Package 'bnma'

August 27, 2020

Type Package

Title Bayesian Network Meta-Analysis using 'JAGS'

Version 1.3.0

Date 2020-07-28

Depends R (>= 2.10)

Imports rjags (>= 4-6), graphics, stats, utils, coda (>= 0.13), ggplot2, grid, igraph

Description Network meta-analyses using Bayesian framework following Dias et al. (2013) <DOI:10.1177/0272989X12458724>. Based on the data input, creates prior, model file, and initial values needed to run models in 'rjags'. Able to handle binomial, normal and multinomial arm-based data. Can handle multi-arm trials and includes methods to incorporate covariate and baseline risk effects. Includes standard diagnostics and visualization tools to evaluate the results.

License GPL-3

LazyData true

RoxygenNote 7.0.2

Encoding UTF-8

VignetteBuilder knitr

Suggests knitr, rmarkdown

NeedsCompilation no

Author Michael Seo [aut, cre], Christopher Schmid [aut]

Maintainer Michael Seo <swj8874@gmail.com>

Repository CRAN

Date/Publication 2020-08-27 05:10:28 UTC

R topics documented:

bnma-package *bnma: A package for network meta analysis using Bayesian methods*

Description

A package for running Bayesian network meta analysis

Details

Network meta-analysis or mixed treatment comparison (MTC) is a method that allows simultaneous comparison of more than two treatments. We use a Bayesian approach to combine both direct and indirect evidence as in Dias et al. 2013a. This package is a user friendly application that can run network meta analysis models without having to code a JAGS model. The program takes the input data and transforms it to a suitable format of analysis, generates a JAGS model and reasonable initial values and runs the model through the rjags package. The focus of this package was inclusion of multinomial response and various options for adding covariates and/or baseline risks effects. Also, while sampling, the package uses Gelman-Rubin convergence criteria to decide whether to continue sampling or not. Furthermore, package includes different models such as contrast based models and unrelated mean effects (UME) model and nodesplitting model to test for inconsistency.

References

A.J. Franchini, S. Dias, A.E. Ades, J.P. Jansen, N.J. Welton (2012), *Accounting for correlation in network meta-analysis with multi-arm trials*, Research Synthesis Methods 3(2):142-160. [https://doi.org/10.1002/jrsm.1049]

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. [https://doi.org/10.1177/0272989X12458724]

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

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

S. Dias, N.J. Welton, A.J. Sutton, D.M. Caldwell, G. Lu, and A.E. Ades (2013), *Evidence synthesis for decision making 4: inconsistency in networks of evidence based on randomized controlled trials*, Medical Decision Making 33(5):641-656. [https://doi.org/10.1177/0272989X12455847]

C.H. Schmid, T.A. Trikalinos, I. Olkin (2014), *Bayesian network meta-analysis for unordered categorical outcomes with incomplete data*, Research Synthesis Methods 5(2):162-185. [https://doi.org/10.1002/jrsm.1103]

A. Gelman, D.B. Rubin (1992), *Inference from iterative simulation using multiple sequences*, Statistical Science 7(4):457-472. [http://dx.doi.org/10.1214/ss/1177011136]

D.J. Spiegelhalter, N.G. Best, and B.P. Carlin (1998), *Bayesian deviance, the effective nunmber of parameters, and the comparison of arbitrarily complex models*, Technical report, MRC Biostatistics Unit, Cambridge, UK.

F.A. Achana, N.J. Cooper, S. Dias, G. Lu, S.J.C. Rice, D. Kendrick, A.J. Sutton (2012), *Extending methods for investigating the relationship between treatment effect and baseline risk from pairwise meta-analysis to network meta-analysis*, Statistics in Medicine 32(5):752-771. [https://doi.org/10.1002/sim.5539]

F.A. Achana, N.J. Cooper, S. Bujkiewicz, S.J. Hubbard, D. Kendrick, D.R. Jones, A.J. Sutton (2014), *Network meta-analysis of multiple outcomes measures accounting for borrowing of information across outcomes*, BMC Medical Research Methodology 14:92. [https://doi.org/10.1186/1471-2288-14-92]

G. Salanti, A.E. Ades, J.P.A. Ioannidisa (2011), *Graphical methods and numerical summaries for presenting results from multiple-treatment meta-analysis: an overview and tutorial*, Journal of Clinical Epidemiology 64(2):163-171. [https://doi.org/10.1016/j.jclinepi.2010.03.016]

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. [https://doi.org/10.1002/jrsm.1054]

N.J. Cooper, A.J. Sutton, D. Morris, A.E. Ades, N.J. Welton (2009), *Addressing between-study heterogeneity and inconsistency in mixed treatment comparisons: Application to stroke prevention treatments in individuals with non-rheumatic atrial fibrillation*, Statistics in Medicine 28:1861-1881. [https://doi.org/10.1002/sim.3594]

W. Viechtbauer (2010), *Conducting meta-analyses in R with the metafor package*, Journal of Statistical Software, 36(3):1-48. [https://doi.org/10.18637/jss.v036.i03]

See Also

network.data, network.run

`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. Control is coded as 1 and beta blocker treatment is coded as 2.

Usage

```
blocker
```

Format

A list of Outcomes, Treat, Study, and N.

`calculate.contrast.deviance` *Find deviance statistics such as DIC and pD.*

Description

Calculates deviance statistics. This function automatically called in `contrast.network.run` and the deviance statistics are stored after sampling is finished.

Usage

```
calculate.contrast.deviance(result)
```

Arguments

`result`	Object created by `contrast.network.run` function

Value

`Dbar`	Overall residual deviance
`pD`	Sum of leverage_arm (i.e. total leverage)
`DIC`	Deviance information criteria (sum of Dbar and pD)
`resdev_study`	Posterior mean of the residual deviance in each study
`devtilda_study`	Deviance at the posterior mean of the fitted values
`leverage_study`	Difference between resdev_study and devtilda_study for each trial

References

A.J. Franchini, S. Dias, A.E. Ades, J.P. Jansen, N.J. Welton (2012), *Accounting for correlation in network meta-analysis with multi-arm trials*, Research Synthesis Methods 3(2):142-160. [https://doi.org/10.1002/jrsm.1049]

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. [https://doi.org/10.1177/0272989X12458724]

Examples

```
network <- with(parkinsons_contrast, {
 contrast.network.data(Outcomes, Treat, SE, na, V)
})

result <- contrast.network.run(network)
calculate.contrast.deviance(result)
```

`calculate.deviance` *Find deviance statistics such as DIC and pD.*

Description

Calculates deviance statistics. This function automatically called in `network.run` and the deviance statistics are stored after sampling is finished.

