Package: bigmatch (via r-universe)

August 30, 2024

Type Package

Title Making Optimal Matching Size-Scalable Using Optimal Calipers

Version 0.6.4

Author Ruoqi Yu

Maintainer Ruoqi Yu <ruoqiyu125@gmail.com>

Description Implements optimal matching with near-fine balance in large observational studies with the use of optimal calipers to get a sparse network. The caliper is optimal in the sense that it is as small as possible such that a matching exists. The main functions in the 'bigmatch' package are optcal() to find the optimal caliper, optconstant() to find the optimal number of nearest neighbors, and nfmatch() to find a near-fine balance match with a caliper and a restriction on the number of nearest neighbors. Yu, R., Silber, J. H., and Rosenbaum, P. R. (2020). <DOI:10.1214/19-sts699>.

License MIT + file LICENSE

Additional_repositories https://errickson.net/rrelaxiv/

Encoding UTF-8

LazyData true

Imports rcbalance, stats, liqueueR, plyr, mvnfast, methods

Suggests optmatch, rrelaxiv

Note The optmatch package, which is useful for running many of the provided functions, may be downloaded from Github at <https://github.com/markmfredrickson/optmatch> if not available on CRAN. 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/>.

NeedsCompilation no

Date/Publication 2022-08-11 07:40:02 UTC

Repository https://ruoqiyu.r-universe.dev

RemoteUrl https://github.com/cran/bigmatch

RemoteRef HEAD

RemoteSha 366f5bbee600f3fb90823f8d3846daffac6fe68b

Contents

`check` *Check SMDs of the matched data set.*

Description

The function is used to create a table of mean and SMDs to check the balance before and after matching.

Usage

```
check(fdata, mdata, fz, mz)
```

Arguments

`fdata`	A full data frame with length(fz) rows and columns being variables that need to check SMDs. fdata and mdata must have the same variables with same column names in the same order.
`mdata`	A matched data frame with length(mz) rows and columns being variables that need to check SMDs. fdata and mdata must have the same variables with same column names in the same order.
`fz`	A vector whose ith coordinate is 1 for a treated unit and is 0 for a control for subjects in the full data set.
`mz`	A vector whose ith coordinate is 1 for a treated unit and is 0 for a control for subjects in the matched data set.

Value

A matrix with a row for each variable and five columns being the mean of treated group, mean of matched control group, mean of full control group, SMD of matched control group and SMD of full control group.

Examples

```
# To run this example, you must load the optmatch package.

# Caliper of .2 on the propensity score, near fine balance of
# education, a robust Mahalanobis distrance for X.
data(nh0506)
attach(nh0506)
X<-cbind(female,age,black,hispanic,education,povertyr,bmi)
m<-nfmatch(z=z,p=propens,fine=education,X=X,caliper=.2,dat=nh0506,rank=FALSE)
matcheddata<-m$data
Xm<-subset(matcheddata, select=c('female','age','black','hispanic','education','povertyr','bmi'))
check(X,Xm,z,matcheddata$z)
detach(nh0506)
```

`edgenum` *Computes the number of edges in the reduced bipartite.*

Description

Computes the number of edges in the reduced bipartite graph after applying the caliper and number of nearest neighbors (constant).

This function can provide users some idea of the required computation time. Smaller caliper and constant removes more edges, hence accelarates computation, but risks infeasibility.

Usage

```
edgenum(z, p, caliper, constant=NULL, exact=NULL,
ties.all=TRUE)
```

Arguments

`z`	A vector whose ith coordinate is 1 for a treated unit and is 0 for a control.
`p`	A vector of length(z)=length(p) giving the variable used to define the caliper. Typically, p is the propensity score or its rank.
`caliper`	If two individuals differ on p by more than caliper, we will not calculate the distance for this pair.
`constant`	If the number of pairs within a caliper is greater than constant, we will select the constant closest ones.
`exact`	If not NULL, then a vector of length(z)=length(p) giving variable that need to be exactly matched.

`ties.all`	If ties.all is True, include all ties while choosing nearest neighbors. In this case, some treated may have more than constant controls. Otherwise, randomly select one or several controls to make sure there are not more than constant controls for each treated.

Details

A given choice of caliper and number of nearest neighbors (constant) removes candidate pairs, so there exists a corresponding reduced bipartite graph.

