Package ‘multinma’

March 18, 2021

Title Bayesian Network Meta-Analysis of Individual and Aggregate Data

Version 0.3.0

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/rssb.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, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (> 2.19.2), rstantools (>= 2.0.0), Rdpack (>= 0.7), tibble (>= 2.1.3), dplyr (>= 0.8.3), rlang, purrr, forcats, glue, randtoolbox, copula, tidyr (>= 1.0.0), stringr, Matrix, igraph, ggplot2, 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 (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements GNU make

RoxygenNote 7.1.1

Suggests testthat (>= 2.1.0), knitr, rmarkdown, parallel, R.rsp, rprojroot, loo (>= 2.0.0), logitnorm, crayon, tidygraph, pkgdown

RdMacros Rdpack


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**  2021-03-18 14:00:03 UTC

---

## R topics documented:

- `multinma-package` .......................... 3
  - `.default` ................................. 5
  - `adapt_delta` ............................. 5
  - `add_integration` .......................... 6
  - `as.array.stan_nma` ...................... 9
  - `as.igraph.nma_data` ..................... 10
  - `as.stanfit` .............................. 11
  - `atrial_fibrillation` ................... 11
  - `bcg_vaccine` ............................ 12
  - `blocker` ................................ 13
  - `combine_network` ....................... 14
  - `dgent` .................................. 16
  - `diabetes` ................................ 17
  - `dic` .................................... 18
  - `dietary_fat` ............................. 19
  - `distr` ................................... 20
  - `example_pso_mlnmr` ...................... 21
  - `example_smk_fe` ......................... 23
  - `example_smk_re` ......................... 24
  - `example_smk_ume` ....................... 24
  - `hta_psoriasis` ........................... 25
  - `is_network_connected` .................. 26
  - `loo.stan_nma` ............................ 27
  - `mcmc_array-class` ...................... 28
  - `multi` ................................... 29
  - `nma` ..................................... 31
  - `nma_data-class` .......................... 37
  - `nma_dic-class` ........................... 38
  - `nma_prior-class` ....................... 39
  - `nma_summary-class` ..................... 39
  - `pairs.stan_nma` ......................... 40
  - `parkinsons` ............................. 41
  - `plaque_psoriasis_ipd` .................. 42
  - `plot.nma_data` ........................... 43
  - `plot.nma_dic` ............................ 45
  - `plot.nma_summary` ....................... 47
  - `plot_integration_error` ............... 50
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
multinma-package

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:


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


References


**.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

```r
.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/](https://mc-stan.org/users/documentation/).
add_integration

Add numerical integration points to aggregate data

Description

The add_integration() generic creates numerical integration points using a Gaussian copula approach, 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() un nests 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, n_int = 1000L, int_args = list())

## S3 method for class 'nma_data'
add_integration(x, ..., cor = NULL, n_int = 1000L, 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 ....
n_int Number of integration points to generate, default 1000
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().
add_integration

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


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 ~ "TNFα 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 = 1000)

## 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,
as.array.stan_nma

```r
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 = 1000)
```

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

```r
## 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.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
```r
## 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
```r
# 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

```r
tidygraph::as_tbl_graph(smk_net)  # Edges combined by default
tidygraph::as_tbl_graph(smk_net, collapse = FALSE)  # Without combining edges
```

---

```
\section*{as.stanfit}
\section*{as.stanfit}
```

---

\section*{Description}

Attempt to turn an object into a \texttt{stanfit} object.

\section*{Usage}

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

\section*{Arguments}

\begin{itemize}
\item \texttt{x} \hspace{1cm} an object
\item \texttt{...} \hspace{1cm} additional arguments
\end{itemize}

\section*{Value}

A \texttt{stanfit} object.

---

```
\section*{atrial_fibrillation}
\textit{Stroke prevention in atrial fibrillation patients}
```

---

\section*{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).

\section*{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


---

**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


---

**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


---

**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**

```r
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**

```r
## 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,
                          ref = 'placebo')
```
```r
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 ~ "TNFα 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


Deviance Information Criterion (DIC)

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

Usage
```
dic(x, ...)
```

Arguments
- `x` A fitted model object, inheriting class `stan_nma`
- `...` 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
```r
## 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

```r
smk_dic_RE
(smk_dic_RE_UME <- dic(smk_fit_RE_UME)) # no difference in fit
```

# Compare residual deviance contributions

```r
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

```r
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


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

```r
## 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))
```
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_rma` object `pso_fit` will then be available in the global environment.

Details

Plaque psoriasis ML-NMR for use in examples.

Examples

```r
# 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 ~ "TNFα 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 ~ "TNFα 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 = 1000)