Usage

```
calculate.deviance(result)
```

Arguments

`result`	Object created by `network.run` function

Value

`Dbar`	Overall residual deviance
`pD`	Sum of leverage_arm (i.e. total leverage)
`DIC`	Deviance information criteria (sum of Dbar and pD)
`data.points`	Total number of arms in the meta analysis
`dev_arm`	Posterior mean of the residual deviance in each trial arm
`devtilda_arm`	Deviance at the posterior mean of the fitted values
`leverage_arm`	Difference between dev_arm and devtilda_arm for each trial
`rtilda_arm`	Posterior mean of the fitted value for binomial and multinomial
`ybar_arm`	Posterior mean of the fitted value for normal

References

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. [https://doi.org/10.1177/0272989X12458724]

Examples

```
#parkinsons
network <- with(parkinsons, {
 network.data(Outcomes, Study, Treat, SE = SE, response = "normal")
})

result <- network.run(network)
calculate.deviance(result)
```

cardiovascular — *Trials of low dose and high dose statins for cardiovascular disease vs. placebo*

Description

A dataset of 17 studies investigating dosage of statin for cardiovascular disease. There are two treatments and a placebo. High dose statin is coded as 3, low dose statin as 2, and placebo is coded as 1 and treated as a baseline treatment. Outcomes are reported as three mutually exclusive unordered outcomes. First column of the outcome is the patients who are still alive (ALIVE). Second column is fatal non-cardiovascular disease (FnCVD). And, the last column is fatal cardiovascular disease (FCVD).

Usage

```
cardiovascular
```

Format

A list of Outcomes, Treat, Study, and N

References

C.H. Schmid, T.A. Trikalinos, I. Olkin (2014), *Bayesian network meta-analysis for unordered categorical outcomes with incomplete data*, Research Synthesis Methods 5(2):162-185. [https://doi.org/10.1002/jrsm.1103]

certolizumab — *Trials of certolizumab pegol (CZP) for the treatment of rheumatoid arthritis in patients*

Description

A dataset of 12 trials for investigating CZP for the treatment for those who had failed on disease-modifying antirheumatic drugs, including methotrexate (MTX). Data provides the number of patients who have improved and there are 6 different treatments with placebo. Mean disease duration (years) is provided as a covariate.

Usage

```
certolizumab
```

Format

A list of Outcomes, Treat, Study, N, covariate, and Treat.order

References

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

`contrast.network.data` *Make a network object for contrast-level data containing data, priors, and a JAGS model file*

Description

This is similar to the function `network.data`, except it uses contrast-level data instead of arms-level data. Contrast-level format uses treatment differences relative to the control arm. Note that in two arm trials there is only one contrast value per trial, but in three arm trials there are two contrast values relative to the control arm.

Usage

```
contrast.network.data(
  Outcomes,
  Treat,
  SE,
  na,
  V = NULL,
  type = "random",
  rank.preference = "higher",
  mean.d = 0,
  prec.d = 1e-04,
  hy.prior = list("dunif", 0, 100)
)
```

Arguments

`Outcomes`	A vector of Contrast-level outcomes. Outcome is assumed to be normally distributed. If there are three arms in a trial, need to include two contrast values for that trial. See parkinsons_contrast data for an example.
`Treat`	A vector of treatments for each arm. Treatments should have positive integer values starting from 1 to total number of treatments.
`SE`	A vector of standard error for each contrasts.
`na`	A vector of number of arms in each study.
`V`	Needed if you have multi-arm trials. Length of this vector should be number of studies. If the study is multi-arm trial, need to specify variance of the baseline treatment in that trial. Denote it with NA if the study only has two-arm trials.
`type`	Type of model fitted: either "random" for random effects model or "fixed" for fixed effects model. Default is "random".

`rank.preference`
: Set it equal to "higher" if higher values are preferred (i.e. assumes events are good). Set it equal to "lower" if lower values are preferred (i.e. assumes events are bad). Default is "higher".

`mean.d`
: Prior mean for the relative effect

`prec.d`
: Prior precision for the relative effect

`hy.prior`
: Prior for the heterogeneity parameter. Supports uniform, gamma, and half normal for normal. It should be a list of length 3, where first element should be the distribution (one of dunif, dgamma, dhnorm, dwish) and the next two are the parameters associated with the distribution. For example, list("dunif", 0, 5) give uniform prior with lower bound 0 and upper bound 5 for the heterogeneity parameter.

Value

Creates list of variables that are used to run the model using `contrast.network.run`

References

A.J. Franchini, S. Dias, A.E. Ades, J.P. Jansen, N.J. Welton (2012), *Accounting for correlation in network meta-analysis with multi-arm trials*, Research Synthesis Methods 3(2):142-160. [https://doi.org/10.1002/jrsm.1049]

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. [https://doi.org/10.1177/0272989X12458724]

Examples

```
network <- with(parkinsons_contrast, {
 contrast.network.data(Outcomes, Treat, SE, na, V)
})
```

---

contrast.network.deviance.plot

Make a contrast network deviance plot

Description

This makes a contrasrt network deviance plot which plots residual deviance (resdev_study) vs. all study.

Usage

```
contrast.network.deviance.plot(result)
```

Arguments

`result`
: Object created by `contrast.network.run` function

Value

None

Examples

```
network <- with(parkinsons_contrast, {
contrast.network.data(Outcomes, Treat, SE, na, V)
})

result <- contrast.network.run(network)
contrast.network.deviance.plot(result)
```

`contrast.network.leverage.plot` *Make a leverage plot*

Description

This function makes a leverage vs. square root of residual deviance plot

Usage

```
contrast.network.leverage.plot(result)
```

Arguments

`result`	Object created by `contrast.network.run` function

Value

None

Examples

```
network <- with(parkinsons_contrast, {
contrast.network.data(Outcomes, Treat, SE, na, V)
})

result <- contrast.network.run(network)
contrast.network.leverage.plot(result)
```

`contrast.network.run` *Run the model using the network object*

Description

This is similar to the function `network.run`, except it uses contrast-level data instead of arms-level data.

Usage

```
contrast.network.run(
  network,
  inits = NULL,
  n.chains = 3,
  max.run = 1e+05,
  setsize = 10000,
  n.run = 50000,
  conv.limit = 1.05,
  extra.pars.save = NULL
)
```

Arguments

`network`	contrast level network object created from `contrast.network.data` function
`inits`	Initial values for the parameters being sampled. If left unspecified, program will generate reasonable initial values.
`n.chains`	Number of chains to run
`max.run`	Maximum number of iterations that user is willing to run. If the algorithm is not converging, it will run up to `max.run` iterations before printing a message that it did not converge
`setsize`	Number of iterations that are run between convergence checks. If the algorithm converges fast, user wouldn't need a big setsize. The number that is printed between each convergence checks is the gelman-rubin diagnostics and we would want that to be below the conv.limit the user specifies.
`n.run`	Final number of iterations that the user wants to store. If after the algorithm converges, user wants less number of iterations, we thin the sequence. If the user wants more iterations, we run extra iterations to reach the specified number of runs
`conv.limit`	Convergence limit for Gelman and Rubin's convergence diagnostic. Point estimate is used to test convergence of parameters for study effect (eta), relative effect (d), and heterogeneity (log variance (logvar)).
`extra.pars.save`	Parameters that user wants to save besides the default parameters saved. See code using `cat(network$code)` to see which parameters can be saved.

