Package ‘rbmiUtils’

May 23, 2025

Title Utility Functions to Support and Extend the 'rbmi' Package

Version 0.1.4

Description Provides utility functions that extend the capabilities of the reference-based multiple imputation package 'rbmi'. It supports clinical trial analysis workflows with functions for managing imputed datasets, applying analysis methods across imputations, and tidying results for reporting.

Maintainer Mark Baillie <`bailliem@gmail.com`>

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0), tidyr, readr, tibble, ggplot2, rstan

Config/testthat/edition 3

Language en-US

Imports assertthat, dplyr, purrr, rbmi (>= 1.4), beeca, rlang

VignetteBuilder knitr

Depends R (>= 4.1)

LazyData true

URL `https://github.com/openpharma/rbmiUtils`

BugReports `https://github.com/openpharma/rbmiUtils/issues`

NeedsCompilation no

Author Mark Baillie [aut, cre, cph] (ORCID: <`https://orcid.org/0000-0002-5618-0667`>),
Tobias Mütze [aut] (ORCID: <`https://orcid.org/0000-0002-4111-1941`>),
Jack Talboys [aut],
Lukas A. Widmer [ctb] (ORCID: <`https://orcid.org/0000-0003-1471-3493`>)

Repository CRAN

Date/Publication 2025-05-23 18:32:01 UTC

Contents

`ADEFF` *Example efficacy trial dataset*

Description

A simplified example of a simulated trial dataset, with missing data.

Usage

```
ADEFF
```

Format

`ADEFF` A data frame with 1,000 rows and 10 columns:

USUBJID Unique subject identifier

AVAL Primary outcome variable

TRT01P Planned treatment

STRATA Stratification at randomisation

REGION Stratification by region

REGIONC Stratification by region, numeric code

BASE Baseline value of primary outcome variable

CHG Change from baseline

AVISIT Visit number

PARAM Analysis parameter name

`ADMI` *Example multiple imputation trial dataset*

Description

A simplified example of a simulated trial ADMI dataset

Usage

```
ADMI
```

Format

`ADMI` A data frame with 100,000 rows and 12 columns:

USUBJID	Unique patient identifier
STRATA	Stratification at randomisation
REGION	Stratification by region
REGIONC	Stratification by region, numeric code
TRT	Planned treatment
BASE	Baseline value of primary outcome variable
CHG	Change from baseline
AVISIT	Visit number
IMPID	Imputation number identifier
CRIT1FLN	Responder criteria (binary)
CRIT1FL	Responder criteria (categorical)
CRIT	Responder criteria (definition)

`analyse_mi_data` *Apply Analysis Function to Multiple Imputed Datasets*

Description

This function applies an analysis function (e.g., ANCOVA) to imputed datasets and stores the results for later pooling. It is designed to work with multiple imputed datasets and apply a given analysis function to each imputation iteration.

Usage

```
analyse_mi_data(
  data = NULL,
  vars = NULL,
  method = NULL,
  fun = rbmi::ancova,
  delta = NULL,
  ...
)
```

Arguments

`data`	A data frame containing the imputed datasets. The data frame should include a variable (e.g., `IMPID`) that identifies distinct imputation iterations.
`vars`	A list specifying key variables used in the analysis (e.g., `subjid`, `visit`, `group`, `outcome`). Required.
`method`	A character string or object specifying the method used for analysis (e.g., Bayesian imputation). Defaults to `NULL`.
`fun`	A function that will be applied to each imputed dataset. Defaults to `rbmi::ancova`. Must be a valid analysis function.
`delta`	A `data.frame` used for delta adjustments, or `NULL` if no delta adjustments are needed. Defaults to `NULL`.
`...`	Additional arguments passed to the analysis function `fun`.

Details

The function loops through distinct imputation datasets (identified by `IMPID`), applies the provided analysis function `fun`, and stores the results for later pooling. If a `delta` dataset is provided, it will be merged with the imputed data to apply the specified delta adjustment before analysis.

Value

An object of class `analysis` containing the results from applying the analysis function to each imputed dataset.