Smaller caliper and constant removes more edges from the original dense graph, hence the computation is faster. However, this risks infeasibility. A smallest caliper that permits a feasible match and its corresponding smallest number of nearest neighbors can be computed by functions optcal() and optconstant().

Value

Number of edges in the reduced bipartite graph with the constraints on caliper and number of nearest neighbors (constant).

Examples

```
data(nh0506)
attach(nh0506)
edgenum(z,propens,0.2)

edgenum(z,propens,0.2,10,exact=female)

detach(nh0506)
```

`findexact` *Build an exact match variable given a matrix of covariates ordered by the user according to their importance.*

Description

The function is used to build an exact match variable given a matrix of covariates ordered by the user according to their importance. It will select as many important covariates as possible.

Usage

```
findexact(z, E, ncontrol=1)
```

Arguments

`z`	A vector whose ith coordinate is 1 for a treated unit and is 0 for a control.
`E`	A matrix or a vector with length(z) rows giving the covariates that need to be exactly matched.
`ncontrol`	The number of controls to be matched to each treated individual.

Value

`miss`	The covariates that cannot be exactly matched.
`variables`	The covariates that can be exactly matched.
`NewExact`	The constructed exact match variable.

Examples

```
data(nh0506)
attach(nh0506)
# The following example uses all of the variables in the propensity score,propens
ex<-findexact(z,cbind(female,age,education,black,hispanic,povertyr))
head(ex$miss)
head(ex$variables)
table(z,ex$NewExact)

# In order to exact match on age as well
# one option is to divide age into several levels
age_quantile<-as.integer(cut(age,quantile(age,c(0,0.25,0.5,0.75,1))))
ex2<-findexact(z,cbind(female,age_quantile,education,black,hispanic,povertyr,bmi))
head(ex2$miss)
head(ex2$variables)
table(z,ex2$NewExact)
detach(nh0506)
```

`glover` *Maximum matching in a convex bipartite graph.*

Description

Uses Glover's (1967) algorithm to find a maximum matching in a doubly convex bipartite graph. The implementation uses a priority queue, not used by Glover, as in Lipski and Preparata (1981). Of limited interest to most users; function glover() would typically be called by other functions.

Usage

```
glover(left, right)
```

Arguments

`left`	Treated person i may be matched to controls j with left[i] <= j <= right[i]. There are length(left) treated individuals and length(left)=length(right). Must have left[i]<=right[i] for every i. Define maxc = max(right). Then there are maxc potential control, labeled 1, 2,..., maxc. The values in left and right are these labels for controls.
`right`	See left.

Details

The match produced by glover is rarely useful in observational studies, because it finds a match, not the closest match, not the minimum distance match.

The glover algorithm may be used to find the smallest feasible caliper on a certain criterion. Each caliper on the criterion creates a different convex bipartite graph. Use glover to check whether a particular caliper is feasible. Use bisection search to find the smallest caliper that permits a match. There are many variations on this theme.

The glover algorithm is much faster than optimal matching. Iterative use of glover's algorithm is often faster than a single minimum distance match.

Value

The maximum matching ratio. A perfect matching has every treated individual i matched to a different control j with left[i] <= j <= right[i]. A perfect matching may not exist. A number smaller than 1 is returned if no perfect matching exists. Otherwise, 1 is returned.

References

Glover, F. (1967). Maximum Matching In Convex Bipartite Graphs. Naval Research Logistics Quarterly, 14, pp 313-316.

Lipski, W., Jr, and Preparata, F. P. (1981). Efficient Algorithms For Finding Maximum Matchings In Convex Bipartite Graphs And Related Problems. Journal Acta Informatica, 15, 4, pp 329-346.

Examples

```
# A perfect matching exists, and glover produces one.
left<-c(2,1,1,4,5)
right<-c(4,3,1,5,5)
glover(left,right)

# No perfect matching exists, and glover returns maximum matching ratio.
# Treated 4 and treated 5 can only be matched to control 5.

left<-c(2,1,1,5,5)
right<-c(4,3,1,5,5)
glover(left,right)
```

`nearfine` *Minimum-distance near-fine matching.*

Description

The program finds an optimal near-fine match given a distance structure with from node (treated), to node (control), and distance between each pair.

