Package: metainsight (via r-universe)

October 14, 2026

Version 7.1.0

Title A 'shiny' Application for Network Meta-Analysis

Description Conduct network meta-analyses through a graphical user interface using 'bnma', 'gemtc' and 'netmeta' with additional analysis provided by 'meta' and 'metafor'. Frequentist, Bayesian, meta-regression and baseline risk meta-regression analyses can all be conducted using a consistent data structure and terminology. Many options are provided for downloading publication-ready outputs and analyses can be reproduced outside of the application by downloading a 'quarto' file. The interface was generated using 'shinyscholar'. The initial version of the app was described by Owen et al. (2018) <doi:10.1002/jrsm.1373>, Bayesian ranking visualisations were described by Nevill et al. (2023) <doi:10.1016/j.jclinepi.2023.02.016> and metaregression was described by Morris et al. (2025) <doi:10.1016/j.jclinepi.2025.111839>.

Depends R (>= 4.1.0)

Imports bayesplot, bnma (>= 1.4.0), bslib, coda, cookies, DT (>= 0.5), dplyr, gargoyle, gemtc (>= 1.1.1), ggiraphExtra, ggplot2, ggrepel, glue, gt, igraph, jsonlite, knitcitations, knitr, MCMCvis, magick, meta (>= 8.1.0), metafor, mirai (>= 2.0.0), netmeta (>= 3.1.0), patchwork, plotly, quarto, R6, rio, rintrojs, rmarkdown, rsvg, shiny (>= 1.8.1), shinyalert, shinybusy, shinyjs, stringr, shinyWidgets (>= 0.6.0), svglite, tidyr, xml2

Suggests jsonvalidate, mockery, pdftools, shinytest2, testthat

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder quarto

NeedsCompilation no

Author Alex Sutton [aut, cre], Naomi Bradbury [aut], Ryan Field [aut], Tom Morris [aut], Clareece Nevill [aut], Janion Nevill [aut], Rhiannon Owen [aut], Simon E. H. Smart [aut], Yiqiao Xin [aut], Nicola Cooper [ctb], Suzanne Freeman [ctb]

Maintainer Alex Sutton <ajs22@leicester.ac.uk>

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev libglpk-dev make libharfbuzz-dev libmagick++-dev gsfonts jags libicu-dev libpng-dev librsvg2-dev libuv1-dev libxml2-dev libssl-dev libnode-dev libx11-dev zlib1g-dev

Repository https://ajsajs22.r-universe.dev

Date/Publication 2026-06-15 14:41:11 UTC

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

RemoteRef HEAD

RemoteSha 726b17dfbd1c592d4f7207c797ab601d7ea7e161

Contents

metainsight-package . . . 3
baseline_compare . . . 3
baseline_deviance . . . 4
baseline_forest . . . 5
baseline_model . . . 6
baseline_regression . . . 8
baseline_summary . . . 9
bayes_compare . . . 9
bayes_details . . . 10
bayes_deviance . . . 11
bayes_forest . . . 12
bayes_mcmc . . . 13
bayes_model . . . 14
bayes_nodesplit . . . 15
bayes_nodesplit_plot . . . 16
bayes_ranking . . . 17
bayes_results . . . 18
covariate_model . . . 19
covariate_regression . . . 21
covariate_summary . . . 22
density_plots . . . 23
dic_table . . . 23
export_cinema . . . 24
freq_compare . . . 25
freq_forest . . . 26
freq_inconsistency . . . 27
freq_summary . . . 28
gelman_plots . . . 28
make_netconnect . . . 29
metaregression_plot . . . 30

network_structure 32
ranking_plot 32
ranking_table 33
run_metainsight 34
setup_configure 34
setup_configure_table 36
setup_exclude 37
setup_exclude_plot 38
setup_load 39
setup_upgrade 40
summary_char 41
summary_network 41
summary_study 42
trace_plots 43
write_plot 44

Index **45**

`metainsight-package` metainsight*: A flexible platform for meta-analysis*

Description

Run the application via the function `run_metainsight`

`baseline_compare` *Compare treatment pairs*

Description

Produce a table of comparisons of all treatment pairs for baseline risk models using `bnma::relative.effects.table()`

Usage

```
baseline_compare(model, logger = NULL)
```

Arguments

`model`	list. Object produced by `baseline_model()`
`logger`	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default `NULL`

Value

Relative effects table

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_iter, max_iter and check_iter are set low to run quickly, but should
# be left as the default values in real use

fitted_baseline_model <- baseline_model(configured_data = configured_data,
                                        regressor_type = "shared",
                                        n_iter = 120,
                                        max_iter = 120,
                                        check_iter = 10)

baseline_compare(model = fitted_baseline_model)
```

---

baseline_deviance *Produce deviance plots for baseline risk models*

Description

Produce deviance plotly plots for baseline risk models. Unlike for `bayes_model` output, only stem and leverage plots are produced.

Usage

```
baseline_deviance(model, async = FALSE)
```

Arguments

`model`	Output model produced by `baseline_model()`
`async`	Whether or not the function is being used asynchronously. Default `FALSE`

Value

list containing:

`deviance_mtc`	equivalent summary to that produced by `gemtc::mtc.deviance()`
`stem_plot`	plotly object
`lev_plot`	plotly object

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_iter, max_iter and check_iter are set low to run quickly, but should
# be left as the default values in real use
```

```
fitted_baseline_model <- baseline_model(configured_data = configured_data,
                                        regressor_type = "shared",
                                        n_iter = 120,
                                        max_iter = 120,
                                        check_iter = 10)

baseline_deviance(model = fitted_baseline_model)
```

`baseline_forest` *Produce a forest plot for baseline risk models*

Description

Produce a forest plot for a baseline risk model using `gemtc::blobbogram()`

Usage

```
baseline_forest(
  model,
  xmin = NULL,
  xmax = NULL,
  title = "Baseline risk regression analysis",
  ranking = FALSE,
  logger = NULL
)
```

Arguments

`model`	Output produced by `baseline_model()`
`xmin`	numeric. Minimum x-axis value. Default `NULL` in which case it is calculated internally
`xmax`	numeric. Maximum x-axis value. Default `NULL` in which case it is calculated internally
`title`	character. Title for the plot. Defaults to `Baseline risk regression analysis`
`ranking`	logical. Whether the function is being used in `baseline_ranking`
`logger`	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default `NULL`

Value

html. Contains the svg string to generate the plot. This will display the plot when at the end of a quarto or rmarkdown chunk. To view in the viewer panel of Rstudio, use `htmltools::browsable()`. The output can be saved using `write_plot()`.

