Package 'triplesmatch'

July 22, 2025

Type Package

Title Match Triples Consisting of Two Controls and a Treated Unit or Vice Versa

Version 1.1.0

Description Attain excellent covariate balance by matching two treated units and one control unit or vice versa within strata. Using such triples, as opposed to also allowing pairs of treated and control units, allows easier interpretation of the two possible weights of observations and better insensitivity to unmeasured bias in the test statistic. Using triples instead of matching in a fixed 1:2 or 2:1 ratio allows for the match to be feasible in more situations. The 'rrelaxiv' package, which provides an alternative solver for the underlying network flow problems, carries an academic license and is not available on CRAN, but may be downloaded from 'GitHub' at <https://github.com/josherrickson/rrelaxiv/>. The 'Gurobi' commercial optimization software is required to use the two functions [infsentrip()] and [triplesIP()]. These functions are not essential to the main purpose of this package. A free academic license can be obtained at <https://www.gurobi.com/features/academic-named-user-license/>. The 'gurobi' R package can then be installed following the instructions at <https://www.gurobi.com/documentation/9.1/refman/ins_the_r_package.html>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Depends rcbalance, rlemon, stats, graphics, MASS, optmatch, utils, rlang

Suggests rrelaxiv, testthat (>= 3.0.0), gurobi, sensitivityfull, informedSen, Matrix

Config/testthat/edition 3

NeedsCompilation no

Author Katherine Brumberg [aut, cre, cph]

Maintainer Katherine Brumberg <kbrum@umich.edu>

Repository CRAN

Date/Publication 2024-07-29 16:50:02 UTC

Contents

`aberrantscores` *Convert outcome to aberrant ranks*

Description

Replaces non-aberrant responses by 0 and ranks the aberrant responses by severity. The more aberrant responses have the highest ranks.

Usage

```
aberrantscores(ymat, cutoff, cutoff_dir = "less", tau = 0, treated1 = NULL)
```

Arguments

`ymat`	A matrix of outcomes. Rows correspond to matched triples and the three columns correspond to the three units in the match. The first unit is the one treated unit if ‘treated1 == TRUE‘ or the one control unit if ‘treated1 == FALSE‘. The other two columns contain the remaining two units in the match. These are control units if ‘treated1 == TRUE‘ or treated units if ‘treated1 == FALSE‘. This can easily be created from the triples match using the [formattrip()] function with ‘type == "wide"‘
`cutoff`	The cutoff for whether an outcome is aberrant. Any outcome more extreme then this cutoff will be considered aberrant
`cutoff_dir`	Either ‘less‘ or ‘greater‘, indicating whether outcomes should be aberrant if they are less than the ‘cutoff‘ or greater than the ‘cutoff‘, respectively

`tau` The null treatment effect to be subtracted from all treated units before aberrant ranking commences. If ‘tau != 0‘, then ‘treated1‘ is required

`treated1` A logical vector with length equal to the number of triples. ‘TRUE‘ if there is only one treated unit in the matched triple; ‘FALSE‘ if there are two treated units and only one control unit. This can easily be created from the triples match using the [formattrip()] function with ‘type == "wide"‘

Details

This can be useful for creating ‘scores‘ in [sentrip()] for an aberrant rank test.

Value

A matrix similar to ‘ymat‘ in all regards other than the outcomes being converted to aberrant ranks

See Also

aberrantscoreslong for the same function with inputs given in the long format as opposed to the wide format

formattrip for formatting the triples match into long or wide format

Examples

```
# Generate some data
set.seed(246)
n <- 30
x <- rnorm(n, 0, 3)
nt <- floor(n * 0.5)
nc <- n - nt
z <- c(rep(1, nt), rep(0, nc))
# Create a distance matrix, everything in one stratum
dist <- dist_mahal(data.frame(x = x), z, rep(1, n))[[1]]
# Create the triples match
triplesm <- triples_st(cost = dist, z = z, solver = "rlemon")
# Create an outcome
y <- 1:40
# Give the outcome some random unit names
names(y) <- sample(1:40)
# Reformat the triples match
ywide <- formattrip(m = triplesm, y = y, type = "wide")
# Turn the outcome into scores, in this case aberrant ranks
ab <- aberrantscores(ywide$ymat, 15, cutoff_dir = "less", tau = 0, treated1 = NULL)
# Conduct a one-sided hypothesis test with a bias of gamma = 1.25
sentrip(scores = ab, treated1 = ywide$treated1, gamma = 1.25, alternative = "greater")
```

`aberrantscoreslong` *Convert outcome to aberrant ranks*

Description

Replaces non-aberrant responses by 0 and ranks the aberrant responses by severity. The more aberrant responses have the highest ranks.