# 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

---

### 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

```r
# 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_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

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 39 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 $50 < \text{PASI} < 75$, $75 < \text{PASI} < 90$, and $90 < \text{PASI} < 100$, and are named by the lower end of the interval. (As opposed to "inclusive" counts, which would represent the overlapping categories $\text{PASI} > 50$, $\text{PASI} > 70$, and $\text{PASI} > 90$.) The count for the fourth category (the lowest), $0 < \text{PASI} < 50$, is equal to $\text{sample_size} - \text{PASI50} - \text{PASI75} - \text{PASI90}$.

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 $50 < \text{PASI} < 75$ and $75 < \text{PASI} < 100$. For a study reporting only one outcome, say PASI 75, the count represents $75 < \text{PASI} < 100$.

References


---

**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)

Description

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

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()

---

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'
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.
## Smoking cessation

```r
# 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)
```

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

```r
# 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**

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 ordered = TRUE. 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

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"),
  trt_effects = c("fixed", "random"),
  regression = NULL,
  class_interactions = c("common", "exchangeable", "independent"),
  likelihood = NULL,
  link = NULL,
  ...
  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(),
  QR = FALSE,
  center = TRUE,
  adapt_delta = NULL,
  int_thin = max(network$n_int/10, 1)
)
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, currently either "consistency" or "ume".
- **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.
- **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 $\tau$ ("sd", the default), variance $\tau^2$ ("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).
- **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.

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.
nma 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.

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, nma.fit() returns a stanfit object.

Likelihoods and link functions

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

<table>
<thead>
<tr>
<th>Data type</th>
<th>Likelihood</th>
<th>Link function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>bernoulli, bernoulli2</td>
<td>logit, probit, cloglog</td>
</tr>
<tr>
<td>Count</td>
<td>binomial, binomial2</td>
<td>logit, probit, cloglog</td>
</tr>
<tr>
<td>Rate</td>
<td>poisson</td>
<td>log</td>
</tr>
<tr>
<td>Continuous</td>
<td>normal</td>
<td>identity, log</td>
</tr>
<tr>
<td>Ordered</td>
<td>ordered</td>
<td>logit, probit, cloglog</td>
</tr>
</tbody>
</table>

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).

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 $\sigma_{jk}$ 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, $c_0 < c_1 < \ldots < c_{M}$. 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.

References


Examples

```r
## 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

## 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(trt == 1 ~ "Placebo",
                                trt %in% c(2, 3, 5, 6) ~ "IL blocker",
                                trt == 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(trt == 1 ~ "Placebo",
                                trt %in% c(2, 3, 5, 6) ~ "IL blocker",...
trtn == 4 ~ "TNFa blocker")

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

psp_net <- combine_network(
  set_ipd(psp_ipd,
    study = studyc,
    trt = trtc,
    r = pasi75,
    trt_class = trtclass),
  set_agd_arm(psp_agd,
    study = studyc,
    trt = trtc,
    r = pasi75_r,
    n = pasi75_n,
    trt_class = trtclass)
)

# Print network details
psp_net

# Add integration points to the network
psp_net <- add_integration(psp_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 = 1000)

# 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.
psp_fit <- nma(psp_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)
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(), 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 event/censoring time, of type Surv (time-to-event)
- .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
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` 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()`. 
The `nma_prior` class is used to specify prior distributions.

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`).

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()`.

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

Objects of class `nma_summary` have the following components:
- `summary` A data frame containing the computed summary statistics. 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 a column named `.study`.
- `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()`.
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**

```r
## 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**

```r
## 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
```
```r
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 Parkinson’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

```r
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


---

**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**


---

### Description

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

Usage

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

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.

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)

# 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

```r
## 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


Examples

```r
# 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", 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 = 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 = 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

```r
## S3 method for class 'nma_summary'
plot(
  x,
  ...,  # Additional arguments passed on to the underlying ggdist plot stat, see Details
  stat = "pointinterval",  # Character string specifying the ggdist plot stat to use, default "pointinterval"
  orientation = c("horizontal", "vertical", "y", "x"),  # Whether the ggdist geom is drawn horizontally ("horizontal") or vertically ("vertical"), default "horizontal"
  ref_line = NA_real_  # Numeric vector of positions for reference lines, by default no reference lines are drawn
)

## S3 method for class 'nma_parameter_summary'
plot(
  x,
  ...,  # Additional arguments passed on to the underlying ggdist plot stat, see Details
  stat = "pointinterval",  # Character string specifying the ggdist plot stat to use, default "pointinterval"
  orientation = c("horizontal", "vertical", "y", "x"),  # Whether the ggdist geom is drawn horizontally ("horizontal") or vertically ("vertical"), default "horizontal"
  ref_line = NA_real_  # Numeric vector of positions for reference lines, by default no reference lines are drawn
)

## S3 method for class 'nma_rank_probs'
plot(x, ...)  # Additional arguments passed on to the underlying ggdist plot stat, see Details
```

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"
- `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").