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_iter, max_iter and check_iter are set low to run quickly, but should
# be left as the default values in real use

fitted_baseline_model <- baseline_model(configured_data = configured_data,
                                        regressor_type = "shared",
                                        n_iter = 120,
                                        max_iter = 120,
                                        check_iter = 10)

baseline_forest(model = fitted_baseline_model)
```

---

baseline_model *Fit a baseline risk regression model*

Description

Fit a baseline risk regression model using `bnma::network.run()`. The output is consistent with outputs produced by **gemtc**.

Usage

```
baseline_model(
  configured_data,
  regressor_type,
  n_iter = 20000,
  max_iter = 60000,
  check_iter = 10000,
  async = FALSE
)
```

Arguments

`configured_data`	list. Input dataset created by `setup_configure()` or `setup_exclude()`
`regressor_type`	character. Type of regression coefficient, either `shared`, `unrelated`, or `exchangeable`
`n_iter`	numeric. Number of simulation iterations. Defaults to `20000` and can normally be left unchanged
`max_iter`	numeric. The maximum number of iterations. Defaults to `60000` and can normally be left unchanged.
`check_iter`	numeric. The number of iterations after which convergence is checked for. Defaults to `10000` and can normally be left unchanged.
`async`	Whether or not the function is being used asynchronously. Default `FALSE`

Value

List of bnma related output:

`mtcResults`	model object itself carried through (needed to match existing code)
`covariate_value`	The mean covariate value, used for centring
`reference_treatment`	character. The `reference_treatment`from `configured_data`
`comparator_names`	Vector containing the names of the comparators
`a`	text output stating whether fixed or random effects
`cov_value_sentence`	text output stating the value for which the covariate has been set to for producing output
`slopes`	named list of slopes for the regression equations (unstandardised - equal to one 'increment')
`intercepts`	named list of intercepts for the regression equations at cov_value
`outcome`	character. The `outcome` from `configured_data`
`outcome_measure`	character. The `outcome_measure`from `configured_data`
`effects`	character. The `effects` from `configured_data`
`covariate_min`	Vector of minimum covariate values directly contributing to the regression
`covariate_max`	Vector of maximum covariate values directly contributing to the regression
`dic`	Summary of model fit
`sumresults`	Output of summary(model)
`regressor`	Type of regression coefficient

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_iter, max_iter and check_iter are set low to run quickly, but should
# be left as the default values in real use

fitted_baseline_model <- baseline_model(configured_data = configured_data,
                                        regressor_type = "shared",
                                        n_iter = 120,
                                        max_iter = 120,
                                        check_iter = 10)
```

baseline_regression *Baseline regression data*

Description

Generate data required to produce a metaregression plot for a baseline risk model.

Usage

```
baseline_regression(model, configured_data, async = FALSE)
```

Arguments

model	Output produced by baseline_model()
configured_data	list. Input dataset created by setup_configure() or setup_exclude()
async	Whether or not the function is being used asynchronously. Default FALSE

Value

List containing:

directness	list. Output from CalculateDirectness()
credible_regions	list. Output from CalculateCredibleRegions()

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_iter, max_iter and check_iter are set low to run quickly, but should
# be left as the default values in real use

fitted_baseline_model <- baseline_model(configured_data = configured_data,
                                        regressor_type = "shared",
                                        n_iter = 120,
                                        max_iter = 120,
                                        check_iter = 10)

regression_data <- baseline_regression(model = fitted_baseline_model,
                                       configured_data = configured_data)
```

baseline_summary *Summarise baseline risk*

Description

Produce a plot summarising baselink risk for each study arm

Usage

```
baseline_summary(configured_data, logger = NULL)
```

Arguments

configured_data	list. Input dataset created by `setup_configure()` or `setup_exclude()`
logger	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default `NULL`

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

baseline_summary(configured_data)
```

bayes_compare *Compare treatment pairs*

Description

Produce a table of comparisons of all treatment pairs for Bayesian models using `gemtc::relative.effect.table()`

Usage

```
bayes_compare(model, logger = NULL)

covariate_compare(...)
```

Arguments

model	list. Object created by `bayes_model()` or `covariate_model()`
logger	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default `NULL`
...	Parameters passed to `bayes_compare()`

Value

Relative effects table

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_adapt and n_iter are set low to run quickly, but should be left as the
# default values in real use

fitted_bayes_model <- bayes_model(configured_data = configured_data,
                                  n_adapt = 100,
                                  n_iter = 100)

bayes_compare(model = fitted_bayes_model)
```

---

bayes_details *Summarise a Bayesian model*

Description

Produce a summary of a Bayesian model

Usage

```
bayes_details(model, logger = NULL)

covariate_details(...)

baseline_details(...)
```

Arguments

model	Output produced by `baseline_model()`, `bayes_model()` or `covariate_model()`.
logger	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default `NULL`
...	Parameters passed to `bayes_details()`

Value

HTML summary of the model

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_adapt and n_iter are set low to run quickly, but should be left as the
# default values in real use

fitted_bayes_model <- bayes_model(configured_data = configured_data,
                                  n_adapt = 100,
                                  n_iter = 100)

bayes_details(model = fitted_bayes_model)
```

---

bayes_deviance *Produce deviance plots*

Description

Produce deviance plots using the output of `gemtc::mtc.deviance()` for Bayesian and covariate models. Because these plots are interactive, it is not currently possible to download them, although they can be included in html reports.