Usage

```
aberrantscoreslong(y, cutoff, cutoff_dir = "less", tau = 0, z = NULL)
```

Arguments

`y`	Vector of outcomes. Length is equal to the number of units
`cutoff`	The cutoff for whether an outcome is aberrant. Any outcome more extreme then this cutoff will be considered aberrant
`cutoff_dir`	Either ‘less‘ or ‘greater‘, indicating whether outcomes should be aberrant if they are less than the ‘cutoff‘ or greater than the ‘cutoff‘, respectively
`tau`	The null treatment effect to be subtracted from all treated units before aberrant ranking commences. If ‘tau != 0‘, then ‘z‘ is required
`z`	Vector with length equal to that of y. Each element specifies whether a unit is treated (1) or not (0)

Details

This function serves the same function as [aberrantscores()] but takes inputs in the ‘long‘ format instead of the ‘wide‘ format (see [formattrip()] for a description of the two formats, their uses, and their creation).

This can be useful for creating a column of ‘sc‘ in [infsentrip()] if the aberrant rank test is desired for that variable.

Value

Vector of aberrant ranks corresponding to ‘y‘

See Also

aberrantscores for the same function with inputs in the wide format instead of long

formattrip for formatting of the triples match into wide or long format

Examples

```
# Generate some data
set.seed(316)
n <- 30
x <- rnorm(n, 0, 1)
nt <- floor(n * 0.2)
nc <- n - nt
z <- c(rep(1, nt), rep(0, nc))
# Create a distance matrix (all units in one stratum here)
dist <- dist_mahal(data.frame(x = x), z, rep(1, n))[[1]]
# Conduct the triples match
triplesm <- triples_st(cost = dist, z = z, solver = "rlemon")
# Create primary and negative outcomes with some random unit names
y <- cbind(rnorm(40), runif(40))
rownames(y) <- sample(1:40)
# Reformat the triples match
ylong <- formattrip(m = triplesm, y = y, type = "long")
# Aberrant ranks for primary outcome
y[, 1] <- aberrantscoreslong(y[, 1], cutoff = 0.5, cutoff_dir = "greater")
```

---

boxplot_diffs *Boxplots of pairwise differences in triples match*

Description

Make boxplots of treated - control pair differences before matching, for the two types of triples, and weighted across triples

Usage

```
boxplot_diffs(m, y, z, yname = NULL)
```

Arguments

m	‘m‘ element of the list returned from ‘triples()‘ function containing information about matched individuals
y	Named vector containing variable to plot on the y axis. Names must correspond to the units specified in ‘m‘
z	Vector of treatment indicators. Must be in same order as ‘y‘
yname	y axis label

Value

Boxplots with treated minus control pair differences for the specified covariate. Boxplots are show for before matching, for the matches with 1 treated individual, for the matches with 2 treated individuals, and for the weighted combination that duplicates the differences for the matches with two treated individuals

Examples

```
# Generate some data
set.seed(8)
n <- 200
nt <- floor(n * 0.5)
nc <- n - nt
x <- c(rnorm(nt, 0, 1), rnorm(nc, 0.6, 1))
z <- c(rep(1, nt), rep(0, nc))
# Create some strata
ps <- glm(z ~ x, family = binomial)$fitted.values
ps_st <- cut(ps, c(0, quantile(ps, 1/3 * 1:2), 1), labels = 1:3)
# Create a distance matrix
dist <- dist_mahal(data.frame(x = x), z, ps_st)
# Construct the triples match
triplesm <- triples(cost = dist, z = z, st = ps_st, solver = "rlemon")
boxplot_diffs(m = triplesm$m, y = ps, z = z, yname = "Propensity score")
```

---

boxplot_matches *Series of boxplots for a given variable characterizing the triples match*

Description

Series of boxplots for a given variable characterizing the triples match

Usage

```
boxplot_matches(m, y, z, yname = NULL)
```

Arguments

m	'm' element of the list returned from 'triples()' function containing information about matched individuals
y	Named vector containing variable to plot on the y axis. Names must correspond to the units specified in 'm'
z	Vector of treatment indicators. Must be in same order as 'y'
yname	y axis label

Value

Display containing three sets of boxplots for the propensity score. First is for all treated vs control units. Second is for the triples that have one treated unit and two controls. Third is for the triples that have two treated units and one control.

