Package 'maicChecks'

July 3, 2026

Type Package

Title Exact Matching and Matching-Adjusted Indirect Comparison (MAIC)

Version 0.3.0

Date 2026-07-02

Maintainer Lillian Yau <maicChecks@gmail.com>

Description The current version (0.3.0) streamlines the underlying code, adds a feasibility check to 'maicWt' and 'maxessWt', extends 'exmWt.2ipd' with options 'target.ipd' and 'method' for one-sided MAIC weighting between two IPDs, and adds 'wtTrtDiff' for the weighted treatment-effect difference with a Wald confidence interval based on Section 5 of Glimm & Yau (2026). The second version (0.2.0) contains implementation for exact matching which is an alternative to propensity score matching (see Glimm & Yau (2026) <doi:10.1080/19466315.2025.2507378>). The initial version (0.1.2) contains a collection of easy-to-implement tools for checking whether a MAIC can be conducted, as well as an alternative way of calculating weights (see Glimm & Yau (2022) <doi:10.1002/pst.2210>.)

Depends R (>= 3.5.0)

Imports data.table, tidyr, ggplot2, lpSolve, quadprog

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Lillian Yau [aut, cre],
Ekkehard Glimm [aut],
Xinlei Deng [aut] (ORCID: <https://orcid.org/0000-0001-8129-6007>)

Repository CRAN

Date/Publication 2026-07-03 21:50:02 UTC

Contents

`eAD` *three AD scenarios*

Description

Three artificial aggregate-data (AD) scenarios used in Glimm & Yau (2022). Columns `y1` and `y2` are the **matching covariate means** from the AD study. Columns `r.cont.mean`, `r.cont.sd`, `r.cont.n`, `r.bin.p`, and `r.bin.n` are **response summary** statistics added in v0.3.0 for use with `wtTrtDiff`; they are the same across the three scenarios.

Usage

```
data(eAD)
```

Format

`scen` scenario A, B, or C. Scenario A is very close to the IPD centre (see `eIPD`) and is inside the IPD convex hull; scenario B is further from the centre but still inside the hull; scenario C is outside the hull.

`y1` numeric, matching-covariate-1 mean.

`y2` numeric, matching-covariate-2 mean.

`r.cont.mean` numeric, AD mean of the continuous response.

`r.cont.sd` numeric, AD standard deviation of the continuous response.

`r.cont.n` integer, AD sample size for the continuous response.

`r.bin.p` numeric, AD event proportion for the binary response.

`r.bin.n` integer, AD sample size for the binary response.

References

Glimm E and Yau L. (2022). 'Geometric approaches to assessing the numerical feasibility for conducting matching-adjusted indirect comparisons.' *Pharmaceutical Statistics*, 21(5):974-987. doi:10.1002/pst.2210.

Examples

```
data(eAD)
```

`eIPD` *an IPD set*

Description

An artificial individual-patient-data (IPD) set used in Glimm & Yau (2022). Columns `y1` and `y2` are the **matching covariates** (despite the `y` prefix they play the role of baseline covariates X in MAIC / exact matching). Columns `r.cont` and `r.bin` are simulated **response variables** added in v0.3.0 for use with `wtTrtDiff`.

Usage

```
data(eIPD)
```

Format

`y1` numeric, matching covariate 1.

`y2` numeric, matching covariate 2.

`r.cont` numeric, simulated *continuous* response (`rnorm` with mean 1.0 and SD 0.5).

`r.bin` integer 0/1, simulated *binary* response (`rbinom` with prob 0.6).

References

Glimm E and Yau L. (2022). 'Geometric approaches to assessing the numerical feasibility for conducting matching-adjusted indirect comparisons.' *Pharmaceutical Statistics*, 21(5):974-987. doi:10.1002/pst.2210.

