. As a result, the output is very flexible. Any plotting stats provided by `ggdist` may be used, via the argument `stat`.

The default uses `ggdist::stat_pointinterval()`, to produce medians and 95% Credible Intervals with 66% inner bands. Additional arguments in `...` are passed to the `ggdist` stat, to customise the output. For example, to produce means and Credible Intervals, specify `point_interval = "mean_qi"`. To produce an 80% Credible Interval with no inner band, specify `.width = c(0, 0.8)`.

Alternative stats can be specified to produce different summaries. For example, specify `stat = "[half]eye"` to produce (half) eye plots, or `stat = "histinterval"` to produce histograms with intervals.

A full list of options and examples is found in the ggdist vignette `vignette("slabinterval", package = "ggdist")`.

For survival/hazard/cumulative hazard curves estimated from survival models, the default uses `ggdist::stat_lineribbon()` which produces curves of posterior medians with 50%, 80%, and 95% Credible Interval bands. Again, additional arguments in `...` are passed to the `ggdist` stat. For example, to produce posterior means and 95% Credible bands, specify `point_interval = "mean_qi"` and `.width = 0.95`.

A `ggplot` object is returned which can be further modified through the usual ggplot2 functions to add further aesthetics, geoms, themes, etc.

Value

A `ggplot` object.

Examples

```
## Smoking cessation

# Run smoking RE NMA example if not already available
if (!exists("smk_fit_RE")) example("example_smk_re", run.donttest = TRUE)


# Produce relative effects
smk_releff_RE <- relative_effects(smk_fit_RE)
plot(smk_releff_RE, ref_line = 0)

# Customise plot options
plot(smk_releff_RE, ref_line = 0, stat = "halfeye")

# Further customisation is possible with ggplot commands
plot(smk_releff_RE, ref_line = 0, stat = "halfeye", slab_alpha = 0.6) +
  ggplot2::aes(slab_fill = ggplot2::after_stat(ifelse(x < 0, "darkred", "grey60")))

# Produce posterior ranks
smk_rank_RE <- posterior_ranks(smk_fit_RE, lower_better = FALSE)
plot(smk_rank_RE)

# Produce rank probabilities
smk_rankprob_RE <- posterior_rank_probs(smk_fit_RE, lower_better = FALSE)
plot(smk_rankprob_RE)

# Produce cumulative rank probabilities
smk_cumrankprob_RE <- posterior_rank_probs(smk_fit_RE, lower_better = FALSE,
                                           cumulative = TRUE)
plot(smk_cumrankprob_RE)

# Further customisation is possible with ggplot commands
plot(smk_cumrankprob_RE) +
  ggplot2::facet_null() +
  ggplot2::aes(colour = Treatment)
```

---

plot.nodesplit_summary

Plots of node-splitting models

Description

Produce summary plots of node-splitting models

Usage

```
## S3 method for class 'nodesplit_summary'
plot(
  x,
  ...,
  pars = "d",
  stat = "dens_overlay",
  orientation = c("horizontal", "vertical", "y", "x"),
  ref_line = NA_real_
)
```

Arguments

`x`	A `nodesplit_summary` object.
`...`	Additional arguments passed on to the underlying `ggdist` plot stat, see Details.
`pars`	Character vector specifying the parameters to include in the plot, choices include `"d"` for the direct, indirect, and network estimates of relative effects, `"omega"` for the inconsistency factor, and `"tau"` for heterogeneity standard deviation in random effects models. Default is `"d"`.
`stat`	Character string specifying the `ggdist` plot stat to use. The default `"dens_overlay"` is a special case, producing an overlaid density plot.
`orientation`	Whether the `ggdist` geom is drawn horizontally (`"horizontal"`) or vertically (`"vertical"`), default `"horizontal"`.
`ref_line`	Numeric vector of positions for reference lines, by default no reference lines are drawn.

Details

Plotting is handled by ggplot2 and the stats and geoms provided in the ggdist package. As a result, the output is very flexible. Any plotting stats provided by `ggdist` may be used, via the argument `stat`. The default `"dens_overlay"` is a special exception, which uses `ggplot2::geom_density()`, to plot overlaid densities. Additional arguments in `...` are passed to the `ggdist` stat, to customise the output.

Alternative stats can be specified to produce different summaries. For example, specify `stat = "[half]eye"` to produce (half) eye plots, or `stat = "pointinterval"` to produce point estimates and credible intervals.

A full list of options and examples is found in the ggdist vignette `vignette("slabinterval", package = "ggdist")`.

A `ggplot` object is returned which can be further modified through the usual ggplot2 functions to add further aesthetics, geoms, themes, etc.

Value

A `ggplot` object.

See Also

`summary.nma_nodesplit_df()`

Examples

```
# Run smoking node-splitting example if not already available
if (!exists("smk_fit_RE_nodesplit")) example("example_smk_nodesplit", run.donttest = TRUE)


# Summarise the node-splitting results
(smk_nodesplit_summary <- summary(smk_fit_RE_nodesplit))

# Plot the node-splitting results
plot(smk_nodesplit_summary)

# Plot the inconsistency factors instead, change the plot stat to half-eye,
# and add a reference line at 0
plot(smk_nodesplit_summary, pars = "omega", stat = "halfeye", ref_line = 0)

# Plot a comparison of the heterogeneity under the node-split models vs.
# the consistency model
plot(smk_nodesplit_summary, pars = "tau")
```

`plot_integration_error` *Plot numerical integration error*

Description

For ML-NMR models, plot the estimated numerical integration error over the entire posterior distribution, as the number of integration points increases. See (Phillippo et al. 2020; Phillippo 2019) for details.

Usage

```
plot_integration_error(
  x,
  ...,
  stat = "violin",
  orientation = c("vertical", "horizontal", "x", "y"),
  show_expected_rate = TRUE
)
```

Arguments

`x`	An object of type `stan_mlnmr`
`...`	Additional arguments passed to the `ggdist` plot stat.
`stat`	Character string specifying the `ggdist` plot stat used to summarise the integration error over the posterior. Default is `"violin"`, which is equivalent to `"eye"` with some cosmetic tweaks.

`orientation` Whether the `ggdist` geom is drawn horizontally (`"horizontal"`) or vertically (`"vertical"`), default `"vertical"`

`show_expected_rate`
Logical, show typical convergence rate $1/N$? Default `TRUE`.

Details

The total number of integration points is set by the `n_int` argument to `add_integration()`, and the intervals at which integration error is estimated are set by the `int_thin` argument to `nma()`. The typical convergence rate of Quasi-Monte Carlo integration (as used here) is $1/N$, which by default is displayed on the plot output.

The integration error at each thinning interval N_{thin} is estimated for each point in the posterior distribution by subtracting the final estimate (using all `n_int` points) from the estimate using only the first N_{thin} points.

Value

A `ggplot` object.

Note for survival models

This function is not supported for survival/time-to-event models. These do not save cumulative integration points for efficiency reasons (both time and memory).

