Package 'rcbsubset'

May 9, 2026

Type Package

Title Optimal Subset Matching with Refined Covariate Balance

Version 1.1.7

Date 2022-3-23

Author Samuel D. Pimentel

Maintainer Samuel D. Pimentel <spi@berkeley.edu>

Description Tools for optimal subset matching of treated units and control units in observational studies, with support for refined covariate balance constraints, (including fine and near-fine balance as special cases). A close relative is the 'rcbalance' package. See Pimentel, et al.(2015) <doi:10.1080/01621459.2014.997879> and Pimentel and Kelz (2020) <doi:10.1080/01621459.2020.1720693>. 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/>.

Depends R (>= 3.2.0), MASS, plyr

Imports rcbalance, rlemon

License MIT + file LICENSE

Suggests testthat, rrelaxiv, optmatch

Additional_repositories https://errickson.net/rrelaxiv/

NeedsCompilation no

Repository CRAN

Date/Publication 2022-03-25 23:20:06 UTC

Contents

rcbsubset-package 2

dist2net 3
rcbsubset 5

Index **9**

`rcbsubset-package` *Optimal Subset Matching with Refined Covariate Balance*

Description

Tools for optimal subset matching of treated units and control units in observational studies, with support for refined covariate balance constraints, (including fine and near-fine balance as special cases). A close relative is the 'rcbalance' package. See Pimentel, et al.(2015) <doi:10.1080/01621459.2014.997879> and Pimentel and Kelz (2020) <doi:10.1080/01621459.2020.1720693>. 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/>.

Details

The DESCRIPTION file:

Package:	rcbsubset
Type:	Package
Title:	Optimal Subset Matching with Refined Covariate Balance
Version:	1.1.7
Date:	2022-3-23
Author:	Samuel D. Pimentel
Maintainer:	Samuel D. Pimentel <spi@berkeley.edu>
Description:	Tools for optimal subset matching of treated units and control units in observational studies, with su
Depends:	R (>= 3.2.0), MASS, plyr
Imports:	rcbalance, rlemon
License:	MIT + file LICENSE
Suggests:	testthat, rrelaxiv, optmatch
Additional_repositories:	https://errickson.net/rrelaxiv/

Index of help topics:

```
dist2net                Building and Manipulating Network Flow Problems
rcbsubset               Optimal Matching with Refined Covariate Balance
rcbsubset-package       Optimal Subset Matching with Refined Covariate
                        Balance
```

This package computes matches that are optimal under a set of refined covariate balance constraints. These constraints, provided by the user, are a set of nested categorical variables of decreasing imporance which must be marginally balanced as closely as possible in the resulting treated and matched control populations. In addition, treated units may be excluded in an optimal manner (using a penalty parameter) to improve the quality of the match. For more detail see the references.

The main function is rcbsubset, which takes a distance/sparsity matrix or matrix-like object containing information about matchability of the treated and control units and a list of fine balance variables and produces a match. The other functions are largely for internal use and should not be needed by the large majority of users. The syntax and code structure is very similar in the closely related antecedent package rcbalance, which provides more helper functions for constructing matches but does not support optimal subset matching.

By default the package uses the R package rlemon to solve the minimum-cost network flow optimization problems by which matches are computed. Alternatively, users may specify that the rrelaxiv package should be used instead. However, this package carries an academic license and is not available on CRAN so users must install it themselves.

Author(s)

Samuel D. Pimentel

Maintainer: Samuel D. Pimentel <spi@berkeley.edu>

References

Pimentel, S.D., Kelz, R.R., Silber, J.H., and Rosenbaum, P.R. (2015) Large, sparse optimal matching with refined covariate balance in an observational study of the health outcomes produced by new surgeons, JASA 110 (510), 515-527.

Pimentel, S.D., and Kelz, R.R. (2020). Optimal tradeoffs in matched designs comparing US-trained and internationally trained surgeons. JASA 115 (532), 1675-1688.

Rosenbaum, P.R. (2012) Optimal matching of an optimally chosen subset in observational studies, JCGS 21.1: 57-71.

---

dist2net *Building and Manipulating Network Flow Problems*

Description

These are internal rcbsubset methods not meant to be called directly by users. They are used to construct a network flow problem from the information about a matching problem that is passed to the rcbsubset method.