Examples

```
# Example usage with an ANCOVA function
library(dplyr)
library(rbmi)
library(rbmiUtils)
set.seed(123)
data("ADMI")

# Convert key columns to factors
ADMI$TRT <- factor(ADMI$TRT, levels = c("Placebo", "Drug A"))
ADMI$USUBJID <- factor(ADMI$USUBJID)
ADMI$AVISIT <- factor(ADMI$AVISIT)
```

```
# Define key variables for ANCOVA analysis
 vars <- set_vars(
  subjid = "USUBJID",
  visit = "AVISIT",
  group = "TRT",
  outcome = "CHG",
  covariates = c("BASE", "STRATA", "REGION")  # Covariates for adjustment
 )

# Specify the imputation method (Bayesian) - need for pool step
 method <- rbmi::method_bayes(
 n_samples = 20,
 control = rbmi::control_bayes(
   warmup = 20,
   thin = 1
   )
 )

# Perform ANCOVA Analysis on Each Imputed Dataset
ana_obj_ancova <- analyse_mi_data(
  data = ADMI,
  vars = vars,
  method = method,
  fun = ancova,  # Apply ANCOVA
  delta = NULL   # No sensitivity analysis adjustment
)
```

`as_analysis2` *Construct an rbmi* `analysis` *object*

Description

This is a helper function to create an analysis object that stores the results from multiple imputation analyses. It validates the results and ensures proper class assignment.

This is a modification of the rbmi::as_analysis function.

Usage

```
as_analysis2(results, method, delta = NULL, fun = NULL, fun_name = NULL)
```

Arguments

`results`	A list containing the analysis results for each imputation.
`method`	The method object used for the imputation.
`delta`	Optional. A delta dataset used for adjustment.
`fun`	The analysis function that was used.
`fun_name`	The name of the analysis function (used for printing).

Value

An object of class `analysis` with the results and associated metadata.

`gcomp_binary` *Utility function for Generalized G-computation for Binary Outcomes*

Description

Wrapper function for targeting a marginal treatment effect using g-computation using the beeca package. Intended for binary endpoints.

Usage

```
gcomp_binary(
  data,
  outcome = "CRIT1FLN",
  treatment = "TRT",
  covariates = c("BASE", "STRATA", "REGION"),
  reference = "Placebo",
  contrast = "diff",
  method = "Ge",
  type = "HC0",
  ...
)
```

Arguments

`data`	A data.frame containing the analysis dataset.
`outcome`	Name of the binary outcome variable (as string).
`treatment`	Name of the treatment variable (as string).
`covariates`	Character vector of covariate names to adjust for.
`reference`	Reference level for the treatment variable (default: "Placebo").
`contrast`	Type of contrast to compute (default: "diff").
`method`	Marginal estimation method for variance (default: "Ge").
`type`	Variance estimator type (default: "HC0").
`...`	Additional arguments passed to `beeca::get_marginal_effect()`.

Value

A named list with treatment effect estimate, standard error, and degrees of freedom (if applicable).

Examples

```
# Load required packages
library(rbmiUtils)
library(beeca)      # for get_marginal_effect()
library(dplyr)
# Load example data
data("ADMI")
# Ensure correct factor levels
ADMI <- ADMI %>%
  mutate(
    TRT = factor(TRT, levels = c("Placebo", "Drug A")),
    STRATA = factor(STRATA),
    REGION = factor(REGION)
  )
# Apply g-computation for binary responder
result <- gcomp_binary(
  data = ADMI,
  outcome = "CRIT1FLN",
  treatment = "TRT",
  covariates = c("BASE", "STRATA", "REGION"),
  reference = "Placebo",
  contrast = "diff",
  method = "Ge",    # from beeca: GEE robust sandwich estimator
  type = "HC0"      # from beeca: heteroskedasticity-consistent SE
)

# Print results
print(result)
```

`gcomp_responder` *G-computation Analysis for a Single Visit*

Description

Performs logistic regression and estimates marginal effects for binary outcomes.

Usage

```
gcomp_responder(
  data,
  vars,
  reference_levels = NULL,
  var_method = "Ge",
  type = "HC0",
  contrast = "diff"
)
```

Arguments

`data`	A data.frame with one visit of data.
`vars`	A list containing `group`, `outcome`, `covariates`, and `visit`.
`reference_levels`	Optional vector specifying reference level(s) of the treatment factor.
`var_method`	Marginal variance estimation method (default: "Ge").
`type`	Type of robust variance estimator (default: "HC0").
`contrast`	Type of contrast to compute (default: "diff").

Value

A named list containing estimates and standard errors for treatment comparisons and within-arm means.

`gcomp_responder_multi` *G-computation for a Binary Outcome at Multiple Visits*

Description

Applies `gcomp_responder()` separately for each unique visit in the data.

Usage

```
gcomp_responder_multi(data, vars, reference_levels = NULL, ...)
```

Arguments

`data`	A data.frame containing multiple visits.
`vars`	A list specifying analysis variables.
`reference_levels`	Optional reference level for the treatment variable.
`...`	Additional arguments passed to `gcomp_responder()`.

Value

A named list of estimates for each visit and treatment group.