Usage

```
nearfine(z, fine, dist, dat, ncontrol=1, penalty=1000, max.cost=penalty/10,
nearexPenalty=max.cost, subX=NULL)
```

Arguments

`z`	A vector whose ith coordinate is 1 for a treated unit and is 0 for a control. Must have treated subjects (z=1) before controls (z=0).
`fine`	A vector of with length(z)=length(fine) giving the nominal levels that are to be nearly-finely balanced.
`dist`	A distance list with the starting node (treated subjec), ending node (control), the distance between them and whether nearexact is needed for each pair.
`dat`	A data frame with length(z) rows. One output of the program is a data frame with some of the rows of dat for the matched sample, together with additional columns describing the match.
`ncontrol`	A positive integer giving the number of controls to be matched to each treated subject.
`penalty`	A numeric penalty imposed for each violation of fine balance.
`max.cost`	The maximum cost for the each pair of treated and control while rounding the cost. 2 times it is the cost for nearexact matching
`nearexPenalty`	The penalty for a mismatch on nearexact. If it is a number, then use the same penalty for all nearexact variables. Otherwise, it should be a vector of length the same as number of nearexact variables, indicating the penalty for mismatch on each nearexact variable. The larger nearexPenalty is, the more priorty the variable get in near-exact match.
`subX`	If a subset matching is required, the variable that the subset matching is based on. That is, for each level of subX, extra treated will be discarded in order to have the number of matched treated subjects being the minimum size of treated and control groups. If exact matching on a variable x is desired and discarding extra treated is fine if there are more treated than controls for a certain level k, set exact=x, subX=x.

Details

The match minimizes the total distance between treated subjects and their matched controls subject to a near-fine balance constraint imposed as a penalty on imbalances.

The distance list only includes pairs closed based on the caliper, i.e. some edges are removed from the network. Because of this, the match may be infeasible. This is reported in feasible.

For discussion of networks for fine-balance, see Rosenbaum (1989, Section 3) and Rosenbaum (2010).

For near-fine balannce balance, see Yang et al. (2012).

You MUST install and load the optmatch package to use nearfine().

Value

`feasible`	feasible=1 if the match is feasible or feasible=0 if the match is infeasible.
`timeinrelax`	Time in RELAX IV spent computing the minimum cost flow.
`timeinnet`	Time in constructing the network.
`timeinmatch`	Time in constructing the matched dataset.
`d`	The matched sample. Selected rows of dat. The first column indicates which matched set the subject belongs to.
`number`	Number of edges between the treated subjects and controls in the reduced network.

References

Bertsekas, D. P. and Tseng, P. (1988) The relax codes for linear minimum cost network flow problems. Annals of Operations Research, 13, 125-190. Fortran and C code: http://www.mit.edu/~dimitrib/home.html. Available in R via the optmatch package.

Rosenbaum, P.R. (1989) Optimal matching in observational studies. Journal of the American Statistical Association, 84, 1024-1032.

Rosenbaum, P. R. (2010) Design of Observational Studies. New York: Springer.

Yang, D., Small, D. S., Silber, J. H., and Rosenbaum, P. R. (2012) Optimal matching with minimal deviation from fine balance in a study of obesity and surgical outcomes. Biometrics, 68, 628-636.

Examples

```
## Not run:
# To run this example, you must load the optmatch package.
data(nh0506)
attach(nh0506)
X<-cbind(female,age,black,hispanic,education,povertyr)
dist<-smahal(z,propens,X,0.2)
fine<-education
m<-nearfine(z,fine,dist,nh0506)
head(m$d)
detach(nh0506)

## End(Not run)
```

`netfine` *Optimal near-fine match from a distance matrix.*

Description

The function creates the network for optimal near-fine matching to be passed via callrelax to the Fortran code for Bertsekas and Tseng's (1988) Relax IV.

Of limited interest to most users; function netfine() would typically be called by some other functions.

