Package 'imt'

May 8, 2026

Title Impact Measurement Toolkit

Version 1.0.0

Description A toolkit for causal inference in experimental and observational studies. Implements various simple Bayesian models including linear, negative binomial, and logistic regression for impact estimation. Provides functionality for randomization and checking baseline equivalence in experimental designs. The package aims to simplify the process of impact measurement for researchers and analysts across different fields. Examples and detailed usage instructions are available at <https://book.martinez.fyi>.

License Apache License (>= 2.0)

Encoding UTF-8

RoxygenNote 7.3.2

Biarch true

Depends R (>= 3.4.0)

Imports R6, bayesplot, caret, dplyr, ggplot2, glue, magrittr, methods, Rcpp, RcppParallel, rlang, rstan, rstantools, tibble, tidyr, vizdraws, purrr, scales, tidyselect

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

Suggests knitr, rmarkdown, roxygen2, testthat

SystemRequirements GNU make

URL https://github.com/google/imt

BugReports https://github.com/google/imt/issues

NeedsCompilation yes

Author Ignacio Martinez [aut, cre] (ORCID: <https://orcid.org/0000-0002-3721-8172>)

Maintainer Ignacio Martinez <martinezig@google.com>

Repository CRAN

Date/Publication 2024-09-02 21:40:10 UTC

Contents

`.combineColumns` *Combine and Unite Columns*

Description

This function takes a data frame, identifies columns that are not specified in the exclusion list, and combines them into a new column called 'group'. The original columns used for the combination are then removed. Finally, it returns a data frame with only the 'group', 'variables', 'std_diff', and 'balance' columns.

Usage

```
.combineColumns(df)
```

Arguments

`df`	A data frame containing the columns to be combined.

Value

A data frame with the combined 'group' column and specified columns.

.randomize_internal *Add a Random Treatment Indicator Column to a Data Frame*

Description

This function takes a data frame and adds a new column named "treated" with randomly assigned TRUE/FALSE values. Randomization can be done either on the entire data frame or stratified by specified columns. The probability of being assigned to the treatment group can be specified, with a default of 0.5.

Usage

```
.randomize_internal(data, group_by = NULL, seed = NULL, pr_treated = 0.5)
```

Arguments

data	The input data frame.
group_by	(Optional) A character vector of column names to stratify the randomization. If provided, the randomization will be done within d each groupefined by the specified columns.
seed	(Optional) An integer to set the random seed for reproducibility.
pr_treated	(Optional) The probability of a row being assigned to the treatment group (TRUE). Default is 0.5.

Value

A new data frame with the added "treated" column.

---

balancePlot *Create a Baseline Balance Plot.*

Description

Create a Baseline Balance Plot.

Usage

```
balancePlot(data)
```

Arguments

data	tibble produced with checkBaseline.

Value

ggplot2 baseline balance plot.

Examples

```
library(imt)
set.seed(123)  # for reproducibility
N <- 1000
fake_data <- tibble::tibble(x1 = rnorm(N), x2 = rnorm(N), t = rbinom(N, 1, 0.5))
baseline <- checkBaseline(data = fake_data, variables = c("x1", "x2"), treatment = "t")
balancePlot(data = baseline)
```

---

blm *Bayesian Linear Model Factory*

Description

Bayesian Linear Model Factory

Bayesian Linear Model Factory

Active bindings

`version` im package version used to fit model

`eta_draws` Posterior draws for the treatment effect

`mcmChecks` MCMC diagnostics

`credible_interval` Credible interval for the treatment effect

Methods

Public methods:

- `blm$new()`
- `blm$ppcDensOverlay()`
- `blm$tracePlot()`
- `blm$posteriorProb()`
- `blm$pointEstimate()`
- `blm$credibleInterval()`
- `blm$priorProb()`
- `blm$priorInterval()`
- `blm$vizdraws()`
- `blm$clone()`

Method `new()`**:** Get the package version

Get the posterior draws for eta

Get the MCMC diagnostics

Get the credible interval

Create a new Bayesian Linear Model object.

Usage:

```
blm$new(
  data,
  y,
  x,
  treatment,
  eta_mean,
  eta_sd,
  generate_fake_data = 0,
  seed = 1982,
  ...
)
```

Arguments:

`data` Data frame to be used

`y` Name of the outcome variable in the data frame

`x` Vector of names of all covariates in the data frame

`treatment` Name of the treatment indicator variable in the data frame

`eta_mean` Prior mean for the treatment effect estimation

`eta_sd` Prior standard deviation for the treatment effect estimation

`generate_fake_data` Flag for generating fake data

`seed` Seed for Stan fitting

`...` Additional arguments for Stan

Returns: invisible

Method `ppcDensOverlay()`**:** This method compares the empirical distribution of the data 'y' to the distributions of simulated/replicated data 'yrep' from the posterior predictive distribution. This is done by creating a density overlay plot using the `bayesplot::ppc_dens_overlay` function.

