Package: gemtc (via r-universe)

October 20, 2026

Version 1.1-2

Date 2026-08-01

Title Network Meta-Analysis Using Bayesian Methods

Author Gert van Valkenhoef [aut, cre], Joel Kuiper [aut]

Maintainer Gert van Valkenhoef <gert@gertvv.nl>

Description Network meta-analyses (mixed treatment comparisons) in the Bayesian framework using JAGS. Includes methods to assess heterogeneity and inconsistency, and a number of standard visualizations. van Valkenhoef et al. (2012) <doi:10.1002/jrsm.1054>; van Valkenhoef et al. (2015) <doi:10.1002/jrsm.1167>.

Depends R (>= 3.5.0), coda (>= 0.13)

Imports igraph (>= 1.5), meta (>= 2.1), plyr (>= 1.8), graphics, grDevices, stats, utils, grid, rjags (>= 4-0), Rglpk, forcats (>= 0.5.0)

Suggests testthat (>= 0.8), Matrix

URL https://github.com/gertvv/gemtc

License GPL-3

LazyData true

Collate 'anohe.R' 'arrayize.R' 'blobbogram.R' 'template.R' 'code.R' 'data.R' 'deviance.R' 'forest.R' 'rtruncnorm.R' 'solveLP.R' 'inits.R' 'likelihoods.R' 'll-helper.counts.R' 'll.binom.cloglog.R' 'll.binom.log.R' 'll.binom.logit.R' 'll.call.R' 'll.normal.identity.R' 'll.normal.smd.R' 'll.poisson.log.R' 'minimum.diameter.spanning.tree.R' 'mtc.data.studyrow.R' 'mtc.hy.prior.R' 'mtc.model.R' 'mtc.model.consistency.R' 'mtc.model.nodesplit.R' 'mtc.model.regression.R' 'mtc.model.ume.R' 'mtc.model.use.R' 'mtc.network.R' 'stopIfNotConsistent.R' 'mtc.result.R' 'mtc.run.R' 'nodesplit.R' 'plotCovariateEffect.R' 'priors.R' 'rank.probability.R' 'regression.R' 'relative.effect.R' 'relative.effect.table.R'

RoxygenNote 7.2.3

NeedsCompilation yes

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

Date/Publication 2026-08-21 06:30:02 UTC

RemoteUrl https://github.com/cran/gemtc

RemoteRef HEAD

RemoteSha 1684d78104d4165b72d32c7bc879c1f343ed94ad

Contents

gemtc-package . . . 2
atrialFibrillation . . . 4
blobbogram . . . 5
blocker . . . 8
certolizumab . . . 9
depression . . . 10
dietfat . . . 10
hfPrevention . . . 11
ll.call . . . 11
mtc.anohe . . . 13
mtc.data.studyrow . . . 14
mtc.deviance . . . 16
mtc.hy.prior . . . 17
mtc.model . . . 18
mtc.network . . . 23
mtc.nodesplit . . . 25
mtc.run . . . 27
parkinson . . . 29
plotCovariateEffect . . . 30
rank.probability . . . 31
relative.effect . . . 32
relative.effect.table . . . 34
smoking . . . 35
thrombolytic . . . 36

Index **37**

`gemtc-package` *GeMTC: Network meta-analysis in R*

Description

An R package for performing network meta-analyses (mixed treatment comparisons).

Details

Network meta-analysis, or mixed treatment comparison (MTC) is a technique to meta-analyze networks of trials comparing two or more treatments at the same time [Dias et al. 2013a]. Using a Bayesian hierarchical model, all direct and indirect comparisons are taken into account to arrive at a single consistent estimate of the effect of all included treatments based on all included studies.

This package allows the automated generation of network meta-analysis models [van Valkenhoef et al. 2012], inclusing both fixed effect and random effects network meta-analysis, node-splitting models to identify inconsistency, and network meta-regression models. Models are estimated using JAGS (through the `rjags` package).

It is possible to get reproducible results, but as JAGS uses its own pseudo-random number generator, this is somewhat more involved. See `mtc.model` for details.

The source for GeMTC is available under the GPL-3 on Github.

Author(s)

Gert van Valkenhoef

References

S. Dias, N.J. Welton, D.M. Caldwell, and A.E. Ades (2010), *Checking consistency in mixed treatment comparison meta-analysis*, Statistics in Medicine 29(7-8, Sp. Iss. SI):932-944. [doi:10.1002/sim.3767]

S. Dias, A.J. Sutton, A.E. Ades, and N.J. Welton (2013a), *A Generalized Linear Modeling Framework for Pairwise and Network Meta-analysis of Randomized Controlled Trials*, Medical Decision Making 33(5):607-617. [doi:10.1177/0272989X12458724]

S. Dias, A.J. Sutton, N.J. Welton, and A.E. Ades (2013b), *Heterogeneity - Subgroups, Meta-Regression, Bias, and Bias-Adjustment*, Medical Decision Making 33(5):618-640. [doi:10.1177/0272989X13485157]

S. Dias, N.J. Welton, A.J. Sutton, D.M. Caldwell, G. Lu, and A.E. Ades (2013c), *Inconsistency in Networks of Evidence Based on Randomized Controlled Trials*, Medical Decision Making 33(5):641-656. [doi:10.1177/0272989X12455847]

A. Gelman, A. Jakulin, M. Grazia Pittau, Y.-S. Su (2008), *A weakly informative default prior distribution for logistic and other regression models*, The Annals of Applied Statistics 2(4):1360-1383. [doi:10.1214/08AOAS191]

R.M. Turner, J. Davey, M.J. Clarke, S.G. Thompson, J.P.T. Higgins (2012), *Predicting the extent of heterogeneity in meta-analysis, using empirical data from the Cochrane Database of Systematic Reviews*, International Journal of Epidemiology 41(3):818-827. [doi:10.1093/ije/dys041]

G. van Valkenhoef, G. Lu, B. de Brock, H. Hillege, A.E. Ades, and N.J. Welton (2012), *Automating network meta-analysis*, Research Synthesis Methods 3(4):285-299. [doi:10.1002/jrsm.1054]

G. van Valkenhoef, S. Dias, A.E. Ades, and N.J. Welton (2015), *Automated generation of node-splitting models for assessment of inconsistency in network meta-analysis*, Research Synthesis Methods, accepted manuscript. [doi:10.1002/jrsm.1167]

G. van Valkenhoef et al. (draft), *Modeling inconsistency as heterogeneity in network meta-analysis*, draft manuscript.

D.E. Warn, S.G. Thompson, and D.J. Spiegelhalter (2002), *Bayesian random effects meta-analysis of trials with binary outcomes: methods for the absolute risk difference and relative risk scales*, Statistics in Medicine 21(11):1601-1623. [doi:10.1002/sim.1189]

See Also

mtc.network, mtc.model, mtc.run

Examples

```
# Load the example network and generate a consistency model:
model <- mtc.model(smoking, type="consistency")

# Load pre-generated samples instead of runing the model:
## Not run: results <- mtc.run(model, thin=10)
results <- readRDS(system.file("extdata/luades-smoking-samples.rds", package="gemtc"))

# Print a basic statistical summary of the results:
summary(results)
## Iterations = 5010:25000
## Thinning interval = 10
## Number of chains = 4
## Sample size per chain = 2000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##         Mean     SD Naive SE Time-series SE
## d.A.B 0.4965 0.4081 0.004563       0.004989
## d.A.C 0.8359 0.2433 0.002720       0.003147
## d.A.D 1.1088 0.4355 0.004869       0.005280
## sd.d  0.8465 0.1913 0.002139       0.002965
##
## 2. Quantiles for each variable:
##
##          2.5%    25%    50%    75% 97.5%
## d.A.B -0.2985 0.2312 0.4910 0.7530 1.341
## d.A.C  0.3878 0.6720 0.8273 0.9867 1.353
## d.A.D  0.2692 0.8197 1.0983 1.3824 2.006
## sd.d   0.5509 0.7119 0.8180 0.9542 1.283
```

---

atrialFibrillation *Prevention of stroke in atrial fibrillation patients*

Description

A dataset of 25 trials investigating 17 treatments for stroke prevention in atrial fibrillation patients. The main outcome is the number of patients with a stroke, and a covariate captures the proportion of patients with a prior stroke.

