Package ‘match2C’

March 29, 2022

Type Package

Title Match One Sample using Two Criteria

Version 1.2.3

Description Multivariate matching in observational studies typically has two goals: 1. to construct treated and control groups that have similar distribution of observed covariates and 2. to produce matched pairs or sets that are homogeneous in a few priority variables. This packages implements a
network-flow-
based method built around a tripartite graph that can simultaneously achieve both goals.
The package also implements a template matching algorithm using a variant of the tripartite graph design. A brief description of the workflow and some examples are given in the vignette. A more elaborated
tutorial can be found at <https://www.researchgate.net/publication/359513837_Tutorial_for_R_Package_match2C>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports ggplot2, mvnfast, rcbalance, Rcpp, stats, utils

Suggests dplyr, knitr, mvtnorm, RItools, rmarkdown

LinkingTo Rcpp

VignetteBuilder knitr

NeedsCompilation yes

Author Bo Zhang [aut, cre]

Maintainer Bo Zhang <bozhan@wharton.upenn.edu>

Repository CRAN

Date/Publication 2022-03-28 23:00:02 UTC

R topics documented:

NewPackage-package *A short title line describing what the package does*

Description

A more detailed description of what the package does. A length of about one to five lines is recommended.

Details

This section should provide a more detailed overview of how to use the package, including the most important functions.

Author(s)

Your Name, email optional.

Maintainer: Your Name <your@email.com>

References

This optional section can contain literature or other references for background information.

See Also

Optional links to other man pages

Examples

```
    ## Not run:
       ## Optional simple examples of the most important functions
       ## These can be in \dontrun{} and \donttest{} blocks.

  ## End(Not run)
```

`check_balance` *Check balance after matching.*

Description

This function checks the overall balance after statistical matching and plots the distribution of the propensity score in the treated group, the control group, and the matched control group.

Usage

```
check_balance(Z, match_object, cov_list, plot_propens, propens)
```

Arguments

`Z`	A vector of treatment indicator.
`match_object`	An object returned by match_2C or match_2C_mat or match_2C_list.
`cov_list`	A vector of names of covariates as appeared in the original dataset.
`plot_propens`	Post-matching distribution of the estimated propensity scores in two groups is plotted if TRUE; FALSE by default.
`propens`	NULL by default. If plot_propens = TRUE, then a vector of estimated propensity scores satisfying length(propens) = length(Z) needs to be supplied.

Value

This function returns a data frame of the overall balance after statistical matching. We tabulate the mean of each covariate in the cov_list in the treated group and control groups after matching, and calculate their standardized differences. Standardized difference is defined as the mean difference divided by the pooled standard error before matching.

check_balance_template *Check balance after template matching.*

Description

This function checks the overall balance after template matching and returns a dataframe with 7 columns: (1) mean of all covariates in the treated group, (2) mean of all covariates in the control group, (3) standardized mean differences of (1) and (2), (4) mean of all covariates in the matched treated group, (5) mean of all covariates in the matched control group, (6) standardized mean differences of (4) and (5), (7) mean of covariates in the template

Usage

```
check_balance_template(dataset, template, template_match_object, cov_list)
```

Arguments

`dataset`	The original dataset.
`template`	A data frame of the template.
`template_match_object`	An object returned by template_match.
`cov_list`	A vector of names of covariates as appeared in the original dataset and the template.

Value

This function returns a data frame of the overall balance after template matching. We tabulate the mean and SMD of each covariate in the cov_list in the template, the matched treated group, and the matched control group.

construct_outcome *Construct an output for matching.*

Description

This function constructs the output given the relaxsolution to the associated network flow problem and the original dataset.

Usage

```
construct_outcome(res, dist_list_1, Z, dataset, controls = 1)
```

Arguments

`res`	A callrelax output.
`dist_list_1`	A possibly sparse representation of the first distance matrix.
`Z`	A vector of treatment status.
`dataset`	The original dataset.
`controls`	Number of controls matched to each treated.

