# Package ‘gemtc’

February 19, 2015

**Version** 0.6-1  
**Date** 2014-12-12  
**Title** Network Meta-Analysis Using Bayesian Methods  
**Author** Gert van Valkenhoef, Joel Kuiper  
**Maintainer** Gert van Valkenhoef <g.h.m.van.valkenhoef@rug.nl>  
**Description** Network meta-analyses (mixed treatment comparisons) in the Bayesian framework. You need either BUGS or JAGS to use this package, preferably JAGS (through the rjags package). Includes methods to assess heterogeneity and inconsistency, and a number of standard visualizations.

**Depends**  
coda (>= 0.13)

**Imports**  
igraph (>= 0.6.4), meta (>= 2.1), XML (>= 3.6)

**Suggests**  
rjags (>= 3-0), BRugs (>= 0.8), R2WinBUGS (>= 2.1), testthat (>= 0.8), Matrix

**URL**  
http://github.com/gertvv/gemtc

**License** GPL-3

**NeedsCompilation** yes  
**Repository** CRAN  
**Date/Publication** 2014-12-12 20:34:43

## R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>gemtc-package</td>
<td>2</td>
</tr>
<tr>
<td>blobbogram</td>
<td>3</td>
</tr>
<tr>
<td>ll.call</td>
<td>6</td>
</tr>
<tr>
<td>mtc.anohe</td>
<td>7</td>
</tr>
<tr>
<td>mtc.data.studyrow</td>
<td>8</td>
</tr>
<tr>
<td>mtc.hy.prior</td>
<td>10</td>
</tr>
<tr>
<td>mtc.model</td>
<td>11</td>
</tr>
<tr>
<td>mtc.network</td>
<td>14</td>
</tr>
<tr>
<td>mtc.nodessplit</td>
<td>17</td>
</tr>
<tr>
<td>mtc.run</td>
<td>19</td>
</tr>
<tr>
<td>rank.probability</td>
<td>20</td>
</tr>
<tr>
<td>relative.effect</td>
<td>22</td>
</tr>
</tbody>
</table>
gemtc-package

Index

24

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. 2013]. 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] that can be run using JAGS (using the rjags package), OpenBUGS (using the BRugs package) or WinBUGS (using the R2WinBUGS package). Note that there is a known issue with BUGS that prevents some models from running, though the remaining models appear to run correctly. It is highly recommended to use JAGS rather than BUGS.

This package is interoperable with GeMTC files that were created by the GeMTC GUI or exported from ADDIS. The source for GeMTC is available under the GPL-3 on Github.

See http://drugis.org/gemtc for more information.

Author(s)

Gert van Valkenhoef

References


G. van Valkenhoef, S. Dias, A.E. Ades, and N.J. Welton (2014a), Automated generation of node-splitting models for the assessment of inconsistency in network meta-analysis, draft manuscript.
G. van Valkenhoef et al. (2014b), *Modeling inconsistency as heterogeneity in network meta-analysis*, draft manuscript.

**See Also**

`mtc.network`, `mtc.model`, `mtc.run`

**Examples**

```r
# Load the example network and generate a consistency model:
file <- system.file("extdata/luades-smoking.gemtc", package="gemtc")
network <- read.mtc.network(file)
model <- mtc.model(network, type="consistency")

# Load pre-generated samples instead of running the model:
## Not run: results <- mtc.run(model, thin=10)
results <- dget(system.file("extdata/luades-smoking.samples.gz", 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:
##
## d.A.B         2.5%  25%  50%   75%  97.5%
##               -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
```

---

**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,
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))

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.
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 en-
compass 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.