Usage:

```
blm$ppcDensOverlay(n = 50)
```

Arguments:

`n` Number of posterior draws to use for the overlay

Returns: ggplot2 visualization

Method `tracePlot()`**:** Plot MCMC trace for the eta and sigma parameters.

Usage:

```
blm$tracePlot(...)
```

Arguments:

`...` Additional arguments for Stan

Returns: A ggplot object.

Method `posteriorProb()`**:** Calculate posterior probability of effect exceeding a threshold

This function calculates the posterior probability of the effect being larger or smaller than a specified threshold.

Usage:

```
blm$posteriorProb(threshold = 0, gt = TRUE)
```

Arguments:

`threshold` A numeric value specifying the threshold.

`gt` A logical value indicating whether to calculate the probability of the effect being greater than (TRUE) or less than (FALSE) the threshold.

Details: This function uses the private$..eta_draws internal variable to obtain draws from the posterior distribution of the effect size. Based on the specified arguments, the function calculates the proportion of draws exceeding/falling below the threshold and returns a formatted statement describing the estimated probability.
Calculate point estimate of the effect
This R6 method calculates the point estimate of the effect size based on the posterior draws of the eta parameter.

Returns: A character string summarizing the estimated probability.

Method `pointEstimate()`**:**

Usage:

```
blm$pointEstimate(median = TRUE)
```

Arguments:

`median` Logical value. If TRUE (default), the median of the eta draws is returned. If FALSE, the mean is returned.

Details: This method uses the private$..eta_draws internal variable which contains MCMC draws of the eta parameter representing the effect size. Based on the specified median argument, the method calculates and returns either the median or the mean of the draws. Calculates credible interval for the effect of the intervention
This R6 method calculates and returns a formatted statement summarizing the credible interval of a specified width for the effect of the intervention.

Returns: A numeric value representing the point estimate.

Method `credibleInterval()`**:**

Usage:

```
blm$credibleInterval(width = 0.75, round = 0)
```

Arguments:

`width` Numeric value between 0 and 1 representing the desired width of the credible interval (e.g., 0.95 for a 95% credible interval).

`round` Integer value indicating the number of decimal places to round the lower and upper bounds of the credible interval.

Details: This method uses the private$..eta_draws internal variable containing MCMC draws of the eta parameter representing the effect size. It calculates the credible interval, stores it internally, and returns a formatted statement summarizing the findings.
Calculate and format probability statement based on prior
This method calculates the probability that the effect is greater than or less than a given threshold based on the prior distribution of the effect. The results are formatted into a statement suitable for reporting.

Returns: A character string with the following information:

- The probability associated with the specified width
- The lower and upper bounds of the credible interval, rounded to the specified number of decimal places

Method `priorProb()`**:**

Usage:

```
blm$priorProb(threshold = 0, gt = TRUE)
```

Arguments:

`threshold` Numerical threshold for comparison.

`gt` Logical indicating whether to calculate probability greater than or less than the threshold.

Returns: A character string containing the formatted probability statement. Calculates the prior for the effedt of the intervention based on the hyperpriors.
This method computes and formats a statement about the probability interval of the effect based on the prior distribution.

Method `priorInterval()`**:**

Usage:

```
blm$priorInterval(width = 0.75, round = 0)
```

Arguments:

`width` Desired probability width of the interval (default: 0.75).

`round` Number of decimal places for rounding the bounds (default: 0).

Details: This method calculates the lower and upper bounds of the interval based on the specified probability width and the prior distribution of the effect. It then formats the results into a clear and informative statement.
Note that the probability is checked to be within valid range (0-1) with consideration of machine precision using .Machine$double.eps.

Returns: A character string containing the formatted probability interval statement.

Method `vizdraws()`**:** Plots impact's prior and posterior distributions.
For more details see `vizdraws::vizdraws()`.

Usage:

```
blm$vizdraws(...)
```

Arguments:

`...` other arguments passed to vizdraws.

Returns: An interactive plot of the prior and posterior distributions.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
blm$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`calcProb` *Calculate Probability of Posterior Draws Falling Within a Range*

Description

This function estimates the probability that a vector of posterior draws, represented by the parameter `eta`, falls within a specified range. It provides flexibility to use either prior distributions or posterior draws, and to specify one-sided or two-sided probability calculations.

Usage

```
calcProb(x, a = NULL, b = NULL, prior = FALSE, group_name = "group average")
```

Arguments

`x`	A numeric vector containing either posterior draws (default) or prior samples of the treatment effect parameter (`eta`).
`a`	(Optional) The lower bound of the range (as a proportion, not percentage).
`b`	(Optional) The upper bound of the range (as a proportion, not percentage).
`prior`	A logical value indicating whether to use prior samples (`TRUE`) or posterior draws (`FALSE`, default) for calculation.
`group_name`	A string describing the group for which the probability is being calculated (default: "group average").