Examples

```
data(eIPD)
```

`exmLP.2ipd` *Checks whether two IPD datasets can be matched with lpSolve::lp*

Description

Checks whether two IPD datasets can be matched with lpSolve::lp

Usage

```
exmLP.2ipd(
  ipd1,
  ipd2,
  vars_to_match = NULL,
  cat_vars_to_01 = NULL,
  mean.constrained = FALSE
)
```

Arguments

`ipd1`	a dataframe with n1 row and p column, where n1 is number of subjects of the first IPD, and p is the number of variables used in standardization.
`ipd2`	a dataframe with n2 row and p column, where n2 is number of subjects of the second IPD, and p is the number of variables used in standardization.
`vars_to_match`	variables used for matching. if NULL, use all variables.
`cat_vars_to_01`	variable names for the categorical variables that need to be converted to indicator variables.
`mean.constrained`	whether to restrict the weighted means to be within the ranges of observed means. Default is FALSE. When it is TRUE, there is a higher chance of not having a solution.

Details

If dummy variables are already created for the categorical variables in the data set, and are present in `ipd1` and `ipd2`, then `cat_vars_to_01` should be left as NULL.

Value

`lp.check`	0 = exact matching is feasible; 2 = otherwise (no solution; see `lp` status codes).

Author(s)

Lillian Yau

References

Glimm E and Yau L. (2026). 'Exact matching as an alternative to propensity score matching.' *Statistics in Biopharmaceutical Research*, 18(1):106-116. doi:10.1080/19466315.2025.2507378.

Examples

```
## Not run:
ipd1 <- sim110[sim110$study == 'IPD A', ]
ipd2 <- sim110[sim110$study == 'IPD B', ]
x <- exmLP.2ipd(ipd1, ipd2,
                vars_to_match    = paste0('X', 1:5),
                cat_vars_to_01   = paste0('X', 1:3),
                mean.constrained = FALSE)

## End(Not run)
```

`exmWt.2ipd` *Exact matching for two IPD's*

Description

Computes exact-matching weights for two individual-patient datasets (IPDs). Two modes are supported:

- **Default** (`target.ipd = NULL`) – the method described in Glimm & Yau (2026): both IPDs are reweighted simultaneously so that their weighted covariate means match each other and the joint effective sample size (ESS) is maximised. The optimisation is solved via `solve.QP`.
- **One-sided MAIC** (`target.ipd = 'ipd1'` **or** `'ipd2'`) – the chosen IPD plays the role of the aggregate reference: its column means (after dummy expansion) are used as the AD, and the other IPD is weighted with either `maicWt` (default) or `maxessWt`. The target IPD is returned with uniform weights of 1.

In all modes the function first calls `exmLP.2ipd` as a feasibility gate. If matching is infeasible, a `message()` is emitted and `NA` weights are returned together with `lp.check = 2`, instead of letting the QP / optimiser fail.

Usage

```
exmWt.2ipd(
  ipd1,
  ipd2,
  vars_to_match = NULL,
  cat_vars_to_01 = NULL,
  mean.constrained = FALSE,
  target.ipd = NULL,
  method = c("maicWt", "maxessWt")
)
```

Arguments

`ipd1`	a dataframe with n1 row and p column.
`ipd2`	a dataframe with n2 row and p column.

vars_to_match	variables used for matching. If NULL (default), all variables shared by ipd1 and ipd2 are used.
cat_vars_to_01	a vector of variable names for the categorical variables that need to be converted to indicator variables. If the dummies are already present in the data, leave as NULL.
mean.constrained	whether to restrict the weighted means to be within the ranges of observed means. Only used when target.ipd = NULL. Default is FALSE. When TRUE, the QP is more likely to have no solution.
target.ipd	one of NULL (default), 'ipd1', or 'ipd2'. When NULL, the default method (Glimm & Yau, 2026) is used. When 'ipd1', the means of ipd1 are used as the AD and ipd2 is weighted via method; vice versa for 'ipd2'.
method	one of 'maicWt' (default) or 'maxessWt'. Ignored when target.ipd = NULL.