Details

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
```r
data <- read.table(textConnection('id group pe ci.l ci.u style value.A value.B
"Study 1" 1 0.35 0.08 0.62 "normal" "2/46" "7/46"
"Study 2" 1 0.43 0.15 1.14 "normal" "4/59" "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)

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/BUGS 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.init: 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.

Value

The return value of the called function.

Author(s)

Gert van Valkenhoef
mtc.anohe

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

Description

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

Usage

```r
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('
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

mtc.data.studyrow
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.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**

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

```r
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 (BUGS/JAGS syntax). Typical ones would be 'dunif' (uniform), 'dgamma' (Gamma), or 'dlnorm' (log-normal).
- `...` 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 BUGS 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

```r
file <- system.file("extdata/luades-smoking.gemtc", package="gemtc")
network <- read.mtc.network(file)

# 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(network, 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(network, 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(network, 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**

```r
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"), ...)
```
### Arguments

- **network**: An object of S3 class `mtc.network`
- **type**: A string literal indicating the type of model (either "consistency", "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`.
- ... 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.
- **binom/logit**: for dichotomous data.
  Required columns `[responders, sampleSize]. Result: (log) odds 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.

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: \( t_1 \) and \( t_2 \) (Dias et al. 2010, van Valkenhoef et al. 2014a (draft))

• **use**: unrelated study effects. Models the effects within each study as if the studies are independent. No additional parameters. [van Valkenhoef et al. 2014b (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. 2014b (draft)]

### 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

These elements determine the model run by JAGS/BUGS:

- **n.chain**: The number of chains
- **code**: Model code in JAGS/BUGS syntax. Use `cat()` for proper formatting.
- **data**: Data in JAGS/BUGS compatible format
- **inits**: Initial values in JAGS/BUGS compatible format
- **monitors**: The nodes of the JAGS/BUGS model to monitor

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. The default up-to version 0.1-2 was `igraph::layout.kamada.kawai` but could produce overlapping edges.

**Author(s)**

Gert van Valkenhoef, Joël Kuiper

**See Also**

`mtc.network`, `mtc.run`

**Examples**

```r
file <- system.file("extdata/parkinson.gemtc", package="gemtc")
network <- read.mtc.network(file)
model <- mtc.model(network)
plot(model)
summary(model)
```

---

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

**Description**

Creates an object of class `mtc.network`
mtc.network

Usage

mtc.network(data.ab, treatments, description, data.re, data)
read.mtc.network(file)
write.mtc.network(network, file)

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.
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.
network    An object of the S3 class mtc.network.
file    Path to the file to read (read.mtc.network) or write (write.mtc.network). For write.mtc.network, if file="", the output is printed to standard output. If it is file="|cmd", the output is piped to the command given by cmd. See base::cat for further details.

Details

One-arm trials are automatically removed, which results in a warning.
read.mtc.network and write.mtc.network deal with the older GeMTC XML format. This format supports dichotomous data as responders/sampleSize and continuous data as mean/std.dev/sampleSize only. Relative effect data, or other forms of arm-based data, are not supported. In general, it may be more convenient to use dput and dget to read and write networks.
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 and read.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)

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 -0.8 0.7 33
## 4 2 B -1.10 0.5 31
# Read an example GeMTC XML file
file <- system.file("extdata/luades-smoking.gemtc", package="gemtc")
network <- read.mtc.network(file)

# Summarize the network (generate some interesting network properties)
summary(network)

## Description
## [1] "MTC dataset: Smoking cessation rates"
##
## 'Studies per treatment'
## A B C D
## 19 6 19 6
##
## 'Number of n-arm studies'
## 2-arm 3-arm
## 22 2

# Write the network to a temporary file in the GeMTC XML format
tmp <- tempfile()
write.mtc.network(network, file=tmp)

---

### 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

```r
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. 2014a (draft)]. 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.eff` Summary of direct effects for each split comparison
- `ind.eff` Summary of indirect effects for each split comparison
- `cons.eff` 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**

```r
# Read the network
fileName <- system.file('extdata/parkinson.gemtc', package='gemtc')
network <- read.mtc.network(fileName)