Details

This function checks the following cases:

- If both `a` and `b` are `NULL`, it returns an empty string.
- If `b` is less than or equal to `a`, it throws an error.

The calculated probability and range are presented in a human-readable string using the `glue` package for formatting.

Value

A formatted string stating the calculated probability and the specified range. The probability is the proportion of samples (either prior or posterior) that fall within the defined range.

calculateEffectSizes *Calculate Effect Sizes for Treatment vs. Control*

Description

This function calculates effect sizes for each variable in a data frame, comparing treatment and control groups. It handles continuous, binary, and categorical variables using appropriate effect size measures.

Usage

```
calculateEffectSizes(data, treatment_column, to_check = NULL)
```

Arguments

`data`	A data frame containing the variables and treatment/control indicator.
`treatment_column`	The name of the column (as a string) in the data frame that indicates whether an observation is in the treatment or control group. This column must be a factor with exactly two levels.
`to_check`	(optional) A character vector specifying the names of the variables for which effect sizes should be calculated. If NULL (default), all variables (except the treatment column) are processed.

Details

- For continuous variables, Hedges' g effect size is calculated. - For binary variables, Cox's Proportional Hazards Index (Cox's C) is calculated. - For categorical variables, the variable is converted into multiple indicator (dummy) variables, and the average Cox's C across these indicators is reported.

Value

A data frame with two columns: * `Variable`: The name of each variable in the original data frame. * `EffectSize`: The calculated effect size for each variable.

checkBaseline *Check Baseline Equivalency.*

Description

Check Baseline Equivalency.

Usage

```
checkBaseline(data, variables, treatment)
```

Arguments

data	dataframe with the pre-intervention variables, and the treatment indicator.
variables	vector of with the names of the pre-intervention variables.
treatment	name of the treatment indicator.

Value

tibble with the standardized difference of the pre-intervention variables. The tibble includes variables: the variable name, std_diff: the standardized difference for that variable as a number balance: the standardized difference for that variable as a factor variable. For more details about this methodology check https://ies.ed.gov/ncee/wwc/Docs/OnlineTraining/wwc_training_m3.pdf.

---

cleanData *Cleans and prepares data for analysis*

Description

This function performs a series of data cleaning and preprocessing steps to ensure the data is suitable for analysis. This includes:

- Missing data handling
- Variable type checks
- Collinearity and zero-variance feature removal

Usage

```
cleanData(data, y, treatment, x = NULL, binary = FALSE)
```

Arguments

data	A data.frame containing the data to be cleaned.
y	Name of the dependent variable (character).
treatment	Name of the treatment variable (character, should be logical).
x	Names of the covariates to include in the model (character vector, optional).
binary	Should the dependent variable be treated as binary? Default is FALSE

Value

A list containing the cleaned dataset and relevant metadata:

- N: The number of observations after cleaning.
- K The number of covariates after cleaning.
- X The cleaned covariate matrix.
- treat_vec: Treatment vector as integers (1 for TRUE, 0 for FALSE).
- Y: The dependent variable vector.

countMissing *Count missing values (NA) in a dataframe*

Description

This function takes a dataframe as input and returns a tibble summarizing the number of missing values (NA) in each column and the number of rows with at least one missing value.

Usage

```
countMissing(df)
```

Arguments

df	A dataframe to analyze.

Value

A tibble with the following columns:

- NA_<column_name>: Number of NAs in each original column
- missing_rows: Number of rows with at least one NA across all columns

coxsIndex *Cox's Proportional Hazards Index (Cox's C)*

Description

Calculates Cox's C, a standardized effect size measure for comparing hazard rates between two groups in survival analysis.

Usage

```
coxsIndex(p_t, p_c, n_t, n_c)
```

Arguments

p_t	Numeric value representing the proportion of events (e.g., failures, deaths) in the treatment group.
p_c	Numeric value representing the proportion of events in the control group.
n_t	Numeric value representing the sample size of the treatment group.
n_c	Numeric value representing the sample size of the control group.

Details

Cox's C is a useful effect size for survival analysis when hazard ratios are not constant over time. It's calculated based on the log odds ratio of events and includes a small sample size correction. The value 1.65 is used to approximate a conversion to a Cohen's d-like scale.

Value

The calculated Cox's C effect size.

References

Cox, D. R. (1972). Regression models and life-tables. Journal of the Royal Statistical Society: Series B (Methodological), 34(2), 187-202.

