Package 'WMAP'

June 9, 2026

Title Weighted Meta-Analysis with Pseudo-Populations

Version 1.3.1

Description Implementation of integrative weighting approaches for multiple observational studies and causal inferences. The package features three weighting approaches, each representing a special case of the unified weighting framework, introduced by Guha and Li (2024) <doi:10.1093/biomtc/ujae070>, which includes an extension of inverse probability weights for data integration settings.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

LazyData true

Imports pkgcond, ggplot2, zeallot, randomForest, ranger, forcats, utils, stats

Depends R (>= 3.5.0)

Suggests future (>= 1.33.0), future.apply (>= 1.11.0), parallelly (>= 1.37.0), knitr, rmarkdown

NeedsCompilation no

Author Subharup Guha [aut, cre],
Mengqi Xu [aut],
Chayce Reed [aut],
Kashish Priyam [aut],
Yi Li [aut]

Maintainer Subharup Guha <Subharup.Guha@dartmouth.edu>

Repository CRAN

Date/Publication 2026-06-09 14:50:02 UTC

VignetteBuilder knitr

Contents

`balancing.weights` *Compute balancing weights using FLEXOR, IC, or IGO methods*

Description

This function calculates balancing weights based on the specified pseudo-population method.

Usage

```
balancing.weights(
  S,
  Z,
  X,
  method,
  naturalGroupProp,
  num.random = 40,
  gammaMin = 0.001,
  gammaMax = (1 - 0.001),
  seed = NULL,
  verbose = TRUE,
  suppress_diagnostics = FALSE,
  nodesize = NULL
)
```

Arguments

`S`	Vector of factor levels representing the study memberships. Takes values in {1, ..., J}.
`Z`	Vector of factor levels representing the group memberships. Takes values in {1, ..., K}.
`X`	Covariate matrix of N rows and p columns.
`method`	Pseudo-population method, i.e., weighting method. Take values in `FLEXOR`, `IC`, or `IGO`.
`naturalGroupProp`	Relevant only for FLEXOR method: a fixed user-specified probability vector θ.
`num.random`	Relevant only for FLEXOR method: number of random starting points of γ in the two-step iterative procedure. Default is 40.
`gammaMin`	Relevant only for FLEXOR method: Lower bound for each γ_s in the two-step iterative procedure. Default is 0.001.

gammaMax	Relevant only for FLEXOR method: Upper bound for each γ_s in the two-step iterative procedure. Default is 0.999.
seed	Seed for random number generation. Default is NULL.
verbose	Logical; Relevant only for FLEXOR method: if TRUE (default), displays progress messages during computation to the console. Set to FALSE to suppress these messages.
suppress_diagnostics	Logical; if TRUE, suppress diagnostic warnings (default: FALSE). Used internally during bootstrapping to avoid console clutter.
nodesize	Integer nodesize for the propensity score random forests; if NULL, selected automatically. Used internally during bootstrapping to reuse the nodesize from the original data.

Value

An S3 object of class `balancing_weights` with the following components:

wt.v N empirically normalized sample weights.

percentESS Percentage sample effective sample size (ESS) for the pseudo-population.

method Weighting method used (`FLEXOR`, `IC`, or `IGO`).

selected_nodesize Selected `min.node.size` for the propensity forest.

diagnostics A list of propensity model diagnostics:

- `frac_truncated`: Fraction of subjects with own-cell propensity below the truncation floor prior to stabilization.
- `crossfit_accuracy`: Out-of-fold classification accuracy of the propensity model (NA during bootstrap).
- `max_weight`: Maximum weight prior to normalization.
- `prob_quantiles`: Quantiles of the subject-level propensity distribution.

Examples

```
data(demo)

# IC Method
weights_ic <- balancing.weights(S, Z, X, method = "IC", naturalGroupProp)
summary(weights_ic)

# IGO Method
weights_igo <- balancing.weights(S, Z, X, method = "IGO", naturalGroupProp)
summary(weights_igo)

# FLEXOR Method
weights_flexor <- balancing.weights(S, Z, X, method = "FLEXOR", naturalGroupProp, num.random = 40)
summary(weights_flexor)
```

`causal.estimate` *Estimate causal effects using FLEXOR, IC, or IGO methods*

Description

This function estimates causal effects based on the specified pseudo-population method. The FLEXOR method involves an iterative two-step procedure.