Value

data_rjags	Data that is put into rjags function jags.model
inits	Initial values that are either specified by the user or generated as a default
pars.save	Parameters that are saved. Add more parameters in extra.pars.save if other variables are desired
burnin	Half of the converged sequence is thrown out as a burnin
n.thin	If the number of iterations user wants (n.run) is less than the number of converged sequence after burnin, we thin the sequence and store the thinning interval
samples	MCMC samples stored using jags. The returned samples have the form of mcmc.list and can be directly applied to coda functions
max.gelman	Maximum Gelman and Rubin's convergence diagnostic calculated for the final sample
deviance	Contains deviance statistics such as pD (effective number of parameters) and DIC (Deviance Information Criterion)
rank.tx	Rank probability calculated for each treatments. rank.preference parameter in contrast.network.data is used to define whether higher or lower value is preferred. The numbers are probabilities that a given treatment has been in certain rank in the sequence.

Examples

```
network <- with(parkinsons_contrast, {
 contrast.network.data(Outcomes, Treat, SE, na, V)
})

result <- contrast.network.run(network)
```

---

draw.network.graph *Draws network graph using igraph package*

Description

This function draws network graph using igraph package

Usage

```
draw.network.graph(network, label.dist = 2)
```

Arguments

network	Object created by network.data function
label.dist	distance of the label from the node. Default is 2.

Value

None

Examples

```
#cardiovascular
network <- with(thrombolytic, {
 network.data(Outcomes, Study, Treat, N=N, response = "binomial")
})
draw.network.graph(network)
```

`network.autocorr.diag` *Generate autocorrelation diagnostics using coda package*

Description

This function generates autocorrelation diagnostics using coda package. User can specify lags and parameters to display. Note that to display extra parameters that are not saved, user needs to first specify parameters in `extra.pars.save` parameter in `network.run` function.

Usage

```
network.autocorr.diag(
  result,
  lags = c(0, 1, 5, 10, 50),
  extra.pars = NULL,
  only.pars = NULL
)
```

Arguments

`result`	Object created by `network.run` function
`lags`	A vector of lags at which to calculate the autocorrelation
`extra.pars`	Extra parameters that the user wants to display other than the default parameters.
`only.pars`	Parameters that user wants to display. This gets rids of other default parameters user doesn't want to show.

Value

Returns autocorrelation diagnostics

Examples

```
network <- with(blocker, {
 network.data(Outcomes, Study, Treat, N = N, response = "binomial")
})

result <- network.run(network)
network.autocorr.diag(result, only.pars = "d")
```

`network.autocorr.plot` *Generate autocorrelation plot using coda package*

Description

This function plots autocorrelation using coda package.

Usage

```
network.autocorr.plot(result, extra.pars = NULL, only.pars = NULL)
```

Arguments

`result`	Object created by `network.run` function
`extra.pars`	Extra parameters that the user wants to plot other than the default parameters.
`only.pars`	Parameters that user wants to display. This gets rids of other default parameters user doesn't want to show

Value

None

Examples

```
network <- with(blocker, {
 network.data(Outcomes, Study, Treat, N = N, response = "binomial")
})

result <- network.run(network)
network.autocorr.plot(result)
```

network.covariate.plot *Make a covariate plot*

Description

This function makes a covariate plot of how the relative effect changes as the covariate value changes. User needs to specify one base treatment and one comparison treatment to make this plot (base category and comparison category is also needed for multinomial). The function uses the relative.effects to calculate the correct relative effect. 2.5%, median, and 97.5% C.I. are drawn.

Usage

```
network.covariate.plot(
  result,
  base.treatment = NULL,
  comparison.treatment = NULL,
  base.category = NULL,
  comparison.category = NULL,
  covariate.name = NULL
)
```

Arguments

result	Object created by network.run function
base.treatment	Base treatment for relative effect
comparison.treatment	Treatment comparing against base treatment
base.category	Base category for multinomial data. Note that category in multinomial denotes which column it is in the Outcomes matrix. Thus, this should be a numeric value.
comparison.category	Comparison category for multinomial data
covariate.name	A vector of covariate names of the covariate that goes into x-axis label

Value

None

Examples

```
########### certolizumab (with covariate)
network <- with(certolizumab, {
 network.data(Outcomes, Study, Treat, N=N, response="binomial", Treat.order,
 covariate = covariate, hy.prior = list("dhnorm", 0, 9.77))
})
```

```
result <- network.run(network)
network.covariate.plot(result, base.treatment = "Placebo", comparison.treatment = "CZP",
covariate.name = "Disease Duration")
```

`network.cumrank.tx.plot` *Create a treatment cumulative rank plot*

Description

This function creates a treatment cumulative rank plot. Rank preference can be specified by the `rank.preference` parameter in `network.data`

Usage

```
network.cumrank.tx.plot(
  result,
  txnames = NULL,
  catnames = NULL,
  legend.position = c(1, 1)
)
```

Arguments

`result`	Object created by `network.run` function
`txnames`	Treatment names used in creating legend
`catnames`	Category names. Only used in multinomial.
`legend.position`	x, y position of the legend

Value

None

See Also

`rank.tx`

Examples

```
network <- with(blocker, {
 network.data(Outcomes, Study, Treat, N = N, response = "binomial")
})

result <- network.run(network)
network.cumrank.tx.plot(result, txnames = c("control", "beta blocker"))
```

`network.data` *Make a network object containing data, priors, and a JAGS model file*

Description

This function makes a network object that can be used to run network meta-analysis using `network.run`. User needs to specify Outcomes, Study, Treat, N or SE, and response. Prior parameters are filled in automatically based on the data type if not specified. The input data should be arm-level so that we have observations for each treatment in each study. The input data is preprocessed to fit the format necessary to run model in JAGS.

Usage

```
network.data(
  Outcomes = NULL,
  Study = NULL,
  Treat = NULL,
  N = NULL,
  SE = NULL,
  response = NULL,
  Treat.order = NULL,
  type = "random",
  rank.preference = "higher",
  baseline = "none",
  baseline.risk = "independent",
  covariate = NULL,
  covariate.type = NULL,
  covariate.model = NULL,
  mean.d = NULL,
  prec.d = NULL,
  mean.Eta = NULL,
  prec.Eta = NULL,
  hy.prior.Eta = NULL,
  mean.bl = NULL,
  prec.bl = NULL,
  hy.prior.bl = NULL,
  mean.cov = NULL,
  prec.cov = NULL,
  hy.prior.cov = NULL,
  hy.prior = NULL,
  mean.A = NULL,
  prec.A = NULL
)
```