# Run all relevant node-splitting models
## Not run: result.ns <- mtc.nodesplit(network, 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)
```
### Description

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

### Usage

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

### Arguments

- **model**: An object of S3 class `mtc.model` describing a network meta-analysis model.
- **sampler**: A string literal indicating which sampler to use. Allowed values are 'JAGS' or 'BUGS'. Alternatively, the specific package to be used can be specified: 'rjags', 'BRugs' or 'R2WinBUGS'. When unspecified it will try 'rjags', 'BRugs', and 'R2WinBUGS', in that order.
- **n.adapt**: Amount of adaptation (or tuning) iterations.
- **n.iter**: Amount of simulation iterations.
- **thin**: Thinning factor.

### 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.
- **sampler**: The sampler (R package) used to produce the samples.

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, rank.probability, coda::gelman.diag, coda::gelman.plot

Examples

```r
file <- system.file("extdata/luades-smoking.gemtc", package="gemtc")
network <- read.mtc.network(file)
model <- mtc.model(network)

## Not run: results <- mtc.run(model, thin=10)
results <- dget(system.file("extdata/luades-smoking.samples.gz", 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 0.004809     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
```

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)`

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.

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

A matrix with the treatments as rows and the ranks as columns. The matrix is given class `mtc.rank.probability`, for which `print` and `plot` are overridden.

Author(s)

Gert van Valkenhoef, Joël Kuiper

See Also

- `relative.effect`

Examples

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

ranks <- rank.probability(results)
print(ranks)
## Rank probability; preferred direction = 1
## A 0.0000000 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
```
relative.effect  

Calculating relative effects

Description
Calculates the relative effects of pairs of treatments.

Usage

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

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.w and sd.d.

Value
Returns an mtc.result object containing the calculated relative effects.

Author(s)
Gert van Valkenhoef, Joël Kuiper

See Also

rank.probability

Examples

file <- system.file("extdata/luades-smoking.gemtc", package="gemtc")
network <- read.mtc.network(file)
model <- mtc.model(network)
# To save computation time we load the samples instead of running the model
## Not run: results <- mtc.run(model)
results <- dget(system.file("extdata/luades-smoking.samples.gz", 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 |
Index

[Dias et al. 2010, van Valkenhoef et al. 2014a (draft)], 13
[Dias et al. 2013a, van Valkenhoef et al. 2012], 12
[Dias et al. 2013b, van Valkenhoef et al. 2014b (draft)], 13
[Turner et al. 2012], 11
[van Valkenhoef et al. 2014a (draft)], 17
[van Valkenhoef et al. 2014b (draft)], 7, 13
as.mcmc.list.mtc.result (mtc.run), 19
print.mtc.result (mtc.run), 19
rank.probability, 19, 20, 20, 22
read.mtc.network (mtc.network), 14
relative.effect, 19–21, 22
summary.mtc.anohe (mtc.anohe), 7
summary.mtc.model (mtc.model), 11
summary.mtc.nodesplit (mtc.nodesplit), 17
summary.mtc.result (mtc.run), 19
write.mtc.network (mtc.network), 14

blobbogram, 3
forest (relative.effect), 22
forest.mtc.result (mtc.run), 19
gemtc (gemtc-package), 2
gemtc-package, 2
ll.call, 6
mtc (gemtc-package), 2
mtc.anohe, 7
mtc.data.studyrow, 8, 15, 16
mtc.hy.empirical.lor (mtc.hy.prior), 10
mtc.hy.prior, 10, 12
mtc.model, 3, 7, 8, 10, 11, 11, 15–18, 20
mtc.network, 3, 7, 9, 14, 14, 17
mtc.nodesplit, 17
mtc.run, 3, 7, 8, 14, 17, 18, 19
plot.mtc.anohe (mtc.anohe), 7
plot.mtc.model (mtc.model), 11
plot.mtc.nodesplit (mtc.nodesplit), 17
plot.mtc.result (mtc.run), 19
print.mtc.anohe (mtc.anohe), 7
print.mtc.model (mtc.model), 11
print.mtc.nodesplit (mtc.nodesplit), 17