Examples

```
# Generate some data
set.seed(8)
n <- 200
nt <- floor(n * 0.5)
nc <- n - nt
x <- c(rnorm(nt, 0, 1), rnorm(nc, 0.6, 1))
z <- c(rep(1, nt), rep(0, nc))
# Create some strata
ps <- glm(z ~ x, family = binomial)$fitted.values
ps_st <- cut(ps, c(0, quantile(ps, 1/3 * 1:2), 1), labels = 1:3)
# Create a distance matrix
dist <- dist_mahal(data.frame(x = x), z, ps_st)
# Construct the triples match
triplesm <- triples(cost = dist, z = z, st = ps_st, solver = "rlemon")
boxplot_matches(m = triplesm$m, y = ps, z = z, yname = "Propensity score")
```

---

cut_quant *Create strata based on quantiles of a score*

Description

Create strata based on quantiles of a score

Usage

```
cut_quant(v, q, int = TRUE)
```

Arguments

v	Vector of scores (typically propensity scores)
q	Vector of desired quantiles (between 0 and 1) at which to cut the strata
int	Boolean whether to return strata as integers. Default is 'TRUE'

Value

Vector of strata

Examples

```
cut_quant(1:9, c(1/3, 2/3), int = TRUE)
```

dist_mahal *Make Mahalanobis distance matrix*

Description

Make Mahalanobis distance matrix

Usage

```
dist_mahal(X, z, st, rank_based = FALSE)
```

Arguments

X	Covariate matrix to be used in calculating distances
z	Vector of treatment assignments
st	Vector of stratum assignments, should be denoted by consecutive integers starting from 1
rank_based	Whether to use rank based Mahalanobis distance or not

Value

List of squared Mahalanobis distance matrices between each pair of treated-control units in a stratum. There is one entry in the list for each stratum. This entry is a distance matrix with a row for each treated unit and a column for each control unit in the stratum.

Examples

```
# Generate some data
set.seed(1)
n <- 40
x <- rnorm(n, 0, 1)
nt <- floor(n * 0.4)
nc <- n - nt
z <- c(rep(1, nt), rep(0, nc))
# Create some strata
ps <- glm(z ~ x, family = binomial)$fitted.values
ps_st <- cut(ps, c(0, quantile(ps, 1/3 * 1:2), 1), labels = 1:3)
# Create a list of distance matrices, one for each stratum
dist <- dist_mahal(data.frame(x = x), z, ps_st)
```

formattrip *Formats the triples match for input to other functions*

Description

Formats the triples match for input to other functions

Usage

```
formattrip(m, y, type = "wide")
```

Arguments

m	The triples match, typically the output of [triples()]
y	A vector or matrix containing the outcome(s). If there are multiple outcomes, there should be a column for each. The rows (or vector elements) correspond to units. The vector should have names or the matrix should have rownames specifying the unit name. These unit names should correspond to those used in ‘m‘
type	Either ‘wide‘ or ‘long‘. ‘wide‘ formats the match for input to [sentrip()] whereas ‘long‘ formats the match for input to [infsentrip()]. ‘wide‘ can only be used if there is exactly one outcome in ‘y‘. ‘wide‘ creates a matrix ‘ymat‘ of outcomes with a row corresponding to each triple and three columns corresponding to the units in the triple. It also creates a logical vector ‘treated1‘ stating whether the first unit in the corresponding row of ‘ymat‘ is the one treated in the triple or not (in which case it would be the one control in the triple). ‘long‘ creates a list of three elements: ‘y‘, ‘z‘, and ‘mset‘. Each of these elements is a vector with one element corresponding to each unit in the triples match. ‘y‘ is the outcome, ‘z‘ is 1 if treated and 0 otherwise, and ‘mset‘ is the number of the triple to which this unit belongs

Value

A list containing either two elements ‘ymat‘ and ‘treated1‘ if ‘type == "wide"‘ or three elements ‘y‘, ‘z‘, and ‘mset‘ if ‘type == "long"‘

Examples

```
# Generate some data
set.seed(316)
n <- 30
x <- rnorm(n, 0, 1)
nt <- floor(n * 0.2)
nc <- n - nt
z <- c(rep(1, nt), rep(0, nc))
# Create a distance matrix (all units in one stratum here)
dist <- dist_mahal(data.frame(x = x), z, rep(1, n))[[1]]
```

```
# Conduct the triples match
triplesm <- triples_st(cost = dist, z = z, solver = "rlemon")
# Create primary and negative outcomes with some random unit names
y <- cbind(rnorm(40), runif(40))
rownames(y) <- sample(1:40)
# Reformat the triples match
ylong <- formattrip(m = triplesm, y = y, type = "long")
# Wide version only works with a single outcome
ywide <- formattrip(m = triplesm, y = y[, 1], type = "wide")
```