Examples

```
library(dplyr)
library(rbmi)
library(rbmiUtils)

data("ADMI")

ADMI <- ADMI |>
  mutate(
    TRT = factor(TRT, levels = c("Placebo", "Drug A")),
    STRATA = factor(STRATA),
    REGION = factor(REGION)
  )

# Note: method must match the original used for imputation
method <- method_bayes(
  n_samples = 100,
  control = control_bayes(warmup = 20, thin = 2)
)

vars_binary <- set_vars(
  subjid = "USUBJID",
  visit = "AVISIT",
  group = "TRT",
  outcome = "CRIT1FLN",
  covariates = c("BASE", "STRATA", "REGION")
)

ana_obj_prop <- analyse_mi_data(
  data = ADMI,
  vars = vars_binary,
  method = method,
  fun = gcomp_responder_multi,
  reference_levels = "Placebo",
  contrast = "diff",
  var_method = "Ge",
  type = "HC0"
)

pool(ana_obj_prop)
```

`get_imputed_data` *Get Imputed Data Sets as a data frame*

Description

This function takes an imputed dataset and a mapping variable to return a dataset with the original IDs mapped back and renamed appropriately.

Usage

```
get_imputed_data(impute_obj)
```

Arguments

`impute_obj`	The imputation object from which the imputed datasets are extracted.

Value

A data frame with the original subject IDs mapped and renamed.

Examples

```
library(dplyr)
library(rbmi)
library(rbmiUtils)

set.seed(1974)
# Load example dataset
data("ADEFF")

# Prepare data
ADEFF <- ADEFF |>
  mutate(
    TRT = factor(TRT01P, levels = c("Placebo", "Drug A")),
    USUBJID = factor(USUBJID),
    AVISIT = factor(AVISIT)
  )

# Define variables for imputation
vars <- set_vars(
  subjid = "USUBJID",
  visit = "AVISIT",
  group = "TRT",
  outcome = "CHG",
  covariates = c("BASE", "STRATA", "REGION")
)

# Define Bayesian imputation method
method <- method_bayes(
  n_samples = 100,
  control = control_bayes(warmup = 200, thin = 2)
)

# Generate draws and perform imputation
draws_obj <- draws(data = ADEFF, vars = vars, method = method)
impute_obj <- impute(draws_obj,
  references = c("Placebo" = "Placebo", "Drug A" = "Placebo"))

# Extract imputed data with original subject IDs
admi <- get_imputed_data(impute_obj)
head(admi)
```

`tidy_pool_obj` *Tidy and Annotate a Pooled Object for Publication*

Description

This function processes a pooled analysis object of class `pool` into a tidy tibble format. It adds contextual information, such as whether a parameter is a treatment comparison or a least squares mean, dynamically identifies visit names from the `parameter` column, and provides additional columns for parameter type, least squares mean type, and visit.

Usage

```
tidy_pool_obj(pool_obj)
```

Arguments

`pool_obj`	A pooled analysis object of class `pool`.

Details

The function rounds numeric columns to three decimal places for presentation. It dynamically processes the `parameter` column by separating it into components (e.g., type of estimate, reference vs. alternative arm, and visit), and provides informative descriptions in the output.

Value

A tibble containing the processed pooled analysis results. The tibble includes columns for the parameter, description, estimates, standard errors, confidence intervals, p-values, visit, parameter type, and least squares mean type.

Examples

```
# Example usage:
library(dplyr)
library(rbmi)

data("ADMI")
N_IMPUTATIONS <- 100
BURN_IN <- 200
BURN_BETWEEN <- 5

# Convert key columns to factors
ADMI$TRT <- factor(ADMI$TRT, levels = c("Placebo", "Drug A"))
ADMI$USUBJID <- factor(ADMI$USUBJID)
ADMI$AVISIT <- factor(ADMI$AVISIT)

# Define key variables for ANCOVA analysis
```

```
 vars <- set_vars(
  subjid = "USUBJID",
  visit = "AVISIT",
  group = "TRT",
  outcome = "CHG",
  covariates = c("BASE", "STRATA", "REGION")  # Covariates for adjustment
 )

# Specify the imputation method (Bayesian) - need for pool step
method <- rbmi::method_bayes(
  n_samples = N_IMPUTATIONS,
  control = rbmi::control_bayes(
    warmup = BURN_IN,
    thin = BURN_BETWEEN
    )
  )

# Perform ANCOVA Analysis on Each Imputed Dataset
ana_obj_ancova <- analyse_mi_data(
  data = ADMI,
  vars = vars,
  method = method,
  fun = ancova,  # Apply ANCOVA
  delta = NULL   # No sensitivity analysis adjustment
)

pool_obj_ancova <- pool(ana_obj_ancova)
tidy_df <- tidy_pool_obj(pool_obj_ancova)

# Print tidy data frames
print(tidy_df)
```

Index