Arguments

`Outcomes`	Arm-level outcomes. If it is a multinomial response, the matrix would have dimensions treatment arms (row) by multinomial categories (column). If it is

	binomial or normal, it would be a vector.
Study	A vector of study indicator for each arm
Treat	A vector of treatment indicator for each arm
N	A vector of total number of observations in each arm. Used for binomial and multinomial responses.
SE	A vector of standard error for each arm. Used only for normal response.
response	Specification of the outcomes type. Must specify one of the following: "normal", "binomial", or "multinomial".
Treat.order	Treatment order which determines how treatments are compared. The first treatment that is specified is considered to be the baseline treatment. Default order is alphabetical. If the treatments are coded 1, 2, etc, then the treatment with a value of 1 would be assigned as a baseline treatment.
type	Type of model fitted: either "random" for random effects model or "fixed" for fixed effects model. Default is "random".
rank.preference	Set it equal to "higher" if higher values are preferred (i.e. assumes events are good). Set it equal to "lower" if lower values are preferred (i.e. assumes events are bad). Default is "higher".
baseline	Three different assumptions for treatment x baseline risk interactions (slopes): "independent", "common", or "exchangeable". Default is "none" which doesn't incorporate baseline risk.
baseline.risk	Two different assumptions for baseline risk: "independent" or "exchangeable". See Achana et al. (2012) for more information about baseline risk.
covariate	A covariate matrix with each row representing each trial and column representing each covariate. This is a study-level data, meaning that the user doesn't need to repeatedly specify covariates for each arm.
covariate.type	Should be a vector indicating the type of the covariate. Covariate can be either "continuous" or "discrete". If it continuous, covariates are centered. If the covariate is discrete it is not centered and it has to be in a dummy integer format (i.e. 0,1,2,...). The code doesn't factor the covariates for the user, so user needs to specify dummy variables if factor is needed.
covariate.model	"independent" allows covariate effects for each treatment. "common" restricts same covariate effect for all treatment. Lastly, "exchangeable" assumes that the covariate effects are different but related and strength is borrowed across them. We set "common" to be default. See Cooper et al. (2009) for more details on covariates.
mean.d	Prior mean for the relative effect
prec.d	Prior precision for the relative effect
mean.Eta	Prior mean for the study effect (baseline risk)
prec.Eta	Prior precision for the study effect (baseline risk)
hy.prior.Eta	Between treatment heterogeneity in baseline risk (for exchangeable assumption only). Format of the parameter is same as hy.prior.

mean.bl	Prior mean for the baseline slope
prec.bl	Prior precision for the baseline slope
hy.prior.bl	Between treatment heterogeneity in baseline slope (for exchangeable regression coefficient only). Format of the parameter is same as hy.prior.
mean.cov	Prior mean for the covariate effect
prec.cov	Prior precision for the covariate effect
hy.prior.cov	Between treatment heterogeneity in covariate effect (for exchangeable regression coefficient only). Format of the parameter is same as hy.prior. Default is set to be dunif(0, 5) for binary, dunif(0, 100) for normal, and wishart with identity scale matrix and (# of categories - 1) degrees of freedom for multinomial.
hy.prior	Prior for the heterogeneity parameter. Supports uniform, gamma, and half normal for normal and binomial response and wishart for multinomial response. It should be a list of length 3, where first element should be the distribution (one of dunif, dgamma, dhnorm, dwish) and the next two are the parameters associated with the distribution. For example, list("dunif", 0, 5) give uniform prior with lower bound 0 and upper bound 5 for the heterogeneity parameter. For wishart distribution, the last two parameter would be the scale matrix and the degrees of freedom.
mean.A	Mean effect of 'standard' treatment (i.e. placebo). It is in logit scale for binomial and continuous scale for normal. For binomial outcome, this additional input is used to calculate the risk difference, relative risk, or number needed to treat. This should be informed from external evidence or can be found by meta-analyzing single proportions. For number needed to treat, we assume that events are "good". Reversal of sign is needed if the events are "bad".
prec.A	Precision of 'standard' treatment. Similarly, it is in logit scale for binomial and continuous scale for normal.

Value

Creates list of variables that are used to run the model using network.run

data	Data combining all the input data. User can check this to insure the data is correctly specified. For modelling purposes, character valued studies or treatment variables are changed to numeric values based on alphabetical order.
nrow	Total number of arms in the meta-analysis
ncat	Number of columns in the Outcomes. Will equal 1 for binary and normal and number of categories for multinomial
nstudy	Number of study
na	Number of arms for each study
ntreat	Number of treatment
b.id	Indicator in sequence of all treatments for which treatment is base treatment in Study
t	Treat transformed into a matrix which has dimensions number of study by max number of arms in studies

r	Outcomes made into an array that is suitable for use in rjags code. For multinomial, it has 3 dimensions: number of study by max number of arms in studies by number of categories.
mx	If the continuous covariate is included, it calculates the mean of the covariates which is used to center the covariates. The numeric indicator after mx refers to column number of the covariates if there are more than one covariates included. Discrete covariates are not centered.
mx_bl	If the baseline effect is specified, it also calculates the mean baseline risk.
prior.data	Prior data created using the user inputs or default values. If no user input is specifies for the prior, it uses default values.
code	Rjags model file code that is generated using information provided by the user. To view model file inside R in a nice format, use cat(network$code).

References

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. [https://doi.org/10.1177/0272989X12458724]

F.A. Achana, N.J. Cooper, S. Dias, G. Lu, S.J.C. Rice, D. Kendrick, A.J. Sutton (2012), *Extending methods for investigating the relationship between treatment effect and baseline risk from pairwise meta-analysis to network meta-analysis*, Statistics in Medicine 32(5):752-771. [https://doi.org/10.1002/sim.5539]

N.J. Cooper, A.J. Sutton, D. Morris, A.E. Ades, N.J. Welton (2009), *Addressing between-study heterogeneity and inconsistency in mixed treatment comparisons: Application to stroke prevention treatments in individuals with non-rheumatic atrial fibrillation*, Statistics in Medicine 28:1861-1881. [https://doi.org/10.1002/sim.3594]

Examples

```
###Blocker data example
blocker
network <- with(blocker, {
 network.data(Outcomes, Study, Treat, N = N, response = "binomial")
})
network
```

---

network.deviance.plot *Make a deviance plot*

Description

This makes a deviance plot which plots residual deviance (dev_arm) vs. all the arms for each study.

Usage

```
network.deviance.plot(result)
```

Arguments

result	Object created by network.run function

Value

None

Examples

```
network <- with(blocker, {
 network.data(Outcomes, Study, Treat, N = N, response = "binomial")
})

result <- network.run(network)
network.deviance.plot(result)
```

---

network.forest.plot *Draws forest plot*

Description

Draws forest plot of pooled treatment effect. Reports odds ratio for binomial and multinomial outcomes and continuous scale for normal outcomes.