Data are take from Table 1 of Cooper et al. (2009), with the following corrections applied: SPAF 3 and AFASAK 2 do not have a treatment 13 arm, and SPAF 1 does not contain treatment 5, but treatment 6. Thanks to prof. Cooper for providing the original analysis dataset.

Format

A network meta-regression dataset containing 63 rows of arm-based data (responders and sample size).

Source

Cooper et al. (2009), *Adressing between-study heterogeneity and inconsistency in mixed treatment comparisons: Application to stroke prevention treatments in individuals with non-rheumatic atrial fibrillation*, Statistics in Medicine 28:1861-1881. [doi:10.1002/sim.3594]

Examples

```
# Build a model similar to Model 4(b) from Cooper et al. (2009):
classes <- list("control"=c("01"),
                "anti-coagulant"=c("02","03","04","09"),
                "anti-platelet"=c("05","06","07","08","10","11","12","16","17"),
                "mixed"=c("13","14","15"))

regressor <- list(coefficient='shared',
                  variable='stroke',
                  classes=classes)

model <- mtc.model(atrialFibrillation,
                   type="regression",
                   regressor=regressor,
                   om.scale=10)

## Not run:
result <- mtc.run(model)
## End(Not run)
```

`blobbogram` *Plot a blobbogram (AKA forest plot)*

Description

`blobbogram` is a flexible function for creating blobbograms (forest plots), making no specific assumptions about the data being plotted. It supports column and row grouping as well as pagination.

Usage

```
blobbogram(data, id.label="Study", ci.label="Mean (95% CI)",
    left.label=NULL, right.label=NULL, center.label=NULL,
    log.scale=FALSE, xlim=NULL, styles=NULL,
```

```
    grouped=TRUE, group.labels=NULL,
    columns=NULL, column.labels=NULL,
    column.groups=NULL, column.group.labels=NULL,
    digits=2,
    ask=dev.interactive(orNone=TRUE),
    draw.no.effect=TRUE)

forest(x, ...)
```

Arguments

data	A data frame containing one row for each confidence interval to be visualized. The data format is described below.
id.label	Label to show above the row-id column.
ci.label	Label to show above the confidence intervals.
left.label	Label to show on the left-hand side of the no-difference line.
right.label	Label to show on the right-hand side of the no-difference line.
center.label	Label to show center-aligned with the no-difference line.
log.scale	If TRUE, the confidence intervals are given on a log scale, and axis labels will be exp() transformed.
xlim	The scale limits of the plot, if the confidence interval exceeds these limits an arrow will be shown at the limit. If unspecified, limits will be chosen that encompass all confidence intervals.
styles	A data frame describing the different row styles. By default, the styles "normal", "pooled" and "group" are defined.
grouped	If TRUE, and group.labels are specified, rows will be grouped according to the "group" column given in the data argument.
group.labels	Vector of group labels.
columns	Additional user-defined columns to be shown (names of columns given in the data argument).
column.labels	A vector of labels for the user-defined columns.
column.groups	Column groups, a numeric vector specifying the column group for each column.
column.group.labels	A vector of labels for the column groups.
digits	The number of (significant) digits to print.
ask	If TRUE, a prompt will be displayed before generating the next page of a multi-page plot.
draw.no.effect	If TRUE, draw the no-effect line.
x	An object to create a forest plot of.
...	Additional arguments.

Details

The `forest` function is a generic S3 method (definition compatible with the `meta` package). This package defines methods for `mtc.result` and `mtc.relative.effect.table`.

The `blobbogram` function creates a blobbogram (forest plot) from the given data (point estimates and confidence intervals) and meta-data (labels, column specifications, column groups, row groups, styles) using the `grid` package. If the plot would not fit the device's graphics region, the content is broken up into multiple plots generated in sequence (pagination).

The `data` argument is given as a data frame containing the following columns:

- id: identifier (label) for this row.
- group (optional): row group this row belongs to (indexes into the `group.labels` argument).
- pe: point estimate.
- ci.l: lower confidence interval limit.
- ci.u: upper confidence interval limit.
- style: the style to apply to this row (defined in the `styles` argument).

Additional user-defined columns can be specified using the `columns` and `column.labels` arguments.

The `styles` argument is given as a data frame containing the following columns:

- style: name of the style.
- weight: font weight.
- pe.style: symbol to draw for the point estimate ("circle" or "square", currently).

Value

None.

Note

This method should not be considered stable. We intend to generalize it further and possibly provide it in a separate package. The interface may change at any time.

Author(s)

Gert van Valkenhoef, Joël Kuiper

See Also

`meta::forest`, `grid::Grid`

Examples

```
data <- read.table(textConnection('
id               group pe       ci.l ci.u style      value.A    value.B
"Study 1"  1         0.35 0.08 0.92 "normal" "2/46"     "7/46"
"Study 2"  1         0.43 0.15 1.14 "normal" "4/50"     "8/49"
"Study 3"  2         0.31 0.07 0.74 "normal" "2/97"     "10/100"
"Study 4"  2         0.86 0.34 2.90 "normal" "9/104"    "6/105"
"Study 5"  2         0.33 0.10 0.72 "normal" "4/74"     "14/74"
"Study 6"  2         0.47 0.23 0.91 "normal" "11/120" "22/129"
"Pooled"     NA      0.42 0.15 1.04 "pooled" NA             NA
'), header=TRUE)
data$pe <- log(data$pe)
data$ci.l <- log(data$ci.l)
data$ci.u <- log(data$ci.u)

blobbogram(data, group.labels=c('GROUP 1', 'GROUP 2'),
    columns=c('value.A', 'value.B'), column.labels=c('r/n', 'r/n'),
    column.groups=c(1, 2), grouped=TRUE,
    column.group.labels=c('Intervention', 'Control'),
    id.label="Trial", ci.label="Odds Ratio (95% CrI)", log.scale=TRUE)
```

`blocker` *Beta blockers to prevent mortality after myocardial infarction*

Description

A dataset of 22 trials investigating beta blockers versus control to prevent mortality after myocardial infarction. Number of events and sample size.

Format

A meta-analysis dataset containing 44 rows of arm-based data (responders and sample size).

Source

S. Dias, A.J. Sutton, A.E. Ades, and N.J. Welton (2013a), *A Generalized Linear Modeling Framework for Pairwise and Network Meta-analysis of Randomized Controlled Trials*, Medical Decision Making 33(5):607-617. [doi:10.1177/0272989X12458724]

J.B. Carlin (1992), *Meta-analysis for 2 × 2 tables: a Bayesian approach*, Statistics in Medicine 11(2):141-158. [doi:10.1002/sim.4780110202]

certolizumab *Certolizumab Pegol (CZP) for Rheumatoid Arthritis*

Description

A dataset of 12 trials investigating 6 treatments and placebo for rheumatoid arthritis. The main outcome is the number of patients who improved by at least 50% on the American College of Rheumatology scale (ACR50) at 6 Months. A covariate is present for the mean disease duration at baseline (years).

Format

A network meta-regression dataset containing 24 rows of arm-based data (responders and sample size).