Usage

```
causal.estimate(
  S,
  Z,
  X,
  Y,
  B = 0,
  method,
  naturalGroupProp = NULL,
  num.random = 40,
  gammaMin = 0.001,
  gammaMax = (1 - 0.001),
  seed = NULL,
  verbose = TRUE,
  parallel = FALSE,
  n_cores = NULL,
  outcome_model = FALSE,
  crossfit_folds = NULL
)
```

Arguments

`S`	Vector of factor levels representing the study memberships. Takes values in {1, ..., J}.
`Z`	Vector of factor levels representing the group memberships. Takes values in {1, ..., K}.
`X`	Covariate matrix of N rows and p columns.
`Y`	Matrix of L outcomes, with dimensions $N \times L$.
`B`	Number of bootstrap samples for variance estimation. Default is 0 (point estimates only).
`method`	Pseudo-population method, i.e., weighting method. Take values in `FLEXOR`, `IC`, or `IGO`.
`naturalGroupProp`	Relevant only for FLEXOR method: a fixed user-specified probability vector θ.
`num.random`	Relevant only for FLEXOR method: number of random starting points of γ in the two-step iterative procedure. Default is 40.

gammaMin	Relevant only for FLEXOR method: Lower bound for each γ_s in the two-step iterative procedure. Default is 0.001.
gammaMax	Relevant only for FLEXOR method: Upper bound for each γ_s in the two-step iterative procedure. Default is 0.999.
seed	Seed for random number generation. Default is NULL.
verbose	Logical; if TRUE (default), displays progress messages during computation to the console. Set to FALSE to suppress these messages.
parallel	Logical; if TRUE, use parallel processing for bootstrap (default: FALSE)
n_cores	Number of cores to use for parallel bootstrap (default: availableCores() - 1). Only used if parallel = TRUE.
outcome_model	Logical; if TRUE, fits a random forest on X, S, Z and substitutes predicted outcomes for observed Y. Default FALSE.
crossfit_folds	Integer >= 2 for K-fold cross-fitting of the outcome model, or NULL (default) to fit on the full data. Only used when outcome_model = TRUE.

Value

An S3 object of class causal_estimates with the following components:

percentESS Percentage sample ESS of the pseudo-population.

moments.ar Array of dimension $3 \times K \times L$: estimated means, SDs, and medians (dim 1) for K groups (dim 2) and L outcomes (dim 3).

otherFeatures.v Estimated mean group differences for L outcomes.

collatedMoments.ar Array of dimension $3 \times K \times L \times B$: moments.ar for each of B bootstrap samples (only present when B > 0).

collatedOtherFeatures.mt Matrix of dimension $L \times B$: otherFeatures.v for each of B bootstrap samples (only present when B > 0).

collatedESS Vector of length B: percentage sample ESS for each bootstrap sample (only present when B > 0).

method Weighting method used (FLEXOR, IC, or IGO).

selected_nodesize Selected min.node.size for the propensity forest.

diagnostics A list of propensity model diagnostics:

- frac_truncated: Fraction of subjects with own-cell propensity below the truncation floor prior to stabilization.
- crossfit_accuracy: Out-of-fold classification accuracy of the propensity model.
- max_weight: Maximum weight prior to normalization.
- prob_quantiles: Quantiles of the subject-level propensity distribution.

