Package 'mcnet'

June 13, 2017

Title Bayesian Network Meta-Analysis

Version 0.5.0

Depends R (>= 2.10)

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

Description

Network meta-analyses using Bayesian framework. Creates data input, 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/baseline effects. Includes standard diagnostic tools to evaluate the results.

License GPL-3

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

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

Maintainer Michael Seo <michael_seo@brown.edu>

R topics documented:

`mcnet-package` *mcnet: 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 the rjags model. It takes the input data and transforms it to a suitable format of analysis, generates a rjags model and reasonable initial values and run the model through the rjags package.

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]

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]

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]

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

Description

Calculates deviance statistics. This function is 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	dev_arm - 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
Outcomes <- parkinsons[["Outcomes"]]
Study <- parkinsons[["Study"]]
Treat <- parkinsons[["Treat"]]
SE <- parkinsons[["SE"]]
network <- 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 affects 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 baseline treatment. Outcomes are reported as three mutually exclusive unordered outcomes. First column of the outcome is the patients who are still 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]

`network.autocorr.diag` *Use coda package to find autocorrelation diagnostics*

Description

Use coda package to find autocorrelation diagnostics

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 plot other than the default parameters.
`only.pars`	parameters that user wants to plot only

Examples

```
Outcomes <- blocker[["Outcomes"]]
Study <- blocker[["Study"]]
Treat <- blocker[["Treat"]]
N <- blocker[["N"]]
network <- network.data(Outcomes, Study, Treat, N = N, response = "binomial")
result <- network.run(network)
network.autocorr.diag(result, only.pars = "d")
```

---

network.autocorr.plot *Use coda package to plot autocorrelation plot*

Description

Use coda package to plot autocorrelation plot

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 plot only

Examples

```
#cardiovascular
Study <- cardiovascular[["Study"]]
Treat <- cardiovascular[["Treat"]]
Outcomes <- cardiovascular[["Outcomes"]]
N <- cardiovascular[["N"]]
network <- network.data(Outcomes, Study, Treat, N, response = "multinomial")
result <- network.run(network)
network.autocorr.plot(result, only.pars = "d")
```

---

network.covariate.plot

Make a covariate plot

Description

Make a covariate plot of how the relative effect changes as the covariate value changes. Plot is created for each one of the covariate. User needs to specify one base treatment and one comparison treatment to make this plot (base category and comparison category is needed for multinomial). It then uses the relative.effects to calculate the correct relative effect. 2.5% and 97.5% C.I. are drawn along with the median value.

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
comparison.category	comparison category for multinomial data
covariate.name	A vector of covariate names naming of the covariate that goes into x axis label

Examples

```
########### certolizumab (with covariate)
Outcomes <- certolizumab[["Outcomes"]]
N <- certolizumab[["N"]]
Study <- certolizumab[["Study"]]
Treat <- certolizumab[["Treat"]]
covariate <- certolizumab[["covariate"]]
Treat.order <- certolizumab[["Treat.order"]]
network <- 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

Creates a treatment cumulative rank plot

Description

Creates a treatment cumulative rank plot

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

See Also

rank.tx

Examples

```
Outcomes <- blocker[["Outcomes"]]
Study <- blocker[["Study"]]
Treat <- blocker[["Treat"]]
N <- blocker[["N"]]
network <- 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

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

Usage

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

Arguments

Outcomes	Arm-level outcomes. If it is multinomial response, the matrix would have arms for the row and categories for the column. Otherwise, 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 for Outcomes type. Must specify one of the following: "normal", "binomial", or "multinomial".
Treat.order	This specifies the treatment order and therefore how treatments are compared. The first treatment that is specified is considered as a base treatment. Default order would be alphabetical. This would hold true for numbers. If the treatments are coded 1,2,etc, then treatment with a value of 1 would be assigned as baseline treatment.
type	Type of model fitted: either "random" for random effects model or "fixed" for fixed effects model. The 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).

`miss.matrix`
: This is parameter for running incomplete multinomial. Still under revision.

`baseline`
: "independent" allows baseline effects for each treatment while "common" restricts same baseline effect for all treatment. Default is "none" which doesn't incorporate baseline effect.

`covariate`
: A covariate matrix with each row representing each trial and column representing each covariate. Covariate information is needed for each study, and so the user doesn't need to repeatly 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 while "common" restricts same covariate effect for all treatment. We set "common" to be default if not specified.

`dic`
: This is an indicator for whether you want to calculate DIC. It stores less information if you set it to FALSE.

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

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

`mean.Eta`
: Prior mean for the study effect

`prec.Eta`
: Prior precision for the study effect

`mean.bl`
: Prior mean for the baseline effect

`prec.bl`
: Prior precision for the baseline effect

`mean.cov`
: Prior mean for the covariate effect

`prec.cov`
: Prior precision for the covariate effect

`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.