Details

If dummy variables are already created for the categorical variables in the data set, and are present in ipd1 and ipd2, then cat_vars_to_01 should be left as NULL.

Value

A list with the following slots:

ipd1	re-scaled exact-matching weights for IPD 1 (column exm.wts) together with the IPD 1 data, categorical variables converted to 0/1 indicators. NA if the matching is infeasible.
ipd2	same for IPD 2.
wtd.summ	a 1-row matrix with ESS for IPD 1, ESS for IPD 2, and the weighted means of the matching variables.
lp.check	the exmLP.2ipd (or maicLP) feasibility status: 0 = feasible, 2 = infeasible.
target.ipd	the value of target.ipd that was used.
method	the weighting method used: 'qp' for the default Glimm & Yau (2026) method, otherwise 'maicWt' or 'maxessWt'.

Author(s)

Lillian Yau

References

Glimm E and Yau L. (2026). 'Exact matching as an alternative to propensity score matching.' *Statistics in Biopharmaceutical Research*, 18(1):106-116. doi:10.1080/19466315.2025.2507378.

Examples

```
## Not run:
ipd1 <- sim110[sim110$study == 'IPD A', ]
ipd2 <- sim110[sim110$study == 'IPD B', ]

## (1) Default: both IPDs reweighted simultaneously (Glimm & Yau, 2026)
x0 <- exmWt.2ipd(ipd1, ipd2,
                 vars_to_match    = paste0('X', 1:5),
                 cat_vars_to_01   = paste0('X', 1:3),
                 mean.constrained = FALSE)

## (2) One-sided MAIC: target = IPD A, weight IPD B via maicWt
x1 <- exmWt.2ipd(ipd1, ipd2,
                 vars_to_match  = paste0('X', 1:5),
                 cat_vars_to_01 = paste0('X', 1:3),
                 target.ipd     = 'ipd1',
                 method         = 'maicWt')

## (3) One-sided MAIC: target = IPD B, weight IPD A via maxessWt
x2 <- exmWt.2ipd(ipd1, ipd2,
                 vars_to_match  = paste0('X', 1:5),
                 cat_vars_to_01 = paste0('X', 1:3),
                 target.ipd     = 'ipd2',
                 method         = 'maxessWt')

## End(Not run)
```

`maicLP` *Checks if AD is within the convex hull of IPD using lp-solve*

Description

Checks if AD is within the convex hull of IPD using lp-solve

Usage

```
maicLP(ipd, ad)
```

Arguments

`ipd`	a dataframe with n row and p column, where n is number of subjects and p is the number of variables used in matching.
`ad`	a dataframe with 1 row and p column. The matching variables should be in the same order as that in `ipd`. The function does not check this.

Value

`lp.check`	0 = AD is inside IPD, and MAIC can be conducted; 2 = otherwise

References

Glimm E and Yau L. (2022). 'Geometric approaches to assessing the numerical feasibility for conducting matching-adjusted indirect comparisons.' *Pharmaceutical Statistics*, 21(5):974-987. doi:10.1002/pst.2210.

Examples

```
## eIPD now contains response columns (r.cont, r.bin) in addition to the
## matching columns y1, y2. Subset to the matching columns explicitly.

## eAD[1,] is the scenario A in the reference paper,
## i.e. when AD is within IPD convex hull
maicLP(eIPD[, c('y1', 'y2')], eAD[1, c('y1', 'y2')])

## eAD[3,] is the scenario C in the reference paper,
## i.e. when AD is outside IPD convex hull
maicLP(eIPD[, c('y1', 'y2')], eAD[3, c('y1', 'y2')])
```

---

maicMD *Checks if AD is within the convex hull of IPD using Mahalanobis distance*

Description

Should only be used when all matching variables are normally distributed

Usage

```
maicMD(ipd, ad, n.ad = Inf)
```

Arguments

ipd	a dataframe with n row and p column, where n is number of subjects and p is the number of variables used in matching.
ad	a dataframe with 1 row and p column. The matching variables should be in the same order as that in ipd. The function does not check this.
n.ad	default is Inf assuming ad is a fixed (known) quantity with infinit accuracy. In most MAIC applications ad is only the sample statistics and n.ad is known.

