Package ‘NetworkMetaAnalysis’

June 16, 2020

Title Network meta analysis of target-comparator-outcomes in OHDSI network studies

Version 0.0.0.9000

Description
A package for using a collection of target-comparator-outcome estimates from OHDSI network studies to estimate relative effectiveness between all target-comparator pairs, even in the absence of head-to-head comparison.

Depends R (>= 3.5.0)

Imports coda (>= 0.19.3),
DatabaseConnector (>= 2.4.4),
dplyr (>= 0.8.5),
foreach (>= 1.5.0),
gemtc (>= 0.8.4),
ggraph (>= 2.0.2),
ggplot2 (>= 3.3.0),
grid (>= 3.6.3),
igraph (>= 1.2.5),
magrittr (>= 1.5),
plyr (>= 1.8.6),
rlang (>= 0.4.6),
scales (>= 1.1.1),
SqlRender (>= 1.6.5),
stringr (>= 1.4),
tibble (>= 3.0.1),
tidyr (>= 1.0.2),
tidyselect (>= 1.0.0)

Suggests doSNOW (>= 1.0.18),
DT (>= 0.13),
shiny (>= 1.4.0.2),
shinydashboard (>= 0.7.1),
shinythemes (>= 1.1.2),
snow (>= 0.4.3),
testthat

License file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.0

Language en_GB

R topics documented:

computePairwiseEstimates 2
computeRankProbabilities 3
computeRhat 3
computeSucra 4
deriveEdges 4
deriveNodes 5
fetchParameterSummaries 5
fetchPosteriorDraws 6
fetchTcoEstimates 6
fetchTidyNodesplitResults 7
fitNetwork 7
launchShinyApp 8
loadFromDb 8
plotForest 9
plotNetwork 9
plotNodesplitResults 10
plotPairwiseEstimates 10
plotPosteriorDensities 11
plotRankogram 12
plotRhat 13
plotTraces 13
runAnalyses 14
saveToDatabase 15

Index **16**

`computePairwiseEstimates` *Compute all pairwise estimates*

Description

After fitting the model, this function pulls out all pairwise network effect (direct and indirect combined) in a tidy format, on the hazard rate scale.

Usage

```
computePairwiseEstimates(fit)
```

Arguments

`fit`	the output of `fitNetwork`

`computeRankProbabilities` *Compute the rank probabilities*

Description

To be used for, i.a., rankograms (that use the cumulative probability).

Usage

```
computeRankProbabilities(fit, preferredDirection = 1)
```

Arguments

`fit`	object returned from GeMTC.
`preferredDirection`	scalar, preferential direction of the outcome. `1` means higher values are preferred, `-1` means lower values are preferred.

Value

Tidy data frame.

`computeRhat` *Compute the Rhat statistic for a given fit*

Description

Returns a tidy tibble with the Rhat statistic for each parameter. Also handles node-splitting models.

Usage

```
computeRhat(fit)
```

Arguments

`fit`	an object of class `"mtc.result"` or `"mtc.nodesplit"`.

Value

Tidy tibble. If a node-splitting result is supplied, the first columns called `modelName` indicates the comparison that was split in the analysis.

`computeSucra` *Compute SUCRA estimates*

Description

Computes surface under the cumulative ranking curve (SUCRA) for each treatment in the network. The ranking curve and rankogram are synonyms. Not sure this deserves its own function. Let's see down the road.

Usage

```
computeSucra(rankProbabilities)
```

Arguments

`rankProbabilities`	output of `computeRankProbabilities`

`deriveEdges` *Use to prepare data needed to draw the edges of the network*

Description

Use to prepare data needed to draw the edges of the network

Usage

```
deriveEdges(networkData, nodes)
```

Arguments

`networkData`	subset of the original network study data.
`nodes`	output from `deriveNodes`

Value

A ggplot object.

deriveNodes *Deriving node data for drawing the network*

Description

Deriving node data for drawing the network

Usage

```
deriveNodes(networkData, labelOrientation)
```

Arguments

`networkData`	tidy data frame, subset of the full data frame with aggregate results at study arm level for specific network meta-analysis.
`labelOrientation`	string giving how to rotate the label. There are two allowed settings: `"radial"` (default) and `"horisontal"`/`"horizontal"`.