Source

S. Dias, A.J. Sutton, N.J. Welton, and A.E. Ades (2013b), *Heterogeneity - Subgroups, Meta-Regression, Bias, and Bias-Adjustment*, Medical Decision Making 33(5):618-640. [doi:10.1177/0272989X13485157]

Examples

```
# Run RE regression model with informative heterogeneity prior

regressor <- list(coefficient='shared',
                  variable='diseaseDuration',
                  control='Placebo')

# sd ~ half-Normal(mean=0, sd=0.32)
hy.prior <- mtc.hy.prior(type="std.dev", distr="dhnorm", 0, 9.77)

model <- mtc.model(certolizumab,
                   type="regression",
                   regressor=regressor,
                   hy.prior=hy.prior)

## Not run:
result <- mtc.run(model)
## End(Not run)
```

depression *Treatment response in major depression*

Description

A dataset of 111 trials investigating 12 treatments for major depression on treatment response. Treatment response was defined as a reduction of at least 50% from the baseline score on the HAM-D or MADRS at week 8 (or, if not available, another time between week 6 and 12).

Format

A network meta-analysis dataset containing 224 rows of arm-based data (responders and sample size).

Source

Cipriani et al. (2009), *Comparative efficacy and acceptability of 12 new-generation antidepressants: a multiple-treatments meta-analysis*, Lancet 373(9665):746-758. [doi:10.1016/S01406736(09)60046-5]

dietfat *Effects of low-fat diets on mortality*

Description

A dataset of 10 trials investigating low-fat diet versus control diet for mortality. Number of events and exposure in person-years.

Format

A meta-analysis dataset containing 20 rows of arm-based data (responders, exposure, and sample size).

Source

S. Dias, A.J. Sutton, A.E. Ades, and N.J. Welton (2013a), *A Generalized Linear Modeling Framework for Pairwise and Network Meta-analysis of Randomized Controlled Trials*, Medical Decision Making 33(5):607-617. [doi:10.1177/0272989X12458724]

Hooper et al. (2000), *Reduced or modified dietary fat for preventing cardiovascular disease*, Cochrane Database of Systematic Reviews 2:CD002137. [doi:10.1002/14651858.CD002137]

hfPrevention — *Statins versus placebo in primary and secondary prevention of heart failure*

Description

A dataset of 19 trials comparing statins versus placebo or usual care for cholesterol lowering. The main outcome is the number of deaths. Trials are either primary prevention (no previous heart disease; secondary = 0) or secondary prevention (previous heart disease; secondary = 1).

Format

A meta-regression dataset containing 38 rows of arm-based data (responders and sample size).

Source

S. Dias, A.J. Sutton, N.J. Welton, and A.E. Ades (2013b), *Heterogeneity - Subgroups, Meta-Regression, Bias, and Bias-Adjustment*, Medical Decision Making 33(5):618-640. [doi:10.1177/0272989X13485157]

Examples

```
# Build a model similar to Program 1(a) from Dias et al. (2013b):
regressor <- list(coefficient='shared',
                  variable='secondary',
                  control='control')

model <- mtc.model(hfPrevention,
                   type="regression",
                   regressor=regressor,
                   hy.prior=mtc.hy.prior("std.dev", "dunif", 0, 5))

## Not run:
result <- mtc.run(model)
## End(Not run)
```

ll.call — *Call a likelihood/link-specific function*

Description

GeMTC implements various likelihood/link combinations. Functionality specific to the likelihood/link is handled by methods with names ending in `.<likelihood>.<link>`. This convenience function calls such methods.

Usage

```
ll.call(fnName, model, ...)
```

Arguments

fnName	The name of the function to call. See details for available functions.
model	An object of S3 class mtc.model describing a network meta-analysis model, or a list containing elements named 'likelihood' and 'link'.
...	Additional arguments to be passed to the function.

Details

The following methods currently need to be implemented to implement a likelihood/link:

- mtc.arm.mle: calculates a (corrected) maximum likelihood estimate for an arm-level effect. Used to generate starting values.
- mtc.rel.mle: calculates a (corrected) maximum likelihood estimate for a relative effect. Used to generate starting values.
- mtc.code.likelihood: generates JAGS code implementing the likelihood.
- scale.log: returns TRUE if plots should use the log scale.
- scale.name: returns the user-facing name of the outcome metric.
- scale.limit.inits: returns an upper and lower bound for the initial values, because some initial values might trigger boundary conditions such as probability 0 or 1 for the binomial.
- required.columns.ab: returns the required columns for arm-based data.

The first two methods can now also be used to selectively apply continuity corrections in case the maximum likelihood estimates are used for other purposes. mtc.arm.mle has an additional k=0.5 argument to specify the correction factor. mtc.rel.mle has arguments correction.force=TRUE to force application of the continuity correction even if unnecessary, correction.type="constant" to specify the type of correction (specify "reciprocal") for a correction proportional to the reciprocal of the size of the other arm, and correction.magnitude=1 to specify the (total) magnitude of the correction. These corrections apply only for count data, and will be ignored for continuous likelihood/links.

Value

The return value of the called function.

Author(s)

Gert van Valkenhoef

See Also

mtc.model

Examples

```
# The "model" may be a stub.
model <- list(likelihood="poisson", link="log")

ll.call("scale.name", model)
# "Hazard Ratio"

ll.call("mtc.arm.mle", model, c('responders'=12, 'exposure'=80))
#      mean         sd
#-1.8562980  0.1118034
```

`mtc.anohe` *Analysis of heterogeneity (ANOHE)*

Description

(EXPERIMENTAL) Generate an analysis of heterogeneity for the given network. Three types of model are estimated: unrelated study effects, unrelated mean effects, and consistency. Output of the `summary` function can passed to `plot` for a visual representation.

Usage

```
mtc.anohe(network, ...)
```

Arguments

`network`	An object of S3 class `mtc.network`.
`...`	Arguments to be passed to `mtc.run` or `mtc.model`. This can be used to set the likelihood/link or the number of iterations, for example.

Details

Analysis of heterogeneity is intended to be a unified set of statistics and a visual display that allows the simultaneous assessment of both heterogeneity and inconsistency in network meta-analysis [van Valkenhoef et al. 2014b (draft)].

`mtc.anohe` returns the MCMC results for all three types of model. To get appropriate summary statistics, call `summary()` on the results object. The summary can be plotted.

To control parameters of the MCMC estimation, see `mtc.run`. To specify the likelihood/link or to control other model parameters, see `mtc.model`. The `...` arguments are first matched against `mtc.run`, and those that do not match are passed to `mtc.model`.

Value

For `mtc.anohe`: an object of class `mtc.anohe`. This is a list with the following elements:

`result.use`	The result for the USE model (see `mtc.run`).
`result.ume`	The result for the UME model (see `mtc.run`).

`result.cons` The result for the consistency model (see `mtc.run`).

For `summary`: an object of class `mtc.anohe.summary`. This is a list with the following elements:

`cons.model`	Generated consistency model.
`studyEffects`	Study-level effect summaries (multi-arm trials downweighted).
`pairEffects`	Pair-wise pooled effect summaries (from the UME model).
`consEffects`	Consistency effect summaries.
`indEffects`	Indirect effect summaries (back-calculated).
`isquared.comp`	Per-comparison I-squared statistics.
`isquared.glob`	Global I-squared statistics.

Note

This method should not be considered stable. It is an experimental feature and heavily work in progress. The interface may change at any time.