Examples

```
## Plaque psoriasis ML-NMR
# Set up plaque psoriasis network combining IPD and AgD
library(dplyr)
pso_ipd <- filter(plaque_psoriasis_ipd,
                  studyc %in% c("UNCOVER-1", "UNCOVER-2", "UNCOVER-3"))

pso_agd <- filter(plaque_psoriasis_agd,
                  studyc == "FIXTURE")

head(pso_ipd)
head(pso_agd)

pso_ipd <- pso_ipd %>%
  mutate(# Variable transformations
    bsa = bsa / 100,
    prevsys = as.numeric(prevsys),
    psa = as.numeric(psa),
    weight = weight / 10,
    durnpso = durnpso / 10,
    # Treatment classes
    trtclass = case_when(trtn == 1 ~ "Placebo",
                         trtn %in% c(2, 3, 5, 6) ~ "IL blocker",
                         trtn == 4 ~ "TNFa blocker"),
    # Check complete cases for covariates of interest
    complete = complete.cases(durnpso, prevsys, bsa, weight, psa)
  )

pso_agd <- pso_agd %>%
  mutate(
    # Variable transformations
    bsa_mean = bsa_mean / 100,
    bsa_sd = bsa_sd / 100,
    prevsys = prevsys / 100,
    psa = psa / 100,
    weight_mean = weight_mean / 10,
    weight_sd = weight_sd / 10,
    durnpso_mean = durnpso_mean / 10,
    durnpso_sd = durnpso_sd / 10,
    # Treatment classes
    trtclass = case_when(trtn == 1 ~ "Placebo",
                         trtn %in% c(2, 3, 5, 6) ~ "IL blocker",
                         trtn == 4 ~ "TNFa blocker")
  )

# Exclude small number of individuals with missing covariates
pso_ipd <- filter(pso_ipd, complete)

pso_net <- combine_network(
  set_ipd(pso_ipd,
          study = studyc,
          trt = trtc,
          r = pasi75,
          trt_class = trtclass),
  set_agd_arm(pso_agd,
              study = studyc,
              trt = trtc,
              r = pasi75_r,
              n = pasi75_n,
              trt_class = trtclass)
)

# Print network details
pso_net

# Add integration points to the network
pso_net <- add_integration(pso_net,
  durnpso = distr(qgamma, mean = durnpso_mean, sd = durnpso_sd),
  prevsys = distr(qbern, prob = prevsys),
  bsa = distr(qlogitnorm, mean = bsa_mean, sd = bsa_sd),
  weight = distr(qgamma, mean = weight_mean, sd = weight_sd),
  psa = distr(qbern, prob = psa),
  n_int = 64)


# Fit the ML-NMR model
pso_fit <- nma(pso_net,
               trt_effects = "fixed",
               link = "probit",
               likelihood = "bernoulli2",
               regression = ~(durnpso + prevsys + bsa + weight + psa)*.trt,
               class_interactions = "common",
               prior_intercept = normal(scale = 10),
               prior_trt = normal(scale = 10),
               prior_reg = normal(scale = 10),
               init_r = 0.1,
               QR = TRUE,
               # Set the thinning factor for saving the cumulative results
               # (This also sets int_check = FALSE)
               int_thin = 8)
pso_fit

# Plot numerical integration error
plot_integration_error(pso_fit)
```

plot_prior_posterior *Plot prior vs posterior distribution*

Description

Produce plots comparing the prior and posterior distributions of model parameters.

Usage

```
plot_prior_posterior(
  x,
  ...,
  prior = NULL,
  post_args = list(),
  prior_args = list(),
  overlay = c("prior", "posterior"),
  ref_line = NA_real_
)
```

Arguments

`x`	A `stan_nma` object
`...`	Additional arguments passed on to methods
`prior`	Character vector selecting the prior and posterior distribution(s) to plot. May include `"intercept"`, `"trt"`, `"het"`, `"reg"`, or `"aux"`, as appropriate.
`post_args`	List of arguments passed on to ggplot2::geom_histogram to control plot output for the posterior distribution
`prior_args`	List of arguments passed on to ggplot2::geom_path to control plot output for the prior distribution. Additionally, `n` controls the number of points the density curve is evaluated at (default `500`), and `p_limits` controls the endpoints of the curve as quantiles (default `c(.001, .999)`).

`overlay`	String, should prior or posterior be shown on top? Default `"prior"`.
`ref_line`	Numeric vector of positions for reference lines, by default no reference lines are drawn

Details

Prior distributions are displayed as lines, posterior distributions are displayed as histograms.

Value

A `ggplot` object.

Examples

```
## Smoking cessation NMA

# Run smoking RE NMA example if not already available
if (!exists("smk_fit_RE")) example("example_smk_re", run.donttest = TRUE)


# Plot prior vs. posterior, by default all parameters are plotted
plot_prior_posterior(smk_fit_RE)

# Plot prior vs. posterior for heterogeneity SD only
plot_prior_posterior(smk_fit_RE, prior = "het")

# Customise plot
plot_prior_posterior(smk_fit_RE, prior = "het",
                     prior_args = list(colour = "darkred", size = 2),
                     post_args = list(alpha = 0.6))
```

`posterior_ranks` *Treatment rankings and rank probabilities*

Description

Produce posterior treatment rankings and rank probabilities from a fitted NMA model. When a meta-regression is fitted with effect modifier interactions with treatment, these will differ by study population.

Usage

```
posterior_ranks(
  x,
  newdata = NULL,
  study = NULL,
  lower_better = TRUE,
  probs = c(0.025, 0.25, 0.5, 0.75, 0.975),
  sucra = FALSE,
  summary = TRUE
)

posterior_rank_probs(
  x,
  newdata = NULL,
  study = NULL,
  lower_better = TRUE,
  cumulative = FALSE,
  sucra = FALSE
)
```

Arguments

`x`	A `stan_nma` object created by `nma()`
`newdata`	Only used if a regression model is fitted. A data frame of study details, one row per study, giving the covariate values at which to produce relative effects. Column names must match variables in the regression model. If `NULL`, relative effects are produced for all studies in the network.
`study`	Column of `newdata` which specifies study names, otherwise studies will be labelled by row number.
`lower_better`	Logical, are lower treatment effects better (`TRUE`; default) or higher better (`FALSE`)? See details.
`probs`	Numeric vector of quantiles of interest to present in computed summary, default `c(0.025, 0.25, 0.5, 0.75, 0.975)`
`sucra`	Logical, calculate the surface under the cumulative ranking curve (SUCRA) for each treatment? Default `FALSE`.
`summary`	Logical, calculate posterior summaries? Default `TRUE`.
`cumulative`	Logical, return cumulative rank probabilities? Default is `FALSE`, return posterior probabilities of each treatment having a given rank. If `TRUE`, cumulative posterior rank probabilities are returned for each treatment having a given rank or better.

Details

The function `posterior_ranks()` produces posterior rankings, which have a distribution (e.g. mean/median rank and 95% Credible Interval). The function `posterior_rank_probs()` produces rank probabilities, which give the posterior probabilities of being ranked first, second, etc. out of all treatments.

The argument `lower_better` specifies whether lower treatment effects or higher treatment effects are preferred. For example, with a negative binary outcome lower (more negative) log odds ratios are preferred, so `lower_better = TRUE`. Conversely, for example, if treatments aim to increase the rate of a positive outcome then `lower_better = FALSE`.

Value

A nma_summary object if `summary = TRUE`, otherwise a list containing a 3D MCMC array of samples and (for regression models) a data frame of study information.

See Also

`plot.nma_summary()` for plotting the ranks and rank probabilities.

Examples

```
## Smoking cessation

# Run smoking RE NMA example if not already available
if (!exists("smk_fit_RE")) example("example_smk_re", run.donttest = TRUE)


# Produce posterior ranks
smk_rank_RE <- posterior_ranks(smk_fit_RE, lower_better = FALSE)
smk_rank_RE
plot(smk_rank_RE)

# Produce rank probabilities
smk_rankprob_RE <- posterior_rank_probs(smk_fit_RE, lower_better = FALSE)
smk_rankprob_RE
plot(smk_rankprob_RE)

# Produce cumulative rank probabilities
smk_cumrankprob_RE <- posterior_rank_probs(smk_fit_RE, lower_better = FALSE,
                                           cumulative = TRUE)
smk_cumrankprob_RE
plot(smk_cumrankprob_RE)

# Further customisation is possible with ggplot commands
plot(smk_cumrankprob_RE) +
  ggplot2::facet_null() +
  ggplot2::aes(colour = Treatment)


## Plaque psoriasis ML-NMR

# Run plaque psoriasis ML-NMR example if not already available
if (!exists("pso_fit")) example("example_pso_mlnmr", run.donttest = TRUE)


# Produce population-adjusted rankings for all study populations in
# the network

# Ranks
pso_rank <- posterior_ranks(pso_fit)
pso_rank
plot(pso_rank)
```

```
# Rank probabilities
pso_rankprobs <- posterior_rank_probs(pso_fit)
pso_rankprobs
plot(pso_rankprobs)

# Cumulative rank probabilities
pso_cumrankprobs <- posterior_rank_probs(pso_fit, cumulative = TRUE)
pso_cumrankprobs
plot(pso_cumrankprobs)

# Produce population-adjusted rankings for a different target
# population
new_agd_means <- data.frame(
  bsa = 0.6,
  prevsys = 0.1,
  psa = 0.2,
  weight = 10,
  durnpso = 3)

# Ranks
posterior_ranks(pso_fit, newdata = new_agd_means)

# Rank probabilities
posterior_rank_probs(pso_fit, newdata = new_agd_means)

# Cumulative rank probabilities
posterior_rank_probs(pso_fit, newdata = new_agd_means,
                     cumulative = TRUE)
```

`predict.stan_nma` *Predictions of absolute effects from NMA models*

Description

Obtain predictions of absolute effects from NMA models fitted with `nma()`. For example, if a model is fitted to binary data with a logit link, predicted outcome probabilities or log odds can be produced. For survival models, predictions can be made for survival probabilities, (cumulative) hazards, (restricted) mean survival times, and quantiles including median survival times. When an IPD NMA or ML-NMR model has been fitted, predictions can be produced either at an individual level or at an aggregate level. Aggregate-level predictions are population-average absolute effects; these are marginalised or standardised over a population. For example, average event probabilities from a logistic regression, or marginal (standardised) survival probabilities from a survival model.