Details

When AD does not have the largest Mahalanobis distance, in the original scale AD can still be outside of the IPD convex hull. On the other hand, when AD does have the largest Mahalanobis distance, in the original scale, AD is for sure outside the IPD convex hull.

Value

Prints a message whether AD is furthest away from 0, i.e. IPD center in terms of Mahalanobis distance. Also returns ggplot object for plotting.

`md.dplot`	dot-plot of AD and IPD in Mahalanobis distance
`md.check`	0 = AD has the largest Mahalanobis distance to the IPD center; 2 = otherwise

References

Glimm E and Yau L. (2022). 'Geometric approaches to assessing the numerical feasibility for conducting matching-adjusted indirect comparisons.' *Pharmaceutical Statistics*, 21(5):974-987. doi:10.1002/pst.2210.

Examples

```
## Not run:
## eIPD contains response columns; subset to matching columns y1, y2.
## eAD[1,] is the scenario A in the reference paper,
## i.e. when AD is perfectly within IPD
md <- maicMD(eIPD[, c('y1', 'y2')], eAD[1, c('y1', 'y2')])
md ## a dot-plot of IPD Mahalanobis distances along with AD in the same metric.

## End(Not run)
```

`maicPCA` *Checks whether AD is outside IPD in PC coordinates*

Description

Checks whether AD is outside IPD in principal component (PC) coordinates

Usage

```
maicPCA(ipd, ad)
```

Arguments

`ipd`	a dataframe with n row and p column, where n is number of subjects in IPD set and p is the number of variables used in matching.
`ad`	a dataframe with 1 row and p column. The matching variables should be in the same order as that in `ipd`. The function does not check this.

Details

When AD is within the IPD PC ranges, AD can still be outside the IPD convex hull in the original scale. On the other hand, if AD is outside the IPD PC ranges, in the original scale AD is for sure outside the IPD convex hull.

Value

Prints a message whether AD is inside or outside IPD PC coordinates. Also returns a ggplot object to be plotted.

`pc.dplot`	dot-plot of AD and IPD both in IPD's PC coordinates
`pca.check`	0 = AD within the ranges of IPD's PC coordinates; 2 = otherwise

References

Glimm E and Yau L. (2022). 'Geometric approaches to assessing the numerical feasibility for conducting matching-adjusted indirect comparisons.' *Pharmaceutical Statistics*, 21(5):974-987. doi:10.1002/pst.2210.

Examples

```
## Not run:
## eIPD contains response columns; subset to matching columns y1, y2.
## eAD[1,] is the scenario A in the reference paper,
## i.e. when AD is perfectly within IPD
a1 <- maicPCA(eIPD[, c('y1', 'y2')], eAD[1, c('y1', 'y2')])
a1 ## the dot plots of PC's for IPD and AD

## eAD[3,] is the scenario C in the reference paper,
## i.e. when AD is outside IPD
a3 <- maicPCA(eIPD[, c('y1', 'y2')], eAD[3, c('y1', 'y2')])
a3 ## the dot plots of PC's for IPD and AD

## End(Not run)
```

`maicT2Test` *Hotelling's T-square test to check whether maic is needed*

Description

Hotelling's T-square test to check whether maic is needed

Usage

```
maicT2Test(ipd, ad, n.ad = Inf)
```

Arguments

`ipd`	a dataframe with n row and p column, where n is number of subjects and p is the number of variables used in matching.
`ad`	a dataframe with 1 row and p column. The matching variables should be in the same order as that in `ipd`. The function does not check this.
`n.ad`	default is Inf assuming `ad` is a fixed (known) quantity with infinit accuracy. In most MAIC applications `ad` is the sample statistics and `n.ad` is known.