`infsentrip` *Sensitivity analysis for triples matches informed by tests for unmeasured bias*

Description

This function is very similar to [informedSen::informedsen()], with a few minor differences. This version allows for matches to contain either two treated units and one control unit or two controls and one treated unit ([informedSen::informedsen()] requires only one treated unit and a fixed number of controls in each match). This version also allows the primary hypothesis test to be one-sided. To use this function, the optimization software 'gurobi' and its R package must be installed.

Usage

```
infsentrip(gamma, sc, z, mset, alpha, alternative = "both")
```

Arguments

`gamma`	The sensitivity parameter $\Gamma \geq 1$. Setting $\Gamma = 1$ performs a randomization test that assumes ignorable treatment assignment given the matched triples
`sc`	A matrix with N rows and at least two columns. The first column is the primary outcome and the remaining columns are unaffected outcomes used to test for bias
`z`	A vector of length N with 0s for control and 1s for treated units
`mset`	A vector of length N indicating the matched triple
`alpha`	A vector with length equal to the number of columns of 'sc'. Each coordinate contains the level of the test applied to the corresponding column of 'sc'. If 'alpha' is a scalar, it is repeated for each column
`alternative`	One of 'greater', 'less' or 'both'. 'greater' implies the alternative hypothesis is that the treatment has a positive effect on the scores of the primary outcome, 'less' implies the alternative hypothesis is that the treatment has a negative effect on the scores of the primary outcome, and 'both' conducts a two-sided hypothesis test. The negative outcomes will always be two-sided tests (since one does not expect an effect in either direction)

Value

result Text indicating whether or not the test for bias rejects all biases of magnitude Gamma or less. If yes, then the conclusion is that you must increase Gamma to continue. If no, then the test on the primary outcome is conducted inside the confidence set defined by a test for bias.

optimization.problem Reiterates the result above, where the word yes means the optimization problem is infeasible, and the word no means it is feasible. See the conclusion for a scientific interpretation of this aspect of the output.

conclusion Text indicating the result of the test for effect on the primary outcome.

deviates A vector of standardized deviates that might be compared with the standard Normal distribution. There is one deviate for each column of ‘sc‘. If ‘sc‘ has column names, then the column names label the deviates. The deviates are computed at the treatment assignment probabilities, theta, that solve the constrained optimization problem.

alphas A vector of significance levels used for the deviates, together with their total. The total is relevant if the Bonferroni inequality is used to ensure joint level of all the tests. The absolute deviates might be compared with qnorm(1-alphas/2) for a two-sided test.

See Also

formattrip for easily creating inputs to this function.

Examples

```
# Generate some data
set.seed(316)
n <- 30
x <- rnorm(n, 0, 1)
nt <- floor(n * 0.2)
nc <- n - nt
z <- c(rep(1, nt), rep(0, nc))
# Create a distance matrix (all units in one stratum here)
dist <- dist_mahal(data.frame(x = x), z, rep(1, n))[[1]]
# Conduct the triples match
triplesm <- triples_st(cost = dist, z = z, solver = "rlemon")
# Create primary and negative outcomes with some random unit names
y <- cbind(rnorm(40), runif(40))
rownames(y) <- sample(1:40)
# Reformat the triples match
ylong <- formattrip(m = triplesm, y = y, type = "long")
# Score the outcomes, in this case aberrant ranks for primary outcome and
#      ranks for unaffected outcome
y[, 1] <- aberrantscoreslong(y[, 1], cutoff = 0.5, cutoff_dir = "greater")
y[, 2] <- rank(y[, 2])
# Run the informed sensitivity analysis at gamma of 1.5
inf1 <- infsentrip(gamma = 1.5, sc = ylong$y, z = ylong$z, ylong$mset,
                   alpha = 0.05, alternative = "both")
```

`make_bal_tab` *Make covariate balance table*

Description

Make covariate balance table

Usage

```
make_bal_tab(X, z, m, cov_names)
```

Arguments

`X`	Covariate matrix
`z`	Vector of treatment indicators. Must be in same order as ‘y‘
`m`	‘m‘ element of the list returned from ‘triples()‘ function containing information about matched individuals
`cov_names`	Row names to use instead of the column names of X when returning the table