`createData` *Converts a dataframe into a named list to provide data to a Stan model*

Description

Converts a dataframe into a named list to provide data to a Stan model

Usage

```
createData(
  data,
  y,
  treatment,
  x = NULL,
  eta_mean = 0,
  eta_sd = 1,
  run_estimation = 1
)
```

Arguments

`data`	The data frame to be used
`y`	The name of the outcome variable in the data frame
`treatment`	The name of the treatment indicator variable in the data frame
`x`	A vector of names of all covariates to be used in the data frame
`eta_mean`	The prior mean to be used for estimating the treatment effect
`eta_sd`	The prior standard deviation to be used for estimating the treatment effect
`run_estimation`	Whether to run the estimation, or merely draw data from the priors

Value

Returns
`stan_data` a named list providing the data for the stan model

`credibleInterval` *Calculate credible interval from MCMC draws*

Description

This function calculates a credible interval of the specified width from a vector of MCMC draws.

Usage

```
credibleInterval(draws, width)
```

Arguments

`draws`	A numeric vector containing MCMC draws.
`width`	A numeric value between 0 and 1 specifying the desired width of the credible interval.

Details

The function calculates the lower and upper bounds of the credible interval using the quantile function based on the specified width.

Value

A named list containing three elements:

- width: The specified width of the credible interval.
- lower_bound: The lower bound of the credible interval.
- upper_bound: The upper bound of the credible interval.

Examples

```
# Generate example draws
draws <- rnorm(1000)

# Calculate 95% credible interval
credibleInterval(draws, width = 0.95)
```

fit *Fits Stan model.*

Description

Fits Stan model.

Usage

```
fit(stan_data, model = "blm", ...)
```

Arguments

stan_data	A named list providing the data for the Stan model.
model	A character string specifying the model type. Must be either "blm" (Bayesian linear model) or "bnb" (Bayesian negative binomial model). Defaults to "blm".
...	Additional options to be passed through to `rstan::sampling`.

Value

The complete StanFit object from the fitted Stan model.

getStanParameter *Extracts parameter from Stan model.*

Description

Extracts parameter from Stan model.

Usage

```
getStanParameter(fit, par)
```

Arguments

fit	A Stan fit model.
par	Name of the parameter you want to extract.

Value

A long tibble with draws for the pamete.

hedgesG *Hedges' g Effect Size with Pooled Standard Deviation*

Description

Calculates Hedges' g, a standardized effect size for comparing means. This version includes a small-sample correction factor (omega) and uses the pooled standard deviation.

Usage

```
hedgesG(n_t, n_c, y_t, y_c, s_t, s_c)
```

Arguments

n_t	Numeric value representing the sample size of the treatment group.
n_c	Numeric value representing the sample size of the control group.
y_t	Numeric value representing the mean of the treatment group.
y_c	Numeric value representing the mean of the control group.
s_t	Numeric value representing the standard deviation of the treatment group.
s_c	Numeric value representing the standard deviation of the control group.

Details

Hedges' g is a variation of Cohen's d that adjusts for small-sample bias. It is calculated as the difference in means divided by the pooled standard deviation, then multiplied by a correction factor.

Value

The calculated Hedges' g effect size.

References

Hedges, L. V. (1981). Distribution theory for Glass's estimator of effect size and related estimators. Journal of Educational Statistics, 6(2), 107-128.

`logit` *Bayesian Logit Model Factory*

Description

A class for creating and managing Bayesian Logit Models

Active bindings

`version` im package version used to fit model

`tau_draws` Posterior draws for the treatment effect

`mcmChecks` MCMC diagnostics

`credible_interval` Credible interval for the treatment effect

`prior_eta` Prior distribution for eta

`prior_tau` Prior distribution for tau

`prior_mean_y` Prior distribution for mean y

`eta_draws` Posterior draws for eta

`predict_list` List of predictions

Methods

Public methods:

- `logit$new()`
- `logit$tracePlot()`
- `logit$calcProb()`
- `logit$pointEstimate()`
- `logit$credibleInterval()`
- `logit$vizdraws()`
- `logit$lollipop()`
- `logit$plotPrior()`
- `logit$predict()`
- `logit$getPred()`
- `logit$predSummary()`
- `logit$predCompare()`
- `logit$clone()`

Method `new()`**:** Get the package version
Get the posterior draws for tau
Get the MCMC diagnostics
Get the credible interval
Get the prior for eta
Get the prior for tau

Get the prior for mean y

Get the posterior draws for eta

Get the list of predictions

Create a new Bayesian Logit Model object.

Usage:

```
logit$new(
  data,
  y,
  x,
  treatment,
  mean_alpha,
  sd_alpha,
  mean_beta,
  sd_beta,
  tau_mean,
  tau_sd,
  seed = 1982,
  fit = TRUE,
  ...
)
```

Arguments:

`data` Data frame to be used

`y` Name of the outcome variable in the data frame

`x` Vector of names of all covariates in the data frame

`treatment` Name of the treatment indicator variable in the data frame

`mean_alpha` Prior mean for alpha

`sd_alpha` Prior standard deviation for alpha

`mean_beta` Prior mean for beta

`sd_beta` Prior standard deviation for beta

`tau_mean` Prior mean for the treatment effect estimation

`tau_sd` Prior standard deviation for the treatment effect estimation

`seed` Seed for Stan fitting

`fit` Flag for fitting the data to the model or not

`...` Additional arguments for Stan

Returns: invisible

Method `tracePlot()`**:** Plot MCMC trace for the eta and sigma parameters.