Value

This function returns a list of three objects: 1) feasible: 0/1 depending on the feasibility of the matching problem; 2) data_with_matched_set_ind: a data frame that is the same as the original data frame, except that a column called "matched_set" and a column called "distance" are appended to it. "matched_set" column assigns 1,2,...,n_t to each matched set, and NA to those not matched to any treated. Variable "distance" records the distance (as specified in the left network) between each matched control and the treated, and assigns NA to all treated and controls that are left unmatched. If matching is not feasible, NULL will be returned; 3) matched_data_in_order:a dataframe organized in the order of matched sets and otherwise the same as data_with_matched_set_ind. Note that the matched_set column assigns 1,2,...,n_t for as indices for matched sets, and NA for those controls that are not paired. Null will be returned if the matching is unfeasible.

`construct_outcome_template` *Construct an output for template matching.*

Description

This function constructs the output for template matching given the relaxsolution to the network flow problem, number of edges in the template-to-treated network, a vector of treatment status, and the original dataset. This function is of little interest to users.

Usage

```
construct_outcome_template(res, num_edges_left, Z, dataset)
```

Arguments

`res`	A callrelax output.
`num_edges_left`	Number of edges in the template-to-treatment network.
`Z`	A vector of treatment status.
`dataset`	The original dataset.

Value

This function returns a list of three objects: 1) feasible: 0/1 depending on the feasibility of the matching problem; 2) match_treated: a data frame of the matched treated units; 3) match_control: a data frame of the matched control units.

create_list_from_mat *Create a list representation of a distance matrix.*

Description

This function creates a “list representation” of a treatment-by-control distance matrix.

Usage

```
create_list_from_mat(
  Z,
  dist_mat,
  p = NULL,
  caliper = NULL,
  k = NULL,
  penalty = Inf
)
```

Arguments

`Z`	A length (n = n_t + n_c) vector of treatment indicators.
`dist_mat`	A treatment-by-control (n_t-by-n_c) distance matrix.
`p`	A vector of length (n_t + n_c) on which caliper applies (e.g. propensity scores)
`caliper`	Size of the caliper.
`k`	Connect each treated to the nearest k controls
`penalty`	Penalty for violating the caliper. Set to Inf by default.

Details

This function creates a list representation of a treatment-by-control network. The list representation can be made sparse using a user-specified caliper. A list representation of a treatment-by-control distance matrix consists of the following arguments:

- start_n: a vector containing the node numbers of the start nodes of each arc in the network.
- end_n: a vector containing the node numbers of the end nodes of each arc in the network.
- d: a vector containing the integer cost of each arc in the network.

Node 1,2,...,n_t are n_t treatment nodes; n_t + 1, n_t + 2, ..., n_t + n_c are n_c control nodes. start_n, end_n, and d should have the same lengths, all of which equal to the number of edges.

There are two options for users to make a network sparse. Option caliper is a value applied to the vector p to avoid connecting treated to controls whose covariate or propensity score defined by p is outside p +/- caliper. Second, within a specified caliper, sometimes there are still too many controls connected to each treated, and we can further trim down this number up to k by restricting our attention to the k nearest (in p) to each treated.

By default a hard caliper is applied, i.e., option penalty is set to Inf by default. Users may make the caliper a soft one by setting penalty to a large yet finite number.

Value

This function returns a list that consists of three arguments: start_n, end_n, and d, as described above.

`create_list_from_scratch`

Create a sparse list representation of treatment-to-control distance matrix with a caliper.

Description

This function takes in a n-by-p matrix of observed covariates, a length-n vector of treatment indicator, a caliper, and construct a possibly sparse list representation of the distance matrix.

Usage

```
create_list_from_scratch(
  Z,
  X,
  exact = NULL,
  soft_exact = FALSE,
  p = NULL,
  caliper_low = NULL,
  caliper_high = NULL,
  k = NULL,
  alpha = 1,
  penalty = Inf,
  method = "maha",
  dist_func = NULL
)
```

Arguments

`Z`	A length-n vector of treatment indicator.
`X`	A n-by-p matrix of covariates.
`exact`	A vector of strings indicating which variables need to be exactly matched.
`soft_exact`	If set to TRUE, the exact constraint is enforced up to a large penalty.
`p`	A length-n vector on which a caliper applies, e.g. a vector of propensity score.
`caliper_low`	Size of caliper low.
`caliper_high`	Size of caliper high.
`k`	Connect each treated to the nearest k controls. See details section.
`alpha`	Tuning parameter.
`penalty`	Penalty for violating the caliper. Set to Inf by default.
`method`	Method used to compute treated-control distance
`dist_func`	A user-specified function that compute treate-control distance. See details section.