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

```r
## 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 = 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_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.
plot_prior_posterior

Examples

```r
## 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)

# 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

```r
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

```r
## 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

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

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

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

```r
# 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

```r
# 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.

**Usage**

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

**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. 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 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`). 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.

  Use the `trt_ref` 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 `type = "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 `type = "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 `type`.

- **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.

- **trt_ref**: 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.

- **type**: Whether to produce predictions on the "link" scale (the default, e.g. log odds) or "response" scale (e.g. probabilities).

- **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 `type` is "individual" or "aggregate", and for all arm-based AgD studies in the network if `type` is "aggregate".

- **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).
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.

probs        Numeric vector of quantiles of interest to present in computed summary, default c(0.025, 0.25, 0.5, 0.75, 0.975)

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 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 probabilities 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 = 1000)

# 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))
print.nma_dic

Description

Print details of networks stored as nma_data objects, as created by set_ipd(), set_agd_arm(), set_agd_contrast(), 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

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.
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

```r
## 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.stan_nma  

Print stan_nma objects

Description

Print stan_nma objects

Usage

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

Arguments

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

---

priors  

Prior distributions

Description

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

Usage

```r
normal(location = 0, scale)

half_normal(scale)

log_normal(location, scale)

dauchy(location = 0, scale)

half_dauchy(scale)
```
student_t(location = 0, scale, df)

half_student_t(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.

<table>
<thead>
<tr>
<th>Prior Type</th>
<th>Intercept</th>
<th>Treatment effects</th>
<th>Heterogeneity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>half-Normal</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>log-Normal</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Cauchy</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>half-Cauchy</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Student t</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>half-Student t</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Exponential</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Flat</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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.
qbern

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 quantile function qbern for a Bernoulli distribution, with success probability prob. This is equivalent to 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

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

\[
qgamma(p, \\
\text{shape}, \\
\text{rate} = 1, \\
\text{scale} = 1/\text{rate}, \\
\text{lower.tail} = \text{TRUE}, \\
\log.p = \text{FALSE}, \\
\text{mean}, \\
\text{sd})
\]

\[
dgamma(x, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate}, \log = \text{FALSE}, \text{mean}, \text{sd})
\]

\[
pgamma(q, \\
\text{shape}, \\
\text{rate} = 1, \\
\text{scale} = 1/\text{rate}, \\
\text{lower.tail} = \text{TRUE}, \\
\log.p = \text{FALSE}, \\
\text{mean}, \\
\text{sd})
\]

Arguments

- \textbf{p} vector of probabilities
- \textbf{shape, rate, scale, log, lower.tail, log.p}
  see \texttt{stats::GammaDist}
- \textbf{mean, sd} mean and standard deviation, overriding \text{shape} and \text{rate} or \text{scale} if specified
- \textbf{x, q} vector of quantiles

\begin{center}
\textit{qlogitnorm} \hspace{2cm} The \textit{logit} Normal distribution
\end{center}

Description

We provide convenient extensions of the \texttt{[dpq]logitnorm} functions in the package \texttt{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

\begin{align*}
q\text{logitnorm}(p, \text{mu} = 0, \text{sigma} = 1, \ldots, \text{mean}, \text{sd}) \\
d\text{logitnorm}(x, \text{mu} = 0, \text{sigma} = 1, \ldots, \text{mean}, \text{sd}) \\
p\text{logitnorm}(q, \text{mu} = 0, \text{sigma} = 1, \ldots, \text{mean}, \text{sd})
\end{align*}

Arguments

\begin{itemize}
  \item \texttt{p, x} \hspace{1cm} \text{vector of quantiles}
  \item \texttt{mu, sigma, ...} \hspace{1cm} \text{see \texttt{logitnorm}}
  \item \texttt{mean, sd} \hspace{1cm} \text{mean and standard deviation, overriding mu and sigma if specified}
  \item \texttt{q} \hspace{1cm} \text{vector of probabilities}
\end{itemize}

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

\begin{verbatim}
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), 
  summary = TRUE
)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{x} \hspace{1cm} A \texttt{stan\_nma} object created by \texttt{nma()}
  \item \texttt{newdata} \hspace{1cm} 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.
  \item \texttt{study} \hspace{1cm} Column of \texttt{newdata} which specifies study names, otherwise studies will be labelled by row number.
  \item \texttt{all_contrasts} \hspace{1cm} Logical, generate estimates for all contrasts (\texttt{TRUE}), or just the "basic" contrasts against the network reference treatment (\texttt{FALSE})? Default \texttt{FALSE}.
\end{itemize}
relative_effects

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)

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.
set_agd_arm

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 \texttt{combine_network()}. 

Usage

\begin{verbatim}
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
)
\end{verbatim}

Arguments

\begin{verbatim}
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
\end{verbatim}

Value

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

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

Examples

\begin{verbatim}
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")
\end{verbatim}
\texttt{set_agd_arm}

\begin{itemize}
\item \texttt{n} \quad column of data specifying Binomial outcome numerator
\item \texttt{E} \quad column of data specifying the total time at risk for Poisson outcomes
\item \texttt{sample_size} \quad column of data giving the sample size in each arm. Optional, see details.
\item \texttt{trt_ref} \quad 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).
\item \texttt{trt_class} \quad column of data specifying treatment classes, coded using integers, strings, or factors. By default, no classes are specified.
\end{itemize}

Details

By default, \texttt{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 \texttt{trt_ref} argument of \texttt{relative_effects()} and \texttt{predict.stan_nma()}. The \texttt{sample_size} argument is optional, but when specified:

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

If a Binomial outcome is specified and \texttt{sample_size} is omitted, \texttt{n} will be used as the sample size by default. If a Multinomial outcome is specified and \texttt{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:

\begin{itemize}
\item A column name as a character string, e.g. \texttt{study = "studyc"}
\item A bare column name, e.g. \texttt{study = studyc}
\item \texttt{dplyr::mutate()} style semantics for inline variable transformations, e.g. \texttt{study = paste(author, year)}
\end{itemize}

Value

An object of class \texttt{nma_data}

See Also

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

\texttt{print.nma_data()} for the print method displaying details of the network, and \texttt{plot.nma_data()} for network plots.
Examples