Usage:

```
logit$tracePlot(...)
```

Arguments:

`...` Additional arguments for Stan

Returns: A ggplot object.

Method `calcProb()`**:** Calculates the posterior of an effect being greater than, less than, or within a range defined by thresholds.

Usage:

```
logit$calcProb(a = 0, b = NULL, prior = FALSE)
```

Arguments:

`a` Optional. Lower bound for the threshold.

`b` Optional. Upper bound for the threshold.

`prior` Logical. If TRUE, calculates probabilities based on the prior distribution. If FALSE (default), uses the posterior distribution.

`x` A numeric vector containing draws from the posterior

Returns: A character string summarizing the estimated probability
Calculate point estimate of the effect
This R6 method calculates the point estimate of the effect size based on the posterior draws of the eta parameter.

Method `pointEstimate()`**:**

Usage:

```
logit$pointEstimate(median = TRUE)
```

Arguments:

`median` Logical value. If TRUE (default), the median of the eta draws is returned. If FALSE, the mean is returned.

Details: This method uses the private$..eta_draws internal variable which contains MCMC draws of the eta parameter representing the effect size. Based on the specified median argument, the method calculates and returns either the median or the mean of the draws. Calculates credible interval for the effect of the intervention
This R6 method calculates and returns a formatted statement summarizing the credible interval of a specified width for the effect of the intervention.

Returns: A numeric value representing the point estimate.

Method `credibleInterval()`**:**

Usage:

```
logit$credibleInterval(width = 0.75, round = 0)
```

Arguments:

`width` Numeric value between 0 and 1 representing the desired width of the credible interval (e.g., 0.95 for a 95% credible interval).

`round` Integer value indicating the number of decimal places to round the lower and upper bounds of the credible interval.

Details: This method uses the private$..eta_draws internal variable containing MCMC draws of the eta parameter representing the effect size. It calculates the credible interval, stores it internally, and returns a formatted statement summarizing the findings.

Returns: A character string with the following information:

- The probability associated with the specified width
- The lower and upper bounds of the credible interval, rounded to the specified number of decimal places

Method `vizdraws()`**:** Plots impact's prior and posterior distributions.

Usage:

```
logit$vizdraws(tau = FALSE, ...)
```

Arguments:

`tau` Logical. If TRUE, plot tau instead of eta

`...` other arguments passed to vizdraws.

Returns: An interactive plot of the prior and posterior distributions.

Method `lollipop()`**:** Plots lollipop chart for the prior and posterior of the impact being greater or less than a threshold.
For more details see `vizdraws::lollipops()`.

Usage:

```
logit$lollipop(threshold = 0, ...)
```

Arguments:

`threshold` cutoff used to calculate the probability. Defaults to zero percent points

`...` other arguments passed to vizdraws.

Returns: A lollipop chart with the prior and posterior probability of the impact being above or below a threshold.

Method `plotPrior()`**:** Plots draws from the prior distribution of the outcome, tau, and impact in percentage points.

Usage:

```
logit$plotPrior()
```

Method `predict()`**:** Predict new data

Usage:

```
logit$predict(new_data, name = NULL, M = NULL, ...)
```

Arguments:

`new_data` Data frame to be predicted

`name` Group name of the prediction

`M` Number of posterior draws to sample from

`...` Additional arguments

Returns: invisible(self)

Method `getPred()`**:** Get posterior predictive draws

Usage:

```
logit$getPred(name = NULL, ...)
```

Arguments:

`name` Group name of the prediction

`...` Additional arguments (not used)

Returns: Matrix of posterior predictive draws

Method `predSummary()`**:** Get point estimate, credible interval and prob summary of predictive draws

Usage:

```
logit$predSummary(
  name = NULL,
  subgroup = NULL,
  median = TRUE,
  width = 0.75,
  round = 0,
  a = NULL,
  b = NULL,
  ...
)
```

Arguments:

`name` Optional. Group name of the prediction

`subgroup` Optional. A boolean vector to get summary on the conditional group average

`median` Optional. Logical value for using median or mean

`width` Optional. Numeric value for credible interval width

`round` Optional. Integer value for rounding

`a` Optional. Lower bound threshold

`b` Optional. Upper bound threshold

`...` Additional arguments

Returns: A character string with summary information

Method `predCompare()`**:** Compare the average of the posterior draws of two groups

Usage:

```
logit$predCompare(
  name1,
  name2,
  subgroup1 = NULL,
  subgroup2 = NULL,
  median = TRUE,
  width = 0.75,
  round = 0,
  a = NULL,
  b = NULL,
  ...
)
```