Details

Currently, there are 4 methods implemented in this function: 'maha' (Mahalanobis distance), robust maha' (robust Mahalanobis distance), '0/1' (distance = 0 if and only if covariates are the same), 'Hamming' (Hamming distance).

Users can also supply their own distance function by setting method = 'other' and using the argument "dist_func". "dist_func" is a user-supplied distance function in the following format: dist_func(controls, treated), where treated is a length-p vector of covaraites and controls is a n_c-by-p matrix of covariates. The output of function dist_func is a length-n_c vector of distance between each control and the treated.

There are two options for users to make a network sparse. Option caliper is a value applied to the vector p to avoid connecting treated to controls whose covariate or propensity score defined by p is outside p +/- caliper. Second, within a specified caliper, sometimes there are still too many controls connected to each treated, and we can further trim down this number up to k by restricting our attention to the k nearest (in p) to each treated.

By default a hard caliper is applied, i.e., option penalty is set to Inf by default. Users may make the caliper a soft one by setting penalty to a large yet finite number.

Value

This function returns a list of three objects: start_n, end_n, and d. See documentation of function "create_list_from_mat" for more details.

Examples

```
## Not run:
# We first prepare the input X, Z, propensity score

attach(dt_Rouse)
X = cbind(female,black,bytest,dadeduc,momeduc,fincome)
Z = IV
propensity = glm(IV~female+black+bytest+dadeduc+momeduc+fincome,
                family=binomial)$fitted.values
detach(dt_Rouse)

# Create distance lists with built-in options.

# Mahalanobis distance with propensity score caliper = 0.05
# and k = 100.

dist_list_pscore_maha = create_list_from_scratch(Z, X, p = propensity,
                               caliper_low = 0.05, k = 100, method = 'maha')


# More examples, including how to use a user-supplied
# distance function, can be found in the vignette.

## End(Not run)
```

`create_list_from_scratch_overall` *Create a sparse list representation of treated-to-control distance matrix with a fixed number caliper with L1-distance.*

Description

This function takes in a n-by-p matrix of observed covariates, a length-n vector of treatment indicator, a caliper, and construct a possibly sparse list representation of the distance matrix with Mahalanobis distance. Note that this function is of limited interest to most users.

Usage

```
create_list_from_scratch_overall(
  Z,
  X,
  exact = NULL,
  soft_exact = FALSE,
  p = NULL,
  caliper_low = NULL,
  caliper_high = NULL,
  k = NULL,
  penalty = Inf,
  dist_func = NULL
)
```

Arguments

`Z`	A length-n vector of treatment indicator.
`X`	A n-by-p matrix of covariates.
`exact`	A vector of strings indicating which variables are to be exactly matched.
`soft_exact`	If set to TRUE, the exact constraint is enforced up to a large penalty.
`p`	A length-n vector on which a caliper applies, e.g. a vector of propensity score.
`caliper_low`	Size of caliper_inf.
`caliper_high`	Size of caliper_sup.
`k`	Connect each treated to the nearest k controls
`penalty`	Penalty for violating the caliper. Set to Inf by default.
`dist_func`	A function used to calculate distance

Value

This function returns a list of three objects: start_n, end_n, and d. See documentation of function “create_list_from_mat” for more details.

`dt_Rouse` *Rouse (1995) dataset*

Description

Variables of the dataset is as follows:

educ86 Years of education since 1986.

twoyr Attending a two-year college immediately after high school.

female Gender: 1 if female and 0 otherwise.

black Race: 1 if African American and 0 otherwise.

hispanic Race: 1 if Hispanic and 0 otherwise.

bytest Test score.

fincome Family income.

fincmiss Missingness indicator for family income.

IV Instrumental variable: encouagement to attend a two-year college.

dadeduc Dad's education: College - 2; Some college - 1; Neither - 0.

momeduc Mom's education: College - 2; Some college - 1; Neither - 0.

Usage

```
data(dt_Rouse)
```

Format

A data frame with 3037 rows, 8 observed variables, 1 binary instrumental variable, 1 treatment, and 1 continuous response.

Source

ss

`force_control` *Force including certain controls in the final matched samples.*

Description

This function processes the given distance list by adding certain zero-cost edges so that the user-specified controls are forced into the final matched samples. This function is of little interest to most users.