Usage

```
netfine(z, fine, dist, ncontrol=1, penalty=1000, max.cost=penalty/10,
nearexPenalty=max.cost, subX=NULL)
```

Arguments

`z`	A vector whose ith coordinate is 1 for a treated unit and is 0 for a control.
`fine`	A vector of with length(z)=length(fine) giving the nominal levels that are to be nearly-finely balanced.
`dist`	A distance list with the starting node (treated subjec), ending node (control), the distance between them and whether nearexact is needed for each pair.
`ncontrol`	A positive integer giving the number of controls to be matched to each treated subject.
`penalty`	A numeric penalty imposed for each violation of fine balance.
`max.cost`	The maximum cost for the each pair of treated and control while rounding the cost.
`nearexPenalty`	The penalty for a mismatch on nearexact. If it is a number, then use the same penalty for all nearexact variables. Otherwise, it should be a vector of length the same as number of nearexact variables, indicating the penalty for mismatch on each nearexact variable. The larger nearexPenalty is, the more priorty the variable get in near-exact match.
`subX`	If a subset matching is required, the variable that the subset matching is based on. That is, for each level of subX, extra treated will be discarded in order to have the number of matched treated subjects being the minimum size of treated and control groups. If exact matching on a variable x is desired and discarding extra treated is fine if there are more treated than controls for a certain level k, set exact=x, subX=x.

Details

The network contains a bipartite graph for treated and control subjects plus additional nodes for fine balance categories, plus additional nodes accept needed deviations from fine balance yielding near-fine balance.

For discussion of fine-balance, see Rosenbaum (1989, Section 3) and Rosenbaum (2010). For near-fine balance balance, see Yang et al. (2012).

Value

A network for optimal near-fine matching.

References

Bertsekas, D. P. and Tseng, P. (1988) The relax codes for linear minimum cost network flow problems. Annals of Operations Research, 13, 125-190. Fortran and C code: http://www.mit.edu/~dimitrib/home.html. Available in R via the optmatch package.

Rosenbaum, P.R. (1989) Optimal matching in observational studies. Journal of the American Statistical Association, 84, 1024-1032.

Rosenbaum, P. R. (2010) Design of Observational Studies. New York: Springer.

Yang, D., Small, D. S., Silber, J. H., and Rosenbaum, P. R. (2012) Optimal matching with minimal deviation from fine balance in a study of obesity and surgical outcomes. Biometrics, 68, 628-636.

`nfmatch` *Minimum-distance near-fine matching.*

Description

The program finds an optimal near-fine match with a given caliper on p or rank of p.

Usage

```
nfmatch(z, p, fine=rep(1,length(z)), X, dat, caliper, constant=NULL, ncontrol=1,
rank=TRUE, exact=NULL, penalty=1000, max.cost=penalty/10, nearexact=NULL,
nearexPenalty=max.cost, Xextra=NULL, weight=NULL, subX=NULL, ties.all=TRUE, seed=1)
```

Arguments

`z`	A vector whose ith coordinate is 1 for a treated unit and is 0 for a control.
`p`	A vector of with length(z)=length(p) giving the numeric values of the variable to be matched with a caliper. Typically, p is the propensity score. If p takes a few levels, exact matching for p is attempted if caliper=0. If no caliper is intended to apply, set p=rep(1,length(z)) and caliper=1.
`fine`	A vector of with length(z)=length(fine) giving the nominal levels that are to be nearly-finely balanced.
`X`	A matrix of covariates used to create a robust Mahalanobis distance. X must have length(z) rows.
`dat`	A data frame with length(z) rows. If the match is feasible, the matched portion of dat is returned with additional columns that define the match.
`caliper`	If two individuals differ on p by more than caliper, we will not calculate the distance for this pair. If caliper is too small, the match may be infeasible. If no caliper is intended to apply, set p=rep(1,length(z)) and caliper=1.
`constant`	If there are more than constant controls for a treated differ on p within caliper, we select the constant closest controls.
`ncontrol`	A positive integer giving the number of controls to be matched to each treated subject. If ncontrol is too large, the match will be infeasible.
`rank`	An indicator of whether we want a caliper on rank of p or p.
`exact`	If not NULL, then a vector of length(z)=length(p) giving variable that need to be exactly matched.
`penalty`	A numeric penalty imposed for each violation of fine balance.