Arguments:

`name1` Group name of the first prediction to be compared

`name2` Group name of the second prediction to be compared

`subgroup1` Optional. A boolean vector for the first group

`subgroup2` Optional. A boolean vector for the second group

`median` Optional. Logical value for using median or mean

`width` Optional. Numeric value for credible interval width

`round` Optional. Integer value for rounding

a Optional. Lower bound threshold

b Optional. Upper bound threshold

... Additional arguments

Returns: A character string with comparison summary

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
logit$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`logitRng` *Calculate logit link and sample from binomial distribution*

Description

This function takes prior / posterior draws of the Bayesian logit model, applies theinverse logit (logistic) transformation to obtain probabilities, and then generates random samples from binomial distributions.

Usage

```
logitRng(alpha, tau, beta, treat, X, N)
```

Arguments

`alpha`	Numeric. A draw of alpha param from the logit model
`tau`	Numeric. A draw of tau param from the logit model
`beta`	Vector. A draw of beta params from the logit model
`treat`	A 0 / 1 vector of treatment indicator
`X`	Data to be predicted
`N`	Numeric. Size of the sample, should depend on size of the data.

Value

A vector of N samples of preditive draws

mcmcChecks *MCMC Checks*

Description

Checks convergence, mixing, effective sample size, and divergent transitions

Methods

`$new(fit, pars)` Runs diagnostics on the supplied `stanfit` object, restricted to parameters identified by the character vector `pars`.
Tests include:
Share of specified parameters with an Rhat less than 1.1. If any have an Rhat > 1.1, `everything_looks_fine` is set to `FALSE`.
Share of specified parameters with an n_eff at least 0.1% of the total number of posterior draws. If any have n_eff < 0.001 * N, `everything_looks_fine` is set to `FALSE`.
Share of specified parameters with an n_eff of at least 100. If any have n_eff < 100, `everything_looks_fine` is set to `FALSE`.
Number of divergent transitions during posterior sampling. If there are any whatsoever, `everything_looks_fine` is set to `FALSE`.
Share of posterior iterations where the sampler reached the maximum treedepth. If more than 25\ `everything_looks_fine` is set to `FALSE`.

Active bindings

`everything_looks_fine` logical indicating whether all MCMC tests passed.

`diagnostics` list of the outcome of each MCMC test

`warnings` list of the warning messages from failed MCMC tests

Methods

Public methods:

- `mcmcChecks$new()`
- `mcmcChecks$clone()`

Method `new()`**:** Initialize a new mcmcChecks object and run diagnostics

Usage:

```
mcmcChecks$new(fit, pars)
```

Arguments:

`fit` A stanfit object to check

`pars` A character vector of parameter names to check

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
mcmcChecks$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`metaAnalysis` *Create a Meta-Analysis Object Using Data From Previous Studies*

Description

Create a Meta-Analysis Object Using Data From Previous Studies

Create a Meta-Analysis Object Using Data From Previous Studies

Details

A meta analysis has raw data and draws from the lift's posterior distribution. This is represented by an R6 Class.

Active bindings

`PosteriorATE` Draws from the posterior distribution of the average treatment effect.

`checks` MCMC diagnostics

`CredibleInterval` Lower and upper bounds of the credible interval

`PointEstimate` Point estimate of the average treatment effect

`fitted` Stan fit object

Methods

Public methods:

- `metaAnalysis$new()`
- `metaAnalysis$PlotRawData()`
- `metaAnalysis$PlotLift()`
- `metaAnalysis$UpdateCI()`
- `metaAnalysis$probability()`
- `metaAnalysis$findings()`
- `metaAnalysis$clone()`

Method `new()`**:** Create a new meta analysis object.

Usage:

```
metaAnalysis$new(
  data,
  point_estimates,
  standard_errors,
  id,
  mean_mu = 0,
  sd_mu = 0.05,
```

```
  ci_width = 0.75,
  X = NULL,
  run_estimation = 1,
  ...
)
```

Arguments:

`data` Data frame with data point estimates and standard errors from studies.

`point_estimates` Name of the variable in the data frame that contains the point estimates.

`standard_errors` Name of the variable in the data frame that contains the standard errors of the point estimates.

`id` Name of the variable in the data frame that contains the id of the studies.

`mean_mu` Prior mean for the true lift in the population.

`sd_mu` Prior mean for the standard deviation of the true lift in the population.

`ci_width` Credible interval's width.

`X` Covariates matrix.

`run_estimation` Integer flag to control whether estimation is run (1) or not (0).

`...` other arguments passed to `rstan::sampling()`

Returns: A new `meta_analysis` object.

Method `PlotRawData()`**:** Plots the raw data.

Usage:

```
metaAnalysis$PlotRawData()
```

Returns: A plot with point estimates and 95% confidence intervals.