Details

When n.ad is not Inf, the covariance matrix is adjusted by the factor n.ad/(n.ipd + n.ad)), where n.ipd is nrow(ipd), the sample size of ipd.

Value

T.sq.f	the value of the T^2 test statistic
p.val	the p-value corresponding to the test statistic. When the p-value is small, matching is necessary.

References

Glimm E and Yau L. (2022). 'Geometric approaches to assessing the numerical feasibility for conducting matching-adjusted indirect comparisons.' *Pharmaceutical Statistics*, 21(5):974-987. doi:10.1002/pst.2210.

Examples

```
## eIPD contains response columns; subset to matching columns y1, y2.
## eAD[1,] is the scenario A in the reference paper,
## i.e. when AD is perfectly within IPD
maicT2Test(eIPD[, c('y1', 'y2')], eAD[1, c('y1', 'y2')])
```

---

maicWt *Estimates the MAIC weights*

Description

Estimates the MAIC weights for each individual in the IPD. The function first calls maicLP as a feasibility gate; if ad is not within the convex hull of ipd the function stops with an informative error rather than letting optim() fail silently.

Usage

```
maicWt(ipd, ad, max.it = 25)
```

Arguments

ipd	a dataframe with n row and p column, where n is number of subjects and p is the number of variables used in matching.
ad	a dataframe with 1 row and p coln. The matching variables should be in the same order as that in ipd. The function does not check this.
max.it	maximum iteration passed to optim(). if ad is within ipd convex hull, then the default 25 iterations of optim() should be enough.

Value

The main code are taken from Philippo (2016). It returns the following:

`optim.out`	results of optim()
`maic.wt`	MAIC un-scaled weights for each subject in the IPD set
`maic.wt.rs`	re-scaled weights which add up to the original total sample size, i.e. nrow(ipd)
`ipd.ess`	effective sample size
`ipd.wtsumm`	weighted summary statistics of the matching variables after matching. they should be identical to the input AD when AD is within the IPD convex hull.

References

Phillippo DM, Ades AE, Dias S, et al. (2016). Methods for population-adjusted indirect comparisons in submissions to NICE. NICE Decision Support Unit Technical Support Document 18.

Examples

```
## eIPD contains response columns (r.cont, r.bin) in addition to the
## matching columns y1, y2. Subset to the matching columns explicitly.
## eAD[1,] is scenario A in the reference manuscript
m1 <- maicWt(eIPD[, c('y1', 'y2')], eAD[1, c('y1', 'y2')])
```

`maxessWt` *Maximum ESS Weights*

Description

Estimates an alternative set of weights which maximizes effective sample size (ESS) for a given set of variates used in the matching. The function first calls `maicLP` as a feasibility gate; if `ad` is not within the convex hull of `ipd` the function stops with an error message.

Usage

```
maxessWt(ipd, ad)
```

Arguments

`ipd`	a dataframe with n row and p column, where n is number of subjects and p is the number of variables used in matching.
`ad`	a dataframe with 1 row and p column. The matching variables should be in the same order as that in `ipd`. The function does not check this.

Details

The weights maximize the ESS subject to the set of baseline covariates used in the matching.

Value

maxess.wt	maximum ESS weights. Scaled to sum up to the total IPD sample size, i.e. nrow(ipd)
ipd.ess	effective sample size. It is no smaller than the ESS given by the MAIC weights.
ipd.wtsumm	weighted summary statistics of the matching variables after matching. they should be identical to the input AD when AD is within the IPD convex hull.

References

Glimm E and Yau L. (2022). 'Geometric approaches to assessing the numerical feasibility for conducting matching-adjusted indirect comparisons.' *Pharmaceutical Statistics*, 21(5):974-987. doi:10.1002/pst.2210.