Usage

```
bayes_deviance(model, n_adapt = 5000, n_iter = 20000, async = FALSE)

covariate_deviance(...)
```

Arguments

`model`	Bayesian model produced by `bayes_model()` or `covariate_model()`
`n_adapt`	numeric. Number of adaptation iterations. Defaults to `5000` and can normally be left unchanged
`n_iter`	numeric. Number of simulation iterations. Defaults to `20000` and can normally be left unchanged
`async`	Whether or not the function is being used asynchronously. Default `FALSE`
`...`	Parameters passed to `bayes_deviance()`

Value

A list containing different elements depending on the input model:

When `model` was created by `bayes_model()` containing:

`deviance_mtc`	results from `gemtc::mtc.deviance()` for model$mtcResults
`deviance_ume`	results from `gemtc::mtc.deviance()` for UME model
`scat_plot`	plotly object

stem_plot	plotly object
lev_plot	plotly object

When model was created by covariate_model() containing:

deviance_mtc	results from gemtc::mtc.deviance() for model$mtcResults
stem_plot	plotly object
lev_plot	plotly object

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_adapt and n_iter are set low to run quickly, but should be left as the
# default values in real use

fitted_bayes_model <- bayes_model(configured_data = configured_data,
                                  n_adapt = 100,
                                  n_iter = 100)

bayes_deviance(model = fitted_bayes_model,
               n_adapt = 100,
               n_iter = 100)
```

---

bayes_forest *Bayesian forest plot*

Description

Produce a Bayesian forest plot with gemtc::forest()

Usage

```
bayes_forest(
  model,
  xmin = NULL,
  xmax = NULL,
  title = "",
  ranking = FALSE,
  logger = NULL
)

covariate_forest(...)
```

Arguments

model	list. Object created by bayes_model() or covariate_model()
xmin	numeric. Minimum x-axis value. Default NULL in which case it is calculated internally
xmax	numeric. Maximum x-axis value. Default NULL in which case it is calculated internally
title	character. Title for the plot. Default is no title
ranking	logical. Whether the function is being used in bayes_ranking
logger	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default NULL
...	Parameters passed to bayes_forest()

Value

html. Contains the svg string to generate the plot. This will display the plot when at the end of a quarto or rmarkdown chunk. To view in the viewer panel of Rstudio, use htmltools::browsable(). The output can be saved using write_plot().

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_adapt and n_iter are set low to run quickly, but should be left as the
# default values in real use

fitted_bayes_model <- bayes_model(configured_data = configured_data,
                                  n_adapt = 100,
                                  n_iter = 100)

bayes_forest(model = fitted_bayes_model)
```

---

bayes_mcmc *Markov chain Monte Carlo plots*

Description

Produce Markov chain Monte Carlo plots for Bayesian models

Usage

```
bayes_mcmc(model, async = FALSE)

covariate_mcmc(...)

baseline_mcmc(...)
```

Arguments

model	Output from baseline_model(), bayes_model() or covariate_model()
async	Whether or not the function is being used asynchronously. Default FALSE
...	Parameters passed to bayes_mcmc()

Value

list containing:

gelman_plots	Gelman plots
trace_plots	Trace plots
density_plots	Density plots
n_rows	The number of rows for each plot

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_adapt and n_iter are set low to run quickly, but should be left as the
# default values in real use

fitted_bayes_model <- bayes_model(configured_data = configured_data,
                                  n_adapt = 100,
                                  n_iter = 100)

bayes_mcmc(model = fitted_bayes_model)
```

---

bayes_model *Fit a Bayesian model*

Description

Fit a Bayesian model using **gemtc**

Usage

```
bayes_model(configured_data, n_adapt = 5000, n_iter = 20000, async = FALSE)
```

Arguments

configured_data	list. Input dataset created by setup_configure() or setup_exclude()
n_adapt	numeric. Number of adaptation iterations. Defaults to 5000 and can normally be left unchanged
n_iter	numeric. Number of simulation iterations. Defaults to 20000 and can normally be left unchanged
async	Whether or not the function is being used asynchronously. Default FALSE

Value

List containing:

mtcResults	mtc.result. Output from gemtc::mtc.run()
mtcRelEffects	mtc.result. Output from gemtc::relative.effect()
rel_eff_tbl	mtc.relative.effect.table. Output from gemtc::relative.effect.table()
sumresults	summary.mtc.result. Output from summary(mtcRelEffects)
mtcNetwork	mtc.network. Output from gemtc::mtc.network()
dic	dataframe. Containing the statistics 'Dbar', 'pD', 'DIC', and 'data points'
outcome	character. The outcome from configured_data
outcome_measure	character. The outcome_measurefrom configured_data
reference_treatment	character. The reference_treatmentfrom configured_data
effects	character. The effects from configured_data
seed	numeric. The seed from configured_data

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_adapt and n_iter are set low to run quickly, but should be left as the
# default values in real use

fitted_bayes_model <- bayes_model(configured_data = configured_data,
                                  n_adapt = 100,
                                  n_iter = 100)
```

---

bayes_nodesplit *Fit a Bayesian nodesplitting model*

Description

Fit a Bayesian nodesplitting model with gemtc::mtc.nodesplit(). This is not possible for all networks and the function will return an error if the nodes cannot be split.

Usage

```
bayes_nodesplit(configured_data, n_adapt = 5000, n_iter = 20000, async = FALSE)
```

Arguments

configured_data	list. Input dataset created by `setup_configure()` or `setup_exclude()`
n_adapt	numeric. Number of adaptation iterations. Defaults to `5000` and can normally be left unchanged
n_iter	numeric. Number of simulation iterations. Defaults to `20000` and can normally be left unchanged
async	Whether or not the function is being used asynchronously. Default `FALSE`

Value

`mtc.nodesplit` object containing an `mtc.result` object for each node

Examples

```
nodesplit_path <- system.file("extdata", "continuous_nodesplit.csv", package = "metainsight")
loaded_data <- setup_load(data_path = nodesplit_path,
                          outcome = "continuous")

configured_data <- setup_configure(loaded_data = loaded_data,
                                   reference_treatment = "Placebo",
                                   effects = "random",
                                   outcome_measure = "MD",
                                   ranking_option = "good",
                                   seed = 123)

# n_adapt and n_iter are set low to run quickly, but should be left as the
# default values in real use

nodesplit_model <- bayes_nodesplit(configured_data,
                                   n_adapt = 100,
                                   n_iter = 100)
```

---

bayes_nodesplit_plot *Bayesian nodesplitting forest plot*

Description

Produce a forest plot from nodesplitting results

Usage

```
bayes_nodesplit_plot(nodesplit, main_analysis = TRUE, logger = NULL)
```

Arguments

nodesplit	mtc.nodesplit object produced by bayes_nodesplit()
main_analysis	logical. Whether the analysis is the main or sensitivity analysis. Default TRUE.
logger	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default NULL

Value

html. Contains the svg string to generate the plot. This will display the plot when at the end of a quarto or rmarkdown chunk. To view in the viewer panel of Rstudio, use htmltools::browsable(). The output can be saved using write_plot().