Value

Table displaying means of the treated and control groups before and after matching, as well as standardized differences before and after matching

Examples

```
# Generate some data
set.seed(8)
n <- 200
nt <- floor(n * 0.5)
nc <- n - nt
x <- c(rnorm(nt, 0, 1), rnorm(nc, 0.6, 1))
z <- c(rep(1, nt), rep(0, nc))
names(z) <- 1:length(z)
names(x) <- 1:length(x)
# Create some strata
ps <- glm(z ~ x, family = binomial)$fitted.values
ps_st <- cut(ps, c(0, quantile(ps, 1/3 * 1:2), 1), labels = 1:3)
# Create a distance matrix
dist <- dist_mahal(data.frame(x = x), z, ps_st)
# Construct the triples match
triplesm <- triples(cost = dist, z = z, st = ps_st, solver = "rlemon")
make_bal_tab(X = cbind(x, ps), z = z, m = triplesm$m, cov_names = c("x", "prop score"))
```

`sentrip` *Sensitivity analysis for a triples match in an observational study*

Description

This function parallels [sensitivityfull::senfm()] for full matches. However, this function does not force the scores used for the test to be Huber's M-statistic. Instead, scores should be calculated ahead of time and then entered here. Performs either a randomization test or the corresponding Rosenbaum sensitivity analysis.

Usage

```
sentrip(scores, treated1, gamma = 1, alternative = "greater")
```

Arguments

`scores`	A matrix of scores. Rows correspond to matched triples and the three columns correspond to the three units in the match. The first unit is the one treated unit if 'treated1 == TRUE' or the one control unit if 'treated1 == FALSE'. The other two columns contain the remaining two units in the match. These are control units if 'treated1 == TRUE' or treated units if 'treated1 == FALSE'. This can easily be created from the triples match using the [formattrip()] function with 'type == "wide"'
`treated1`	A logical vector with length equal to the number of triples. 'TRUE' if there is only one treated unit in the matched triple; 'FALSE' if there are two treated units and only one control unit. This can easily be created from the triples match using the [formattrip()] function with 'type == "wide"'
`gamma`	The sensitivity parameter $\Gamma \geq 1$. Setting $\Gamma = 1$ performs a randomization test that assumes ignorable treatment assignment given the matched triples
`alternative`	One of 'greater', 'less' or 'both'. 'greater' implies the alternative hypothesis is that the treatment has a positive effect on the scores, 'less' implies the alternative hypothesis is that the treatment has a negative effect on the scores, and 'both' conducts a two-sided hypothesis test

Value

Named list with 5 elements: 'pval' is the upper bound on the one or two-sided P-value depending on 'alternative', 'deviate' is the deviate that was compared to the Normal distribution to produce pval, 'statistic' is the value of the statistic that is the sum of scores among treated units, 'expectation' is the maximum expectation of this statistic for the given Γ, and 'variance' is the maximum variance of this statistic among treatment assignments that achieve the maximum expectation

See Also

sensitivityfull::senfm for more details, especially for the interpretation of the 'expectation' and 'variance' components of the output and relevant references.

formattrip for easily creating inputs to this function.

Examples

```
# Generate some data
set.seed(246)
n <- 30
x <- rnorm(n, 0, 3)
nt <- floor(n * 0.5)
nc <- n - nt
z <- c(rep(1, nt), rep(0, nc))
# Create a distance matrix, everything in one stratum
dist <- dist_mahal(data.frame(x = x), z, rep(1, n))[[1]]
# Create the triples match
triplesm <- triples_st(cost = dist, z = z, solver = "rlemon")
# Create an outcome
y <- 1:40
# Give the outcome some random unit names
names(y) <- sample(1:40)
# Reformat the triples match
ywide <- formattrip(m = triplesm, y = y, type = "wide")
# Turn the outcome into scores, in this case aberrant ranks
ab <- aberrantscores(ywide$ymat, 15, cutoff_dir = "less", tau = 0, treated1 = NULL)
# Conduct a one-sided hypothesis test with a bias of gamma = 1.25
sentrip(scores = ab, treated1 = ywide$treated1, gamma = 1.25, alternative = "greater")
```

`triples` *Create a triples match*

Description

Create a triples match

Usage

```
triples(cost, z, st, solver = "rrelaxiv")
```

Arguments

`cost`	List of matrices of distances between treated (rows) and control (columns) units within a stratum with one entry in the list per stratum
`z`	Vector of treatment assignment (0 for control, 1 for treated)
`st`	Vector of stratum assignments
`solver`	Solver to use for the network problem. Either 'rrelaxiv' or 'rlemon'. 'rrelaxiv' can be downloaded from "https://github.com/josherrickson/rrelaxiv/"

Value

Named list with three elements: ‘m‘ contains the triples match. This is in the form of a data.frame with number of rows equal to the number of triples and 8 columns specifying the match number, the names of the three units within the match, the costs of the two treated-control pairs within the match, the number of treated units, and the stratum. ‘obj‘ contains the total objective from the network optimization and ‘bound‘ contains a loose lower bound on the objective of the optimal match.