Usage

```
force_control(dist_list, Z, include)
```

Arguments

`dist_list`	A distance_list object.
`Z`	A length-n vector of treatment indicator.
`include`	A binary vector indicating which controls must be included (length(include) = sum(1-Z).

Value

This function returns a distance list object with added edges.

---

match_2C — *Optimal Matching with Two Criteria.*

Description

This function performs an optimal statistical matching that sequentially balances the nominal levels (near-fine balance), the marginal distribution of the propensity score, and the total within-matched-pair Mahalanobis distance.

Usage

```
match_2C(
  Z,
  X,
  propensity,
  dataset,
  method = "maha",
  exact = NULL,
  caliper_left = 1,
  caliper_right = 1,
  k_left = NULL,
  k_right = NULL,
  fb_var = NULL,
  controls = 1,
  include = NULL
)
```

Arguments

`Z`	A length-n vector of treatment indicator.
`X`	A n-by-p matrix of covariates with column names.
`propensity`	A vector of estimated propensity score (length(propensity) = length(Z)).
`dataset`	Dataset to be matched.
`method`	Method used to compute treated-control distance on the left. The default is the Mahalanobis distance.

exact	A vector of strings indicating which variables need to be exactly matched.
caliper_left	Size of caliper on the left network.
caliper_right	Size of caliper on the right network.
k_left	Connect each treated to k_left controls closest in the propensity score in the left network.
k_right	Connect each treated to k_right controls closest in the propensity score in the right network.
fb_var	A vector giving names of variables in matrix X to be finely balanced.
controls	Number of controls matched to each treated. Default is 1.
include	A binary vector indicating which controls must be included (length(include) = sum(1-Z)).

Value

This function returns a list of three objects including the feasibility of the matching problem and the matched controls organized in different formats. See the documentation of the function construct_outcome or the vignette for more details.

Examples

```
# We first prepare the input X, Z, propensity score

attach(dt_Rouse)
X = cbind(female,black,bytest,dadeduc,momeduc,fincome)
Z = IV
propensity = glm(IV~female+black+bytest+dadeduc+momeduc+fincome,
                family=binomial)$fitted.values
detach(dt_Rouse)

matching_output_double_calipers = match_2C(Z = Z, X = X,
propensity = propensity,
caliper_left = 0.05, caliper_right = 0.05,
k_left = 100, k_right = 100,
dataset = dt_Rouse)

# Please refer to the vignette for many more examples.
```

---

match_2C_list *Perform a pair (or 1:k) matching with two user-specified list representations of distance matrices.*

Description

This function performs a pair-matching using (at most) two user-specified distance matrices in their (possibly sparse) list representations. For more details on “list representations” of a treatment-by-control distance matrix, see the documentation of the function “create_list_from_mat”.

Usage

```
match_2C_list(
  Z,
  dataset,
  dist_list_1,
  dist_list_2 = NULL,
  lambda = 1000,
  controls = 1,
  overflow = FALSE
)
```

Arguments

`Z`	A length-n vector of treatment indicator.
`dataset`	dataset to be matched.
`dist_list_1`	A (possibly sparse) list representation of treatment-by-control distance matrix.
`dist_list_2`	A second (possibly sparse) list representation of treatment-by-control distance matrix.
`lambda`	A penalty that does a trade-off between two parts of the network.
`controls`	Number of controls matched to each treated. Default is set to 1.
`overflow`	A logical value indicating if overflow protection is turned on.

Details

This function is designed for more experienced and sophisticated R users. Instead of providing possibly dense treatment-by-control distance matrices that take up a lot of memories, users may simply provide two lists that specifies information of edges: their starting points, ending points, capacity, and cost. For more information on list representations of a distance matrix, see the documentation of the function “create_list_from_mat” and “create_list_from_scratch”. Note that by setting dist_list_2 = NULL, the usual matching framework is restored.

Value

This function returns the same object as function match_2C_mat.

`match_2C_mat` *Perform a pair matching using two user-specified distance matrices.*

Description

This function performs a pair-matching using two user-specified distance matrices and two calipers. Typically one distance matrix is used to minimize matched-pair differences, and a second distance matrix is used to enforce constraints on marginal distributions of certain variables.