Usage

```
## S3 method for class 'stan_nma'
predict(
  object,
  ...,
  baseline = NULL,
  newdata = NULL,
  study = NULL,
  type = c("link", "response"),
  level = c("aggregate", "individual"),
  baseline_trt = NULL,
  baseline_type = c("link", "response"),
  baseline_level = c("individual", "aggregate"),
  probs = c(0.025, 0.25, 0.5, 0.75, 0.975),
  predictive_distribution = FALSE,
  summary = TRUE,
  progress = FALSE,
  trt_ref = NULL
)

## S3 method for class 'stan_nma_surv'
predict(
  object,
  times = NULL,
  ...,
  baseline_trt = NULL,
  baseline = NULL,
  aux = NULL,
  newdata = NULL,
  study = NULL,
  type = c("survival", "hazard", "cumhaz", "mean", "median", "quantile", "rmst", "link"),
  quantiles = c(0.25, 0.5, 0.75),
  level = c("aggregate", "individual"),
  times_seq = NULL,
  probs = c(0.025, 0.25, 0.5, 0.75, 0.975),
  predictive_distribution = FALSE,
  summary = TRUE,
  progress = interactive(),
  trt_ref = NULL
)
```

Arguments

object	A `stan_nma` object created by `nma()`.
...	Additional arguments, passed to `uniroot()` for regression models if `baseline_level = "aggregate"`.
baseline	An optional `distr()` distribution for the baseline response (i.e. intercept), about which to produce absolute effects. Can also be a character string naming a study in the network to take the estimated baseline response distribution from. If `NULL`, predictions are produced using the baseline response for each study in the network with IPD or arm-based AgD. For regression models, this may be a list of `distr()` distributions (or study

	names in the network to use the baseline distributions from) of the same length as the number of studies in `newdata`, possibly named by the studies in `newdata` or otherwise in order of appearance in `newdata`. Use the `baseline_type` and `baseline_level` arguments to specify whether this distribution is on the response or linear predictor scale, and (for ML-NMR or models including IPD) whether this applies to an individual at the reference level of the covariates or over the entire `newdata` population, respectively. For example, in a model with a logit link with `baseline_type = "link"`, this would be a distribution for the baseline log odds of an event. For survival models, `baseline` always corresponds to the intercept parameters in the linear predictor (i.e. `baseline_type` is always `"link"`, and `baseline_level` is `"individual"` for IPD NMA or ML-NMR, and `"aggregate"` for AgD NMA). Use the `baseline_trt` argument to specify which treatment this distribution applies to.
`newdata`	Only required if a regression model is fitted and `baseline` is specified. A data frame of covariate details, for which to produce predictions. Column names must match variables in the regression model. If `level = "aggregate"` this should either be a data frame with integration points as produced by `add_integration()` (one row per study), or a data frame with individual covariate values (one row per individual) which are summarised over. If `level = "individual"` this should be a data frame of individual covariate values, one row per individual. If `NULL`, predictions are produced for all studies with IPD and/or arm-based AgD in the network, depending on the value of `level`.
`study`	Column of `newdata` which specifies study names or IDs. When not specified: if `newdata` contains integration points produced by `add_integration()`, studies will be labelled sequentially by row; otherwise data will be assumed to come from a single study.
`type`	Whether to produce predictions on the `"link"` scale (the default, e.g. log odds) or `"response"` scale (e.g. probabilities). For survival models, the options are `"survival"` for survival probabilities (the default), `"hazard"` for hazards, `"cumhaz"` for cumulative hazards, `"mean"` for mean survival times, `"quantile"` for quantiles of the survival time distribution, `"median"` for median survival times (equivalent to `type = "quantile"` with `quantiles = 0.5`), `"rmst"` for restricted mean survival times, or `"link"` for the linear predictor. For `type = "survival"`, `"hazard"` or `"cumhaz"`, predictions are given at the times specified by `times` or at the event/censoring times in the network if `times = NULL`. For `type = "rmst"`, the restricted time horizon is specified by `times`, or if `times = NULL` the earliest last follow-up time amongst the studies in the network is used. When `level = "aggregate"`, these all correspond to the standardised survival function (see details).
`level`	The level at which predictions are produced, either `"aggregate"` (the default), or `"individual"`. If `baseline` is not specified, predictions are produced for all IPD studies in the network if `level` is `"individual"` or `"aggregate"`, and for all arm-based AgD studies in the network if `level` is `"aggregate"`.

`baseline_trt`	Treatment to which the `baseline` response distribution refers, if `baseline` is specified. By default, the baseline response distribution will refer to the network reference treatment. Coerced to character string.
`baseline_type`	When a `baseline` distribution is given, specifies whether this corresponds to the `"link"` scale (the default, e.g. log odds) or `"response"` scale (e.g. probabilities). For survival models, `baseline` always corresponds to the intercept parameters in the linear predictor (i.e. `baseline_type` is always `"link"`).
`baseline_level`	When a `baseline` distribution is given, specifies whether this corresponds to an individual at the reference level of the covariates (`"individual"`, the default), or from an (unadjusted) average outcome on the reference treatment in the `newdata` population (`"aggregate"`). Ignored for AgD NMA, since the only option is `"aggregate"` in this instance. For survival models, `baseline` always corresponds to the intercept parameters in the linear predictor (i.e. `baseline_level` is `"individual"` for IPD NMA or ML-NMR, and `"aggregate"` for AgD NMA).
`probs`	Numeric vector of quantiles of interest to present in computed summary, default `c(0.025, 0.25, 0.5, 0.75, 0.975)`
`predictive_distribution`	Logical, when a random effects model has been fitted, should the predictive distribution for absolute effects in a new study be returned? Default `FALSE`.
`summary`	Logical, calculate posterior summaries? Default `TRUE`.
`progress`	Logical, display progress for potentially long-running calculations? Population-average predictions from ML-NMR models are computationally intensive, especially for survival outcomes. Currently the default is to display progress only when running interactively and producing predictions for a survival ML-NMR model.
`trt_ref`	Deprecated, renamed to `baseline_trt`.
`times`	A numeric vector of times to evaluate predictions at. Alternatively, if `newdata` is specified, `times` can be the name of a column in `newdata` which contains the times. If `NULL` (the default) then predictions are made at the event/censoring times from the studies included in the network (or according to `times_seq`). Only used if `type` is `"survival"`, `"hazard"`, `"cumhaz"` or `"rmst"`.
`aux`	An optional `distr()` distribution for the auxiliary parameter(s) in the baseline hazard (e.g. shapes). Can also be a character string naming a study in the network to take the estimated auxiliary parameter distribution from. If `NULL`, predictions are produced using the parameter estimates for each study in the network with IPD or arm-based AgD. For regression models, this may be a list of `distr()` distributions (or study names in the network to use the auxiliary parameters from) of the same length as the number of studies in `newdata`, possibly named by the study names or otherwise in order of appearance in `newdata`.
`quantiles`	A numeric vector of quantiles of the survival time distribution to produce estimates for when `type = "quantile"`.
`times_seq`	A positive integer, when specified evaluate predictions at this many evenly-spaced event times between 0 and the latest follow-up time in each study, instead of every observed event/censoring time. Only used if `newdata = NULL`

and `type` is `"survival"`, `"hazard"` or `"cumhaz"`. This can be useful for plotting survival or (cumulative) hazard curves, where prediction at every observed even/censoring time is unnecessary and can be slow. When a call from within `plot()` is detected, e.g. like `plot(predict(fit, type = "survival"))`, `times_seq` will default to 50.

Value

A nma_summary object if `summary = TRUE`, otherwise a list containing a 3D MCMC array of samples and (for regression models) a data frame of study information.

Aggregate-level predictions from IPD NMA and ML-NMR models

Population-average absolute effects can be produced from IPD NMA and ML-NMR models with `level = "aggregate"`. Predictions are averaged over the target population (i.e. standardised/marginalised), either by (numerical) integration over the joint covariate distribution (for AgD studies in the network for ML-NMR, or AgD `newdata` with integration points created by `add_integration()`), or by averaging predictions for a sample of individuals (for IPD studies in the network for IPD NMA/ML-NMR, or IPD `newdata`).