Usage

```
dist2net(dist.struct, k, exclude.treated = FALSE, exclude.penalty = NULL, ncontrol = NULL,
tol = 1e-3)

dist2net.matrix(dist.struct, k, exclude.treated = FALSE, exclude.penalty = NULL,
tol = 1e-3)

add.layer(net.layers, new.layer)

penalty.update(net.layers, newtheta, newp = NA)

penalize.near.exact(net.layers, near.exact)
```

Arguments

dist.struct	an object specifying the sparsity structure of the match. For the dist2net method it is a list of vectors, and for the dist2net.matrix method it is a matrix or InfinitySparseMatrix. See rcbalance documentation for more details.
k	a nonnegative integer. The number of control units to which each treated unit will be matched.
exclude.treated	if TRUE, then when there is no feasible match using all treated units, a minimal number of treated units may be dropped so that a match can be formed. Specifying this argument adds penalized edges to the network so that such a match can be computed. NOTE: this argument is incompatible with values of k greater than 1.
exclude.penalty	a parameter that gives the cost of excluding a treated unit. If left NULL it will be set to a very large value designed to ensure treated units are never excluded if they can be matched. Lower values may result in subsets of treated units being excluded.
ncontrol	the total number of controls in the matching problem. If left NULL the function will attempt to compute it by counting controls referenced in the distance object provided.
tol	edge cost tolerance. This is the smallest tolerated difference between matching costs. It is used in these functions only when exclude.penalty is NULL, to choose a default penalty that is small enough not to cause integer overflows.
net.layers	a layered network object of the type produced by the dist2net function.
new.layer	a vector equal in length to the number of treated and control units in the matching problem. Each coordinate contains the value of a new fine balance variable for the corresponding unit.
newtheta	optional argument giving a new value for the theta field of the net.layers object (see value section for description of this field).
newp	optional argument giving a new value for the p field of the net.layers object (see value section for description of this field).
near.exact	a vector equal in length to the number of treated and control units in the matching problem. Edges between units with different values of this variable will be penalized.

Details

dist2net and dist2net.matrix take the distance structure given to rcbalance encoding information about the matching problem and converts it into a network flow problem. add.layer adds network structure to handle an individual fine balance variable (it can be called iteratively to add many such variables). penalty.update is used to change the penalties for each layer (and the penalties for edges used to exclude treated units if they are present) and penalize.near.exact is used to add penalties to the treated-control edges to allow near-exact matching. See the references for a detailed description of how the matching problem is transformed into a network.

Value

A layered network object, formatted as a list with the following arguments (where narcs is the number of arcs and nnodes is the number of nodes in the network):

`startn`	a vector of length narc containing the node numbers of the start nodes of each arc in the network.
`endn`	a vector of length narc containing the node numbers of the end nodes of each arc in the network.
`ucap`	a vector of length narc containing the (integer) upper capacity of each arc in the network.
`cost`	a vector of length narc containing the (integer) cost of each arc in the network.
`b`	a vector of length nnode containing the (integer) supply or demand of each node in the network. Supplies are given as positive numbers and demands as negative numbers.
`tcarcs`	an integer giving the total number of arcs between the treated and control nodes in the network.
`layers`	a list object containing information about the refined covariate balance layers of the network.
`z`	a vector of treatment indicators.
`fb.structure`	a matrix containing information about the membership of the treated and control units in the different classes of refined balance covariates.
`penalties`	a vector of integer penalties, one for each fine balance layer.
`theta`	a value no less than 1 giving the ratio by which the penalty is increased with each additional layer of fine balance.
`p`	a nonnegative value giving the penalty for the finest level of fine balance.

Author(s)

Samuel D. Pimentel

---

rcbsubset *Optimal Matching with Refined Covariate Balance*

Description

This function computes an optimal match with refined covariate balance.

Usage

```
rcbsubset(distance.structure, near.exact = NULL, fb.list = NULL,
treated.info = NULL, control.info = NULL, exclude.penalty = NULL,
penalty = 2, tol = 1e-3, solver = 'rlemon')
```

Arguments

`distance.structure`
: A distance matrix, a sparse distance matrix of class InfinitySparseMatrix (produced by various functions in the `optmatch` package), or a a list of vectors that encodes information about covariate distances between treated and control units (produced by the `build.dist.struct` function in the `rcbalance` package). If a matrix is given, rows should correspond to treated units and columns to control units. Distances of `Inf` should be used to indicate that the units in question must never be matched to each other.

`near.exact`
: an optional character vector specifying names of covariates for near-exact matching. This argument takes precedence over any refined covariate balance constraints, so the match will produce the best refined covariate balance subject to matching exactly on this variable wherever possible. If multiple covariates are named, near-exact matching will be done on their interaction.

`fb.list`
: an optional list of character vectors specifying covariates to be used for refined balance. Each element of the list corresponds to a level of refined covariate balance, and the levels are assumed to be in decreasing order of priority. Each character vector should contain one or more names of categorical covariates on which the user would like to enforce near fine balance. If multiple covariates are specified, an interaction is created between the categories of the covariates and near fine balance is enforced on the interaction. IMPORTANT: covariates or interactions coming later in the list must be nested within covariates coming earlier in the list; if this is not the case the function will stop with an error. An easy way to ensure that this occurs is to include in each character vector all the variables named in earlier list elements. If the `fb.list` argument is specified, the `treated.info` and `control.info` arguments must also be specified.