Author(s)

Gert van Valkenhoef, Joël Kuiper

See Also

`mtc.model` `mtc.run`

---

mtc.data.studyrow *Convert one-study-per-row datasets*

Description

Converts datasets in the one-study-per-row format to one-arm-per-row format used by GeMTC

Usage

```
mtc.data.studyrow(data,
  armVars=c('treatment'='t', 'responders'='r', 'sampleSize'='n'),
  nArmsVar='na',
  studyVars=c(),
  studyNames=1:nrow(data),
  treatmentNames=NA,
  patterns=c('%s..', '%s..%d.'))
```

Arguments

`data`	Data in one-study-per-row format.
`armVars`	Vector of per-arm variables. The name of each component will be the column name in the resulting dataset. The column name in the source dataset is derived from the value of each component.
`nArmsVar`	Variable holding the number of arms for each study.
`studyVars`	Vector of per-study variables. The name of each component will be the column name in the resulting dataset. The column name in the source dataset is derived from the value of each component.
`studyNames`	Vector of study names.
`treatmentNames`	Vector of treatment names.
`patterns`	Patterns to generate column names in the source dataset. The first is for per-study variables, the second for per-arm variables.

Details

Maps the one-study-per-row format that is widely used and convenient for BUGS models to the one-arm-per-row format used by GeMTC. As the primary purpose is to input datasets from BUGS models, the defaults work for the standard BUGS data table format. In most cases, it should be possible to just copy/paste the BUGS data table (without the final 'END') and `read.table` it into R, then apply `mtc.data.studyrow`. In many cases, the resulting table can be processed directly by `mtc.network`.

Value

A data table with the requested columns.

Author(s)

Gert van Valkenhoef

See Also

`mtc.network`

Examples

```
## Example taken from the NICE DSU TSD series in Evidence Synthesis, #2
## Dopamine agonists for the treatment of Parkinson's

# Read the bugs-formatted data
data.src <- read.table(textConnection('
t[,1] t[,2] t[,3] y[,1] y[,2] y[,3] se[,1] se[,2] se[,3] na[]
1 3 NA -1.22 -1.53 NA 0.504 0.439 NA 2
1 2 NA -0.7 -2.4 NA 0.282 0.258 NA 2
1 2 4 -0.3 -2.6 -1.2 0.505 0.510 0.478 3
3 4 NA -0.24 -0.59 NA 0.265 0.354 NA 2
3 4 NA -0.73 -0.18 NA 0.335 0.442 NA 2
```

```
4 5 NA -2.2 -2.5 NA 0.197 0.190 NA 2
4 5 NA -1.8 -2.1 NA 0.200 0.250 NA 2'), header=TRUE)

# Convert the data, setting treatment names
data <- mtc.data.studyrow(data.src,
  armVars=c('treatment'='t', 'mean'='y', 'std.err'='se'),
  treatmentNames=c('Placebo', 'DA1', 'DA2', 'DA3', 'DA4'))

# Check that the data are correct
print(data)

# Create a network
network <- mtc.network(data)
```

`mtc.deviance` *Inspect residual deviance*

Description

Inspect the posterior residual deviance and summarize it using plots

Usage

```
mtc.deviance(result)

mtc.devplot(x, ...)
mtc.levplot(x, ...)

## S3 method for class 'mtc.deviance'
plot(x, auto.layout=TRUE, ...)
```

Arguments

`result`	An object of class `mtc.result`.
`x`	An object of class `mtc.deviance`.
`auto.layout`	If `TRUE`, the separate plots will be shown as panels on a single page.
`...`	Graphical parameters.

Details

`mtc.devplot` will generate a stem plot of the posterior deviance per arm (if there are only arm-based data) or the mean per data point deviance per study (if there are contrast-based data).

`mtc.levplot` will plot the leverage versus the square root of the residual deviance (mean per data point for each study).

The generic plot function will display both on a single page (unless `auto.layout=FALSE`).

Value

`mtc.deviance` returns the deviance statistics of a `mtc.result`.

Author(s)

Gert van Valkenhoef

See Also

`mtc.run`

`mtc.hy.prior` *Set priors for the heterogeneity parameter*

Description

These functions generate priors for the heterogeneity parameter in `mtc.model`. Priors can be set explicitly or, for outcomes on the log odds-ratio scale, based on empirical research.

Usage

```
mtc.hy.prior(type, distr, ...)

mtc.hy.empirical.lor(outcome.type, comparison.type)
```

Arguments

`type`	Type of heterogeneity prior: 'std.dev', 'var', or 'prec' for standard deviation, variance, or precision respectively.
`distr`	Prior distribution name (JAGS syntax). Typical ones would be 'dunif' (uniform), 'dgamma' (Gamma), or 'dlnorm' (log-normal). Use 'dhnorm' for the half-normal. Note that, as in JAGS, the precision (and not the variance or standard deviation) is used for the normal distribution and its derivatives.
`...`	Arguments to the `distr`. Can be numerical values or "om.scale" for the estimated outcome measure scale (see `mtc.model`)
`outcome.type`	The type of outcome to get an empirical prior for. Can be one of 'mortality' (all-cause mortality), 'semi-objective' (e.g. cause-specific mortality, major morbidity event, drop-outs), or 'subjective' (e.g. pain, mental health, dichotomous biomarkers).
`comparison.type`	The type of comparison to get an empirical prior for. Can be one of 'pharma-control' (pharmacological interventions versus control), 'pharma-pharma' (pharmacological versus pharmacological interventions) and 'non-pharma' (any other comparisons).

Details

The generated prior is a list, the structure of which may change without notice. It can be converted to JAGS compatible code using `as.character`.

Empirical priors for the log odds-ratio (LOR) are taken from [Turner et al. 2012].

Value

A value to be passed to `mtc.model`.

Author(s)

Gert van Valkenhoef

See Also

`mtc.model`

Examples

```
# NOTE: the mtc.run commands below are for illustrative purposes, such a small
# number of iterations should obviously not be used in practice.

# set a uniform prior standard deviation
model1 <- mtc.model(smoking, hy.prior=mtc.hy.prior("std.dev", "dunif", 0, 2))
result <- mtc.run(model1, n.adapt=10, n.iter=10)

# set an empirical (log-normal) prior on the variance
model2 <- mtc.model(smoking, hy.prior=mtc.hy.empirical.lor("subjective", "non-pharma"))
result <- mtc.run(model2, n.adapt=10, n.iter=10)

# set a gamma prior on the precision
model3 <- mtc.model(smoking, hy.prior=mtc.hy.prior("prec", "dgamma", 0.01, 0.01))
result <- mtc.run(model3, n.adapt=10, n.iter=10)
```

`mtc.model` *Generate network meta-analysis models*

Description

The `mtc.model` function generates network meta-analysis models from an `mtc.network` object.

Usage

```
mtc.model(network, type = "consistency", factor = 2.5, n.chain = 4,
  likelihood=NULL, link=NULL, linearModel="random",
  om.scale=NULL, hy.prior=mtc.hy.prior("std.dev", "dunif", 0, "om.scale"),
  re.prior.sd=15 * om.scale, dic=TRUE, powerAdjust=NA, ...)
```

Arguments

network	An object of S3 class mtc.network
type	A string literal indicating the type of model (allowed values are "consistency", "regression", "nodesplit", "ume", or "use").
factor	Variance scaling factor for the starting values
n.chain	Number of chains in the model
likelihood	The likelihood to be used. If unspecified, a suitable likelihood will be inferred for the given data.
link	The link function to be used. If unspecified, a suitable link function will be inferred for the given data.
linearModel	The type of linear model to be generated. Can be "random" for a random effects model, or "fixed" for a fixed effect model.
om.scale	Outcome measure scale. Represents a "very large" difference on the analysis' outcome scale. This is used to set vague priors. For the log odds-ratio, values between 2 and 5 are considered reasonable. For continuous outcomes, this depends heavily on the specific outcome. If left unspecified, it is determined from the data.
hy.prior	Heterogeneity prior. See mtc.hy.prior.
re.prior.sd	Standard deviation for the relative effects prior (normal distribution).
dic	When set to TRUE, deviance and fitted values will be monitored to allow computation of the Deviance Information Criterion (DIC) at residual.
powerAdjust	Optional: the name of a column in the studies data frame of the mtc.network. This column must contain values between 0 and 1. The likelihood for each study will be adjusted by inflating the variance, where 0 means the study is excluded and 1 means it receives full weight. See details for more.
...	Additional arguments to be passed to the type-specific model generation function.