`max.cost`	The maximum cost for the each pair of treated and control while rounding the cost.
`nearexact`	If not NULL, then a vector of length length(z) or matrix with length(z) rows giving variables that need to be exactly matched. If it is not possible to exactly match all variables, we will exactly match as many variables as we can.
`nearexPenalty`	The penalty for a mismatch on nearexact. If it is a number, then use the same penalty for all nearexact variables. Otherwise, it should be a vector of length the same as number of nearexact variables, indicating the penalty for mismatch on each nearexact variable. The larger nearexPenalty is, the more priorty the variable get in near-exact match.
`Xextra`	If not NULL, another robust Mahalanobis distance based on Xextra is calculated. The distance between treated-control pair is a weighted sum of the two distances.
`weight`	The weight for Mahalanobis distance of Xextra.
`subX`	If a subset matching is required, the variable that the subset matching is based on. That is, for each level of subX, extra treated will be discarded in order to have the number of matched treated subjects being the minimum size of treated and control groups. If exact matching on a variable x is desired and discarding extra treated is fine if there are more treated than controls for a certain level k, set exact=x, subX=x.
`ties.all`	If ties.all is True, include all ties while choosing nearest neighbors. In this case, some treated may have more than constant controls. Otherwise, randomly select one or several controls to make sure there are not more than constant controls for each treated.
`seed`	When ties.all is False, seed for randomly select one or several controls when there are ties.

Details

The match minimizes the total distance between treated subjects and their matched controls subject to a near-fine balance constraint imposed as a penalty on imbalances.

For discussion of networks for fine-balance, see Rosenbaum (1989, Section 3) and Rosenbaum (2010). For near-fine balannce balance, see Yang et al. (2012).

You MUST install and load the optmatch package to use nearfine.

Value

If the match is infeasible, a warning is issued. Otherwise, a list of results is returned.

A match may be infeasible if the caliper is too small, or ncontrol is too large, or if exact matching for exact is impossible.

`data`	The matched sample. Selected rows of dat.
`timeinrelax`	Time in RELAX IV spent computing the minimum cost flow.
`edgenum`	Number of edges between the treated subjects and controls in the reduced network.

`timeind`	Time in calculating robust Mahalanobis distance between connected pairs.
`timeinnet`	Time in constructing the network.
`timeinmatch`	Time in constructing the matched dataset.

References

Bertsekas, D. P. and Tseng, P. (1988) The relax codes for linear minimum cost network flow problems. Annals of Operations Research, 13, 125-190. Fortran and C code: http://www.mit.edu/~dimitrib/home.html. Available in R via the optmatch package.

Rosenbaum, P.R. (1989) Optimal matching in observational studies. Journal of the American Statistical Association, 84, 1024-1032.

Rosenbaum, P. R. (2010) Design of Observational Studies. New York: Springer.

Yang, D., Small, D. S., Silber, J. H., and Rosenbaum, P. R. (2012) Optimal matching with minimal deviation from fine balance in a study of obesity and surgical outcomes. Biometrics, 68, 628-636.

Examples

```
# To run this example, you must load the optmatch package.

# Caliper of .3 on the propensity score, near fine balance of
# education, a robust Mahalanobis distrance for X.
data(nh0506)
attach(nh0506)
X<-cbind(female,age,black,hispanic,education,povertyr,bmi)
m<-nfmatch(z=z,p=propens,fine=education,X=X,caliper=.3,dat=nh0506,rank=FALSE)
matcheddata=m$data
table(matcheddata$z,matcheddata$education)
head(matcheddata)
detach(nh0506)

  #finds the optimal caliper for the propensity score while exact matching on female
  #near fine balance for education and hispanic jointly.
  data(nh0506)
  attach(nh0506)
  X<-cbind(female,age,black,hispanic,education,povertyr,bmi)

  oc<-optcal(z,propens,exact=female,tol=0.1,rank=FALSE)
  oc
  oco<-optconstant(z,propens,oc$caliper,exact=female,rank=FALSE)
  oco
 m2<-nfmatch(z,propens,factor(hispanic):factor(education),X,nh0506,oc$caliper,oco$constant,
             exact=female,rank=FALSE)

  matcheddata2=m2$data
  table(matcheddata2$z,matcheddata2$female)
  table(matcheddata2$z,matcheddata2$education)
  table(matcheddata2$z,matcheddata2$education,matcheddata2$hispanic)

  #finds the optimal caliper for the propensity score while exact matching on female
  #nearexact on quantiles of povertyr and bmi
  #near fine balance for education and hispanic jointly.
  pq=cut(povertyr,c(-0.1,1,2,3,4,5))
  bq=cut(bmi,(0:7)*20)
  #first assume povertyr and bmi are of the same importance
 m3<-nfmatch(z,propens,factor(hispanic):factor(education),X,nh0506,oc$caliper,oco$constant,
              exact=female,nearexact=cbind(pq,bq),rank=FALSE)
  matcheddata3=m3$data
  head(matcheddata3)

  #then assume povertyr is more important than bmi
 m4<-nfmatch(z,propens,factor(hispanic):factor(education),X,nh0506,oc$caliper,oco$constant,
              exact=female,nearexact=cbind(pq,bq),nearexPenalty=c(100,50),rank=FALSE)
  matcheddata4=m4$data
  head(matcheddata4)
  detach(nh0506)
```