Examples

```
## eIPD contains response columns (r.cont, r.bin) in addition to the
## matching columns y1, y2. Subset to the matching columns explicitly.
## eAD[1,] is scenario A in the reference manuscript
m0 <- maxessWt(eIPD[, c('y1', 'y2')], eAD[1, c('y1', 'y2')])
```

---

sim110 *Simulated data used for exact matching*

Description

`sim110` is one of the simulated data sets presented in the simulation study of Glimm & Yau (2026). The **matching covariates** are `X1` to `X15`. The **continuous response** `Y` is simulated to depend on 6 of the 15 covariates; the **binary response** `Y.bin` (added in v0.3.0) is the indicator `Y > median(Y)`.

Usage

```
data(sim110)
```

Format

`X1` **to** `X15` matching covariates.

`Y` numeric, continuous response.

`Y.bin` integer 0/1, binary response equal to `as.integer(Y > median(Y))`.

`study` factor with levels `'IPD A'` and `'IPD B'`.

References

Glimm E and Yau L. (2026). 'Exact matching as an alternative to propensity score matching.' *Statistics in Biopharmaceutical Research*, 18(1):106-116. doi:10.1080/19466315.2025.2507378.

Examples

```
data(sim110)
```

wtTrtDiff *Weighted treatment-effect difference with Wald confidence interval*

Description

Computes the weighted mean response in each of two arms (IPD vs IPD or IPD vs AD), their difference, and a Wald (normal-based) confidence interval for the difference, see Section 5 of Glimm & Yau (2026). Two modes are supported:

- **IPD vs IPD** – supply per-subject treatment-effect responses and weights for both arms (`ipd1.te`, `w1`, `ipd2.te`, `w2`). Weights are typically obtained from `exmWt.2ipd`.
- **IPD vs AD** – supply per-subject responses and weights for one arm (`ipd1.te`, `w1`) together with an aggregate summary for the other (`ad.mean`, optionally `ad.sd` and `ad.n`). IPD weights are typically obtained from `maicWt` or `maxessWt`.

The function is intended for **binary (0/1) or continuous** responses only. It is *not* suitable for time-to-event / survival endpoints. This is not checked by the function – the user is responsible for supplying an appropriate response type.

Usage

```
wtTrtDiff(
  ipd1.te,
  w1,
  ipd2.te = NULL,
  w2 = NULL,
  ad.mean = NULL,
  ad.sd = NULL,
  ad.n = NULL,
  conf.level = 0.95,
  var.method = c("paper", "weighted_ess", "sq_residual", "all")
)
```

Arguments

`ipd1.te`	numeric vector of per-subject treatment-effect responses for IPD 1 (0/1 or continuous). The suffix `.te` stands for *treatment effect*.
`w1`	numeric vector of weights for IPD 1 (e.g. `maic.wt.rs` from `maicWt`, `maxess.wt` from `maxessWt`, or `exm.wts` from `exmWt.2ipd`). Must have the same length as `ipd1.te`.
`ipd2.te`	optional numeric vector of per-subject treatment-effect responses for IPD 2.
`w2`	optional numeric vector of weights for IPD 2. Must have the same length as `ipd2.te`.
`ad.mean`	optional scalar, the aggregate-data (AD) mean response. Used only when `ipd2.te` and `w2` are not supplied.

`ad.sd`	optional scalar, the AD standard deviation of the response. If `NULL` the AD mean is treated as a constant.
`ad.n`	optional scalar, the AD sample size.
`conf.level`	confidence level for the Wald CI. Default `0.95`.
`var.method`	character string selecting the arm-level variance estimator: `'paper'` (default; the conservative Glimm & Yau (2026, Section 5) estimator), `'weighted_ess'`, `'sq_residual'`, or `'all'` to return all three side by side. Partial matching is allowed. See **Details**.