For example, with a binary outcome, the population-average event probabilities on treatment k in study/population j are

$$\bar{p}_{jk} = \int_{\mathfrak{X}} p_{jk}(\mathbf{x}) f_{jk}(\mathbf{x}) d\mathbf{x}$$

for a joint covariate distribution $f_{jk}(\mathbf{x})$ with support $\mathfrak{X}$ or

$$\bar{p}_{jk} = \sum_i p_{jk}(\mathbf{x}_i)$$

for a sample of individuals with covariates $\mathbf{x}_i$.

Population-average absolute predictions follow similarly for other types of outcomes, however for survival outcomes there are specific considerations.

Standardised survival predictions:

Different types of population-average survival predictions, often called standardised survival predictions, follow from the **standardised survival function** created by integrating (or equivalently averaging) the individual-level survival functions at each time t:

$$\bar{S}_{jk}(t) = \int_{\mathfrak{X}} S_{jk}(t|\mathbf{x}) f_{jk}(\mathbf{x}) d\mathbf{x}$$

which is itself produced using `type = "survival"`.

The **standardised hazard function** corresponding to this standardised survival function is a weighted average of the individual-level hazard functions

$$\bar{h}_{jk}(t) = \frac{\int_{\mathfrak{X}} S_{jk}(t|\mathbf{x}) h_{jk}(t|\mathbf{x}) f_{jk}(\mathbf{x}) d\mathbf{x}}{\bar{S}_{jk}(t)}$$

weighted by the probability of surviving to time t. This is produced using `type = "hazard"`.

The corresponding **standardised cumulative hazard function** is

$$\bar{H}_{jk}(t) = -\log(\bar{S}_{jk}(t))$$

and is produced using `type = "cumhaz"`.

Quantiles and medians of the standardised survival times are found by solving

$$\bar{S}_{jk}(t) = 1 - \alpha$$

for the $\alpha\%$ quantile, using numerical root finding. These are produced using `type = "quantile"` or `"median"`.

(Restricted) means of the standardised survival times are found by integrating

$$\mathrm{RMST}_{jk}(t^*) = \int_0^{t^*} \bar{S}_{jk}(t) dt$$

up to the restricted time horizon t^*, with $t^* = \infty$ for mean standardised survival time. These are produced using `type = "rmst"` or `"mean"`.

See Also

`plot.nma_summary()` for plotting the predictions.

Examples

```
## Smoking cessation

# Run smoking RE NMA example if not already available
if (!exists("smk_fit_RE")) example("example_smk_re", run.donttest = TRUE)


# Predicted log odds of success in each study in the network
predict(smk_fit_RE)

# Predicted probabilities of success in each study in the network
predict(smk_fit_RE, type = "response")

# Predicted probabilities in a population with 67 observed events out of 566
# individuals on No Intervention, corresponding to a Beta(67, 566 - 67)
# distribution on the baseline probability of response, using
# `baseline_type = "response"`
(smk_pred_RE <- predict(smk_fit_RE,
                        baseline = distr(qbeta, 67, 566 - 67),
                        baseline_type = "response",
                        type = "response"))
plot(smk_pred_RE, ref_line = c(0, 1))

# Predicted probabilities in a population with a baseline log odds of
# response on No Intervention given a Normal distribution with mean -2
# and SD 0.13, using `baseline_type = "link"` (the default)
# Note: this is approximately equivalent to the above Beta distribution on
# the baseline probability
(smk_pred_RE2 <- predict(smk_fit_RE,
                         baseline = distr(qnorm, mean = -2, sd = 0.13),
                         type = "response"))
plot(smk_pred_RE2, ref_line = c(0, 1))
```

```
## Plaque psoriasis ML-NMR

# Run plaque psoriasis ML-NMR example if not already available
if (!exists("pso_fit")) example("example_pso_mlnmr", run.donttest = TRUE)


# Predicted probabilities of response in each study in the network
(pso_pred <- predict(pso_fit, type = "response"))
plot(pso_pred, ref_line = c(0, 1))

# Predicted probabilites of response in a new target population, with means
# and SDs or proportions given by
new_agd_int <- data.frame(
  bsa_mean = 0.6,
  bsa_sd = 0.3,
  prevsys = 0.1,
  psa = 0.2,
  weight_mean = 10,
  weight_sd = 1,
  durnpso_mean = 3,
  durnpso_sd = 1
)

# We need to add integration points to this data frame of new data
# We use the weighted mean correlation matrix computed from the IPD studies
new_agd_int <- add_integration(new_agd_int,
                           durnpso = distr(qgamma, mean = durnpso_mean, sd = durnpso_sd),
                               prevsys = distr(qbern, prob = prevsys),
                               bsa = distr(qlogitnorm, mean = bsa_mean, sd = bsa_sd),
                             weight = distr(qgamma, mean = weight_mean, sd = weight_sd),
                               psa = distr(qbern, prob = psa),
                               cor = pso_net$int_cor,
                               n_int = 64)

# Predicted probabilities of achieving PASI 75 in this target population, given
# a Normal(-1.75, 0.08^2) distribution on the baseline probit-probability of
# response on Placebo (at the reference levels of the covariates), are given by
(pso_pred_new <- predict(pso_fit,
                         type = "response",
                         newdata = new_agd_int,
                         baseline = distr(qnorm, -1.75, 0.08)))
plot(pso_pred_new, ref_line = c(0, 1))


## Progression free survival with newly-diagnosed multiple myeloma

# Run newly-diagnosed multiple myeloma example if not already available
if (!exists("ndmm_fit")) example("example_ndmm", run.donttest = TRUE)


# We can produce a range of predictions from models with survival outcomes,
# chosen with the type argument to predict

# Predicted survival probabilities at 5 years
predict(ndmm_fit, type = "survival", times = 5)

# Survival curves
plot(predict(ndmm_fit, type = "survival"))

# Hazard curves
# Here we specify a vector of times to avoid attempting to plot infinite
# hazards for some studies at t=0
plot(predict(ndmm_fit, type = "hazard", times = seq(0.001, 6, length.out = 50)))

# Cumulative hazard curves
plot(predict(ndmm_fit, type = "cumhaz"))

# Survival time quantiles and median survival
predict(ndmm_fit, type = "quantile", quantiles = c(0.2, 0.5, 0.8))
plot(predict(ndmm_fit, type = "median"))

# Mean survival time
predict(ndmm_fit, type = "mean")

# Restricted mean survival time
# By default, the time horizon is the shortest follow-up time in the network
predict(ndmm_fit, type = "rmst")

# An alternative restriction time can be set using the times argument
predict(ndmm_fit, type = "rmst", times = 5)  # 5-year RMST
```

`print.nma_data` *Print* `nma_data` *objects*

Description

Print details of networks stored as nma_data objects, as created by `set_ipd()`, `set_agd_arm()`, `set_agd_contrast()`, `set_agd_surv()`, or `combine_network()`.

Usage

```
## S3 method for class 'nma_data'
print(x, ..., n = 10)

## S3 method for class 'mlnmr_data'
print(x, ..., n = 10)
```

Arguments

`x`	`nma_data` object
`...`	other options (not used)
`n`	number of studies of each type to print

Value

`x` is returned invisibly.

`print.nma_dic` *Print DIC details*

Description

Print details of DIC model fit statistics, computed by `dic()` function.

Usage

```
## S3 method for class 'nma_dic'
print(x, digits = 1, ...)
```

Arguments

`x`	An object of class nma_dic
`digits`	An integer passed to `round()`
`...`	Ignored

Value

`x` is returned invisibly.

`print.nma_nodesplit_df` *Print* `nma_nodesplit_df` *objects*

Description

Print `nma_nodesplit_df` objects

Usage

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

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

Arguments

`x`	A nma_nodesplit_df object
`...`	Further arguments passed to `print.stanfit()`

Value

`x` is returned invisibly.

See Also

The summary method `summary.nma_nodesplit_df()` summarises the node-splitting results.

`print.nma_summary` *Methods for* `nma_summary` *objects*

Description

The `as.data.frame()`, `as_tibble()`, and `as.tibble()` methods return the posterior summary statistics in a data frame or tibble. The `as.matrix()` method returns a matrix of posterior draws. The `as.array()` method returns a 3D array [Iteration, Chain, Parameter] of posterior draws (as class mcmc_array).