Method `PlotLift()`**:** Plots lift's prior and posterior distributions.
For more details see `vizdraws::vizdraws()`.

Usage:

```
metaAnalysis$PlotLift(...)
```

Arguments:

`...` other arguments passed to vizdraws.

Returns: An interactive plot of the prior and posterior distributions.

Method `UpdateCI()`**:** Update the width of the credible interval.

Usage:

```
metaAnalysis$UpdateCI(ci_width)
```

Arguments:

`ci_width` New width for the credible interval. This number in the (0,1) interval.

Method `probability()`**:** Calculates that probability that lift is between a and b.

Usage:

```
metaAnalysis$probability(a = -Inf, b = Inf, percent = TRUE)
```

Arguments:

a Lower bound. By default -Inf.

b Upper bound. By default Inf.

percent A logical that indicates that a and b should be converted to percentage.

Returns: A string with the probability.

Method `findings()`**:** Calculates the point estimate a credible interval for the meta analysis.

Usage:

```
metaAnalysis$findings(percent = TRUE)
```

Arguments:

percent A logical that indicates that the point estimate should be converted to percent.

Returns: A string with the findings

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
metaAnalysis$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`negativeBinomial` *Bayesian Negative Binomial Model Factory*

Description

Bayesian Negative Binomial Model Factory

Bayesian Negative Binomial Model Factory

Active bindings

`version` Package version used to fit the model

`mcmChecks` MCMC diagnostics

`credible_interval` Credible interval for the treatment effect

`tau_draws` Posterior draws for the treatment effect

Methods

Public methods:

- `negativeBinomial$new()`
- `negativeBinomial$tracePlot()`
- `negativeBinomial$posteriorProb()`
- `negativeBinomial$pointEstimate()`
- `negativeBinomial$credibleInterval()`
- `negativeBinomial$vizdraws()`

- negativeBinomial$clone()

Method `new()`**:** Create a new Bayesian Negative Binomial Model object.

Usage:

```
negativeBinomial$new(
  data,
  y,
  x,
  treatment,
  tau_mean,
  tau_sd,
  run_estimation = 1,
  seed = 1982,
  ...
)
```

Arguments:

`data` Data frame to be used

`y` Name of the outcome variable in the data frame

`x` Vector of names of all covariates in the data frame

`treatment` Name of the treatment indicator variable in the data frame

`tau_mean` Prior mean for the treatment effect estimation

`tau_sd` Prior standard deviation for the treatment effect estimation

`run_estimation` Integer flag to control whether estimation is run (1) or not (0)

`seed` Seed for Stan fitting

`...` Additional arguments for Stan

Returns: invisible

Method `tracePlot()`**:** Plot MCMC trace for the eta and sigma parameters.

Usage:

```
negativeBinomial$tracePlot(...)
```

Arguments:

`...` Additional arguments for Stan

Returns: A ggplot object.

Method `posteriorProb()`**:** Calculate posterior probability of effect exceeding a threshold

This function calculates the posterior probability of the effect being larger or smaller than a specified threshold.

Usage:

```
negativeBinomial$posteriorProb(threshold = 0, gt = TRUE)
```

Arguments:

`threshold` A numeric value specifying the threshold.

`gt` A logical value indicating whether to calculate the probability of the effect being greater than (TRUE) or less than (FALSE) the threshold.

Details: This function uses the private$..tau_draws internal variable to obtain draws from the posterior distribution of the effect size. Based on the specified arguments, the function calculates the proportion of draws exceeding/falling below the threshold and returns a formatted statement describing the estimated probability.
Calculate point estimate of the effect
This R6 method calculates the point estimate of the effect size based on the posterior draws of the tau parameter.

Returns: A character string summarizing the estimated probability.

Method `pointEstimate()`**:**

Usage:

```
negativeBinomial$pointEstimate(median = TRUE)
```

Arguments:

`median` Logical value. If TRUE (default), the median of the tau draws is returned. If FALSE, the mean is returned.

Details: This method uses the private$..tau_draws internal variable which contains MCMC draws of the tau parameter representing the effect size. Based on the specified median argument, the method calculates and returns either the median or the mean of the draws. Calculates credible interval for the effect of the intervention
This R6 method calculates and returns a formatted statement summarizing the credible interval of a specified width for the effect of the intervention.

Returns: A numeric value representing the point estimate.

Method `credibleInterval()`**:**

Usage:

```
negativeBinomial$credibleInterval(width = 0.75, round = 0)
```

Arguments:

`width` Numeric value between 0 and 1 representing the desired width of the credible interval (e.g., 0.95 for a 95% credible interval).

`round` Integer value indicating the number of decimal places to round the lower and upper bounds of the credible interval.