Details

The mtc.model function generates an object of S3 class mtc.model, which can be visualized by the generic plot function or summarized by the generic summary function.

These likelihood/links are supported:

- normal/identity: for continuous (mean difference) data.
  Required columns: [mean, std.err] or [mean, std.dev, sampleSize].
  Result: relative mean difference.
- normal/smd: for continuous (mean difference) data.
  Required columns: [mean, std.dev, sampleSize].
  Result: relative standardized mean difference.
- binom/logit: for dichotomous data.
  Required columns [responders, sampleSize].
  Result: (log) odds ratio.

- binom/log: for dichotomous data.
  Required columns `[responders, sampleSize]`.
  Result: (log) risk ratio.
- binom/cloglog: for rate (survival) data - equal follow-up in each arm.
  Required columns `[responders, sampleSize]`.
  Result: (log) hazard ratio.
- poisson/log: for rate (survival) data.
  Required columns `[responders, exposure]`.
  Result: (log) hazard ratio.

Most likelihood/links follow [Dias et al. 2013a], and the binom/log model follows [Warn et al. 2002].

The following model types are supported:

- `consistency`: ordinary consistency model. No additional parameters. [Dias et al. 2013a, van Valkenhoef et al. 2012]
- `nodesplit`: node-splitting model. Removes both arms used to estimate the direct evidence from the network of indirect evidence, rather than just one of those arms. This means that three-arm trials do not contribute any evidence in the network of indirect evidence. When relative effect data are present, these are transformed appropriately (using an assumption of normality) to enable this direct/indirect evidence split. Additional parameters: `t1` and `t2`, which indicate the comparison to be split. [Dias et al. 2010, van Valkenhoef et al. 2015]
- `regression`: meta-regression model. Additional parameters: `regressor`, which indicates how to structure the treatment-interaction model for the regression. See below for details.
- `use`: unrelated study effects. Models the effects within each study as if the studies are independent. No additional parameters. [van Valkenhoef et al. (draft)]
- `ume`: unrelated mean effects. Models the effects within each comparison as if they are independent. Does not properly handle multi-arm trials, and warns when they are present in the network. No additional parameters. [Dias et al. 2013b, van Valkenhoef et al. (draft)]

Regressor specification: a list with elements: `variable`, `coefficient`, and either `control` or `classes`. The `variable` is the name of the covariate to include in the regression analysis, and must be a column of the studies data frame in the network. The regressor variable is automatically centered and standardized using the method recommended by Gelman et al. (2008). The `coefficient` indicates the type of treatment-interaction model: "shared", "unrelated", or "exchangeable". `control`, if specified, must be the ID of a treatment in the network. All other treatments have a coefficient relative to the control, which can be the same for all treatments ("shared"), different for all treatments ("unrelated") or exchangeable between treatments ("exchangeable"). `classes` is a named list of treatment classes, the first of which will act as the control class. Each class is a vector of treatment IDs from the network. Only "shared" coefficients can currently be used, meaning a single coefficient per class (except the control class). See also `atrialFibrillation`, `certolizumab`, and `hfPrevention` for examples of meta-regression analyses.

Studies can be downweighted by using the `powerAdjust` argument, which applies a variance inflation (also known as "power prior") to the likelihood. This allows a weight $\alpha_i \in [0, 1]$ to be specified for each study i. The log-likelihood will be multiplied by a factor α_i, or equivalently for normal distributions the variance will be multiplied by $1/\alpha_i$. Setting $\alpha_i = 0$ will completely exclude that

study, whereas setting $\alpha_i = 1$ will weight it fully. Essentially, down-weighted models modify the data and hence model fit statistics such as DIC can not be compared between models with different weightings.

Value

An object of class `mtc.model`. The following elements are descriptive:

`type`	The type of model
`network`	Network the model was generated from
`tree`	Spanning tree formed by the basic parameters
`var.scale`	The scaling factor used to over-disperse starting values
`likelihood`	The likelihood used
`link`	The link function used
`om.scale`	The scale for the variance parameters
`regressor`	Regressor specification (regression models only): includes additional elements "center" and "scale" describing how the regressor input was standardized

These elements determine the model run by JAGS:

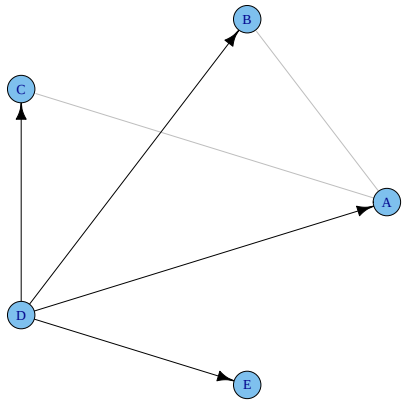
`n.chain`	The number of chains
`code`	Model code in JAGS syntax. Use `cat()` for proper formatting.
`data`	Data in JAGS compatible format
`inits`	Initial values in JAGS compatible format
`monitors`	The nodes of the JAGS model to monitor
`dic`	Whether to compute the DIC

These latter fields can be modified to alter the statistical model, but such changes may break the model or assumptions made elsewhere in the package.

Visualization

Calling the generic plot method on an S3 `mtc.model` object will show a graph with the treatments as vertices and the comparisons as edges. The lines with solid arrows represent basic parameters, and the other lines represent comparisons that are not associated with any parameter but do have direct evidence from trials.

The example code will generate the following graph:



The default layout algorithm is `igraph::layout.circle`, other layout algorithms can be used by passing them as an optional argument called `layout` to `plot`. The `igraph::layout.fruchterman.reingold` algorithm also seems to produce nice results and may be better for large graphs.

Reproducible results

It is possible to get reproducible results, but as JAGS uses its own pseudo-random number generator, this is somewhat more involved.

`mtc.model` generates random numbers using the R functions (for example to generate initial values). Therefore, `set.seed` must be called before calling `mtc.model`.

Then, before calling `mtc.run`, the random number generator type and seed must be set for each of the chains. This can be done as documented in `jags.model`, by setting the `.RNG.name` and `.RNG.seed` for each chain in `model$inits`. See below for an example.

Author(s)

Gert van Valkenhoef, Joël Kuiper

See Also

`mtc.network`, `mtc.run`

Examples

```
# Random effects consistency model for Parkinson network
model <- mtc.model(parkinson)
plot(model)
summary(model)

# Fixed effect meta-regression for heart failure prevention
regressor <- list(coefficient='shared',
```

```
                  variable='secondary',
                  control='control')

model <- mtc.model(hfPrevention,
                   type="regression",
                   regressor=regressor,
                   linearModel="fixed")

# Reproducible results
# Set the R RNG seed
set.seed(42)
model <- mtc.model(parkinson, likelihood='normal', link='identity')
# By default, the model will have 4 chains - generate a seed for each
seeds <- sample.int(4, n = .Machine$integer.max)
# Apply JAGS RNG settings to each chain
model$inits <- mapply(c, model$inits, list(
  list(.RNG.name="base::Wichmann-Hill", .RNG.seed=seeds[1]),
  list(.RNG.name="base::Marsaglia-Multicarry", .RNG.seed=seeds[2]),
  list(.RNG.name="base::Super-Duper", .RNG.seed=seeds[3]),
  list(.RNG.name="base::Mersenne-Twister", .RNG.seed=seeds[4])), SIMPLIFY=FALSE)
```

`mtc.network` *Create an mtc.network*

Description

Creates an object of class `mtc.network`

Usage

```
mtc.network(data.ab=data, treatments=NULL, description="Network",
            data.re=NULL, studies=NULL, data=NULL)

## S3 method for class 'mtc.network'
plot(x, layout=igraph::layout.circle, dynamic.edge.width=TRUE, use.description=FALSE, ...)
```