Usage

```
network.forest.plot(
  result,
  level = 0.95,
  ticks.position = NULL,
  label.multiplier = 0.2,
  label.margin = 10,
  title = "Network Meta-analysis Forest plot",
  only.reference.treatment = FALSE
)
```

Arguments

result	Object created by network.run function
level	Confidence level. Default is 0.95 denoting 95 percent C.I.
ticks.position	Position of the x-axis tick marks. If left unspecified, the function tries to set it at sensible values
label.multiplier	This is a multiplying factor to move the position of the text associated with median[lower, upper] values. This number is multiplied by the range of x-axis and added to the x-axis limit. Default multiplier is set to 0.2.

label.margin	This is how much margin space you specify to assign space for the median[lower, upper] values. Default margin is set to 10.
title	Header name which you can modify
only.reference.treatment	Indicator for plotting only the comparison to the reference treatment

Value

None

References

W. Viechtbauer (2010), *Conducting meta-analyses in R with the metafor package*, Journal of Statistical Software, 36(3):1-48. [https://doi.org/10.18637/jss.v036.i03]

Examples

```
network <- with(certolizumab, {
 network.data(Outcomes, Study, Treat, N=N, response="binomial", Treat.order,
 covariate = covariate, hy.prior = list("dhnorm", 0, 9.77))
})

result <- network.run(network)
network.forest.plot(result)
```

---

network.gelman.diag *Use coda package to find Gelman-Rubin diagnostics*

Description

This function uses coda package to find Gelman-Rubin diagnostics.

Usage

```
network.gelman.diag(result, extra.pars = NULL, only.pars = NULL)
```

Arguments

result	Object created by network.run function
extra.pars	Extra parameters that the user wants to display other than the default parameters.
only.pars	Parameters that user wants to display. This gets rids of other default parameters user doesn't want to show.

Value

Returns gelman-rubin diagnostics

Examples

```
network <- with(statins, {
 network.data(Outcomes, Study, Treat, N = N, response = "binomial",
 Treat.order = c("Placebo", "Statin"), covariate = covariate, covariate.type = "discrete")
})

result <- network.run(network)
network.gelman.diag(result, extra.pars = "Eta")
```

`network.gelman.plot` *Use coda package to plot Gelman-Rubin diagnostic plot*

Description

This function plots Gelman-Rubin diagnostic using coda package.

Usage

```
network.gelman.plot(result, extra.pars = NULL, only.pars = NULL)
```

Arguments

`result`	Object created by `network.run` function
`extra.pars`	Extra parameters that the user wants to plot other than the default parameters.
`only.pars`	Parameters that user wants to display. This gets rids of other default parameters user doesn't want to show.

Value

None

Examples

```
network <- with(statins, {
 network.data(Outcomes, Study, Treat, N = N, response = "binomial",
 Treat.order = c("Placebo", "Statin"), covariate = covariate, covariate.type = "discrete")
})

result <- network.run(network)
network.gelman.plot(result)
```

network.leverage.plot *Make a leverage plot*

Description

This function makes a leverage vs. square root of residual deviance plot

Usage

```
network.leverage.plot(result)
```

Arguments

result	Object created by network.run function

Value

None

Examples

```
network <- with(blocker, {
 network.data(Outcomes, Study, Treat, N = N, response = "binomial")
})

result <- network.run(network)
network.leverage.plot(result)
```

network.rank.tx.plot *Create a treatment rank plot*

Description

This plot displays how each treatment is ranked. For each rank, we show how likely each treatment will be at that rank.

Usage

```
network.rank.tx.plot(
  result,
  txnames = NULL,
  catnames = NULL,
  legend.position = c(1, 1)
)
```

Arguments

result	Object created by network.run function
txnames	Treatment names used in creating legend
catnames	Category names. Only used in multinomial.
legend.position	x,y position of the legend

Value

None

See Also

rank.tx

Examples

```
network <-with(blocker, {
 network.data(Outcomes, Study, Treat, N = N, response = "binomial")
})

result <- network.run(network)
network.rank.tx.plot(result, txnames = c("a", "b"))
```

---

network.run *Run the model using the network object*

Description

This is the core function that runs the model in our program. Before running this function, we need to specify data, prior, JAGS code, etc. using network.data.

Usage

```
network.run(
  network,
  inits = NULL,
  RNG.inits = NULL,
  n.chains = 3,
  max.run = 1e+05,
  setsize = 10000,
  n.run = 50000,
  conv.limit = 1.05,
  extra.pars.save = NULL
)
```

Arguments

network	Network object created from network.data function
inits	Initial values for the parameters being sampled. If left unspecified, program will generate reasonable initial values.
RNG.inits	List of .RNG.name and .RNG.seed that control the JAGS RNGs. Please refer to jags.model function in rjags for more information.
n.chains	Number of chains to run
max.run	Maximum number of iterations that user is willing to run. If the algorithm is not converging, it will run up to max.run iterations before printing a message that it did not converge
setsize	Number of iterations that are run between convergence checks. If the algorithm converges fast, user wouldn't need a big setsize. The number that is printed between each convergence checks is the gelman-rubin diagnostics and we would want that to be below the conv.limit the user specifies.
n.run	Final number of iterations that the user wants to store. If after the algorithm converges, user wants less number of iterations, we thin the sequence. If the user wants more iterations, we run extra iterations to reach the specified number of runs
conv.limit	Convergence limit for Gelman and Rubin's convergence diagnostic. Point estimate is used (instead of 95 percent C.I.) to test convergence of parameters for study effect (eta), relative effect (d), and heterogeneity (log variance (logvar)).
extra.pars.save	Parameters that user wants to save besides the default parameters saved. See code using cat(network$code) to see which parameters can be saved.

Value

data_rjags	Data that is put into rjags function jags.model
inits	Initial values that are either specified by the user or generated as a default
RNG.inits	List of .RNG.name and .RNG.seed used for reproducibility
pars.save	Parameters that are saved. Add more parameters in extra.pars.save if other variables are desired
burnin	Half of the converged sequence is thrown out as a burnin
n.thin	If the number of iterations user wants (n.run) is less than the number of converged sequence after burnin, we thin the sequence and store the thinning interval
samples	MCMC samples stored using jags. The returned samples have the form of mcmc.list and can be directly applied to coda functions
max.gelman	Maximum Gelman and Rubin's convergence diagnostic calculated for the final sample
deviance	Contains deviance statistics such as pD (effective number of parameters) and DIC (Deviance Information Criterion)

rank.tx — Rank probability calculated for each treatments. rank.preference parameter in network.data is used to define whether higher or lower value is preferred. The numbers are probabilities that a given treatment has been in certain rank in the sequence.

Examples

```
#parkinson's example (normal)
network <- with(parkinsons,{
  network.data(Outcomes, Study, Treat, SE = SE, response = "normal")
})

result <- network.run(network)
```

---

nodesplit.network.data

Make a network object containing data, priors, and a JAGS model file

---

Description

This function makes a network object that can be used to run network meta-analysis using nodesplit.network.run. User needs to specify Outcomes, Study, Treat, N or SE, and response. Prior parameters are filled in automatically based on the data type if not specified. The input data should be arm-level so that we have observations for each treatment in each study. The input data is preprocessed to fit the format necessary to run model in JAGS.

Usage

```
nodesplit.network.data(
  Outcomes = NULL,
  Study = NULL,
  Treat = NULL,
  N = NULL,
  SE = NULL,
  response = NULL,
  Treat.order = NULL,
  pair = NULL,
  type = "random",
  dic = FALSE
)
```

Arguments

Outcomes — Arm-level outcomes. If it is a multinomial response, the matrix would have dimensions treatment arms (row) by multinomial categories (column). If it is binomial or normal, it would be a vector.