Details: This method uses the private$..tau_draws internal variable containing MCMC draws of the tau parameter representing the effect size. It calculates the credible interval, stores it internally, and returns a formatted statement summarizing the findings.

Returns: A character string with the following information:

- The probability associated with the specified width
- The lower and upper bounds of the credible interval, rounded to the specified number of decimal places

Method `vizdraws()`**:** Plots impact's prior and posterior distributions.
For more details see `vizdraws::vizdraws()`.

Usage:

```
negativeBinomial$vizdraws(...)
```

Arguments:

... other arguments passed to vizdraws.

Returns: An interactive plot of the prior and posterior distributions.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
negativeBinomial$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`pointEstimate` *Calculate Point Estimate (Median or Mean) as Percentage*

Description

This function computes a point estimate from a numeric vector, returning either the median or the mean as a percentage.

Usage

```
pointEstimate(x, median = TRUE)
```

Arguments

`x`	A numeric vector containing the data from which to calculate the point estimate.
`median`	A logical value indicating whether to use the median (default: `TRUE`) or the mean (`FALSE`) as the point estimate.

Details

This function provides a simple way to obtain either the median or mean of a numeric vector as a percentage. The choice between these two measures of central tendency can be controlled by the `median` argument.

Value

A numeric value representing the chosen point estimate (median or mean) of the input vector `x`, multiplied by 100 to express it as a percentage.

`randomize` *Randomly Assign Treatment While Controlling for Baseline Equivalency*

Description

This function repeatedly randomizes treatment assignment (using `.randomizer`) until baseline equivalency is achieved across specified variables, as measured by the `checkBaseline` function from the `im` package. It can optionally stratify the randomization by specified groups.

Usage

```
randomize(
  data,
  variables,
  standard = "Not Concerned",
  seed = NULL,
  max_attempts = 100,
  pr_treated = 0.5,
  group_by = NULL
)
```

Arguments

`data`	The input data frame containing pre-intervention variables.
`variables`	A vector of the names of the pre-intervention variables to check for baseline equivalency.
`standard`	The desired level of baseline equivalence. Must be one of "Not Concerned", "Concerned", or "Very Concerned". Default is "Not Concerned". ("Not Concerned", "Concerned", or "Very Concerned"). Must be one of "Not Concerned", "Concerned", or "Very Concerned".
`seed`	(Optional) An integer to set the random seed for reproducibility of the initial randomization attempt. Subsequent attempts will use new random seeds.
`max_attempts`	The maximum number of randomization attempts to make before stopping and returning an error.
`pr_treated`	(Optional) The probability of a row being assigned to the treatment group (TRUE). Default is 0.5.
`group_by`	(Optional) A character vector of column names to stratify the randomization. If provided, the randomization will be done within each group defined by the specified columns.

Value

A new data frame with the added "treated" column, if baseline equivalency is achieved within the specified number of attempts. Otherwise, an error is thrown.

randomizer *Randomization Class for Treatment Assignment*

Description

This class provides methods to randomly assign treatments to a dataset while ensuring baseline covariate balance. It can handle both simple and stratified randomization.

Active bindings

`version` The version of the `im` package used for randomization.

`data` The data frame with the assigned treatment.

`seed` The random seed used for reproducibility.

`balance_summary` A summary (or list of summaries) of the balance assessment after randomization.

`balance_plot` A plot (or list of plots) of the balance assessment after randomization.

Methods

Public methods:

- `randomizer$new()`
- `randomizer$clone()`

Method `new()`**:** Initialize a new Randomizer object.

Usage:

```
randomizer$new(
  data,
  variables,
  standard = "Not Concerned",
  seed = NULL,
  max_attempts = 100,
  group_by = NULL
)
```

Arguments:

`data` The input data frame.

`variables` A vector of covariate names to check for balance.

`standard` The desired level of baseline equivalence. Must be one of "Not Concerned", "Concerned", or "Very Concerned". Default is "Not Concerned". ("Not Concerned", "Concerned", or "Very Concerned").

`seed` (Optional) An integer to set the random seed.

`max_attempts` (Optional) Maximum number of randomization attempts.

`group_by` (Optional) A character vector of column names to stratify randomization.

Returns: A new `randomizer` object.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
randomizer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`validate_logical_vector`

Validate a Logical Subgroup Vector

Description

This function checks if a given vector is a logical vector (`TRUE`/`FALSE`) and whether its length matches the number of rows in a specified matrix. It is designed to validate subgroup vectors used for subsettin data.

Usage

```
validate_logical_vector(subgroup, N, name = NULL)
```

Arguments

`subgroup`	A logical vector representing the subgroup to be validated.
`N`	Length the subgroup should have.
`name`	(Optional) A string indicating the name of group.

Details

This function performs two key validations:

1. Checks if the `subgroup` vector is logical.
2. Checks if the length of the `subgroup` vector matches the N.

Value

The original `subgroup` vector if it passes all validation checks.

Index