Arguments

`data.ab`	Arm-level data. A data frame defining the arms of each study, containing the columns 'study' and 'treatment', where 'treatment' must refer to an existing treatment ID if treatments were specified. Further columns define the data per arm, and depend on the likelihood/link to be used. See `mtc.model` for supported likelihood/links and their data requirements.
`data.re`	Relative effect data. A data frame defining the arms of each study, containing the columns 'study' and 'treatment', where 'treatment' must refer to an existing treatment ID if treatments were specified. The column 'diff' specifies the mean difference between the current arm and the baseline arm; set 'diff=NA' for the baseline arm. The column 'std.err' specifies the standard error of the mean difference (for non-baseline arms). For trials with more than two arms, specify

	the standard error of the mean of the baseline arm in 'std.err', as this determines the covariance of the differences.
`treatments`	Optional. A data frame with columns 'id' and 'description' defining the treatments or a vector giving the treatment IDs.
`studies`	Optional. A data frame with a 'study' column naming the studies and additional columns containing covariate values.
`description`	Optional. Short description of the network.
`data`	Deprecated. Arm-level data; automatically assigned to data.ab if it is not specified. Present for compatibility with older versions.
`x`	An mtc.network object.
`layout`	An igraph-compatible layout.
`dynamic.edge.width`	If set to TRUE, dynamically set the edge width based on the number of studies.
`use.description`	Display treatment descriptions instead of treatment IDs.
`...`	Additional arguments passed to `plot.igraph`.

Details

One-arm trials are automatically removed, which results in a warning.

Also see `mtc.data.studyrow` for a convenient way to import data from the one-study-per-row format, which is very popular for BUGS code.

Value

For `mtc.network`, an object of the class `mtc.network` which is a list containing:

`description`	A short description of the network
`treatments`	A data frame describing the treatments
`data.ab`	A data frame containing the network data (arm-level)
`data.re`	A data frame containing the network data (relative effects)
`studies`	A data frame containing study-level information (covariates)

These are cleaned up and standardized versions of the arguments provided, or generated defaults for 'treatments' if the argument was omitted.

Author(s)

Gert van Valkenhoef, Joël Kuiper

See Also

`mtc.data.studyrow`, `mtc.model`

Examples

```
# Create a new network by specifying all information.
treatments <- read.table(textConnection('
  id  description
  A   "Treatment A"
  B   "Treatment B"
  C   "Treatment C"'), header=TRUE)
data <- read.table(textConnection('
  study  treatment  responders  sampleSize
  01     A          2           100
  01     B          5           100
  02     B          6           110
  02     C          1           110
  03     A          3           60
  03     C          4           80
  03     B          7           80'), header=TRUE)
network <- mtc.network(data, description="Example", treatments=treatments)
plot(network)

# Create a new network by specifying only the data.
data <- read.table(textConnection('
  study  treatment  mean   std.dev  sampleSize
  01     A          -1.12  0.6      15
  01     B          -1.55  0.5      16
  02     A          -0.8   0.7      33
  02     B          -1.1   0.5      31'), header=TRUE)
network <- mtc.network(data)

# Print the network
print(network)
## MTC dataset: Network
##   study treatment  mean std.dev sampleSize
## 1     1         A -1.12     0.6         15
## 2     1         B -1.55     0.5         16
## 3     2         A -0.80     0.7         33
## 4     2         B -1.10     0.5         31
```

`mtc.nodesplit` *Node-splitting analysis of inconsistency*

Description

Generate and run an ensemble of node-splitting models, results of which can be jointly summarized and plotted.

Usage

```
mtc.nodesplit(network, comparisons=mtc.nodesplit.comparisons(network), ...)
mtc.nodesplit.comparisons(network)
```

Arguments

network	An object of S3 class mtc.network.
comparisons	Data frame specifying the comparisons to be split. The frame has two columns: 't1' and 't2'.
...	Arguments to be passed to mtc.run or mtc.model. This can be used to set the likelihood/link or the number of iterations, for example.

Details

mtc.nodesplit returns the MCMC results for all relevant node-splitting models [van Valkenhoef et al. 2015]. To get appropriate summary statistics, call summary() on the results object. The summary can be plotted. See mtc.model for details on how the node-splitting models are generated.

To control parameters of the MCMC estimation, see mtc.run. To specify the likelihood/link or to control other model parameters, see mtc.model. The ... arguments are first matched against mtc.run, and those that do not match are passed to mtc.model.

mtc.nodesplit.comparisons returns a data frame enumerating all comparisons that can reasonably be split (i.e. have independent indirect evidence).

Value

For mtc.nodesplit: an object of class mtc.nodesplit. This is a list with the following elements:

d.X.Y	For each comparison (t1=X, t2=Y), the MCMC results
consistency	The consistency model results

For summary: an object of class mtc.nodesplit.summary. This is a list with the following elements:

dir.effect	Summary of direct effects for each split comparison
ind.effect	Summary of indirect effects for each split comparison
cons.effect	Summary of consistency model effects for each split comparison
p.value	Inconsistency p-values for each split comparison
cons.model	The generated consistency model

Author(s)

Gert van Valkenhoef, Joël Kuiper

See Also

mtc.model mtc.run

Examples

```
# Run all relevant node-splitting models
## Not run:  result.ns <- mtc.nodesplit(parkinson, thin=50)
# (read results from file instead of running:)
result.ns <- readRDS(system.file('extdata/parkinson.ns.rds', package='gemtc'))

# List the individual models
names(result.ns)

# Time series plots and convergence diagnostics for d.A.C model
plot(result.ns$d.A.C)
gelman.diag(result.ns$d.A.C, multivariate=FALSE)

# Overall summary and plot
summary.ns <- summary(result.ns)
print(summary.ns)
plot(summary.ns)
```

`mtc.run` *Running an* `mtc.model` *using an MCMC sampler*

Description

The function `mtc.run` is used to generate samples from a object of type `mtc.model` using a MCMC sampler. The resulting `mtc.result` object can be coerced to an `mcmc.list` for further analysis of the dataset using the coda package.

Usage

```
mtc.run(model, sampler = NA, n.adapt = 5000, n.iter = 20000, thin = 1, n.burnin = 0)

## S3 method for class 'mtc.result'
summary(object, ...)
## S3 method for class 'mtc.result'
plot(x, ...)
## S3 method for class 'mtc.result'
forest(x, use.description=FALSE, ...)
## S3 method for class 'mtc.result'
print(x, ...)
## S3 method for class 'mtc.result'
as.mcmc.list(x, ...)
```

Arguments

`model`	An object of S3 class `mtc.model` describing a network meta-analysis model.
`sampler`	Deprecated: gemtc now only supports the JAGS sampler. Specifying a sampler will result in a warning or error. This argument will be removed in future versions.

n.adapt	Amount of adaptation (or tuning) iterations.
n.iter	Amount of simulation iterations.
thin	Thinning factor.
n.burnin	Amount of burn-in iterations. Has to be positive number.
object	Object of S3 class mtc.result.
x	Object of S3 class mtc.result.
use.description	Display treatment descriptions instead of treatment IDs.
...	Additional arguments.

Value

An object of class mtc.result. This is a list with the following elements:

samples	The samples resulting from running the MCMC model, in mcmc.list format.
model	The mtc.model used to produce the samples.
deviance	Residual deviance statistics, a list with the following elements. DIC: deviance information criterion at residual (Dbar + pD). Dbar: mean sum of residual deviance. pD: sum of leverage, also known as the effective number of parameters. dev.ab: mean posterior residual deviance of each arm (for arm-based data). fit.ab: deviance at the posterior mean of te fitted values (for arm-based data). dev.re: mean posterior residual deviance of each study (for relative-effect data). fit.re: deviance at the posterior mean of the fitted values (for relative-effect data).