```r
# 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

```r
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 \( \text{NA} \). 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 (\( \text{center} = \text{TRUE} \)) 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)


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,
    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
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)

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,
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


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)

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


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))
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**

```r
## 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
thrombolytics

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; tenecteplase, 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 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


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:

- study: study name
- trtc: treatment name
- r: total number of deaths
- n: total number of individuals

References


## Index

### datasets
- atrial_fibrillation, 11
- bcg_vaccine, 12
- blocker, 13
- diabetes, 17
- dietary_fat, 19
- hta_psoriasis, 25
- parkinsons, 41
- plaque_psoriasis_ipd, 42
- smoking, 74
- statins, 75
- thrombolytics, 79
- transfusion, 80

### examples
- example_pso_mlnmr, 21
- example_smk_fe, 23
- example_smk_re, 24
- example_smk_ume, 24
- .default, 5
- .is_default(.default), 5
- .priors, 33

- adapt_delta, 5, 32
- add_integration, 6
- add_integration(), 20, 37, 50, 56
- as.array.nma_rank_probs
  - (print.nma_summary), 60
- as.array.nma_summary
  - (print.nma_summary), 60
- as.array.stan_nma, 9
- as.array.stanfit, 9
- as.data.frame.nma_summary
  - (print.nma_summary), 60
- as.data.frame.stan_nma
  - (print.nma_summary), 60
- as.igraph.nma_data, 10
- as.matrix.nma_rank_probs
  - (print.nma_summary), 60
- as.matrix.nma_summary
  - (print.nma_summary), 60
- as.matrix.stan_nma(as.array.stan_nma), 9
- as.stanfit, 11
- as.tibble.nma_summary
  - (print.nma_summary), 60
- as_tbl_graph.nma_data
  - (as.igraph.nma_data), 10
- as_tibble.nma_summary
  - (print.nma_summary), 60
- atrial_fibrillation, 11

- bayesplot, 9
- bayesplot::mcmc_pairs(), 40
- bcg_vaccine, 12
- blocker, 13

- cauchy(priors), 61
- combine_network, 14
- combine_network(), 37, 59, 68–73

- dbern(qbern), 63
- dgamma(qgamma), 63
- dgent, 16
- diabetes, 17
- dic, 18
- dic(), 38, 45, 59
- dietary_fat, 19
- distr, 20, 20
- distr(), 6, 56
- dlogitnorm(qlogitnorm), 64

- example_pso_mlnmr, 21
- example_smk_fe, 23
- example_smk_re, 24
- example_smk_ume, 24
- exponential(priors), 61