Usage

```
match_2C_mat(
  Z,
  dataset,
  dist_mat_1,
  dist_mat_2,
  lambda,
  controls = 1,
  p_1 = NULL,
  caliper_1 = NULL,
  k_1 = NULL,
  p_2 = NULL,
  caliper_2 = NULL,
  k_2 = NULL,
  penalty = Inf,
  overflow = FALSE
)
```

Arguments

`Z`	A length-n vector of treatment indicator.
`dataset`	The original dataset.
`dist_mat_1`	A user-specified treatment-by-control (n_t-by-n_c) distance matrix.
`dist_mat_2`	A second user-specified treatment-by-control (n_t-by-n_c) distance matrix.
`lambda`	A penalty that controls the trade-off between two parts of the network.
`controls`	Number of controls matched to each treated.
`p_1`	A length-n vector on which caliper_1 applies, e.g. a vector of propensity score.
`caliper_1`	Size of caliper_1.
`k_1`	Maximum number of controls each treated is connected to in the first network.
`p_2`	A length-n vector on which caliper_2 applies, e.g. a vector of propensity score.
`caliper_2`	Size of caliper_2.
`k_2`	Maximum number of controls each treated is connected to in the second network.
`penalty`	Penalty for violating the caliper. Set to Inf by default.
`overflow`	A logical value indicating if overflow protection is turned on.

Details

This function performs a pair matching via a two-part network. The first part is a network whose treatment-to-control distance matrix is supplied by dist_mat_1. The second part of the network is constructed using distance matrix specified by dist_mat_2. Often, the first part of the network is used to minimize total treated-to-control matched pair distances, and the second part is used to enforce certain marginal constraints.

The function constructs two list representations of distance matrices, possibly using the caliper. caliper_1 is applied to p_1 (caliper_2 applied to p_2) in order to construct sparse list representations. For instance, a caliper equal to 0.2 (caliper_1 = 0.2) applied to the propensity score (p_1).

lambda is a penalty, or a tuning parameter, that balances these two objectives. When lambda is very large, the network will first minimize the second part of network and then the first part.

Value

This function returns a list of three objects including the feasibility of the matching problem and the matched controls organized in different formats. See the documentation of the function construct_outcome or the tutorial for more details.

Examples

```
## Not run:
To run the following code, one needs to first install
and load the package optmatch.

# We first prepare the input X, Z, propensity score

#attach(dt_Rouse)
#X = cbind(female,black,bytest,dadeduc,momeduc,fincome)
#Z = IV
#propensity = glm(IV~female+black+bytest+dadeduc+momeduc+fincome,
#family=binomial)$fitted.values
#n_t = sum(Z)
#n_c = length(Z) - n_t
#dt_Rouse$propensity = propensity
#detach(dt_Rouse)

# Next, we use the match_on function in optmatch
to create two treated-by-control distance matrices.

#library(optmatch)
# dist_mat_1 = match_on(IV~female+black+bytest+dadeduc+momeduc+fincome,
# method = 'mahalanobis', data = dt_Rouse)

# dist_mat_2 = match_on(IV ~ female, method = 'euclidean', data = dt_Rouse)


# Feed two distance matrices to the function match_2C_mat without caliper
# and a large penalty lambda to enforce (near-)fine balance.

#matching_output = match_2C_mat(Z, dt_Rouse, dist_mat_1, dist_mat_2,
#                                 lambda = 10000, p_1 = NULL, p_2 = NULL)

# For more examples, please consult the RMarkdown tutorial.

## End(Not run)
```

revert_dist_list_cpp *Revert a treated-to-control distance list.*

Description

Revert a treated-to-control distance list.

Usage

```
revert_dist_list_cpp(n_t, n_c, startn, endn, d)
```

Arguments

n_t	Number of treated units
n_c	Number of control units
startn	Vector of starting nodes of edges
endn	Vector of ending nodes of edges
d	Vector of cost associated with edges

solve_network_flow *Solve a network flow problem.*

Description

This function solves network flow optimization problems by calling the RELAX-IV algorithm implemented in FORTRAN by Dimitri Bertsekas and Paul Tseng, and made available by Sam Pimentel in the package rcbalance.

Usage

```
solve_network_flow(net)
```

Arguments

net	A list of five vectors: startn, endn, ucap, cost, b.

Details

This function is of limited interest to users.

Value

If the problem is feasible, function returns a list with the following elements: crash: an integer, equal to zero if the algorithm ran correctly and equal to 1 if it crashed. feasible: an integer, equal to zero if the problem is not feasible. x: a vector equal in length to the number of arcs in argument problem net, giving in each coordinate the number of units of flow passing across the corresponding edge in the optimal network flow. If the problem is not feasible, it returns "Not feasible."