Study — A vector of study indicator for each arm

`Treat`	A vector of treatment indicator for each arm
`N`	A vector of total number of observations in each arm. Used for binomial and multinomial responses
`SE`	A vector of standard error for each arm. Used only for normal response
`response`	Specification of the outcomes type. Must specify one of the following: "normal", "binomial", or "multinomial"
`Treat.order`	Treatment order which determines how treatments are compared. The first treatment that is specified is considered to be the baseline treatment. Default order is alphabetical. If the treatments are coded 1, 2, etc, then the treatment with a value of 1 would be assigned as a baseline treatment.
`pair`	Define a pair to split. It has to be a vector of length 2 with treatment names
`type`	Type of model fitted: either "random" for random effects model or "fixed" for fixed effects model. Default is "random".
`dic`	This is an indicator for whether user wants to calculate DIC. Model stores less information if you set it to FALSE. Default is set to FALSE.

Value

Creates list of variables that are used to run the model using `nodesplit.network.run`

`data`	Data combining all the input data. User can check this to insure the data is correctly specified. For modelling purposes, character valued studies or treatment variables are changed to numeric values based on alphabetical order.
`nrow`	Total number of arms in the meta-analysis
`ncat`	Number of columns in the Outcomes. Will equal 1 for binary and normal and number of categories for multinomial
`nstudy`	Number of study
`na`	Number of arms for each study
`ntreat`	Number of treatment
`b.id`	Indicator in sequence of all treatments for which treatment is base treatment in Study
`t`	`Treat` transformed into a matrix which has dimensions number of study by max number of arms in studies
`r`	`Outcomes` made into an array that is suitable for use in rjags code. For multinomial, it has 3 dimensions: number of study by max number of arms in studies by number of categories.
`code`	Rjags model file code that is generated using information provided by the user. To view model file inside R in a nice format, use `cat(network$code)`.

References

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

Examples

```
###Thrombolytic data example
network <- with(thrombolytic,{
 nodesplit.network.data(Outcomes, Study, Treat, N, response = "binomial", pair = c(3,9))
})
network
```

`nodesplit.network.run` *Run the model using the nodesplit network object*

Description

This is similar to the function `network.run`, except this is used for the nodesplitting model.

Usage

```
nodesplit.network.run(
  network,
  inits = NULL,
  n.chains = 3,
  max.run = 1e+05,
  setsize = 10000,
  n.run = 50000,
  conv.limit = 1.05,
  extra.pars.save = NULL
)
```

Arguments

`network`	network object created from `nodesplit.network.data` function
`inits`	Initial values for the parameters being sampled. If left unspecified, program will generate reasonable initial values.
`n.chains`	Number of chains to run
`max.run`	Maximum number of iterations that user is willing to run. If the algorithm is not converging, it will run up to `max.run` iterations before printing a message that it did not converge
`setsize`	Number of iterations that are run between convergence checks. If the algorithm converges fast, user wouldn't need a big setsize. The number that is printed between each convergence checks is the gelman-rubin diagnostics and we would want that to be below the conv.limit the user specifies.
`n.run`	Final number of iterations that the user wants to store. If after the algorithm converges, user wants less number of iterations, we thin the sequence. If the user wants more iterations, we run extra iterations to reach the specified number of runs

`conv.limit` Convergence limit for Gelman and Rubin's convergence diagnostic. Point estimate is used to test convergence of parameters for study effect (eta), relative effect (d), and heterogeneity (log variance (logvar)).

`extra.pars.save`
Parameters that user wants to save besides the default parameters saved. See code using `cat(network$code)` to see which parameters can be saved.

Value

`data_rjags` Data that is put into rjags function jags.model

`inits` Initial values that are either specified by the user or generated as a default

`pars.save` Parameters that are saved. Add more parameters in extra.pars.save if other variables are desired

`burnin` Half of the converged sequence is thrown out as a burnin

`n.thin` If the number of iterations user wants (n.run) is less than the number of converged sequence after burnin, we thin the sequence and store the thinning interval

`samples` MCMC samples stored using jags. The returned samples have the form of mcmc.list and can be directly applied to coda functions

`max.gelman` Maximum Gelman and Rubin's convergence diagnostic calculated for the final sample

Examples

```
###Thrombolytic data example
network <- with(thrombolytic,{
 nodesplit.network.data(Outcomes, Study, Treat, N, response = "binomial", pair = c(3,9))
})

result <- nodesplit.network.run(network)
```

---

parkinsons *Dopamine agonists as adjunct therapy in Parkinson's disease*

Description

A dataset of 7 studies investigating the mean lost work-time reduction in patients given 4 dopamine agonists and placebo as adjunct therapy for Parkinson's disease. There is placebo and four active drugs coded 2 to 5.

Usage

```
parkinsons
```

Format

A list of Outcomes, Treat, Study, N, covariate, and Treat.order

References

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. [https://doi.org/10.1177/0272989X12458724]

---

parkinsons_contrast *Dopamine agonists as adjunct therapy in Parkinson's disease*

Description

A contrast level (i.e. treatment difference) dataset of 7 studies investigating the mean lost worktime reduction in patients given 4 dopamine agonists and placebo as adjunct therapy for Parkinson's disease. Placebo is coded as 1, and four active drugs are coded 2 to 5. There is placebo, coded as 1, and four active drugs coded 2 to 5.

Usage

```
parkinsons_contrast
```

Format

A list of Outcomes, Treat, SE, na, and V

References

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. [https://doi.org/10.1177/0272989X12458724]

---

plot.contrast.network.result *Plot traceplot and posterior density of the result using contrast data*

Description

This function uses plotting function in coda package to plot mcmc.list object

Usage

```
## S3 method for class 'contrast.network.result'
plot(x, ...)
```

Arguments

x	Result object created by contrast.network.run function
...	Additional arguments affecting the plot produced

Value

None

Examples

```
network <- with(parkinsons_contrast, {
 contrast.network.data(Outcomes, Treat, SE, na, V)
})

result <- contrast.network.run(network)
plot(result)
```

---

plot.network.result *Plot traceplot and posterior density of the result*

Description

This function uses plotting function in coda package to plot mcmc.list object

Usage

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

Arguments

x	Result object created by network.run function
...	Additional arguments affecting the plot produced

Value

None

Examples

```
network <- with(statins, {
 network.data(Outcomes, Study, Treat, N = N, response = "binomial",
 Treat.order = c("Placebo", "Statin"), covariate = covariate, covariate.type = "discrete")
})

result <- network.run(network)
plot(result, only.pars = "sd")
```

plot.ume.network.result *Plot traceplot and posterior density of the result using contrast data*

Description

This function uses plotting function in coda package to plot mcmc.list object

Usage

```
## S3 method for class 'ume.network.result'
plot(x, ...)
```

Arguments

x	Result object created by ume.network.run function
...	Additional arguments affecting the plot produced

Value

None

Examples

```
network <- with(smoking, {
 ume.network.data(Outcomes, Study, Treat, N = N, response = "binomial", type = "random")
})

result <- ume.network.run(network)
plot(result)
```

rank.tx *Create a treatment rank table*

Description

This function makes a table of ranking for each treament. Each number in the cell represents a probability certain treatment was in such rank. This table is also stored as an output from network.run.