- flat(priors), 61
- ggdist, 48
- ggdist::stat_pointinterval(), 48

---

81
ggplot2, 48, 49
ggplot2::geom_histogram, 51
ggplot2::geom_path, 51
ggplot2::theme(, 79
ggplot2::theme_light(, 78
ggplot2::theme_set(, 79
ggraph, 10
ggraph(), 44
half_cauchy (priors), 61
half_normal (priors), 61
half_student_t (priors), 61
hta_psoriasis
igraph, 10, 44
is_network_connected, 26
log_normal (priors), 61
logitnorm, 64, 65
loo (loo.stan_nma), 27
loo(), 27, 28
loo.stan_nma, 27
mcmc_array, 9, 60
mcmc_array (mcmc_array-class), 28
mcmc_array-class, 28
mlnmr_data, 75
mlnmr_data (nma_data-class), 37
mlnmr_data-class (nma_data-class), 37
model.matrix(), 32
multi, 29
multinma (multinma-package), 3
multinma-package, 3
names.mcmc_array (mcmc_array-class), 28
names<-.mcmc_array (mcmc_array-class), 28
nma, 31
nma(), 5, 18, 46, 50, 53, 55, 56, 65, 69, 71, 74
nma.data, 7, 10, 14, 44, 59, 69, 71, 73, 75
nma.data (nma_data-class), 37
nma_data-class, 37
nma_dic, 18, 45, 59
nma_dic (nma_dic-class), 38
nma_dic-class, 38
nma_prior, 63, 75
nma_prior (nma_prior-class), 39
nma_prior-class, 39
nma_rank_probs (nma_summary-class), 39
nma_summary, 28, 53, 57, 66, 78
nma_summary (nma_summary-class), 39
nma_summary-class, 39
normal (priors), 61
pairs(), 40
pairs.stan_nma, 40
parkinsons, 41
pbern (qbern), 63
pgamma (qgamma), 63
gagent (dgent), 16
plaque_psoriasis
(plaque_psoriasis_ipd), 42
plaque_psoriasis_agd
(plaque_psoriasis_ipd), 42
plogitnorm (qlogitnorm), 64
plot.nma_data, 43
plot.nma_data(), 14, 38, 69, 71, 73
plot.nma_dic, 45
plot.nma_dic(), 18, 38
plot.nma_rank_probs_summary
(plot.nma_summary), 47
plot.nma_rank_probs (plot.nma_summary), 47
plot.nma_summary, 47
plot.nma_summary(), 53, 57, 61, 66, 77, 78
plot.stan_nma (summary.stan_nma), 77
plot_integration_error, 50
plot_prior_posterior, 51
plot_prior_posterior(), 33, 63
posterior_rank_probs (posterior_ranks), 52
posterior_rank_probs(), 39, 48, 77, 78
posterior_ranks, 52
posterior_ranks(), 39, 48, 77, 78
predict.stan_nma, 55
predict.stan_nma(), 20, 48, 69, 71, 73, 77, 78
print.mcmc_array (mcmc_array-class), 28
print.mlnmr_data (print.nma_data), 58
print.nma_data, 58
print.nma_data(), 14, 38, 69, 71, 73
print.nma_dic, 59
print.nma_dic(), 18, 38
print.nma_summary, 60
print.stan_nma, 61
print.stanfit(), 61
priors, 39, 61
INDEX

qbern, 63
qgamma, 63
qgent (dgent), 16
qlogitnorm, 64

RE_cor, 67
relative_effects, 65
relative_effects(), 47, 69, 71, 73, 77, 78
round(), 59
rstan::extract(), 77

sampling(), 32, 75
set_agd_arm, 68
set_agd_arm(), 14, 29, 37, 59, 71, 73
set_agd_contrast, 70
set_agd_contrast(), 14, 37, 41, 59, 69, 73
set_ipd, 72
set_ipd(), 14, 29, 37, 59, 69, 71
smoking, 74
sobol(), 6
stan_mlnmr, 27, 28
stan_mlnmr (stan_nma-class), 74
stan_nma, 18, 27, 28, 33, 61
stan_nma (stan_nma-class), 74
stan_nma-class, 74
stanfit, 11, 33
statins, 75
stats::Binomial, 63
stats::GammaDist, 64
student_t (priors), 61
summary(), 33
summary.mcmc_array (mcmc_array-class), 28
summary.nma_prior, 76
summary.nma_prior(), 63
summary.stan_nma, 77

theme_multinma, 78
thrombolytics, 79
tibble, 7
tidygraph, 10
transfusion, 80

uniroot(), 56
unnest_integration (add_integration), 6

waic (loo.stan_nma), 27
waic(), 27, 28
which_RE (RE_cor), 67