Usage

```
## S3 method for class 'nma_summary'
print(x, ..., digits = 2, pars, include = TRUE)

## S3 method for class 'nma_summary'
as.data.frame(x, ...)

## S3 method for class 'nma_summary'
as.tibble(x, ...)

## S3 method for class 'nma_summary'
as_tibble(x, ...)

## S3 method for class 'nma_summary'
as.array(x, ...)

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

## S3 method for class 'nma_rank_probs'
as.array(x, ...)

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

Arguments

`x`	A `nma_summary` object
`...`	Additional arguments passed on to other methods
`digits`	Integer number of digits to display
`pars`	Character vector of parameters to display in the printed summary
`include`	Logical, are parameters named in `pars` included (`TRUE`) or excluded (`FALSE`)

Value

A `data.frame` for `as.data.frame()`, a `tbl_df` for `as.tibble()` and `as_tibble()`, a `matrix` for `as.matrix()`, and an `mcmc_array` for `as.array()`.

The `print()` method returns `x` invisibly.

See Also

`plot.nma_summary()`

---

print.nodesplit_summary

Methods for `nodesplit_summary` *objects*

Description

The `as.data.frame()`, `as_tibble()`, and `as.tibble()` methods return the node-splitting summaries in a data frame or tibble.

Usage

```
## S3 method for class 'nodesplit_summary'
print(x, ..., digits = 2)

## S3 method for class 'nodesplit_summary'
as_tibble(x, ..., nest = FALSE)

as.tibble.nodesplit_summary(x, ..., nest = FALSE)

## S3 method for class 'nodesplit_summary'
as.data.frame(x, ...)
```

Arguments

`x`	A `nodesplit_summary` object
`...`	Additional arguments passed on to other methods
`digits`	Integer number of digits to display
`nest`	Whether to return a nested tibble, with the full nma_summary and nma_dic objects, or to unnest their summaries, default `FALSE`

Value

A data.frame for as.data.frame(), a tbl_df for as.tibble() and as_tibble().

The print() method returns x invisibly.

See Also

plot.nodesplit_summary()

---

print.stan_nma *Print* stan_nma *objects*

Description

Print stan_nma objects

Usage

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

Arguments

x	A stan_nma object
...	Further arguments passed to print.stanfit()

Value

x is returned invisibly.

---

priors *Prior distributions*

Description

These functions are used to specify prior distributions for the model parameters.

Usage

```
normal(location = 0, scale)

half_normal(scale)

log_normal(location, scale)

cauchy(location = 0, scale)

half_cauchy(scale)

student_t(location = 0, scale, df)

half_student_t(scale, df)

log_student_t(location, scale, df)

exponential(scale = 1/rate, rate = 1/scale)

flat()
```

Arguments

`location`	Prior location. Typically prior mean (see details).
`scale`	Prior scale. Typically prior standard deviation (see details).
`df`	Prior degrees of freedom.
`rate`	Prior rate.

Details

The `location` and `scale` parameters are typically the prior mean and standard deviation, with the following exceptions:

- For the Cauchy distribution `location` is the prior median and `scale` is the prior scale.
- For the log-Normal distribution, `location` and `scale` are the prior mean and standard deviation of the logarithm.

Compatibility with model parameters:

The following table summarises which prior distributions may be used with which model parameters. Essentially, priors that take only non-negative values (e.g. half-Normal) may only be used for non-negative parameters (heterogeneity SD/variance/precision, and any auxiliary parameter). If a real-valued prior distribution is specified for a non-negative parameter, it will be truncated at 0 to be non-negative.

	Intercept `prior_intercept`	**Treatment effects** `prior_trt`	**Heterogeneity** `prior_`
Normal `normal()`	Yes	Yes	Yes
half-Normal `half_normal()`	-	-	Yes
log-Normal `log_normal()`	-	-	Yes

Cauchy `cauchy()`	Yes	Yes	Yes
half-Cauchy `half_cauchy()`	-	-	Yes
Student t `student_t()`	Yes	Yes	Yes
half-Student t `half_student_t()`	-	-	Yes
log-Student t `log_student_t()`	-	-	Yes
Exponential `exponential()`	-	-	Yes
Flat `flat()`	Yes	Yes	Yes

The `flat()` prior is a special case where no prior information is added to the model, resulting in an implicit flat uniform prior distribution over the entire support for a parameter. This will be an improper prior if the parameter is unbounded, and is not generally advised. See the Stan user's guide for more details.

Value

Object of class nma_prior.

See Also

`summary.nma_prior()` for summarising details of prior distributions. `plot_prior_posterior()` for plots comparing the prior and posterior distributions of model parameters.

`qbern` *The Bernoulli Distribution*

Description

The density function `dbern()`, distribution function `pbern()`, and quantile function `qbern()` for a Bernoulli distribution, with success probability `prob`. These are equivalent to `dbinom(p, 1, prob)`, `pbinom(p, 1, prob)` and `qbinom(p, 1, prob)`.

Usage

```
qbern(p, prob, lower.tail = TRUE, log.p = FALSE)

pbern(q, prob, lower.tail = TRUE, log.p = FALSE)

dbern(x, prob, log = FALSE)
```

Arguments

`p`	vector of probabilities
`prob`	probability of success
`lower.tail, log.p, log`	see stats::Binomial
`x, q`	vector of quantiles

Value

Numeric vector of length equal to the maximum of the lengths of the input arguments.

`qgamma` *The Gamma distribution*

Description

We provide convenient extensions of the `[dpq]gamma` functions, which allow the distribution to be specified in terms of its mean and standard deviation, instead of shape and rate/scale.

Usage

```
qgamma(
  p,
  shape,
  rate = 1,
  scale = 1/rate,
  lower.tail = TRUE,
  log.p = FALSE,
  mean,
  sd
)

dgamma(x, shape, rate = 1, scale = 1/rate, log = FALSE, mean, sd)

pgamma(
  q,
  shape,
  rate = 1,
  scale = 1/rate,
  lower.tail = TRUE,
  log.p = FALSE,
  mean,
  sd
)
```

Arguments

`p`	vector of probabilities
`shape, rate, scale, log, lower.tail, log.p`	see stats::GammaDist
`mean, sd`	mean and standard deviation, overriding shape and `rate` or `scale` if specified
`x, q`	vector of quantiles

Value

Numeric vector of length equal to the maximum of the lengths of the input arguments.

---

qlogitnorm *The logit Normal distribution*

Description

We provide convenient extensions of the `[dpq]logitnorm` functions in the package logitnorm, which allow the distribution to be specified in terms of its mean and standard deviation, instead of its logit-mean and logit-sd.

Usage

```
qlogitnorm(p, mu = 0, sigma = 1, ..., mean, sd)

dlogitnorm(x, mu = 0, sigma = 1, ..., mean, sd)

plogitnorm(q, mu = 0, sigma = 1, ..., mean, sd)
```

Arguments

`p, x`	vector of quantiles
`mu, sigma, ...`	see `logitnorm`
`mean, sd`	mean and standard deviation, overriding `mu` and `sigma` if specified
`q`	vector of probabilities

Value

Numeric vector of length equal to the maximum of the lengths of the input arguments.

---

relative_effects *Relative treatment effects*

Description

Generate (population-average) relative treatment effects. If a ML-NMR or meta-regression model was fitted, these are specific to each study population.

Usage

```
relative_effects(
  x,
  newdata = NULL,
  study = NULL,
  all_contrasts = FALSE,
  trt_ref = NULL,
  probs = c(0.025, 0.25, 0.5, 0.75, 0.975),
  predictive_distribution = FALSE,
  summary = TRUE
)
```

Arguments

`x`	A `stan_nma` object created by `nma()`
`newdata`	Only used if a regression model is fitted. A data frame of study details, one row per study, giving the covariate values at which to produce relative effects. Column names must match variables in the regression model. If `NULL`, relative effects are produced for all studies in the network.
`study`	Column of `newdata` which specifies study names, otherwise studies will be labelled by row number.
`all_contrasts`	Logical, generate estimates for all contrasts (`TRUE`), or just the "basic" contrasts against the network reference treatment (`FALSE`)? Default `FALSE`.
`trt_ref`	Reference treatment to construct relative effects against, if `all_contrasts = FALSE`. By default, relative effects will be against the network reference treatment. Coerced to character string.
`probs`	Numeric vector of quantiles of interest to present in computed summary, default `c(0.025, 0.25, 0.5, 0.75, 0.975)`
`predictive_distribution`	Logical, when a random effects model has been fitted, should the predictive distribution for relative effects in a new study be returned? Default `FALSE`.
`summary`	Logical, calculate posterior summaries? Default `TRUE`.

Value

A nma_summary object if `summary = TRUE`, otherwise a list containing a 3D MCMC array of samples and (for regression models) a data frame of study information.

See Also

`plot.nma_summary()` for plotting the relative effects.

Examples