Value

A tidy data frame.

fetchParameterSummaries *Table of parameter estimates and deviance statistics for reconciled results*

Description

Table of parameter estimates and deviance statistics for reconciled results

Usage

```
fetchParameterSummaries(
  parameterSummaries,
  devianceStatistics,
  networkMetaAnalysisId = NULL,
  nDigits = 1
)
```

Arguments

`parameterSummaries`	output of `deriveParameterQuantiles`.
`devianceStatistics`	output of `fetchDevianceStatistics`.
`networkMetaAnalysisId`	scalar, what analysis to use. If `NULL` (default), the function assumes pre-filtered data to be supplied in the first two arguments.

`nDigits`	scalar, how many digits to show for parameter estimates. Won't affect deviance statistics.

`fetchPosteriorDraws` *Fetches and tidies the posterior draws*

Description

Potentially, with some thinning to keep the amount of space required down.

Usage

```
fetchPosteriorDraws(fit, drawsThin = 10)
```

Arguments

`fit`	the result of `fitNetwork`.
`drawsThin`	scalar, thinning factor. The default value (`10`) keeps every tenth observation. This is mainly used when producing output to be saved to the server, to keep the storage requirements at a reasonable level.

Value

Tidy `tibble` with columns for chain and MCMC iteration indicators and one column per parameter.

`fetchTcoEstimates` *Load target-comparator-outcome estimates from server*

Description

Load TCO estimates from the server and wrangle them into a format appropriate for network visualisation and analysis.

Usage

```
fetchTcoEstimates(
  conn,
  resultsCdm,
  resultsTable,
  excludedDatabases = "Meta-analysis"
)
```

Arguments

`conn`	a connection to a database, e.g., the result of calling `DatabaseConnector::connect`.
`resultsCdm`	string, the name of the schema in which the table lives.
`resultsTable`	string, the name of the table holding the estimates.
`excludedDatabases`	character vector, names of the databases whose results should be ignored. The default (`"Meta-analysis"`) is obvious and should be kept if other databases are added.

`fetchTidyNodesplitResults` *Get the node-splitting analyses results in tidy format*

Description

Returns a tidy data frame with direct, indirect and consistency (network) effect estimates for each target-comparator pair along with the probability of inconsistency between the direct and indirect estimates.

Usage

```
fetchTidyNodesplitResults(fit)
```

Arguments

`fit`	the result of calling `fitNetwork` with `includeNodesplittingAnalysis` = `TRUE`.

Value

Tidy data frame.

`fitNetwork` *Run model on network to estimate parameters*

Description

Runs a GeMTC-powered network meta-analysis on a subset of the results of calling `fetchTcoEstimates` (i.e., setting analysisId and outcomeId) and

Usage

```
fitNetwork(
  networkData,
  includeNodesplittingAnalysis = FALSE,
  modelType = "random",
  nWarmup = 1000,
  nIter = 3000,
  fitThin = 1,
  nChains = 4
)
```

Arguments

`networkData`	a `dataframe` holding the subset of a target-comparator-outcome estimates from the result of `fetchTcoEstimates`, with specified `outcomeId` and `analysisId`.
`includeNodesplittingAnalysis`	boolean, should also node-splitting analyses be run? NB: This can be quite time-consuming, so is by default set to `FALSE`.