`treated.info`
: an optional data frame containing covariate information for the treated units in the problem. The row count of this data frame must be equal to the length of the `distance.structure` argument, and it is assumed that row `i` contains covariate information for the treated unit described by element `i` of `distance.structure`. In addition, the column count and column names must be identical to those of the `control.info` argument, and the column names must include all of the covariate names mentioned in the `near.exact` and `fb.list` arguments.

`control.info`
: an optional data frame containing covariate information for the control units in the problem. The row count of this data frame must be no smaller than the maximum control index in the `distance.structure` argument, and it is assumed that row `i` contains the covariate information for the control indexed by `i` in distance.structure. In addition, the column count and column names must be identical to those of the `treated.info` argument.

`exclude.penalty`
: A parameter that gives the cost of excluding a treated unit. If left NULL it will be set to a very large value designed to ensure treated units are never excluded if they can be matched. Lower values may result in subsets of treated units being excluded.

`penalty`
: a nonnegative value. This is a tuning parameter that helps ensure the different levels of refined covariate balance are prioritized correctly. Setting the penalty

higher tends to improve the guarantee of match optimality up to a point, but penalties above a certain level cause integer overflows and throw errors. It is not recommended that the user change this parameter from its default value.

`tol` edge cost tolerance. This is the smallest tolerated difference between matching costs; cost differences smaller than this will be considered zero. Match distances will be scaled by inverse tolerance, so when matching with large edge costs or penalties the tolerance may need to be increased.

`solver` the name of the package used to solve the network flow optimization problem underlying the match, one of 'rlemon' (which uses the Lemon Optimization Library) and 'rrelaxiv' (which uses the RELAX-IV algorithm).

Details

To use the option `solver = 'rrelaxiv'`, the user must install the `rrelaxiv` manually; it is not hosted on CRAN because it carries an academic license.

Value

A list with the following components:

`matches` a nt by k matrix containing the matched sets produced by the algorithm (where nt is the number of treated units). The rownames of this matrix are the numbers of the treated units (indexed by their position in distance.structure), and the elements of each row contain the indices of the control units to which this treated unit has been matched.

`fb.tables` a list of matrices, equal in length to the fb.list argument. Each matrix is a contingency table giving the counts among treated units and matched controls for each level of the categorical variable specified by the corresponding element of fb.list.

Author(s)

Samuel D. Pimentel

References

Pimentel, S.D., Kelz, R.R., Silber, J.H., and Rosenbaum, P.R. (2015) Large, sparse optimal matching with refined covariate balance in an observational study of the health outcomes produced by new surgeons, JASA 110 (510), 515-527.

Pimentel, S.D., and Kelz, R.R. (2020). Optimal tradeoffs in matched designs comparing US-trained and internationally trained surgeons. JASA 115 (532), 1675-1688.

Rosenbaum, P.R. (2012) Optimal matching of an optimally chosen subset in observational studies, JCGS 21.1: 57-71.

Examples

```
## Not run:
library(optmatch)
data(nuclearplants)
```

```
#match exactly on 3 binaries
exact.mask <- exactMatch(pr ~ pt + ct + bw, data = nuclearplants)
my.dist.matrix <- match_on(pr ~ date + t1 + t2 + cap + ne + cum.n,
within = exact.mask, data = nuclearplants)

#one treated unit out of 10 is excluded
rcbsubset(my.dist.matrix)

#repeat under a refined balance constraint
rcbsubset(my.dist.matrix, fb.list = list('ne'),
 treated.info = nuclearplants[which(nuclearplants$pr ==1),],
 control.info = nuclearplants[which(nuclearplants$pr == 0),])

#specifying a low exclude.penalty leads to more individuals excluded
rcbsubset(my.dist.matrix, fb.list = list('ne'),
 treated.info = nuclearplants[which(nuclearplants$pr ==1),],
 control.info = nuclearplants[which(nuclearplants$pr == 0),], exclude.penalty = 1.5)


#match using distance objects created by rcbalance package
library(rcbalance)

my.dist.struct <- build.dist.struct(z = nuclearplants$pr,
X = subset(nuclearplants[c('date','t1','t2','cap','ne','cum.n')]),
exact = paste(nuclearplants$pt, nuclearplants$ct, nuclearplants$bw, sep = '.'))

rcbsubset(my.dist.struct, fb.list = list('ne'),
 treated.info = nuclearplants[which(nuclearplants$pr ==1),],
 control.info = nuclearplants[which(nuclearplants$pr == 0),], exclude.penalty = 15)


## End(Not run)
```

Index

add.layer *(*dist2net*)*, 3

dist2net, 3

penalize.near.exact *(*dist2net*)*, 3
penalty.update *(*dist2net*)*, 3

rcbsubset, 5
rcbsubset-package, 2
remove.layer *(*dist2net*)*, 3