Usage

```
rank.tx(result)
```

Arguments

result	Object created by network.run function

Value

Returns a table of ranking

See Also

network.rank.tx.plot

Examples

```
network <- with(blocker, {
 network.data(Outcomes, Study, Treat, N = N, response = "binomial")
})

result <- network.run(network)
rank.tx(result)
```

`relative.effects` *Find relative effects for base treatment and comparison treatments*

Description

This function calculates relative effects for base treatment and comparison treatments.

Usage

```
relative.effects(
  result,
  base.treatment = NULL,
  comparison.treatments = NULL,
  base.category = NULL,
  comparison.categories = NULL,
  covariate = NULL
)
```

Arguments

`result`	Object created by `network.run` function
`base.treatment`	Base treatment user wants for the relative effects. Base treatment is initially set by `Treat.order` parameter in `network.data` (first one in the list). If set to null, default is to use base treatment.
`comparison.treatments`	Treatments that user wants to compare against base treatment. If set to null, all the treatments besides base treatment is considered as comparison treatments.
`base.category`	Base category user wants for the relative effects. Only used for multinomial data.

comparison.categories
: Category that user wants to compare against base.category. Only used for multinomial data.

covariate
: Covariate value at which to compute relative effects. Only used if covariate value is specified in the model.

Value

This returns a mcmc.list sample of relative effects for the base treatment specified. This allows user to obtain relative effects of different base.treatment after the sampling has been done. For a simple summary, use relative.effects.table.

See Also

relative.effects.table

Examples

```
network <- with(parkinsons, {
 network.data(Outcomes, Study, Treat, SE = SE, response = "normal")
})

result <- network.run(network)
summary(relative.effects(result, base.treatment = "Placebo"))
```

---

relative.effects.table

Make a summary table for relative effects

Description

This function creates a summary table of relative effects. Relative effects are in units of log odds ratio for binomial and multinomial data and real number scale for normal data.

Usage

```
relative.effects.table(
  result,
  summary_stat = "mean",
  probs = NULL,
  base.category = NULL
)
```

Arguments

result	Object created by network.run function
summary_stat	Specifies what type of statistics user wants. Options are: "mean", "ci", "quantile", "sd", "p-value". "ci" gives 95 "p-value" is the probability relative effect (in binomial, log odds ratio) is less than 0.
probs	Used only for the quantile summary. Specifies which quantile user wants the summary of (should be one numeric value between 0 to 1)
base.category	Specifies for which base category user wants for the summary. Used only for multinoimal.

Value

Returns relative effects table

See Also

relative.effects

Examples

```
#cardiovascular
network <- with(cardiovascular,{
 network.data(Outcomes, Study, Treat, N, response = "multinomial")
})

result <- network.run(network)
exp(relative.effects.table(result)) #look at odds ratio instead of log odds ratio
```

---

smoking *Smoking cessation counseling programs*

Description

Twenty-four studies, including 2 three-arm trials, compared 4 smoking cessation counseling programs and recorded the number of individuals with successful smoking cessation at 6 to 12 month. Counseling programs include 1 = no intervention, 2 = self-help, 3 = individual counseling, and 4 - group counseling.

Usage

```
smoking
```

Format

A list of Outcomes, Treat, Study, and N

References

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. [https://doi.org/10.1177/0272989X12458724]

`statins` *Trials of statins for cholesterol lowering vs. placebo or usual care*

Description

A dataset of 19 trials of statins for cholesterol lowering vs. placebo. Each trial has a subgroup indicator for primary prevention (patients included had no previous heart disease) or secondary prevention (patients had previous heart disease). Dummy variable is coded such that covariate is equal to 1 if a study is a secondary prevention study and 0 if a study is a primary prevention study.

Usage

```
statins
```

Format

A list of Outcomes, Treat, Study, N, covariate, and Treat.order

References

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

`sucra` *Calculate SUCRA*

Description

SUCRA is the surface under the cumulative ranking distribution defined in Salanti et al. (2011)

Usage

```
sucra(result, txnames = NULL, catnames = NULL)
```

Arguments

`result`	Object created by `network.run` function
`txnames`	Treatment names used in creating legend
`catnames`	Category names. Only used in multinomial.

Value

Returns SUCRA for each treatment

References

G. Salanti, A.E. Ades, J.P.A. Ioannidisa (2011), *Graphical methods and numerical summaries for presenting results from multiple-treatment meta-analysis: an overview and tutorial*, Journal of Clinical Epidemiology 64(2):163-71. [https://doi.org/10.1016/j.jclinepi.2010.03.016]

See Also

rank.tx

Examples

```
########### certolizumab (with baseline risk)
network <- with(certolizumab, {
 network.data(Outcomes, Study, Treat, N=N, response = "binomial", Treat.order,
 baseline = "common", hy.prior = list("dhnorm", 0, 9.77))
})

result <- network.run(network)
sucra(result)
```

---

summary.contrast.network.result

Summarize result run by contrast.network.run

Description

This function uses summary function in coda package to summarize mcmc.list object. Monte carlo error (Time-series SE) is also obtained using the coda package and is printed in the summary as a default.

Usage

```
## S3 method for class 'contrast.network.result'
summary(object, ...)
```

Arguments

object	Result object created by contrast.network.run function
...	Additional arguments affecting the summary produced

Value

Returns summary of the contrast network model result

Examples

```
network <- with(parkinsons_contrast, {
 contrast.network.data(Outcomes, Treat, SE, na, V)
})

result <- contrast.network.run(network)
summary(result)
```

---

summary.network.result

Summarize result run by network.run

Description

This function uses summary function in coda package to summarize mcmc.list object. Monte carlo error (Time-series SE) is also obtained using the coda package and is printed in the summary as a default.

Usage

```
## S3 method for class 'network.result'
summary(object, ...)
```

Arguments

object	Result object created by network.run function
...	Additional arguments affecting the summary produced

Value

Returns summary of the network model result

Examples

```
network <- with(statins, {
 network.data(Outcomes, Study, Treat, N = N, response = "binomial",
 Treat.order = c("Placebo", "Statin"), covariate = covariate, covariate.type = "discrete")
})

result <- network.run(network)
summary(result)
```

summary.nodesplit.network.result

Summarize result run by nodesplit.network.run

Description

This function uses summary function in coda package to summarize mcmc.list object. Monte carlo error (Time-series SE) is also obtained using the coda package and is printed in the summary as a default.

Usage

```
## S3 method for class 'nodesplit.network.result'
summary(object, ...)
```

Arguments

object	Result object created by nodesplit.network.run function
...	Additional arguments affecting the summary produced

Value

Returns summary of the nodesplit network model result

Examples

```
###Parkinsons data example
network <- with(parkinsons, {
 nodesplit.network.data(Outcomes, Study, Treat, SE = SE, response = "normal",
 Treat.order = Treat.order, pair = c("Placebo", "Ropinirole"))
})

result <- nodesplit.network.run(network)
summary(result)
```

summary.ume.network.result

Summarize result run by ume.network.run

Description

This function uses summary function in coda package to summarize mcmc.list object. Monte carlo error (Time-series SE) is also obtained using the coda package and is printed in the summary as a default.