`stitch_two_nets` *Stitch two treated-to-control networks into one two-part networks.*

Description

This function takes as inputs two networks and one penalty lambda, and constructs one two-part network out of them.

Usage

```
stitch_two_nets(net1, net2, lambda, controls = 1, overflow = FALSE)
```

Arguments

`net1`	A list of five vectors: startn, endn, ucap, cost, b.
`net2`	A list of five vectors: startn, endn, ucap, cost, b.
`lambda`	A penalty.
`controls`	Number of controls matched to each treated.
`overflow`	A logical value indicating if overflow protection is turned on.

Details

This function is of limited interest to users. Once overflow is set to TRUE, each cotnrol in the first network will be directly connected to the sink at a large cost, so that the network flow problem is feasible as long as the first part is feasible.

Value

This function returns a list of five vectors: startn, endn, ucap, cost, b.

`stitch_two_nets_template` *Stitch a template-to-treated network and a treated-to-control network into one two-part network.*

Description

This function takes as inputs a template-to-treated network, one treated-to-control network, a tuning parameter lambda, and number of controls, and constructs one two-part network out of them.

Usage

```
stitch_two_nets_template(net1, net2, n_c, lambda, multiple = 1)
```

Arguments

`net1`	A list of five vectors: startn, endn, ucap, cost, b.
`net2`	A list of five vectors: startn, endn, ucap, cost, b.
`n_c`	Number of control units.
`lambda`	A penalty.
`multiple`	Number of treated units matched to each unit in the template

Details

This function is of limited interest to users. Parameter lambda is a weight given to the first part of the network, and a large lambda value emphasizes resemblance to the template. Parameter multiple could be taken as any integer number between 1 and floor(treated size / template size).

Value

This function returns a list of five vectors: startn, endn, ucap, cost, b.

`template_match` *Optimal Matching with Two Criteria.*

Description

This function takes as arguments a dataset to be matched and a template, and outputs matched pairs that are closely matched, well balanced, and mimicking the user-supplied template in covariates' distributions of the given template.

Usage

```
template_match(
  template,
  X,
  Z,
  dataset,
  multiple = 1,
  lambda = 1,
  caliper_gscore = 1,
  k_gscore = NULL,
  penalty_gscore = Inf,
  caliper_pscore = 1,
  k_pscore = NULL,
  penalty_pscore = Inf
)
```

Arguments

`template`	A dataframe of template units.
`X`	A n-by-p matrix of covariates with column names.
`Z`	A length-n vector of treatment indicator.
`dataset`	Dataset to be matched.
`multiple`	Number of treated units matched to each template unit. Default is 1.
`lambda`	A tuning parameter controlling the trade-off between internal and external validity. A large lambda favors resemblance to the template.
`caliper_gscore`	Size of generalizability caliper.
`k_gscore`	Connect each template unit to k_gscore treated units closest in the generalizability score.
`penalty_gscore`	Penalty for violating the generalizability caliper. Set to Inf by default.
`caliper_pscore`	Size of propensity score caliper.
`k_pscore`	Connect each treated to k_pscore control units closest in the propensity score.
`penalty_pscore`	Penalty for violating the propensity score caliper. Set to Inf by default.

Details

Please refer to the vignette for reproducible examples.

Value

This function returns a list of three objects: 1) feasible: 0/1 depending on the feasibility of the matching problem; 2) match_treated: a data frame of the matched treated units; 3) match_control: a data frame of the matched control units.

`treated_control_net` *Create a treate-to-control network to be solved via a network flow algorithm.*

Description

This function takes in a list representation of distance matrix and create a network structure to be solved.

Usage

```
treated_control_net(n_t, n_c, dist_list, controls = 1)
```

Arguments

`n_t`	Number of treated subjects.
`n_c`	Number of controls.
`dist_list`	A list representation of the distance matrix.
`controls`	Number of controls matched to each treated.

Details

dist_list is a list consisting of the following three elements: start_n: the starting nodes for all edges, end_n: the ending nodes for all edges, d: distance of all treated-control edges. Function create_dist_list in this package constructs such a list representation given a user-specified distance function.

Value

This function returns a list of five vectors: startn, endn, ucap, cost, b.

Index