Examples

```
# Generate some data
set.seed(1)
n <- 40
x <- rnorm(n, 0, 1)
nt <- floor(n * 0.4)
nc <- n - nt
z <- c(rep(1, nt), rep(0, nc))
# Create some strata
ps <- glm(z ~ x, family = binomial)$fitted.values
ps_st <- cut(ps, c(0, quantile(ps, 1/3 * 1:2), 1), labels = 1:3)
# Create a distance matrix
dist <- dist_mahal(data.frame(x = x), z, ps_st)
# Construct the triples match
triplesm <- triples(cost = dist, z = z, st = ps_st, solver = "rlemon")
```

---

triplesIP *Integer program for finding optimal triples match*

Description

This finds the optimal triples match using a quadratic program. The 'gurobi' package should be installed if using this function. This function should not be used for large problems. Note that this solver may find a good solution even if not optimal; setting ‘time_limit‘ is recommended. For most problems, [triples()] should be used instead to find a good approximate solution very quickly.

Usage

```
triplesIP(z, cost, mt, mc, time_limit = Inf, threads = 1, verbose = 0)
```

Arguments

z	Treatment indicator vector. 0 for control, 1 for treated
cost	Matrix of costs. Rows correspond to treated units; columns to controls
mt	The number of treated units to be used
mc	The number of control units to be used
time_limit	The amount of time in seconds before the solver should abort

threads	The number of threads that should be allocated
verbose	Whether the output of the 'gurobi' solver should be printed. 0 if not, 1 if so

Value

A named list with two elements: ‘match‘ and ‘opt_info‘. ‘match‘ contains the triples match. Similarly to the [triples()] function, this is in the form of a data.frame with number of rows equal to the number of triples and 8 columns specifying the match number, the names of the three units within the match, the costs of the two treated-control pairs within the match, the number of treated units, and the stratum. ‘opt_info‘ contains technical output from the optimization solver.

See Also

triples for an approximate solution

Examples

```
# Generate some data
set.seed(1)
n <- 40
x <- rnorm(n, 0, 1)
nt <- floor(n * 0.4)
nc <- n - nt
z <- c(rep(1, nt), rep(0, nc))
# Create some strata
ps <- glm(z ~ x, family = binomial)$fitted.values
ps_st <- cut(ps, c(0, quantile(ps, 1/3 * 1:2), 1), labels = 1:3)
# Create a distance matrix
dist <- dist_mahal(data.frame(x = x), z, ps_st)
# Construct the triples match using integer program for stratum 1
mIP <- triplesIP(z = z[ps_st == 1], cost = dist[[1]],
                 mt = 5, mc = 7, time_limit = 30, threads = 1, verbose = 0)
```

---

triples_st *Create a triples match for a single stratum*

Description

Create a triples match for a single stratum

Usage

```
triples_st(cost, z, solver = "rrelaxiv")
```

Arguments

cost	Matrix of distances between treated (rows) and control (columns) units within the stratum
z	Vector of treatment assignments for units within the stratum (0 for control, 1 for treated)
solver	Solver to use for the network problem. Either 'rrelaxiv' or 'rlemon'. 'rrelaxiv' can be downloaded from "https://github.com/josherrickson/rrelaxiv/"

Value

Named list with three elements: 'm' contains the triples match. This is in the form of a data.frame with number of rows equal to the number of triples and 7 columns specifying the match number, the names of the three units within the match, the costs of the two treated-control pairs within the match, and the number of treated units. 'obj' contains the total objective from the network optimization and 'bound' contains a loose lower bound on the objective of the optimal match.

Examples

```
set.seed(10)
n <- 20
x <- rnorm(n, 0, 1)
nt <- floor(n * 0.62)
nc <- n - nt
z <- c(rep(1, nt), rep(0, nc))
dist <- dist_mahal(data.frame(x = x), z, rep(1, n))[[1]]
triples_st(cost = dist, z = z, solver = "rlemon")
```

Index