`modelType`	string defining whether to run a random-effects (`"random"`, default) or a fixed-effects (`"fixed"`) mode.
`nWarmup`	scalar, the number of iterations the sampler should do before it starts sampling from the posterior. This should be sufficient to the chains to have converged.
`nIter`	scalar, the number of posterior samples to draw per chain.
`fitThin`	scalar, thinning factor. `1` (default) keeps all samples whereas e.g. `10` would keep only every tenth sample.
`nChains`	scalar, the number of JAGS chains to run

`launchShinyApp` *Launch a Shiny app to explore the results*

Description

Launch a Shiny app to explore the results

Usage

```
launchShinyApp(results, rstudio = TRUE, ...)
```

Arguments

`results`	the output of `reconcileResults` directly or `loadFromDb` if the results live on a server.
`rstudio`	boolean, should the app run in the Viewer window (`TRUE`, default) or in an internet browser (`FALSE`)?
`...`	passed on to `shiny::runApp`

`loadFromDb` *Loads results stored in a database following its data model*

Description

Loads results stored in a database following its data model

Usage

```
loadFromDb(conn, schema)
```

Arguments

`conn`	connection to database, e.g. result of calling `DatabaseConnector::connect`
`schema`	string, the name of the schema where results tables live.

plotForest *Draw forest plot of total network estimates*

Description

Using a `fitNetwork` object, this functions draws a classic forest plot of the combined (direct + indirect) effects estimated.

Usage

```
plotForest(
  relativeEffectsEstimates,
  reference = NULL,
  lineSize = 0.5,
  pointSize = 1
)
```

Arguments

`relativeEffectsEstimates`	the output of `computeRelativeEffects`.
`reference`	string, which exposure to use as the reference. Choice is arbitrary. If nothing is supplied, the first value in the data frame is used.
`lineSize, pointSize`	scalars

plotNetwork *Visualise network*

Description

Draw the head-to-head comparisons as a network. The function relies on **ggraph** to do the heavy lifting.

Usage

```
plotNetwork(
  edges,
  nodes,
  maxNodeRadius = 0.1,
  nodeColour = "dodgerblue",
  labelOrientation = "radial",
  labelColour = "black",
  edgeColour = "dodgerblue",
  edgeAlpha = 0.2,
  treatmentMetaData = NULL,
  edgeMetaData = NULL
)
```

Arguments

`edges, nodes`	data frames with data on edges and nodes in the study graph.
`maxNodeRadius`	the radius of the largest node in the network on the original coordinate scales (= [-1, [1). Default = 0.1.
`labelOrientation`	string giving how to rotate the label. There are two allowed settings: `"radial"` (default) and `"horisontal"`/`"horizontal"`.
`edgeColour, nodeColour, labelColour`	strings given the colours of edges, nodes and label text.
`edgeAlpha`	scalar in [0, 1] giving the opacity of the edges (default: 0.2). 1 = complete opaque, 0 = completely transparent.
`treatmentMetaData`	NOT IN USE a `dataframe` with information about each node in the network, e.g., name and databases with data for this treatment.
`edgeMetaData`	NOT IN USE a `dataframe` with information about the head-to-head comparisons. Note that the number of head-to-head comparisons is computed so doesn't need to be supplied.

`plotNodesplitResults` *Visualise results of node-splitting analysis*

Description

Pending, but could be useful for getting an overview.

Usage

```
plotNodesplitResults(nodesplitResults)
```

Arguments

`nodesplitResults`	tidy data frame with consistency, direct and indirect HR estimates with CrIs.

`plotPairwiseEstimates` *Creates a heatmap plot of all pairwise estimates*

Description

Some customisation possible, but this is still pretty basic.

Usage

```
plotPairwiseEstimates(
  tidyEstimates,
  nDigits = 1,
  colour_low = "blue",
  colour_high = "red",
  contrastThreshold = 0.5,
  mid = "hr",
  lo = "cri95Lb",
  hi = "cri95Ub",
  textSize = 12,
  sucraEstimates = NULL
)
```

Arguments

`tidyEstimates`	data frame/tibble with all pairwise combinations to feature in the heatmap.
`nDigits`	scalar, how many digits should the shown estimates have? Default is `1`
`colour_low, colour_high`	valid colour specifications for the lower and higher bounds of the diverging colour scale
`contrastThreshold`	scalar, absolute values on the log-scale above this threshold will be white, otherwise they will be black
`mid, lo, hi`	strings, the names of the columns in the `tidyEstimates` data frame holding the point estimate (usually median) and lower and upper bounds of the credibility intervals (usually 2.5 and 97.5 percentiles)
`textSize`	scalar.
`sucraEstimates`	data frame with two columns: exposureId and SUCRA estimate as numeric value. Output from `computeSucra`. Default: `NULL`. If valid data supplied, the SUCRA estimates will be shown in the diagonal, and the exposures will be ordered by descending SUCRA estimate.