Details

The weighted point estimate in arm k is

$$\hat{\mu}_k = \frac{\sum_i w_i y_i}{\sum_i w_i},$$

with effective sample size $\mathrm{ESS}_k = (\sum_i w_i)^2 / \sum_i w_i^2$. The arm-level variance $\widehat{\mathrm{var}}(\hat{\mu}_k)$ is selected by `var.method`. This offers three estimators:

`'paper'` **(default)** The estimator described in Glimm & Yau (2026, Section 5),

$$\widehat{\mathrm{var}}(\hat{\mu}_k) = \frac{\sum_i w_i^2}{(\sum_i w_i)^2}\, s_k^2 = \frac{s_k^2}{\mathrm{ESS}_k}, \qquad s_k^2 = \frac{1}{n_k}\sum_i (y_i - \bar{y}_k)^2,$$

where s_k^2 uses the *unweighted* sample mean $\bar{y}_k$. Under the assumptions stated in Section 5 (no hidden confounders, within-study exchangeability, and $\mathrm{var}(Y) \geq \mathrm{var}(Y \mid X)$) this is a *conservative* estimator of the conditional variance, and is the recommended default.

`'weighted_ess'` As `'paper'`, but the sample variance is taken about the *weighted* mean $\hat{\mu}_k$ and is itself weighted,

$$\widehat{\mathrm{var}}(\hat{\mu}_k) = \frac{s_{w,k}^2}{\mathrm{ESS}_k}, \qquad s_{w,k}^2 = \frac{\sum_i w_i (y_i - \hat{\mu}_k)^2}{\sum_i w_i}.$$

This estimator arises from treating w_i as the weight in a pseudo-population where w_i identical repetitions of $(\mathbf{x}_i, y_i)$ had been observed. For non-integer weights, this is an analogy rather than a literal sampling interpretation.

`'sq_residual'` A linearization / sandwich-type estimator formed directly from the weighted residuals,

$$\widehat{\mathrm{var}}(\hat{\mu}_k) = \frac{\sum_i w_i^2 (y_i - \hat{\mu}_k)^2}{(\sum_i w_i)^2}.$$

This has the form of a sandwich or robust variance estimator for a weighted mean, as commonly used in survey sampling and generalized estimating equations. In the present setting, however, the weights are covariate-balancing weights rather than precision weights. Therefore the squared residuals may reflect both random outcome variation and systematic outcome differences associated with covariates. This estimator is included as a sensitivity option rather than as the recommended default.

For 'paper' and 'weighted_ess', the estimated outcome variance is divided by the ESS to obtain the variance estimate of the treatment effect estimate. For 'sq_residual', no further division by the ESS is made, because the weights enter the variance formula directly through both the numerator and the denominator.

Setting var.method = 'all' returns all three estimators side by side (see **Value**) for sensitivity comparison.

For the **IPD vs AD** mode, if both ad.sd and ad.n are supplied the AD variance is $\hat{\sigma}^2_{ad}/n_{ad}$; otherwise the AD mean is treated as a fixed constant ($\widehat{\mathrm{var}} = 0$). For binary AD endpoints the user may supply ad.sd = sqrt(p * (1 - p)).

Assuming the two arms are independent (e.g. they come from different studies),

$$\mathrm{SE}(\hat{\Delta}) = \sqrt{\widehat{\mathrm{var}}(\hat{\mu}_1) + \widehat{\mathrm{var}}(\hat{\mu}_2)},$$

and the $(1-\alpha)$ Wald confidence interval for $\hat{\Delta} = \hat{\mu}_1 - \hat{\mu}_2$ is $\hat{\Delta} \pm z_{1-\alpha/2} \cdot \mathrm{SE}(\hat{\Delta})$.

Value

When var.method is one of 'paper', 'weighted_ess' or 'sq_residual', a list with the following slots:

wt.y1	weighted mean response in arm 1.
wt.y2	weighted mean response in arm 2 (or ad.mean).
wt.diff	the difference wt.y1 - wt.y2.
se1	standard error of wt.y1.
se2	standard error of wt.y2 (0 when AD is treated as a constant).
se.diff	standard error of wt.diff.
ci.lower	lower Wald confidence limit for wt.diff.
ci.upper	upper Wald confidence limit for wt.diff.
conf.level	the confidence level used.
ess1	effective sample size for arm 1.
ess2	effective sample size for arm 2 (ad.n when supplied, NA otherwise).
var.method	the variance estimator used.
var1	arm-1 variance $\widehat{\mathrm{var}}(\hat{\mu}_1)$ under the chosen var.method.
var2	arm-2 variance $\widehat{\mathrm{var}}(\hat{\mu}_2)$ under the chosen var.method.