Value

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

`data`
: Data combining all the input data. User can check this to insure the data is correctly specified. For modelling purposes, any character valued study or treatment variable is changed to numeric values based on alphabetical order

`nrow`
: Total number of arms in the meta-analysis

`ncol`
: 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 made into a matrix which has dimensions number of study by max number of arms in studies
r	Outcomes made into array that is suitable for use in our 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 is more than one covariates included. For discrete covariate, 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, 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]

Examples

```
###Blocker data example
blocker
Outcomes <- blocker[["Outcomes"]]
Study <- blocker[["Study"]]
Treat <- blocker[["Treat"]]
N <- blocker[["N"]]
network <- network.data(Outcomes, Study, Treat, N = N, response = "binomial")
str(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

Examples

```
Outcomes <- blocker[["Outcomes"]]
Study <- blocker[["Study"]]
Treat <- blocker[["Treat"]]
N <- blocker[["N"]]
network <- network.data(Outcomes, Study, Treat, N = N, response = "binomial")
result <- network.run(network)
network.deviance.plot(result)
```

---

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

Description

Use coda package to find gelman-diagnostic 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 plot other than the default parameters.
only.pars	parameters that user wants to plot only

Examples

```
#statins example (binomial)
Outcomes <- statins[["Outcomes"]]
Study <- statins[["Study"]]
Treat <- statins[["Treat"]]
N <- statins[["N"]]
covariate <- statins[["covariate"]]
Treat.order <- statins[["Treat.order"]]
network <- 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-diagnostic plot*

Description

Use coda package to plot gelman-diagnostic plot

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 plot only

Examples

```
#blocker
Outcomes <- blocker[["Outcomes"]]
Study <- blocker[["Study"]]
Treat <- blocker[["Treat"]]
N <- blocker[["N"]]
network <- network.data(Outcomes, Study, Treat, N = N, response = "binomial")
result <- network.run(network)
network.gelman.plot(result, only.pars = "d")
```

network.leverage.plot *make a leverage plot*

Description

Make a leverage vs. square root of residual deviance plot

Usage

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

Arguments

result	object created by network.run function

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

Description

Creates a treatment rank plot

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

See Also

rank.tx

Examples

```
Outcomes <- blocker[["Outcomes"]]
Study <- blocker[["Study"]]
Treat <- blocker[["Treat"]]
N <- blocker[["N"]]
network <- 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

Run the model using the network object

Usage

```
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 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.

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 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)
parkinsons
Outcomes <- parkinsons[[1]]
```

```
Study <- parkinsons[[2]]
Treat <- parkinsons[[3]]
SE <- parkinsons[[4]]
network <- network.data(Outcomes, Treat, Study, SE = SE, response = "normal")
result <- network.run(network)
str(result)
```

`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, coded as 1, 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]

`rank.tx` *Creates a treatment rank table*

Description

Creates a treatment rank table

Usage

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

Arguments

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

Value

This 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`.

See Also

network.rank.tx.plot

Examples

```
Outcomes <- blocker[["Outcomes"]]
Study <- blocker[["Study"]]
Treat <- blocker[["Treat"]]
N <- blocker[["N"]]
network <- network.data(Outcomes, Study, Treat, N = N, response = "binomial")
result <- network.run(network)
rank.tx(result)
```

---

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

Description

Find relative effects for different 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. This is only used for multinomial data.
comparison.categories	category that user wants to compare against base.category
covariate	covariate value at which to compute relative effects.

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

```
#We can fit two different models with different base treatment and we can
#obtain same relative effects estimate using this function
#parkinsons
Outcomes <- parkinsons[["Outcomes"]]
Study <- parkinsons[["Study"]]
Treat <- parkinsons[["Treat"]]
SE <- parkinsons[["SE"]]
network <- network.data(Outcomes, Study, Treat, SE = SE, response = "normal")
result <- network.run(network)

Treat.order <- c(2,1,3,4,5)
network2 <- network.data(Outcomes, Study, Treat, SE = SE, response = "normal",
Treat.order = Treat.order)
result2 <- network.run(network2)

summary(result2)
summary(relative.effects(result2, base.treatment = 2))

#This also works for comparing different base.category for multinomial.
#We fit two different models and compare the estimates again.
#cardiovascular

Study <- cardiovascular[["Study"]]
Treat <- cardiovascular[["Treat"]]
Outcomes <- cardiovascular[["Outcomes"]]
N <- cardiovascular[["N"]]
network3 <- network.data(Outcomes, Study, Treat, N, response = "multinomial")
result3 <- network.run(network3)

Outcomes <- Outcomes[,c(2,1,3)]
network4 <- network.data(Outcomes, Study, Treat, N, response = "multinomial")
result4 <- network.run(network4)

summary(result3)
summary(relative.effects(result4, base.category = 2))
```

---

relative.effects.table

Make a summary table for relative effects

Description

Make a summary table for relative effects

Usage

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

Arguments

result	object created by network.run function

summary_stat — specifies what type of statistics user wants. Options are: "mean", "quantile", "sd", "p-value". Quantile use c(0.025, 0.5, 0.975). P-value is the probability relative effect is less than 0.

base.category — specifies for which base category user wants for the summary. Used only for multinoimal.

See Also

relative.effects

Examples

```
#cardiovascular
Study <- cardiovascular[["Study"]]
Treat <- cardiovascular[["Treat"]]
Outcomes <- cardiovascular[["Outcomes"]]
N <- cardiovascular[["N"]]
network <- 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
```

---

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 *Creates a treatment rank plot*

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.

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)
Outcomes <- certolizumab[["Outcomes"]]
N <- certolizumab[["N"]]
Study <- certolizumab[["Study"]]
Treat <- certolizumab[["Treat"]]
covariate <- certolizumab[["covariate"]]
Treat.order <- certolizumab[["Treat.order"]]
network <- 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.network.result *summarize result run by* network.run

Description

Use summary function in coda to summarize mcmc.list object

Usage

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

Arguments

object	result object created by network.run function
...	additional arguments affecting the summary produced

variance.tx.effects *Calculates correlation matrix for multinomial heterogeneity parameter.*

Description

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

Examples

```
#cardiovascular
Study <- cardiovascular[["Study"]]
Treat <- cardiovascular[["Treat"]]
Outcomes <- cardiovascular[["Outcomes"]]
N <- cardiovascular[["N"]]
network <- network.data(Outcomes, Study, Treat, N, response = "multinomial")
result <- network.run(network)
variance.tx.effects(result)
```

Index