Examples

```
nodesplit_path <- system.file("extdata", "continuous_nodesplit.csv", package = "metainsight")
loaded_data <- setup_load(data_path = nodesplit_path,
                          outcome = "continuous")

configured_data <- setup_configure(loaded_data = loaded_data,
                                   reference_treatment = "Placebo",
                                   effects = "random",
                                   outcome_measure = "MD",
                                   ranking_option = "good",
                                   seed = 123)

nodesplit_model <- bayes_nodesplit(configured_data,
                                   n_adapt = 100,
                                   n_iter = 100)

bayes_nodesplit_plot(nodesplit_model)
```

---

bayes_ranking *Treatment rankings*

Description

Generate treatment ranking data required to produce SUCRA plots from Bayesian models

Usage

```
bayes_ranking(model, configured_data, logger = NULL)

baseline_ranking(...)

covariate_ranking(...)
```

Arguments

model	list. Output produced by baseline_model(), bayes_model() or covariate_model().
configured_data	list. Input dataset created by setup_configure() or setup_exclude()
logger	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default NULL
...	Parameters passed to bayes_ranking()

Value

List of output created by rankdata()

SUCRA	Dataframe of SUCRA data
Colour	Dataframe of colours
Cumulative	Dataframe of cumulative ranking probabilities
Probabilities	Dataframe of ranking probabilities
Network	Dataframe of network characteristics

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_adapt and n_iter are set low to run quickly, but should be left as the
# default values in real use

fitted_bayes_model <- bayes_model(configured_data = configured_data,
                                  n_adapt = 100,
                                  n_iter = 100)

ranking_data <- bayes_ranking(fitted_bayes_model, configured_data)
```

---

bayes_results *Summarise a Bayesian model*

Description

Produce a table summarising Bayesian models

Usage

```
bayes_results(model, logger = NULL)

covariate_results(...)

baseline_results(...)
```

Arguments

model	list. Output produced by baseline_model(), bayes_model() or covariate_model().
logger	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default NULL
...	Parameters passed to bayes_results()

Value

HTML summary of the model

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_adapt and n_iter are set low to run quickly, but should be left as the
# default values in real use

fitted_bayes_model <- bayes_model(configured_data = configured_data,
                                  n_adapt = 100,
                                  n_iter = 100)

bayes_results(fitted_bayes_model)
```

---

covariate_model *Fit a covariate regression model*

Description

Fit a covariate regression model using **gemtc**.

Usage

```
covariate_model(
  configured_data,
  covariate_value,
  regressor_type,
  covariate_model_output = NULL,
  n_adapt = 5000,
  n_iter = 20000,
  async = FALSE
)
```

Arguments

configured_data	list. Input dataset created by `setup_configure()` or `setup_exclude()`
covariate_value	numeric. The value at which to fit the model. Must be greater than or equal to the minimum value and less than or equal to the maximum value in `configured_data`
regressor_type	character. Type of regression coefficient, either `shared`, `unrelated`, or `exchangeable`
covariate_model_output	list. The output of the function. Default `NULL`. When supplied, only the output is recalculated for a given covariate value, rather than refitting the model.
n_adapt	numeric. Number of adaptation iterations. Defaults to `5000` and can normally be left unchanged
n_iter	numeric. Number of simulation iterations. Defaults to `20000` and can normally be left unchanged
async	Whether or not the function is being used asynchronously. Default `FALSE`

Value

List of **gemtc** related output:

mtcResults	model object from `gemtc::mtc.run()` carried through (needed to match existing code)
mtcRelEffects	data relating to presenting relative effects
rel_eff_tbl	table of relative effects for each comparison
covariate_value	The covariate value originally passed into this function
reference_treatment	character. The `reference_treatment`from `configured_data`
comparator_names	Vector containing the names of the comparators
a	text output stating whether fixed or random effects
sumresults	summary output of relative effects
dic	data frame of model fit statistics
cov_value_sentence	text output stating the value for which the covariate has been set to for producing output
slopes	named list of slopes for the regression equations (unstandardised - equal to one 'increment')
intercepts	named list of intercepts for the regression equations at covariate_value
outcome	character. The `outcome` from `configured_data`
outcome_measure	character. The `outcome_measure`from `configured_data`
effects	character. The `effects` from `configured_data`
mtcNetwork	The network object from GEMTC
covariate_min	Vector of minimum covariate values directly contributing to the regression
covariate_max	Vector of maximum covariate values directly contributing to the regression

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_adapt and n_iter are set low to run quickly, but should be left as the
# default values in real use

# initial model
fitted_covariate_model <- covariate_model(configured_data = configured_data,
                                          covariate_value = 98,
                                          regressor_type = "shared",
                                          n_adapt = 100,
                                          n_iter = 100)

# updated for new covariate value
updated_covariate_model <- covariate_model(configured_data = configured_data,
                                          covariate_value = 97,
                                          regressor_type = "shared",
                                        covariate_model_output = fitted_covariate_model,
                                          n_adapt = 100,
                                          n_iter = 100)
```

---

covariate_regression *Covariate regression data*

Description

Generate data from a covariate model required to produce a metaregression plot

Usage

```
covariate_regression(model, configured_data, async = FALSE)
```

Arguments

model	list. Output created by `covariate_model()`
configured_data	list. Input dataset created by `setup_configure()` or `setup_exclude()`
async	Whether or not the function is being used asynchronously. Default `FALSE`

Value

List containing:

directness	list. Output from `CalculateDirectness()`
credible_regions	list. Output from `CalculateCredibleRegions()`

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_adapt and n_iter are set low to run quickly, but should be left as the
# default values in real use

fitted_covariate_model <- covariate_model(configured_data = configured_data,
                                          covariate_value = 98,
                                          regressor_type = "shared",
                                          n_adapt = 100,
                                          n_iter = 100)

regression_data <- covariate_regression(model = fitted_covariate_model,
                                        configured_data = configured_data)
```

---

covariate_summary *Summarise covariate data*

Description

Produce a plot summarising the covariate value for each study arm

Usage

```
covariate_summary(configured_data, logger = NULL)
```

Arguments

configured_data	list. Input dataset created by `setup_configure()` or `setup_exclude()`
logger	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default `NULL`

Value

html. Contains the svg string to generate the plot. This will display the plot when at the end of a quarto or rmarkdown chunk. To view in the viewer panel of Rstudio, use `htmltools::browsable()`. The output can be saved using `write_plot()`.