Examples

```
data(demo)

# Point estimates only (no CIs)
result <- causal.estimate(S, Z, X, Y, B = 0, method = "IC", naturalGroupProp)
```

```
summary(result)

# Bootstrap CIs
set.seed(1)
result <- causal.estimate(S, Z, X, Y, B = 100, method = "IC", naturalGroupProp)
summary(result)
plot(result)

# FLEXOR method
result <- causal.estimate(S, Z, X, Y, B = 100, method = "FLEXOR", naturalGroupProp,
                          num.random = 40, seed = 1)

# Outcome model: substitute RF predictions for observed Y
result <- causal.estimate(S, Z, X, Y, B = 0, method = "FLEXOR", naturalGroupProp,
                          outcome_model = TRUE)

# Parallel bootstrap
result <- causal.estimate(S, Z, X, Y, B = 100, method = "FLEXOR", naturalGroupProp,
                          parallel = TRUE, n_cores = 2)
```

---

demo *Demo Dataset*

Description

A dataset containing example data for demonstration purposes.

Usage

```
data(demo)
```

Format

An rda object, with 450 observations and the following variables:

S A vector of factor levels, representing the study memberships.

Z A vector of factor levels, representing the group memberships.

X A covariate matrix.

Y An outcome matrix.

naturalGroupProp The relative group prevalences of the larger natural population. Necessary only for FLEXOR weights; it should be skipped for IC and IGO weights.

groupNames Disease subtype names "IDC" or "ILC"

Details

Demo Dataset for WMAP Examples

A demonstration dataset containing multi-site, multi-group observational data for testing WMAP weighting methods. The data simulates a study of breast cancer subtypes across mutliple hospitals.

See Also

balancing.weights for computing weights causal.estimate for estimating causal effects

Examples

```
# Load the demo dataset

data(demo)

# Examine the structure
str(S) # Study memberships
str(Z) # Group memberships
table(S, Z) # Cross-tabulation of studies and groups

# Basic usage with IC method
weights <- balancing.weights(S, Z, X, method = "IC", naturalGroupProp)
summary(weights)

# Full causal analysis
results <- causal.estimate(S, Z, X, Y, B = 10, method = "IC", naturalGroupProp)
summary(results)
plot(results)
```

---

plot.causal_estimates *Plot method for objects of class 'causal_estimates'*

Description

Creates a boxplot showing the distribution of effective sample size (ESS) across bootstrap samples for the specified weighting method.

Usage

```
## S3 method for class 'causal_estimates'
plot(x, ...)
```

Arguments

x	An object of class 'causal_estimates' returned by causal.estimate().
...	Additional arguments:
	y_limit The y-axis range. Default is c(0,50).
	color The boxplot color. Default is "red".

Value

A boxplot of percent sample ESS for a specific weighting method (FLEXOR, IC, or IGO

Examples

```
## Not run:
data(demo)
result <- causal.estimate(S, Z, X, Y, B = 5, method = "FLEXOR", naturalGroupProp)
plot(result)
plot(result, y_limit = c(0, 100), color = "steelblue")

## End(Not run)
```

`summary.balancing_weights` *Summary method for objects of class 'balancing_weights'*

Description

Displays a summary of balancing weights including the numnber of weights. their distribution, and the effective sample size.

Usage

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

Arguments

`object`	An object of class 'balancing_weights' returned by `balancing.weights()`.
`...`	Additional arguments (currently unused, reserved for future extensions).

Value

No return value, called for side effects (prints summary to console)

- The total number of weights
- Statistical summary of weight values
- Percentage sample effective sample size (ESS) for the pseudo-population

Examples

```
data(demo)
result <- balancing.weights(S, Z, X, method = "IC", naturalGroupProp)
summary(result)
```

summary.causal_estimates *Summary method for objects of class 'causal_estimates'*

Description

Displays a comprehensive summary of causal estimates including effective sample size, mean differences between groups with confidence intervals, and standard deviation ratios.

Usage

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

Arguments

object	An object of class 'causal_estimates' returned by causal.estimate()
...	Additional arguments (currently unused, reserved for future extensions).

Value

No return value, called for side effects (prints summary to console):

- Percentage sample effective sample size (ESS) for the pseudo-population
- The mean differences between two groups with their corresponding 95
- The ratios of standard deviations between two groups with their corresponding 95

Examples

```
data(demo)

set.seed(1)
result <- causal.estimate(S, Z, X, Y, B = 5, method = "IC", naturalGroupProp)
summary(result)
```

Index