```
## Smoking cessation

# Run smoking RE NMA example if not already available
```

```
if (!exists("smk_fit_RE")) example("example_smk_re", run.donttest = TRUE)


# Produce relative effects
smk_releff_RE <- relative_effects(smk_fit_RE)
smk_releff_RE
plot(smk_releff_RE, ref_line = 0)

# Relative effects for all pairwise comparisons
relative_effects(smk_fit_RE, all_contrasts = TRUE)

# Relative effects against a different reference treatment
relative_effects(smk_fit_RE, trt_ref = "Self-help")

# Transforming to odds ratios
# We work with the array of relative effects samples
LOR_array <- as.array(smk_releff_RE)
OR_array <- exp(LOR_array)

# mcmc_array objects can be summarised to produce a nma_summary object
smk_OR_RE <- summary(OR_array)

# This can then be printed or plotted
smk_OR_RE
plot(smk_OR_RE, ref_line = 1)


## Plaque psoriasis ML-NMR

# Run plaque psoriasis ML-NMR example if not already available
if (!exists("pso_fit")) example("example_pso_mlnmr", run.donttest = TRUE)


# Produce population-adjusted relative effects for all study populations in
# the network
pso_releff <- relative_effects(pso_fit)
pso_releff
plot(pso_releff, ref_line = 0)

# Produce population-adjusted relative effects for a different target
# population
new_agd_means <- data.frame(
  bsa = 0.6,
  prevsys = 0.1,
  psa = 0.2,
  weight = 10,
  durnpso = 3)

relative_effects(pso_fit, newdata = new_agd_means)
```

RE_cor *Random effects structure*

Description

Use RE_cor to generate the random effects correlation matrix, under the assumption of common heterogeneity variance (i.e. all within-study correlations are 0.5). Use which_RE to return a vector of IDs for the RE deltas (0 means no RE delta on this arm).

Usage

```
RE_cor(study, trt, contrast, type = c("reftrt", "blshift"))

which_RE(study, trt, contrast, type = c("reftrt", "blshift"))
```

Arguments

study	A vector of study IDs (integer, character, or factor)
trt	A factor vector of treatment codes (or coercible as such), with first level indicating the reference treatment
contrast	A logical vector, of the same length as study and trt, indicating whether the corresponding data are in contrast rather than arm format.
type	Character string, whether to generate RE structure under the "reference treatment" parameterisation, or the "baseline shift" parameterisation.

Value

For RE_cor(), a correlation matrix of dimension equal to the number of random effects deltas (excluding those that are set equal to zero).

For which_RE(), an integer vector of IDs indexing the rows and columns of the correlation matrix returned by RE_cor().

Examples

```
RE_cor(smoking$studyn, smoking$trtn, contrast = rep(FALSE, nrow(smoking)))
RE_cor(smoking$studyn, smoking$trtn, contrast = rep(FALSE, nrow(smoking)), type = "blshift")
which_RE(smoking$studyn, smoking$trtn, contrast = rep(FALSE, nrow(smoking)))
which_RE(smoking$studyn, smoking$trtn, contrast = rep(FALSE, nrow(smoking)), type = "blshift")
```

set_agd_arm *Set up arm-based aggregate data*

Description

Set up a network containing arm-based aggregate data (AgD), such as event counts or mean outcomes on each arm. Multiple data sources may be combined once created using `combine_network()`.

Usage

```
set_agd_arm(
  data,
  study,
  trt,
  y = NULL,
  se = NULL,
  r = NULL,
  n = NULL,
  E = NULL,
  sample_size = NULL,
  trt_ref = NULL,
  trt_class = NULL
)
```

Arguments

`data`	a data frame
`study`	column of `data` specifying the studies, coded using integers, strings, or factors
`trt`	column of `data` specifying treatments, coded using integers, strings, or factors
`y`	column of `data` specifying a continuous outcome
`se`	column of `data` specifying the standard error for a continuous outcome
`r`	column of `data` specifying a binary or Binomial outcome count
`n`	column of `data` specifying Binomial outcome numerator
`E`	column of `data` specifying the total time at risk for Poisson outcomes
`sample_size`	column of `data` giving the sample size in each arm. Optional, see details.
`trt_ref`	reference treatment for the network, as a single integer, string, or factor. If not specified, a reasonable well-connected default will be chosen (see details).
`trt_class`	column of `data` specifying treatment classes, coded using integers, strings, or factors. By default, no classes are specified.

Details

By default, `trt_ref = NULL` and a network reference treatment will be chosen that attempts to maximise computational efficiency and stability. If an alternative reference treatment is chosen and the model runs slowly or has low effective sample size (ESS) this may be the cause - try letting the default reference treatment be used instead. Regardless of which treatment is used as the network reference at the model fitting stage, results can be transformed afterwards: see the `trt_ref` argument of `relative_effects()` and `predict.stan_nma()`.

The `sample_size` argument is optional, but when specified:

- Enables automatic centering of predictors (`center = TRUE`) in `nma()` when a regression model is given for a network combining IPD and AgD
- Enables production of study-specific relative effects, rank probabilities, etc. for studies in the network when a regression model is given
- Nodes in `plot.nma_data()` may be weighted by sample size

If a Binomial outcome is specified and `sample_size` is omitted, `n` will be used as the sample size by default. If a Multinomial outcome is specified and `sample_size` is omitted, the sample size will be determined automatically from the supplied counts by default.

All arguments specifying columns of `data` accept the following:

- A column name as a character string, e.g. `study = "studyc"`
- A bare column name, e.g. `study = studyc`
- `dplyr::mutate()` style semantics for inline variable transformations, e.g. `study = paste(author, year)`

Value

An object of class nma_data

See Also

`set_ipd()` for individual patient data, `set_agd_contrast()` for contrast-based aggregate data, and `combine_network()` for combining several data sources in one network.

`print.nma_data()` for the print method displaying details of the network, and `plot.nma_data()` for network plots.

Examples

```
# Set up network of smoking cessation data
head(smoking)

smk_net <- set_agd_arm(smoking,
                       study = studyn,
                       trt = trtc,
                       r = r,
                       n = n,
                       trt_ref = "No intervention")
```

```
# Print details
smk_net


# Plot network
plot(smk_net)
```

---

set_agd_contrast *Set up contrast-based aggregate data*

Description

Set up a network containing contrast-based aggregate data (AgD), i.e. summaries of relative effects between treatments such as log Odds Ratios. Multiple data sources may be combined once created using `combine_network()`.

Usage

```
set_agd_contrast(
  data,
  study,
  trt,
  y = NULL,
  se = NULL,
  sample_size = NULL,
  trt_ref = NULL,
  trt_class = NULL
)
```

Arguments

`data`	a data frame
`study`	column of `data` specifying the studies, coded using integers, strings, or factors
`trt`	column of `data` specifying treatments, coded using integers, strings, or factors
`y`	column of `data` specifying a continuous outcome
`se`	column of `data` specifying the standard error for a continuous outcome
`sample_size`	column of `data` giving the sample size in each arm. Optional, see details.
`trt_ref`	reference treatment for the network, as a single integer, string, or factor. If not specified, a reasonable well-connected default will be chosen (see details).
`trt_class`	column of `data` specifying treatment classes, coded using integers, strings, or factors. By default, no classes are specified.

Details

Each study should have a single reference/baseline treatment, against which relative effects in the other arm(s) are given. For the reference arm, include a data row with continuous outcome `y` equal to `NA`. If a study has three or more arms (so two or more relative effects), set the standard error `se` for the reference arm data row equal to the standard error of the mean outcome on the reference arm (this determines the covariance of the relative effects, when expressed as differences in mean outcomes between arms).

All arguments specifying columns of `data` accept the following:

- A column name as a character string, e.g. `study = "studyc"`
- A bare column name, e.g. `study = studyc`
- `dplyr::mutate()` style semantics for inline variable transformations, e.g. `study = paste(author, year)`

By default, `trt_ref = NULL` and a network reference treatment will be chosen that attempts to maximise computational efficiency and stability. If an alternative reference treatment is chosen and the model runs slowly or has low effective sample size (ESS) this may be the cause - try letting the default reference treatment be used instead. Regardless of which treatment is used as the network reference at the model fitting stage, results can be transformed afterwards: see the `trt_ref` argument of `relative_effects()` and `predict.stan_nma()`.

The `sample_size` argument is optional, but when specified:

- Enables automatic centering of predictors (`center = TRUE`) in `nma()` when a regression model is given for a network combining IPD and AgD
- Enables production of study-specific relative effects, rank probabilities, etc. for studies in the network when a regression model is given
- Nodes in `plot.nma_data()` may be weighted by sample size

Value

An object of class nma_data

See Also

`set_ipd()` for individual patient data, `set_agd_arm()` for arm-based aggregate data, and `combine_network()` for combining several data sources in one network.

`print.nma_data()` for the print method displaying details of the network, and `plot.nma_data()` for network plots.

Examples