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

covariate_summary(configured_data)
```

density_plots *Creates posterior density plots of MCMC samples.*

Description

Creates posterior density plots of MCMC samples.

Usage

```
density_plots(model, parameters)
```

Arguments

model	Model output.
parameters	Vector of parameters to create density plots for.

Value

List of ggplot density plots.

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_adapt and n_iter are set low to run quickly, but should be left as the
# default values in real use

fitted_bayes_model <- bayes_model(configured_data = configured_data,
                                  n_adapt = 100,
                                  n_iter = 100)

mcmc <- bayes_mcmc(model = fitted_bayes_model)

density_plots(fitted_bayes_model$mtcResults, mcmc$parameters)
```

dic_table *Create a summary table of deviance information criterion stats for Bayesian models*

Description

Create a summary table of deviance information criterion stats for Bayesian models

Usage

```
dic_table(dic, analysis = "all")
```

Arguments

dic	dataframe of DIC stats from baseline_model, bayes_model() or covariate_model()
analysis	Whether the analysis is using all studies (all) or a subset (sub)

---

export_cinema *export_cinema*

Description

Prepare project into a JSON format that CINeMA can read.

Usage

```
export_cinema(configured_data, gemtc_results = NULL, logger = NULL)
```

Arguments

configured_data	list. Input dataset created by setup_configure() or setup_exclude()
gemtc_results	Output from gemtc::mtc.run(), as returned in the mtcResults list element from bayes_model(). If this parameter is NULL then the frequentist analysis results found in 'contributions' are used. If it is not NULL then the Bayesian analysis results contained in this parameter are used. Defaults to NULL.
logger	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default NULL

Value

JSON string with the following structure: A named list of lists. The top level list contains items:

- "project" Information for CINeMA project
  - "CM" Contribution matrices
    - "contributionMatrices" output from .PrepareAnalysisForCinema()
  - "format" Data format. Always "long"
  - "type" Outcome type. Either "binary" or "continuous"
  - "Studies" Study data
    - "long" Output from .PrepareDataForCinema()

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

cinema_project <- export_cinema(configured_data = configured_data)

writeLines(cinema_project, tempfile(fileext = ".json"))
```

`freq_compare` *Compare treatments for frequentist models*

Description

Produce a comparison table of treatments using `netmeta::netleague()`.

Usage

```
freq_compare(configured_data, logger = NULL)
```

Arguments

`configured_data`	list. Input dataset created by `setup_configure()` or `setup_exclude()`
`logger`	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default `NULL`

Value

Dataframe of comparisons with one row and one column per treatment

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

freq_compare(configured_data = configured_data)
```

`freq_forest` *Produce a frequentist forest plot*

Description

Produce an annotated frequentist forest plot using `meta::forest()`

Usage

```
freq_forest(
  configured_data,
  xmin = NULL,
  xmax = NULL,
  title = "",
  logger = NULL
)
```

Arguments

`configured_data`	list. Input dataset created by `setup_configure()` or `setup_exclude()`
`xmin`	numeric. Minimum x-axis value. Default `NULL` in which case it is calculated internally
`xmax`	numeric. Maximum x-axis value. Default `NULL` in which case it is calculated internally
`title`	character. Title for the plot.
`logger`	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default `NULL`

Value

html. Contains the svg string to generate the plot. This will display the plot when at the end of a quarto or rmarkdown chunk. To view in the viewer panel of Rstudio, use `htmltools::browsable()`. The output can be saved using `write_plot()`.

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

freq_forest(configured_data = configured_data)
```

`freq_inconsistency` *Inconsistency tables for frequentist models*

Description

Produce inconsistency tables using `netmeta::netsplit()`

Usage

```
freq_inconsistency(configured_data, logger = NULL)
```

Arguments

`configured_data`	list. Input dataset created by `setup_configure()` or `setup_exclude()`
`logger`	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default `NULL`

Value

Dataframe of inconsistency data:

`Comparison`	Treatment comparison
`No.Studies`	Number of studies
`NMA`	NMA treatment effect estimate
`Direct`	Direct treatment effect estimate
`Indirect`	Indirect treatment effect estimate
`Difference`	Difference between treatment effects
`Diff_95CI_lower`	2.5% limit of difference in treatment effects
`Diff_95CI_upper`	97.5% limit of difference in treatment effects
`pValue`	p-value for test of "difference in treatment effects == 0"

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

freq_inconsistency(configured_data = configured_data)
```

freq_summary *Summary forest plot matrix*

Description

Produce a summary forest plot matrix with treatments ranked by SUCRA score, determined by `netmeta::rankogram()`. This function can only be used when `configured_data` contains between 3 and 10 treatments.

Usage

```
freq_summary(configured_data, plot_title = "", logger = NULL)
```

Arguments

`configured_data`	list. Input dataset created by `setup_configure()` or `setup_exclude()`
`plot_title`	character. Title of the plot. Default is no title.
`logger`	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default `NULL`

Value

html. Contains the svg string to generate the plot. This will display the plot when at the end of a quarto or rmarkdown chunk. To view in the viewer panel of Rstudio, use `htmltools::browsable()`. The output can be saved using `write_plot()`.

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

freq_summary(configured_data = configured_data)
```

gelman_plots *Creates Gelman plots for a gemtc or bnma model.*

Description

Creates Gelman plots for a gemtc or bnma model.

Usage

```
gelman_plots(gelman_data, parameters)
```

Arguments

gelman_data	List of outputs from gelman_preplot
parameters	Vector of parameters mentioned in the previous argument.

Value

List of ggplot Gelman plots

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_adapt and n_iter are set low to run quickly, but should be left as the
# default values in real use

fitted_bayes_model <- bayes_model(configured_data = configured_data,
                                  n_adapt = 100,
                                  n_iter = 100)

mcmc <- bayes_mcmc(model = fitted_bayes_model)

gelman_plots(mcmc$gelman_data, mcmc$parameters)
```

---

make_netconnect *Creates network connectivity info displayed under network plots*

Description

Creates network connectivity info displayed under network plots

Usage

```
make_netconnect(freq)
```

Arguments

freq	List of NMA results created by freq_wrap().