The object can be coerced to an mcmc.list from the coda package by the generic S3 method as.mcmc.list.

Analysis of the results

Convergence of the model can be assessed using methods from the coda package. For example the Brooks-Gelman-Rubin method (coda::gelman.diag, coda::gelman.plot). The summary also provides useful information, such as the MCMC error and the time series and densities given by plot should also be inspected.

The forest function can provide forest plots for mtc.result objects. This is especially useful in combination with the relative.effect function that can be used to calculate relative effects compared to any baseline for consistency models. The rank.probability function calculates rank probabilities for consistency models.

Author(s)

Gert van Valkenhoef, Joël Kuiper

See Also

mtc.model

relative.effect.table, relative.effect, rank.probability

coda::gelman.diag, coda::gelman.plot

Examples

```
model <- mtc.model(smoking)

## Not run: results <- mtc.run(model, thin=10)
results <- readRDS(system.file("extdata/luades-smoking-samples.rds", package="gemtc"))

# Convergence diagnostics
gelman.plot(results)

# Posterior summaries
summary(results)
## Iterations = 5010:25000
## Thinning interval = 10
## Number of chains = 4
## Sample size per chain = 2000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##         Mean     SD Naive SE Time-series SE
## d.A.B 0.4965 0.4081 0.004563       0.004989
## d.A.C 0.8359 0.2433 0.002720       0.003147
## d.A.D 1.1088 0.4355 0.004869       0.005280
## sd.d  0.8465 0.1913 0.002139       0.002965
##
## 2. Quantiles for each variable:
##
##          2.5%    25%    50%    75% 97.5%
## d.A.B -0.2985 0.2312 0.4910 0.7530 1.341
## d.A.C  0.3878 0.6720 0.8273 0.9867 1.353
## d.A.D  0.2692 0.8197 1.0983 1.3824 2.006
## sd.d   0.5509 0.7119 0.8180 0.9542 1.283

plot(results) # Shows time-series and density plots of the samples
forest(results) # Shows a forest plot
```

---

parkinson *Mean off-time reduction in Parkinson's disease*

Description

A dataset of seven trials investigating four treatments and placebo for Parkinson's disease. The outcome is mean off-time reduction.

Format

parkinson: A network meta-analysis dataset containing fifteen rows of arm-based data (mean, standard deviation, and sample size).

parkinson_diff: A network meta-analysis dataset containing fifteen rows of contrast-based data.

parkinson_shared: A network meta-analysis dataset containing mixed arm-based and contrast-based data.

Source

Franchini et al. (2012), *Accounting for correlation in network meta-analysis with multi-arm trials*, Research Synthesis Methods, 3(2):142-160. [doi:10.1002/jrsm.1049]

`plotCovariateEffect` *Plot treatment effects versus covariate values*

Description

The plot will show the median treatment effect and the 95% credible interval on the y-axis and the covariate value on the x-axis. One plot page will be generated per pair of treatments.

Usage

```
plotCovariateEffect(result, t1, t2, xlim=NULL, ylim=NULL,
                    ask=dev.interactive(orNone=TRUE))
```

Arguments

`result`	Results object - created by `mtc.result`
`t1`	A list of baseline treatments to calculate treatment effects against. Will be extended to match the length of t2.
`t2`	A list of treatments to calculate the effects for. Will be extended to match the length of t1. If left empty and t1 is a single treatment, effects of all treatments except t1 will be calculated.
`xlim`	The x-axis limits.
`ylim`	The y-axis limits.
`ask`	If TRUE, a prompt will be displayed before generating the next page of a multi-page plot.

Details

Default x-axis limits will be set to three standard deviations above and below the centering value of the covariate. The y-axis limits will be set based on the minimum and maximum 95% CrI limits among the set of effects computed.

Value

None.

Author(s)

Gert van Valkenhoef

See Also

relative.effect, mtc.run

---

rank.probability *Calculating rank-probabilities*

Description

Rank probabilities indicate the probability for each treatment to be best, second best, etc.

Usage

```
rank.probability(result, preferredDirection=1, covariate=NA)

## S3 method for class 'mtc.rank.probability'
print(x, ...)
## S3 method for class 'mtc.rank.probability'
plot(x, ...)

sucra(ranks)
rank.quantiles(ranks, probs=c("2.5%"=0.025, "50%"=0.5, "97.5%"=0.975))
```

Arguments

result	Object of S3 class mtc.result to be used in creation of the rank probability table
preferredDirection	Preferential direction of the outcome. Set 1 if higher values are preferred, -1 if lower values are preferred.
covariate	(Regression analyses only) Value of the covariate at which to compute rank probabilities.
x	An object of S3 class rank.probability.
...	Additional arguments.
ranks	A ranking matrix where the treatments are the rows (e.g. the result of rank.probability).
probs	Probabilities at which to give quantiles.

Details

For each MCMC iteration, the treatments are ranked by their effect relative to an arbitrary baseline. A frequency table is constructed from these rankings and normalized by the number of iterations to give the rank probabilities.

Value

`rank.probability`: A matrix (with class `mtc.rank.probability`) with the treatments as rows and the ranks as columns. `sucra`: A vector of SUCRA values. `rank.quantiles`: A matrix with treatments as rows and quantiles as columns, giving the quantile ranks (by default, the median and 2.5% and 97.5% ranks).

Author(s)

Gert van Valkenhoef, Joël Kuiper

See Also

`relative.effect`

Examples

```
model <- mtc.model(smoking)
# To save computation time we load the samples instead of running the model
## Not run: results <- mtc.run(model)
results <- readRDS(system.file("extdata/luades-smoking-samples.rds", package="gemtc"))

ranks <- rank.probability(results)
print(ranks)
## Rank probability; preferred direction = 1
##       [,1]     [,2]     [,3]     [,4]
## A 0.000000 0.003000 0.105125 0.891875
## B 0.057875 0.175875 0.661500 0.104750
## C 0.228250 0.600500 0.170875 0.000375
## D 0.713875 0.220625 0.062500 0.003000

print(sucra(ranks))
##          A          B          C          D
## 0.03670833 0.39591667 0.68562500 0.88175000

print(rank.quantiles(ranks))
##   2.5% 50% 97.5%
## A    3   4     4
## B    1   3     4
## C    1   2     3
## D    1   1     3

plot(ranks) # plot a cumulative rank plot
plot(ranks, beside=TRUE) # plot a 'rankogram'
```

---

relative.effect *Calculating relative effects*

Description

Calculates the relative effects of pairs of treatments.

Usage

```
relative.effect(result, t1, t2 = c(), preserve.extra = TRUE, covariate = NA)
```

Arguments

result	An object of S3 class mtc.result to derive the relative effects from.
t1	A list of baselines to calculate a relative effects against. Will be extended to match the length of t2.
t2	A list of treatments to calculate the relative effects for. Will be extended to match the length of t1. If left empty and t1 is a single treatment, relative effects of all treatments except t1 will be calculated.
preserve.extra	Indicates whether to preserve extra parameters such as the sd.d.
covariate	(Regression analyses only) Value of the covariate at which to compute relative effects.

Value

Returns an mtc.results object containing the calculated relative effects.

Note that this method stores the raw samples, which may result in excessive memory usage. You may want to consider using relative.effect.table instead.