```
# Set up network of Parkinson's contrast data
head(parkinsons)

park_net <- set_agd_contrast(parkinsons,
                             study = studyn,
                             trt = trtn,
                             y = diff,
                                se = se_diff,
                                sample_size = n)

# Print details
park_net

# Plot network
plot(park_net)
```

set_agd_surv *Set up aggregate survival data*

Description

Set up a network containing aggregate survival data (AgD) in the form of event/censoring times (e.g. reconstructed from digitized Kaplan-Meier curves) and covariate summary statistics from each study. Multiple data sources may be combined once created using `combine_network()`.

Usage

```
set_agd_surv(
  data,
  study,
  trt,
  Surv,
  covariates = NULL,
  trt_ref = NULL,
  trt_class = NULL
)
```

Arguments

`data`	a data frame
`study`	column of `data` specifying the studies, coded using integers, strings, or factors
`trt`	column of `data` specifying treatments, coded using integers, strings, or factors
`Surv`	column of `data` specifying a survival or time-to-event outcome, using the `Surv()` function. Right/left/interval censoring and left truncation (delayed entry) are supported.
`covariates`	data frame of covariate summary statistics for each study or study arm, with corresponding `study` and `trt` columns to match to those in `data`
`trt_ref`	reference treatment for the network, as a single integer, string, or factor. If not specified, a reasonable well-connected default will be chosen (see details).
`trt_class`	column of `data` specifying treatment classes, coded using integers, strings, or factors. By default, no classes are specified.

Details

By default, `trt_ref = NULL` and a network reference treatment will be chosen that attempts to maximise computational efficiency and stability. If an alternative reference treatment is chosen and the model runs slowly or has low effective sample size (ESS) this may be the cause - try letting the default reference treatment be used instead. Regardless of which treatment is used as the network reference at the model fitting stage, results can be transformed afterwards: see the `trt_ref` argument of `relative_effects()` and `predict.stan_nma()`.

All arguments specifying columns of `data` accept the following:

- A column name as a character string, e.g. `study = "studyc"`
- A bare column name, e.g. `study = studyc`
- `dplyr::mutate()` style semantics for inline variable transformations, e.g. `study = paste(author, year)`

Value

An object of class nma_data

See Also

`set_ipd()` for individual patient data, `set_agd_contrast()` for contrast-based aggregate data, and `combine_network()` for combining several data sources in one network.

`print.nma_data()` for the print method displaying details of the network, and `plot.nma_data()` for network plots.

Examples

```
## Newly diagnosed multiple myeloma

head(ndmm_agd)  # Reconstructed Kaplan-Meier data
ndmm_agd_covs   # Summary covariate information on each arm

set_agd_surv(ndmm_agd,
             study = studyf,
             trt = trtf,
             Surv = Surv(eventtime, status),
             covariates = ndmm_agd_covs)
```

`set_ipd` *Set up individual patient data*

Description

Set up a network containing individual patient data (IPD). Multiple data sources may be combined once created using `combine_network()`.

Usage

```
set_ipd(
  data,
  study,
  trt,
  y = NULL,
  r = NULL,
  E = NULL,
  Surv = NULL,
  trt_ref = NULL,
  trt_class = NULL
)
```

Arguments

`data`	a data frame
`study`	column of `data` specifying the studies, coded using integers, strings, or factors
`trt`	column of `data` specifying treatments, coded using integers, strings, or factors
`y`	column of `data` specifying a continuous outcome
`r`	column of `data` specifying a binary outcome or Poisson outcome count
`E`	column of `data` specifying the total time at risk for Poisson outcomes
`Surv`	column of `data` specifying a survival or time-to-event outcome, using the `Surv()` function. Right/left/interval censoring and left truncation (delayed entry) are supported.
`trt_ref`	reference treatment for the network, as a single integer, string, or factor. If not specified, a reasonable well-connected default will be chosen (see details).
`trt_class`	column of `data` specifying treatment classes, coded using integers, strings, or factors. By default, no classes are specified.

Details

By default, `trt_ref = NULL` and a network reference treatment will be chosen that attempts to maximise computational efficiency and stability. If an alternative reference treatment is chosen and the model runs slowly or has low effective sample size (ESS) this may be the cause - try letting the default reference treatment be used instead. Regardless of which treatment is used as the network reference at the model fitting stage, results can be transformed afterwards: see the `trt_ref` argument of `relative_effects()` and `predict.stan_nma()`.

All arguments specifying columns of `data` accept the following:

- A column name as a character string, e.g. `study = "studyc"`
- A bare column name, e.g. `study = studyc`
- `dplyr::mutate()` style semantics for inline variable transformations, e.g. `study = paste(author, year)`

Value

An object of class nma_data

See Also

set_agd_arm() for arm-based aggregate data, set_agd_contrast() for contrast-based aggregate data, and combine_network() for combining several data sources in one network.

print.nma_data() for the print method displaying details of the network, and plot.nma_data() for network plots.

Examples

```
# Set up network of plaque psoriasis IPD
head(plaque_psoriasis_ipd)

pso_net <- set_ipd(plaque_psoriasis_ipd,
                   study = studyc,
                   trt = trtc,
                   r = pasi75)

# Print network details
pso_net

# Plot network
plot(pso_net)

# Setting a different reference treatment
set_ipd(plaque_psoriasis_ipd,
        study = studyc,
        trt = trtc,
        r = pasi75,
        trt_ref = "PBO")
```

---

smoking *Smoking cessation data*

Description

Data frame containing the results of 24 trials of 4 smoking cessation treatments (Hasselblad 1998; Dias et al. 2011).

Usage

```
smoking
```

Format

A data frame with 50 rows and 5 variables:

studyn numeric study ID

trtn numeric treatment code

trtc treatment name

r total number of events

n total number of individuals

References

Dias S, Welton NJ, Sutton AJ, Caldwell DM, Lu G, Ades AE (2011). "NICE DSU Technical Support Document 4: Inconsistency in networks of evidence based on randomised controlled trials." National Institute for Health and Care Excellence. `https://www.sheffield.ac.uk/nice-dsu`.

Hasselblad V (1998). "Meta-analysis of Multitreatment Studies." *Medical Decision Making*, **18**(1), 37–43. doi:10.1177/0272989x9801800110.

`stan_nma-class` *The stan_nma class*

Description

The `stan_nma` and `stan_mlnmr` classes contains the results from running a model with the function `nma()`.

Details

Objects of class `stan_nma` and `stan_mlnmr` have the following components:

`network` The network data from which the model was run (class nma_data for `stan_nma`, or class mlnmr_data for `stan_mlnmr`)

`stanfit` The `stanfit` object returned by calling `sampling()` for the model

`trt_effects` Whether fixed or random effects were used (character string)

`consistency` The consistency/inconsistency model used (character string)

`regression` The regression model used (formula)

`class_interactions` If treatment classes and a regression model are specified, the model used for interactions within each class (common, exchangeable, or independent)

`xbar` A named vector of values used for centering

`likelihood` The likelihood used (character string)

`link` The link function used (character string)

`priors` A list containing the priors used (as nma_prior objects)

`basis` For `mspline` and `pexp` models, a named list of spline bases for each study

The `stan_mlnmr` sub-class inherits from `stan_nma`, and differs only in the class of the `network` object.

`statins` *Statins for cholesterol lowering*

Description

Data frame containing the results of 19 trials comparing statins to placebo or usual care (Dias et al. 2011). The number of deaths (all-cause mortality) are recorded. In some studies the aim was primary prevention (patients had no previous heart disease), and in others the aim was secondary prevention (patients had previous heart disease).

Usage