Value

Vector summarising network connectivity created by netmeta::netconnection().

`metaregression_plot` *Produce a meta-regression plot*

Description

Produce a composite meta-regression plot which comprises plots showing direct and indirect evidence from baseline or covariate models. The design was adapted from Donegan et al. (2018) https://onlinelibrary.wiley.com/doi/10.1002/jrsm.1292

Usage

```
metaregression_plot(
  model,
  configured_data,
  regression_data,
  comparators,
  include_covariate = FALSE,
  include_ghosts = FALSE,
  include_extrapolation = FALSE,
  include_credible = FALSE,
  credible_opacity = 0.2,
  covariate_symbol = "circle open",
  covariate_symbol_size = 10,
  legend_position = "BR",
  logger = NULL
)
```

Arguments

`model`	Output from `baseline_model()` or `covariate_model()`
`configured_data`	list. Input dataset created by `setup_configure()` or `setup_exclude()`
`regression_data`	Output from `baseline_regression()` or `covariate_regression()`
`comparators`	Vector of treatments to plot in colour. Cannot include the `reference_treatment` used in the model.
`include_covariate`	logical. Whether the value of the covariate should be plotted as a vertical line. Defaults to `FALSE`.
`include_ghosts`	logical. Whether the other comparator studies should be plotted in grey in the background of the plot. Defaults to `FALSE`.
`include_extrapolation`	logical. Whether the regression lines should be extrapolated beyond the range of the given data as dashed lines. Defaults to `FALSE`.

include_credible
: logical. Whether the credible regions should be plotted for the specified comparators. These will be partially transparent regions. Defaults to FALSE.

credible_opacity
: numeric. The opacity of the credible regions. Can be any value between 0 and 1, inclusive. Defaults to 0.2.

covariate_symbol
: character. The selected symbol for displaying covariates. Defaults to circle open. Possible values are:

  cross Crosses

  circle_open Open circles

  none No symbols in which case only the plot of direct evidence is produced

covariate_symbol_size
: numeric. Size of the covariate symbols. Defaults to 10.

legend_position
: character. The position of the legend. Defaults to BR. Possible values are:

  BR Bottom-right of the plot area

  BL Bottom-left of the plot area

  TR Top-right of the plot area

  TL Top-left of the plot area

logger
: Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default NULL

Value

html. Contains the svg string to generate the plot. This will display the plot when at the end of a quarto or rmarkdown chunk. To view in the viewer panel of Rstudio, use htmltools::browsable(). The output can be saved using write_plot().

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_iter, max_iter and check_iter are set low to run quickly, but should
# be left as the default values in real use

fitted_baseline_model <- baseline_model(configured_data = configured_data,
                                        regressor_type = "shared",
                                        n_iter = 120,
                                        max_iter = 120,
                                        check_iter = 10)

regression_data <- baseline_regression(model = fitted_baseline_model,
                                       configured_data = configured_data)

metaregression_plot(model = fitted_baseline_model,
                    configured_data = configured_data,
```

```
                    regression_data = regression_data,
                    comparators = c("the_Younger", "the_Little"))
```

`network_structure` *Calculate edge.weights for network*

Description

Calculate edge.weights for network

Usage

```
network_structure(freq, order = NA)
```

Arguments

`freq`	list. Output from `frequentist()`
`order`	character. Vector of treatments names in rank order.

Value

data.frame containing the number of studies that compare each treatment against the reference treatment.

`ranking_plot` *Treatment ranking plots*

Description

Produce either a rankogram or radial SUCRA plot ranking the treatments

Usage

```
ranking_plot(
  ranking_data,
  style,
  colourblind = FALSE,
  simple = FALSE,
  regression_text = "",
  logger = NULL
)
```

Arguments

ranking_data	list created by baseline_ranking(), bayes_ranking() or covariate_ranking().
style	character. The style of plot to produce. Either rankogram or radial
colourblind	logical. Whether to use a colourblind-friendly palette. Defaults to FALSE.
simple	logical. Whether to display a simplified version of the radial plot. Does not affect the rankogram plot. Defaults to FALSE
regression_text	Text to show for regression. Defaults to no text.
logger	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default NULL

Value

html. Contains the svg string to generate the plot. This will display the plot when at the end of a quarto or rmarkdown chunk. To view in the viewer panel of Rstudio, use htmltools::browsable(). The output can be saved using write_plot().

---

ranking_table *Ranking probability table*

Description

Ranking probability table

Usage

```
ranking_table(ranking_data)
```

Arguments

ranking_data	list created by bayes_ranking().

Value

dataframe

run_metainsight *Run* "metainsight" *Application*

Description

This function runs the *"metainsight"* application in the user's default web browser.

Usage

```
run_metainsight(
  launch.browser = TRUE,
  port = getOption("shiny.port"),
  load_file = NULL
)
```

Arguments

launch.browser	Whether or not to launch a new browser window.
port	The port for the shiny server to listen on. Defaults to a random available port.
load_file	Path to a saved session file which will be loaded when the app is opened

Author(s)

Jamie Kass jkass@gradcenter.cuny.edu

Gonzalo E. Pinilla-Buitrago gpinillabuitrago@gradcenter.cuny.edu

Simon E. H. Smart simon.smart@cantab.net

Examples

```
if(interactive()) {
run_metainsight()
}
```

setup_configure *Configure the analysis*

Description

Checks the connectivity of the loaded data and converts it into formats for later analyses. Conducts a frequentist analysis using `netmeta::netmeta()`. The output can be passed to many other functions - all `summary_` and `freq_` functions and `bayes_model()`, `baseline_model()` and `covariate_model()`.

Usage