`plotPosteriorDensities` *Produces density plots of parameter estimates from the fitted models*

Description

Produces a **ggplot2** object so can be further customised as desired. If the results of a node-splitting analysis is supplied, the output is a list of plots, one for each analysis and one for the consistency fit.

Usage

```
plotPosteriorDensities(
  posteriorDraws,
  separateChains = TRUE,
  lineAlpha = 0.7,
  lineSize = 0.2,
  densityResolution = 256,
  wrapByModelName = FALSE
)
```

Arguments

`posteriorDraws`	tidy data frame, all posterior draws with (at least) the following columns: `modelName`, `chainId`, `mcmcIterationId`, `parameterName`, `parameterValue`.
`separateChains`	boolean, should chains be plotted as separate distributions (`TRUE`, default) or as one (`FALSE`)? Not plotting chains separately somewhat defeats the purpose of the visualisation but is anyway left to the user.
`lineAlpha`	the opacity of drawn lines. If `alpha` = `1`, lines will be entirely opaque masking overlain lines, so the default is `0.7`.
`lineSize`	scalar, thickness of the plot lines.
`densityResolution`	scalar, the number of grid points over which to estimate the densities. Higher values yield smoother curves but produces a larger object.
`wrapByModelName`	boolean, if the `modelName` column contains more than one distinct value, this should be set to `TRUE` to show separate plots for each model. Otherwise, the plot will combine samples for the same parameter from different models, which will give misleading results.

Value

A ggplot object, or a list of ggplot objects.

`plotRankogram` *Rank treatments by effectiveness*

Description

Produces a **ggplot2** object so can be further customised as desired. If the results of a node-splitting analysis is supplied, the output is a list of plots, one for each analysis and one for the consistency fit.

Usage

```
plotRankogram(rankProbabilities, lineSize = 1, lineAlpha = 0.9)
```

Arguments

`rankProbabilities`	output of `computeRankProbabilities`
`lineSize`	scalar, thickness of the plot lines.
`lineAlpha`	the opacity of drawn lines. If `alpha` = `1`, lines will be entirely opaque masking overlain lines, so the default is `0.7`.

Value

A ggplot object, or a list of ggplot objects.

plotRhat — *Plot the Rhat values of the fitted model*

Description

Instead of looking at the Rhat values in table format, it's usually easier to quickly scan a plot, and this function allows you to do just that.

Usage

```
plotRhat(
  rhat,
  threshold = 1.05,
  showParameters = FALSE,
  pointSize = 1,
  lineSize = 0.5
)
```

Arguments

`rhat`	output from `computeRhat`
`threshold`	scalar, what is the threshold below which you want all Rhat values to lie? Default is `1.05`. Set to `NULL` to remove the indicator in the plot.
`showParameters`	boolean, should parameters be distinguishable with colours? Potentially useful if any are above the threshold, so by default set to `FALSE`.
`pointSize`	scalar, size of points representing Rhat values.
`lineSize`	scalar, thickness of vertical indicator of threshold, if shown.

Value

A `ggplot` object.

plotTraces — *Produces density plots of parameter estimates from the fitted models*

Description

Produces a **ggplot2** object so can be further customised as desired. If the results of a node-splitting analysis is supplied, the output is a list of plots, one for each analysis and one for the consistency fit.