When var.method = 'all', a list instead containing:

summary	a data frame with one row per variance method ('paper', 'weighted_ess', 'sq_residual') and columns var.method, wt.y1, wt.y2, wt.diff, se1, se2, se.diff, ci.lower, ci.upper, conf.level, ess1, ess2.
arm1.var	named numeric vector of the three arm-1 variance estimates.
arm2.var	named numeric vector of the three arm-2 variance estimates.

Author(s)

Lillian Yau

References

Glimm E and Yau L. (2026). 'Exact matching as an alternative to propensity score matching.' *Statistics in Biopharmaceutical Research*, 18(1):106-116. doi:10.1080/19466315.2025.2507378.

Examples

```
## ------------------------------------------------------------------
## IPD vs AD: weight eIPD onto scenario A of eAD with maicWt, then
## compare the IPD continuous / binary responses (eIPD$r.cont,
## eIPD$r.bin) to the AD summaries stored in the same scenario row of
## eAD. Note y1, y2 in eIPD / eAD are the *matching covariates*, while
## r.cont, r.bin (and r.cont.mean/sd/n, r.bin.p/n in eAD) are the
## response data.
## ------------------------------------------------------------------
w.out <- maicWt(eIPD[, c('y1', 'y2')], eAD[1, c('y1', 'y2')])

## continuous response
wtTrtDiff(ipd1.te = eIPD$r.cont, w1 = w.out$maic.wt.rs,
          ad.mean = eAD$r.cont.mean[1],
          ad.sd   = eAD$r.cont.sd[1],
          ad.n    = eAD$r.cont.n[1])

## binary response, treating AD as a fixed constant
wtTrtDiff(ipd1.te = eIPD$r.bin, w1 = w.out$maic.wt.rs,
          ad.mean = eAD$r.bin.p[1])

## binary response, supplying ad.sd = sqrt(p * (1 - p))
wtTrtDiff(ipd1.te = eIPD$r.bin, w1 = w.out$maic.wt.rs,
          ad.mean = eAD$r.bin.p[1],
          ad.sd   = sqrt(eAD$r.bin.p[1] * (1 - eAD$r.bin.p[1])),
          ad.n    = eAD$r.bin.n[1])

## compare all three variance estimators side by side
wtTrtDiff(ipd1.te = eIPD$r.cont, w1 = w.out$maic.wt.rs,
          ad.mean = eAD$r.cont.mean[1],
          ad.sd   = eAD$r.cont.sd[1],
          ad.n    = eAD$r.cont.n[1],
          var.method = 'all')$summary

## Not run:
## ------------------------------------------------------------------
## IPD vs IPD: symmetric exact-matching weights on sim110 IPD A vs B,
## then compare the simulated continuous (Y) and binary (Y.bin)
## responses.
## ------------------------------------------------------------------
ipd1  <- sim110[sim110$study == 'IPD A', ]
ipd2  <- sim110[sim110$study == 'IPD B', ]
w.out <- exmWt.2ipd(ipd1, ipd2,
                    vars_to_match  = paste0('X', 1:5),
                    cat_vars_to_01 = paste0('X', 1:3))
## continuous response
wtTrtDiff(ipd1.te = ipd1$Y,     w1 = w.out$ipd1$exm.wts,
```

```
          ipd2.te = ipd2$Y,     w2 = w.out$ipd2$exm.wts)
## binary response
wtTrtDiff(ipd1.te = ipd1$Y.bin, w1 = w.out$ipd1$exm.wts,
          ipd2.te = ipd2$Y.bin, w2 = w.out$ipd2$exm.wts)

## End(Not run)
```

Index