```
setup_configure(
  loaded_data,
  reference_treatment,
  effects,
  outcome_measure,
  ranking_option,
  seed,
  logger = NULL
)
```

Arguments

loaded_data	list. Output from setup_load()
reference_treatment	character. The reference treatment of the dataset
effects	character. Type of model to fit, either random or fixed
outcome_measure	character. Outcome measure of the dataset. Either OR, RR or RD when outcome is binary or MD or SMD when outcome is continuous
ranking_option	character. good if the treatment effect is desirable, else bad
seed	numeric. Seed used to fit the models.
logger	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default NULL

Value

configured_data containing:

treatments	dataframe. Treatment names and IDs
reference_treatment	character. The selected reference treatment
disconnected_indices	vector. Indices of studies that are not connected to the main network
connected_data	dataframe. A subset of the data containing only connected studies
non_covariate_data	dataframe. The uploaded data with covariates removed
covariate	A list containing these items if covariate data exists or else empty: **column** character. Name of the column containing covariate data **name** character. Name of the covariate **type** character. Whether the covariate is binary or continuous
freq	list. Processed data for frequentist analyses created by frequentist()
outcome	character. Whether the data is binary or continuous
outcome_measure	character. Outcome measure of the dataset.

effects	character. Whether the models are fixed or random effects
ranking_option	character. Whether higher values in the data are good or bad
seed	numeric. A seed value to be passed to models

Examples

```
minimal_data_path <- system.file("extdata", "continuous_minimal.csv", package = "metainsight")
loaded_data <- setup_load(data_path = minimal_data_path,
                          outcome = "continuous")

configured_data <- setup_configure(loaded_data = loaded_data,
                                   reference_treatment = "the Great",
                                   effects = "random",
                                   outcome_measure = "MD",
                                   ranking_option = "good",
                                   seed = 123)
```

---

setup_configure_table *Summarise the analysis configuration*

Description

Create a table summarising how the analysis has been configured

Usage

```
setup_configure_table(configured_data)
```

Arguments

configured_data	list. Input dataset created by setup_configure() or setup_exclude()

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

setup_configure_table(configured_data)
```

setup_exclude *Takes the configured data, removes any excluded studies and returns subsets of the data to be passed to other functions.*

Description

Takes the configured data, removes any excluded studies and returns subsets of the data to be passed to other functions.

Usage

```
setup_exclude(configured_data, exclusions, async = FALSE)
```

Arguments

configured_data	list. Input dataset created by `setup_configure()` or `setup_exclude()`
exclusions	character. Vector of study names to exclude.
async	Whether or not the function is being used asynchronously. Default `FALSE`

Value

`configured_data` containing:

treatments	dataframe. Treatment names and IDs
reference_treatment	character. The selected reference treatment
connected_data	dataframe. A subset of the data containing only connected studies
covariate	A list containing these items if covariate data exists or else empty:

- `cross`: Crosses
- `circle_open`: Open circles
- `none`: No symbols in which case only the plot of direct evidence is

freq	list. Processed data for frequentist analyses created by `frequentist()`
outcome	character. Whether the data is `binary` or `continuous`
outcome_measure	character. Outcome measure of the dataset.
effects	character. Whether the models are `fixed` or `random` effects
ranking_option	character. Whether higher values in the data are good or bad
seed	numeric. A seed value to be passed to models

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

subsetted_data <- setup_exclude(configured_data = configured_data,
                                exclusions = c("Leo", "Minerva"))
```

`setup_exclude_plot` *Produce an version of the* `summary_study()` *plot for use in the interface for excluding studies. Inside the app this is interactive, but it can also be rendered for non-interactive use.*

Description

Produce an version of the `summary_study()` plot for use in the interface for excluding studies. Inside the app this is interactive, but it can also be rendered for non-interactive use.

Usage

```
setup_exclude_plot(configured_data, exclusions = NULL, hover = FALSE)
```

Arguments

`configured_data`	list. Input dataset created by `setup_configure()` or `setup_exclude()`
`exclusions`	character. Vector of excluded studies. Defaults to `NULL`, but can be used to reset on loading
`hover`	logical. Whether change the cursor on clickable lines. Defaults to `FALSE`

Value

html. Contains the svg string to generate the plot. This will display the plot when at the end of a quarto or rmarkdown chunk. To view in the viewer panel of Rstudio, use `htmltools::browsable()`. The output can be saved using `write_plot()`.

setup_load *Load data*

Description

Load data from a spreadsheet or a default dataset and assess the data for validity. This checks the column names for required columns and balanced wide format numbered columns. Data can be in either a long or wide format; long data has one row per study arm whereas wide data has one row per study. For continuous outcomes, long data should contain the columns: `Study` - an identifier, e.g. author and year, `T` - treatment, `N` - number of participants, `Mean` - mean value of the outcome, `SD` - standard deviation of the outcome. Wide data for continuous outcomes should contain: `Study`, `N.1`, `N.2`, `Mean.1`, `Mean.2`, `SD.1`, `SD.2` where the number refers to the arm of the study and extra columns should be added depending on the number of arms. For binary outcomes, long data should contain: `Study`, `T`, `N` (as for continuous data) and `R` - the number of participants with the outcome of interest. Wide data for binary outcomes should follow the same convention: `Study`, `T.1`, `T.2`, `R.1`, `R.2`, `N.1`, `N.2`. Additionally, a `covar.<name>` column can be added to all formats containing covariate data where `<name>` should be replaced with the name of the covariate. For long data, covariate values must be equal for every study arm. Risk of bias data can also be included with all columns containing values ranging from 1 (low risk) to 3 (high risk): `rob` for the overall risk of bias, `indirectness` for indirectness and `rob.<name>` for up to ten individual components.

Usage

```
setup_load(data_path = NULL, outcome, logger = NULL)
```

Arguments

`data_path`	character. Path to the file (either a `.csv` or `.xlsx`) to be loaded or if `NULL` load the default data
`outcome`	character. Outcome type for the dataset. Either `binary` or `continuous`.
`logger`	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default `NULL`

Value

List containing:

`is_data_valid`	logical. Whether the data is valid
`is_data_uploaded`	logical. Whether the data is uploaded
`data`	dataframe. The data that was uploaded or the default data if no data_path was provided
`treatments`	Dataframe of the treatments in the data. `NULL` if `is_data_valid` is `FALSE`
`outcome`	character. Whether the data is `binary` or `continuous`

Examples

```
# load data from a file
minimal_data_path <- system.file("extdata", "continuous_minimal.csv", package = "metainsight")
loaded_data <- setup_load(data_path = minimal_data_path,
                          outcome = "continuous")

# load default data
loaded_data <- setup_load(outcome = "binary")
```

`setup_upgrade` *Upgrade old data formats*

Description

Loads a `.csv` file and converts it into a suitable format for use by `setup_load`.

Usage