Usage

```
plotTraces(
  posteriorDraws,
  separateChains = TRUE,
  lineAlpha = 0.7,
  lineSize = 0.2,
  wrapByModelName = FALSE
)
```

Arguments

`posteriorDraws`	tidy data frame, all posterior draws with (at least) the following columns: `modelName`, `chainId`, `mcmcIterationId`, `parameterName`, `parameterValue`.
`separateChains`	boolean, should chains be plotted as separate distributions (`TRUE`, default) or as one (`FALSE`)? Not plotting chains separately somewhat defeats the purpose of the visualisation but is anyway left to the user.
`lineAlpha`	the opacity of drawn lines. If `alpha = 1`, lines will be entirely opaque masking overlain lines, so the default is `0.7`.
`lineSize`	scalar, thickness of the plot lines.
`wrapByModelName`	boolean, if the `modelName` column contains more than one distinct value, this should be set to `TRUE` to show separate plots for each model. Otherwise, the plot will combine samples for the same parameter from different models, which will give misleading results.

Value

A ggplot object, or a list of ggplot objects.

`runAnalyses` *Runs the analysis and returns results ready to be stored in database or used directly*

Description

Runs the analysis and returns results ready to be stored in database or used directly

Usage

```
runAnalyses(
  aggregatedResults,
  includeNodesplittingAnalysis = FALSE,
  nCores = 4,
  nWarmup = 1000,
  nIter = 3000,
  nChains = 4,
  fitThin = 1,
  referenceTreatmentId = NULL,
  preferredDirection = 1,
  modelType = "random",
  drawsThin = 10,
  nmaIds = NULL
)
```

Arguments

`aggregatedResults`	tidy data frame, result of calling `fetchTcoEstimates`.

`includeNodesplittingAnalysis`	boolean, should also node-splitting analyses be run? NB: This can be quite time-consuming, so is by default set to `FALSE`.
`nCores`	scalar, how many cores to use when running the analyses. If > `1`, **foreach** is used to parallel processing.
`nWarmup`	scalar, the number of iterations the sampler should do before it starts sampling from the posterior. This should be sufficient to the chains to have converged.
`nIter`	scalar, the number of posterior samples to draw per chain.
`nChains`	scalar, the number of JAGS chains to run
`fitThin`	scalar, thinning factor. `1` (default) keeps all samples whereas e.g. `10` would keep only every tenth sample.
`referenceTreatmentId`	string, the name of the exposure to use as the reference when computing pairwise estimates. If none given, the first in alphabetical order will be used.
`preferredDirection`	scalar, preferential direction of the outcome. `1` means higher values are preferred, `-1` means lower values are preferred.
`modelType`	string defining whether to run a random-effects (`"random"`, default) or a fixed-effects (`"fixed"`) mode.
`drawsThin,`	scalar, thinning factor used when fetching posterior draws to keep in the database. Used to produce further derived quantities, density plots of posterior distributions and trace plots.
`nmaIds`	scalar vector, if only specific network meta analyses should be run, their id's are supplied here. If none supplied, all will be run.

`saveToDatabase` *Saves the reconciled results to database*

Description

If desired, this function takes the reconciled results and saves them to the database for downstream use.

Usage

```
saveToDatabase(results, conn, schema, overwriteExistingTables = FALSE)
```

Arguments

`results`	output of `runAnalyses`.
`conn`	connection object.
`schema`	string, the name of the schema where the tables will live. Should probably be a dedicated schema to prevent conflicting names.
`overwriteExistingTables`	boolean, should tables be overridden? This is a bit dangerous and so set to `FALSE` by default. See note on dedicated schema above.

Index

computePairwiseEstimates, 2
computeRankProbabilities, 3
computeRhat, 3
computeSucra, 4

deriveEdges, 4
deriveNodes, 5

fetchParameterSummaries, 5
fetchPosteriorDraws, 6
fetchTcoEstimates, 6
fetchTidyNodesplitResults, 7
fitNetwork, 7

launchShinyApp, 8
loadFromDb, 8

plotForest, 9
plotNetwork, 9
plotNodesplitResults, 10
plotPairwiseEstimates, 10
plotPosteriorDensities, 11
plotRankogram, 12
plotRhat, 13
plotTraces, 13

runAnalyses, 14

saveToDatabase, 15