Author(s)

Gert van Valkenhoef, Joël Kuiper

See Also

rank.probability, relative.effect.table

Examples

```
model <- mtc.model(smoking)
# To save computation time we load the samples instead of running the model
## Not run: results <- mtc.run(model)
results <- readRDS(system.file("extdata/luades-smoking-samples.rds", package="gemtc"))

# Creates a forest plot of the relative effects
forest(relative.effect(results, "A"))

summary(relative.effect(results, "B", c("A", "C", "D")))
## Iterations = 5010:25000
## Thinning interval = 10
## Number of chains = 4
## Sample size per chain = 2000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean     SD Naive SE Time-series SE
```

```
## d.B.A -0.4965 0.4081 0.004563       0.004989
## d.B.C  0.3394 0.4144 0.004634       0.004859
## d.B.D  0.6123 0.4789 0.005354       0.005297
## sd.d   0.8465 0.1913 0.002139       0.002965
##
## 2. Quantiles for each variable:
##
##          2.5%     25%     50%     75%  97.5%
## d.B.A -1.3407 -0.7530 -0.4910 -0.2312 0.2985
## d.B.C -0.4809  0.0744  0.3411  0.5977 1.1702
## d.B.D -0.3083  0.3005  0.6044  0.9152 1.5790
## sd.d   0.5509  0.7119  0.8180  0.9542 1.2827
```

`relative.effect.table` *Table of relative effects*

Description

Generates a table of the relative effects of all pairs of treatments. Unlike `relative.effect`, this method stores summaries only, not raw samples.

Usage

```
relative.effect.table(result, covariate=NA)

## S3 method for class 'mtc.relative.effect.table'
print(x, ...)
## S3 method for class 'mtc.relative.effect.table'
forest(x, t1, use.description=FALSE, ...)
## S3 method for class 'mtc.relative.effect.table'
as.data.frame(x, ...)
```

Arguments

`result`	An object of S3 class `mtc.result` to derive the relative effects from.
`covariate`	(Regression analyses only) Value of the covariate at which to compute relative effects.
`x`	An object of S3 class `mtc.relative.effect.table`.
`t1`	Baseline treatment for the Forest plot.
`use.description`	Display treatment descriptions instead of treatment IDs.
`...`	Additional arguments.

Value

Returns an `mtc.relative.effect.table` object containing the quantiles of the calculated relative effects of all pair-wise comparisons among the treatments.

The result will be pretty printed as an n-by-n table of relative treatment effects. It can also be used to produce Forest plots against any arbitrary baseline. Finally, the `as.data.frame` generic method makes it possible to export the table for use in Excel or other spreadsheet software, using the core R methods `write.csv` or `write.csv2`.

Author(s)

Gert van Valkenhoef

See Also

`relative.effect`

Examples

```
model <- mtc.model(smoking)
# To save computation time we load the samples instead of running the model
## Not run: results <- mtc.run(model)
results <- readRDS(system.file("extdata/luades-smoking-samples.rds", package="gemtc"))

# Creates a forest plot of the relative effects
tbl <- relative.effect.table(results)

# Print the n*n table
print(tbl)

# Plot effect relative to treatment "C"
forest(tbl, "C")

# Write to CSV (e.g. to import to Excel, then use in a Word table)
## Not run: write.csv(tbl, "smoking-effects.csv")
# Note: use write.csv2 for Western European locales
```

---

smoking *Psychological treatments to aid smoking cessation*

Description

A dataset of 24 trials investigating four psychological treatments and no treatment for smoking cessation. The outcome is the number of people who stopped smoking.

Format

A network meta-analysis dataset containing 50 rows of arm-based data (responders and sample size).

Source

Lu and Ades (2006), *Assessing Evidence Inconsistency in Mixed Treatment Comparisons*, Journal of the American Statistical Society, 101(474):447-459. [doi:10.1198/016214505000001302]

Hasselblad (1998), *Meta-analysis of multitreatment studies*, Medical Decision Making 18(1):37-43. [doi:10.1177/0272989X9801800110]

`thrombolytic` *Thrombolytic treatment after acute myocardial infarction*

Description

A dataset of 28 trials investigating eight thrombolytic treatments administered after a myocardial infarction. The outcome is mortality after 30-35 days.

Format

A network meta-analysis dataset containing 58 rows of arm-based data (responders and sample size).

Source

Lu and Ades (2006), *Assessing Evidence Inconsistency in Mixed Treatment Comparisons*, Journal of the American Statistical Society, 101(474):447-459. [doi:10.1198/016214505000001302]

Boland et al. (2003), *Early thrombolysis for the treatment of acute myocardial infarction: a systematic review and economic evaluation*, Health Technology Assessment 7(15):1-136. [doi:10.3310/hta7150]

Index

[Dias et al. 2010, van Valkenhoef et al. 2015], *20*
[Dias et al. 2013a, van Valkenhoef et al. 2012], *20*
[Dias et al. 2013a], *20*
[Dias et al. 2013b, van Valkenhoef et al. (draft)], *20*
[Turner et al. 2012], *18*
[Warn et al. 2002], *20*
[van Valkenhoef et al. (draft)], *20*
[van Valkenhoef et al. 2014b (draft)], *13*
[van Valkenhoef et al. 2015], *26*

as.data.frame.mtc.relative.effect.table *(*relative.effect.table*)*, 34
as.mcmc.list.mtc.result *(*mtc.run*)*, 27
atrialFibrillation, 4, *20*

blobbogram, 5
blocker, 8

certolizumab, 9, *20*

depression, 10
dietfat, 10

forest *(*blobbogram*)*, 5
forest.mtc.relative.effect.table *(*relative.effect.table*)*, 34
forest.mtc.result *(*mtc.run*)*, 27

Gelman et al. (2008), *20*
gemtc *(*gemtc-package*)*, 2
gemtc-package, 2

hfPrevention, 11, *20*

ll.call, 11

mtc *(*gemtc-package*)*, 2
mtc.anohe, 13
mtc.data.studyrow, 14, *24*
mtc.deviance, 16, *16*
mtc.devplot *(*mtc.deviance*)*, 16
mtc.hy.empirical.lor *(*mtc.hy.prior*)*, 17
mtc.hy.prior, 17, *19*
mtc.levplot *(*mtc.deviance*)*, 16
mtc.model, *3*, *4*, *12–14*, *17*, 18, *18*, *23*, *24*, *26*, *28*
mtc.network, *4*, *13*, *15*, *22*, 23, *26*
mtc.nodesplit, 25
mtc.result, *16*
mtc.result *(*mtc.run*)*, 27
mtc.run, *4*, *13*, *14*, *17*, *22*, *26*, 27, *31*

parkinson, 29
parkinson_diff *(*parkinson*)*, 29
parkinson_shared *(*parkinson*)*, 29
plot.mtc.anohe *(*mtc.anohe*)*, 13
plot.mtc.deviance *(*mtc.deviance*)*, 16
plot.mtc.model *(*mtc.model*)*, 18
plot.mtc.network *(*mtc.network*)*, 23
plot.mtc.nodesplit *(*mtc.nodesplit*)*, 25
plot.mtc.rank.probability *(*rank.probability*)*, 31
plot.mtc.result *(*mtc.run*)*, 27
plotCovariateEffect, 30
print.mtc.anohe *(*mtc.anohe*)*, 13
print.mtc.model *(*mtc.model*)*, 18
print.mtc.nodesplit *(*mtc.nodesplit*)*, 25
print.mtc.rank.probability *(*rank.probability*)*, 31
print.mtc.relative.effect.table *(*relative.effect.table*)*, 34
print.mtc.result *(*mtc.run*)*, 27

rank.probability, *28*, 31, *33*
rank.quantiles *(*rank.probability*)*, 31
relative.effect, *28*, *31*, 32, *32*, *34*, *35*
relative.effect.table, *28*, *33*, 34

smoking, 35
sucra *(*rank.probability*)*, 31
summary.mtc.anohe *(*mtc.anohe*)*, 13
summary.mtc.model *(*mtc.model*)*, 18
summary.mtc.nodesplit *(*mtc.nodesplit*)*, 25
summary.mtc.result *(*mtc.run*)*, 27

thrombolytic, 36