```
setup_upgrade(data_path, treatments, logger = NULL)
```

Arguments

`data_path`	character. Path to the file to be upgraded
`treatments`	character. The treatments in the data separated by commas.
`logger`	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default `NULL`

Value

Dataframe containing the upgraded data

Examples

```
old_data_path <- system.file("extdata", "old_data.csv", package = "metainsight")
upgraded_data <- setup_upgrade(data_path = old_data_path,
                               treatments = "A,B,C,D,E")
```

summary_char *Characterise the data*

Description

Produce summaries of network characteristics, treatments and treatment pairs such as the number of participants and and the mean outcome. Inspired by `BUGSnet::net.tab()`

Usage

```
summary_char(configured_data, logger = NULL)
```

Arguments

`configured_data`	list. Input dataset created by `setup_configure()` or `setup_exclude()`
`logger`	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default `NULL`

Value

List containing:

`network`	network characteristics
`treatments`	treatment characteristics
`pairs`	treatment pair characteristics

summary_network *Produce a plot of the network using* `netmeta::netgraph()`

Description

Produce a plot of the network using `netmeta::netgraph()`

Usage

```
summary_network(
  configured_data,
  style,
  label_size = 1,
  title = "",
  logger = NULL
)
```

Arguments

configured_data	list. Input dataset created by `setup_configure()` or `setup_exclude()`
style	character. The plot to produce, either `netgraph` or `netplot`
label_size	numeric. The size of labels in the plots. Default of `1`.
title	character. Title of plot. Default of no title.
logger	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default `NULL`

Value

html. Contains the svg string to generate the plot. This will display the plot when at the end of a quarto or rmarkdown chunk. To view in the viewer panel of Rstudio, use `htmltools::browsable()`. The output can be saved using `write_plot()`.

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

summary_network(configured_data = configured_data,
                style = "netgraph")
```

---

summary_study *Produce a forest plot of pairwise comparisons, grouped by treatment pairs. If risk of bias data was loaded these are also included.*

Description

Produce a forest plot of pairwise comparisons, grouped by treatment pairs. If risk of bias data was loaded these are also included.

Usage

```
summary_study(
  configured_data,
  plot_area_width = 6,
  colourblind = FALSE,
  x_min = NULL,
  x_max = NULL,
  interactive = FALSE,
  logger = NULL
)
```

Arguments

configured_data	list. Input dataset created by `setup_configure()` or `setup_exclude()`
plot_area_width	numeric. The width of the plot area containing the treatment effects in inches. Defaults to 6.
colourblind	logical. Whether to use a colourblind-friendly palette. Defaults to FALSE
x_min	numeric. Minimum value for the x-axis. Defaults to NULL. For binary outcomes values should be wrapped in `log()`
x_max	numeric. Maximum value for the x-axis. Defaults to NULL. For binary outcomes values should be wrapped in `log()`
interactive	logical. Whether the plot should be altered for preparation into an interactive interface. Defaults to FALSE
logger	Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default NULL

Value

html. Contains the svg string to generate the plot. This will display the plot when at the end of a quarto or rmarkdown chunk. To view in the viewer panel of Rstudio, use `htmltools::browsable()`. The output can be saved using `write_plot()`.

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

summary_study(configured_data = configured_data)
```

---

trace_plots *Creates trace plots of MCMC samples.*

Description

Creates trace plots of MCMC samples.

Usage

```
trace_plots(model, parameters)
```

Arguments

model	Model output.
parameters	Vector of parameters to create trace plots for.

Value

List of ggplot trace plots.

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

# n_adapt and n_iter are set low to run quickly, but should be left as the
# default values in real use

fitted_bayes_model <- bayes_model(configured_data = configured_data,
                                  n_adapt = 100,
                                  n_iter = 100)

mcmc <- bayes_mcmc(model = fitted_bayes_model)

trace_plots(fitted_bayes_model$mtcResults, mcmc$parameters)
```

---

write_plot *Write plots to a file*

Description

Write an svg plot to either a png, pdf or svg file.

Usage

```
write_plot(svg, file)
```

Arguments

svg	html. containing the svg string, returned from `crop_svg()`
file	character. The file to which to write.

Examples

```
configured_data_path <- system.file("extdata", "configured_data.Rds", package = "metainsight")
configured_data <- readRDS(configured_data_path)

tmp <- tempfile(fileext = ".png")
summary_network(configured_data = configured_data,
                style = "netgraph") |>
                write_plot(tmp)
unlink(tmp)
```

Index

baseline_compare, 3
baseline_details *(*bayes_details*)*, 10
baseline_deviance, 4
baseline_forest, 5
baseline_mcmc *(*bayes_mcmc*)*, 13
baseline_model, 6
baseline_ranking *(*bayes_ranking*)*, 17
baseline_regression, 8
baseline_results *(*bayes_results*)*, 18
baseline_summary, 9
bayes_compare, 9
bayes_details, 10
bayes_deviance, 11
bayes_forest, 12
bayes_mcmc, 13
bayes_model, 14
bayes_nodesplit, 15
bayes_nodesplit_plot, 16
bayes_ranking, 17
bayes_results, 18

covariate_compare *(*bayes_compare*)*, 9
covariate_details *(*bayes_details*)*, 10
covariate_deviance *(*bayes_deviance*)*, 11
covariate_forest *(*bayes_forest*)*, 12
covariate_mcmc *(*bayes_mcmc*)*, 13
covariate_model, 19
covariate_ranking *(*bayes_ranking*)*, 17
covariate_regression, 21
covariate_results *(*bayes_results*)*, 18
covariate_summary, 22

density_plots, 23
dic_table, 23

export_cinema, 24

freq_compare, 25
freq_forest, 26
freq_inconsistency, 27
freq_summary, 28

gelman_plots, 28

make_netconnect, 29
metainsight *(*metainsight-package*)*, 3
metainsight-package, 3
metaregression_plot, 30

network_structure, 32

ranking_plot, 32
ranking_table, 33
run_metainsight, *3*, 34

setup_configure, 34
setup_configure_table, 36
setup_exclude, 37
setup_exclude_plot, 38
setup_load, 39
setup_upgrade, 40
summary_char, 41
summary_network, 41
summary_study, 42

trace_plots, 43

write_plot, 44