Usage

```
## S3 method for class 'ume.network.result'
summary(object, ...)
```

Arguments

object	Result object created by ume.network.run function
...	Additional arguments affecting the summary produced

Value

Returns summary of the ume network model result

Examples

```
network <- with(smoking, {
 ume.network.data(Outcomes, Study, Treat, N = N, response = "binomial", type = "random")
})

result <- ume.network.run(network)
summary(result)
```

---

thrombolytic *Thrombolytic drugs and percutaneous transluminal coronary angioplasty*

Description

A dataset consisting of 50 trials comparing 8 thrombolytic drugs and percutaneous transluminal coronary angioplasty, following acute myocardial infarction. Data consist of the number of deaths in 30 or 35 days and the number of patients in each treatment arm. There are 9 treatments in total: streptokinase (1), alteplase (2), accelerated alteplase (3), streptokinase + alteplase (4), reteplase (5), tenecteplase (6), percutaneous transluminal coronary angioplasty (7), urokinase (8), anistreptilase (9)

Usage

```
thrombolytic
```

Format

A list of Outcomes, Treat, Study, and N

References

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. [https://doi.org/10.1177/0272989X12458724]

`ume.network.data` *Make a network object for the unrelated mean effects model (inconsistency model) containing data, priors, and a JAGS model file*

Description

This is similar to the function `network.data`, except this is used for the unrelated mean effects model.

Usage

```
ume.network.data(
  Outcomes,
  Study,
  Treat,
  N = NULL,
  SE = NULL,
  response = NULL,
  type = "random",
  mean.mu = NULL,
  prec.mu = NULL,
  mean.d = NULL,
  prec.d = NULL,
  hy.prior = list("dunif", 0, 5),
  dic = TRUE
)
```

Arguments

`Outcomes`	Arm-level outcomes. If it is a multinomial response, the matrix would be arms (row) by multinomial categories (column). If it is binomial or normal, it would be a vector.
`Study`	A vector of study indicator for each arm
`Treat`	A vector of treatment indicator for each arm. Treatments should have positive integer values starting from 1 to total number of treatments. In a study, lowest number is taken as the baseline treatment.
`N`	A vector of total number of observations in each arm. Used for binomial and multinomial responses.
`SE`	A vector of standard error for each arm. Used only for normal response.
`response`	Specification of the outcomes type. Must specify one of the following: "normal", "binomial", or "multinomial".
`type`	Type of model fitted: either "random" for random effects model or "fixed" for fixed effects model. Default is "random".
`mean.mu`	Prior mean for the study effect (baseline risk)
`prec.mu`	Prior precision for the study effect (baseline risk)

mean.d	Prior mean for the relative effect
prec.d	Prior precision for the relative effect
hy.prior	Prior for the heterogeneity parameter. Supports uniform, gamma, and half normal for normal. It should be a list of length 3, where first element should be the distribution (one of dunif, dgamma, dhnorm, dwish) and the next two are the parameters associated with the distribution. For example, list("dunif", 0, 5) give uniform prior with lower bound 0 and upper bound 5 for the heterogeneity parameter.
dic	This is an indicator for whether user wants to calculate DIC. Model stores less information if you set it to FALSE.

Value

Creates list of variables that are used to run the model using ume.network.run

References

S. Dias, N.J. Welton, A.J. Sutton, D.M. Caldwell, G. Lu, and A.E. Ades (2013), *Evidence synthesis for decision making 4: inconsistency in networks of evidence based on randomized controlled trials*, Medical Decision Making 33(5):641-656. [https://doi.org/10.1177/0272989X12455847]

Examples

```
network <- with(thrombolytic, {
 ume.network.data(Outcomes, Study, Treat, N = N, response = "binomial")
})
network
```

---

ume.network.run *Run the model using the network object*

Description

This is similar to the function network.run, except this is used for the unrelated mean effects model.

Usage

```
ume.network.run(
  network,
  inits = NULL,
  n.chains = 3,
  max.run = 1e+05,
  setsize = 10000,
  n.run = 50000,
  conv.limit = 1.05,
  extra.pars.save = NULL
)
```

Arguments

network	network object created from ume.network.data function
inits	Initial values for the parameters being sampled. If left unspecified, program will generate reasonable initial values.
n.chains	Number of chains to run
max.run	Maximum number of iterations that user is willing to run. If the algorithm is not converging, it will run up to max.run iterations before printing a message that it did not converge
setsize	Number of iterations that are run between convergence checks. If the algorithm converges fast, user wouldn't need a big setsize. The number that is printed between each convergence checks is the gelman-rubin diagnostics and we would want that to be below the conv.limit the user specifies.
n.run	Final number of iterations that the user wants to store. If after the algorithm converges, user wants less number of iterations, we thin the sequence. If the user wants more iterations, we run extra iterations to reach the specified number of runs
conv.limit	Convergence limit for Gelman and Rubin's convergence diagnostic. Point estimate is used to test convergence of parameters for study effect (eta), relative effect (d), and heterogeneity (log variance (logvar)).
extra.pars.save	Parameters that user wants to save besides the default parameters saved. See code using cat(network$code) to see which parameters can be saved.

Value

data_rjags	Data that is put into rjags function jags.model
inits	Initial values that are either specified by the user or generated as a default
pars.save	Parameters that are saved. Add more parameters in extra.pars.save if other variables are desired
burnin	Half of the converged sequence is thrown out as a burnin
n.thin	If the number of iterations user wants (n.run) is less than the number of converged sequence after burnin, we thin the sequence and store the thinning interval
samples	MCMC samples stored using jags. The returned samples have the form of mcmc.list and can be directly applied to coda functions
max.gelman	Maximum Gelman and Rubin's convergence diagnostic calculated for the final sample
deviance	Contains deviance statistics such as pD (effective number of parameters) and DIC (Deviance Information Criterion)
rank.tx	Rank probability calculated for each treatments. rank.preference parameter in ume.network.data is used to define whether higher or lower value is preferred. The numbers are probabilities that a given treatment has been in certain rank in the sequence.

Examples

```
network <- with(thrombolytic, {
 ume.network.data(Outcomes, Study, Treat, N = N, response = "binomial")
})

result <- ume.network.run(network, n.run = 10000)
```

`variance.tx.effects` *Calculate correlation matrix for multinomial heterogeneity parameter.*

Description

This function calculates correlation matrix from the variance matrix for heterogeneity parameter. Only used for multinomial.

Usage

```
variance.tx.effects(result)
```

Arguments

`result`	Object created by `network.run` function

Value

Returns correlation matrix

Examples

```
#cardiovascular
network <- with(cardiovascular, {
 network.data(Outcomes, Study, Treat, N, response = "multinomial")
})

result <- network.run(network)
variance.tx.effects(result)
```

Index