```
statins
```

Format

A data frame with 38 rows and 7 variables:

studyn numeric study ID

studyc study name

trtn numeric treatment code

trtc treatment name

prevention primary or secondary prevention study

r number of deaths

n sample size

References

Dias S, Sutton AJ, Welton NJ, Ades AE (2011). "NICE DSU Technical Support Document 3: Heterogeneity: subgroups, meta-regression, bias and bias-adjustment." National Institute for Health and Care Excellence. https://www.sheffield.ac.uk/nice-dsu.

`summary.nma_nodesplit_df` *Summarise the results of node-splitting models*

Description

Posterior summaries of node-splitting models (`nma_nodesplit` and `nma_nodesplit_df` objects) can be produced using the `summary()` method, and plotted using the `plot()` method.

Usage

```
## S3 method for class 'nma_nodesplit_df'
summary(
  object,
  consistency = NULL,
  ...,
  probs = c(0.025, 0.25, 0.5, 0.75, 0.975)
)

## S3 method for class 'nma_nodesplit'
summary(
  object,
  consistency = NULL,
  ...,
  probs = c(0.025, 0.25, 0.5, 0.75, 0.975)
)

## S3 method for class 'nma_nodesplit'
plot(x, consistency = NULL, ...)

## S3 method for class 'nma_nodesplit_df'
plot(x, consistency = NULL, ...)
```

Arguments

`consistency`	Optional, a `stan_nma` object for the corresponding fitted consistency model, to display the network estimates alongside the direct and indirect estimates. The fitted consistency model present in the `nma_nodesplit_df` object will be used if this is present (see `get_nodesplits()`).
`...`	Additional arguments passed on to other methods
`probs`	Numeric vector of specifying quantiles of interest, default `c(0.025, 0.25, 0.5, 0.75, 0.975)`
`x, object`	A `nma_nodesplit` or `nma_nodesplit_df` object

Details

The `plot()` method is a shortcut for `plot(summary(nma_nodesplit))`. For details of plotting options, see `plot.nodesplit_summary()`.

Value

A nodesplit_summary object

See Also

`plot.nodesplit_summary()`

Examples

```
# Run smoking node-splitting example if not already available
if (!exists("smk_fit_RE_nodesplit")) example("example_smk_nodesplit", run.donttest = TRUE)


# Summarise the node-splitting results
summary(smk_fit_RE_nodesplit)

# Plot the node-splitting results
plot(smk_fit_RE_nodesplit)
```

---

summary.nma_prior *Summary of prior distributions*

Description

Print a summary of prior distribution details.

Usage

```
## S3 method for class 'nma_prior'
summary(object, ..., probs = c(0.5, 0.95), digits = 2, trunc = NULL)
```

Arguments

object	Prior distribution as a `nma_prior` object
...	Additional arguments, not used
probs	Numeric vector of probabilities to calculate prior intervals
digits	Number of digits to display
trunc	Optional numeric vector of length 2, giving the truncation limits of the prior distribution. Useful if a real-valued prior is assigned to a positive-valued parameter, then `trunc = c(0, Inf)` will give the correct prior intervals. By default, truncation is not used.

Value

A data frame is returned invisibly, giving the prior intervals

Examples

```
summary(normal(location = 0, scale = 1))
summary(half_normal(scale = 1))
summary(log_normal(location = -3.93, scale = 1.51))

# Truncation limits may be set, for example to restrict a prior to positive values
summary(normal(location = 0.5, scale = 1), trunc = c(0, Inf))
```

summary.stan_nma *Posterior summaries from stan_nma objects*

Description

Posterior summaries of model parameters in stan_nma objects may be produced using the summary() method and plotted with the plot() method. NOTE: To produce relative effects, absolute predictions, or posterior ranks, see relative_effects(), predict.stan_nma(), posterior_ranks(), posterior_rank_probs().

Usage

```
## S3 method for class 'stan_nma'
summary(object, ..., pars, include, probs = c(0.025, 0.25, 0.5, 0.75, 0.975))

## S3 method for class 'stan_nma'
plot(
  x,
  ...,
  pars,
  include,
  stat = "pointinterval",
  orientation = c("horizontal", "vertical", "y", "x"),
  ref_line = NA_real_
)
```

Arguments

...	Additional arguments passed on to other methods
pars, include	See rstan::extract()
probs	Numeric vector of specifying quantiles of interest, default c(0.025, 0.25, 0.5, 0.75, 0.975)
x, object	A stan_nma object
stat	Character string specifying the ggdist plot stat to use, default "pointinterval"
orientation	Whether the ggdist geom is drawn horizontally ("horizontal") or vertically ("vertical"), default "horizontal"
ref_line	Numeric vector of positions for reference lines, by default no reference lines are drawn
summary	Logical, calculate posterior summaries? Default TRUE.

Details

The plot() method is a shortcut for plot(summary(stan_nma)). For details of plotting options, see plot.nma_summary().

Value

A nma_summary object

See Also

`plot.nma_summary()`, `relative_effects()`, `predict.stan_nma()`, `posterior_ranks()`, `posterior_rank_probs()`

Examples

```
## Smoking cessation

# Run smoking RE NMA example if not already available
if (!exists("smk_fit_RE")) example("example_smk_re", run.donttest = TRUE)


# Summary and plot of all model parameters
summary(smk_fit_RE)
plot(smk_fit_RE)

# Summary and plot of heterogeneity tau only
summary(smk_fit_RE, pars = "tau")
plot(smk_fit_RE, pars = "tau")

# Customising plot output
plot(smk_fit_RE,
     pars = c("d", "tau"),
     stat = "halfeye",
     ref_line = 0)
```

`theme_multinma` *Plot theme for multinma plots*

Description

A simple `ggplot2` theme for plots in the `multinma` package.

Usage

```
theme_multinma(...)
```

Arguments

`...`	Arguments passed to `ggplot2::theme_light()`

Value

A `ggplot2` theme

See Also

`ggplot2::theme()`, `ggplot2::theme_set()`

Examples

```
library(ggplot2)
theme_set(theme_multinma())
```

`thrombolytics` *Thrombolytic treatments data*

Description

Data frame containing the results of 50 trials of 8 thrombolytic drugs (streptokinase, SK; alteplase, t-PA; accelerated alteplase, Acc t-PA; streptokinase plus alteplase, SK+tPA; reteplase, r-PA; tenocteplase, TNK; urokinase, UK; anistreptilase, ASPAC) plus per-cutaneous transluminal coronary angioplasty (PTCA) (Boland et al. 2003; Lu and Ades 2006; Dias et al. 2011). The number of deaths in 30 or 35 days following acute myocardial infarction are recorded.

Usage

```
thrombolytics
```

Format

A data frame with 102 rows and 5 variables:

studyn numeric study ID

trtn numeric treatment code

trtc treatment name

r total number of events

n total number of individuals

References

Boland A, Dundar Y, Bagust A, Haycox A, Hill R, Mota RM, Walley T, Dickson R (2003). “Early thrombolysis for the treatment of acute myocardial infarction: a systematic review and economic evaluation.” *Health Technology Assessment*, **7**(15). doi:10.3310/hta7150.

Dias S, Welton NJ, Sutton AJ, Caldwell DM, Lu G, Ades AE (2011). “NICE DSU Technical Support Document 4: Inconsistency in networks of evidence based on randomised controlled trials.” National Institute for Health and Care Excellence. `https://www.sheffield.ac.uk/nice-dsu`.

Lu GB, Ades AE (2006). “Assessing evidence inconsistency in mixed treatment comparisons.” *Journal of the American Statistical Association*, **101**(474), 447–459. doi:10.1198/016214505000001302.

`transfusion` *Granulocyte transfusion in patients with neutropenia or neutrophil dysfunction*

Description

Data frame containing the number of deaths in 6 trials comparing transfusion of granulocytes (white blood cells) to control (Stanworth et al. 2005). Previously used to demonstrate informative prior distributions for the heterogeneity variance by Turner et al. (2012).

Usage

```
transfusion
```

Format

A data frame with 12 rows and 4 variables:

studyc study name

trtc treatment name

r total number of deaths

n total number of individuals

References

Stanworth S, Massey E, Hyde C, Brunskill SJ, Navarette C, Lucas G, Marks D, Paulus U (2005). "Granulocyte transfusions for treating infections in patients with neutropenia or neutrophil dysfunction." *Cochrane Database of Systematic Reviews*. ISSN 1465-1858, doi:10.1002/14651858.CD005339.

Turner RM, Davey J, Clarke MJ, Thompson SG, Higgins JPT (2012). "Predicting the extent of heterogeneity in meta-analysis, using empirical data from the Cochrane Database of Systematic Reviews." *International Journal of Epidemiology*, **41**(3), 818–827. doi:10.1093/ije/dys041.

Index