`nh0506` *Smoking and homocysteine levels in NHANES 2005-2006.*

Description

Bazzano et al. (2003) noted higher homocysteine levels in smokers than in nonsmokers. See also Pimentel et al. (2016) for a related analysis.

Usage

```
data("nh0506")
```

Format

A data frame with 2475 observations on the following 32 variables.

`Row` a numeric vector

`SEQN` NHANES id number

`female` 1 if female, 0 if male

`age` age in years, >=20

`black` 1 if black, 0 otherwise

`hispanic` 1 if hispanic, 0 otherwise

`education` Education

`povertyr` Ratio of family income to the poverty level, capped at 5x

`creactiveprotein` creactive protein

`homocysteine` homocysteine

`cotinine` cotinine in blood

`cadmium` cadmium in blood

`lead` lead in blood

`bmi` Body mass index

`cigs100life` 1 if smoked more than 100 cigarettes in lifetime, 0 otherwise

`smokenow` 1 if smokes now, 0 otherwise

`cigsdays30` Days smoked in last 30 days: 0 if never smoker, 30 if daily smoker

`cigsperday30` Cigarettes smoked per day in last 30 days

`tobacco5days` 1 = used tobacco in the last 30 days, 0 otherwise

`dailysmoker` 1 = daily smoker, 0 = never smoker

`neversmoker` 1 = never smoker, 0 = daily smoker

`z` 1 if daily smoker, 0 if never smoker

`propens` Estimated propensity score. The score was formed by logit regression of z on female, age, education, black, hispanic, povertyr, and bmi.

`pstrat` Propensity score strata: `(0,0.0733] (0.0733,0.131] (0.131,0.204] (0.204,0.33] (0.33,1]`

`age3` Age in 3 categories

`ed3` Education in 3 categories

`bmi3` BMI in 3 categories

`pov2` Income above 2 times poverty, TRUE or FALSE

`stf` A factor defining strata using female, age3, ed3, bmi3 pov2.

`st` A numeric version of stf

`stfp` A factor defining strata using stf and pstrat

`stp` A numeric version of stp

Details

Data from NHANES 2005-2006 concerning homocysteine levels in daily smokers (z=1) and never smokers (z=0), aged 20 and older. Daily smokers smoked every day for the last 30 days, smoking an average of at least 10 cigarettes per day. Never smokers smoked fewer than 100 cigarettes in their lives, do not smoke now, and had no tobacco use in the previous 5 days.

Source

NHANES, the US National Health and Nutrition Examination Survey, 2005-2006.

References

Bazzano, L. A., He, J., Muntner, P., Vupputuri, S. and Whelton, P. K. (2003) Relationship between cigarette smoking and novel risk factors for cardiovascular disease in the United States. Annals of Internal Medicine, 138, 891-897.

Pimentel, S. D., Small, D. S. and Rosenbaum, P. R. (2016) Constructed second control groups and attenuation of unmeasured biases. Journal of the American Statistical Association, 111, 1157-1167.

Examples

```
data(nh0506)
```

`optcal` *Finds the optimal caliper width.*

Description

Finds the smallest caliper on variable p or rank of p such that a treated-control matching with that caliper exists. If exact is not NULL, then finds the smallest caliper on p or rank of p such that a treated-control matching with that caliper exists while also matching exactly for the variable exact.

Usage

```
optcal(z, p, exact=NULL, ncontrol=1, tol=NULL, rank=TRUE, subX=NULL)
```

Arguments

`z`	A vector whose ith coordinate is 1 for a treated unit and is 0 for a control.
`p`	A vector with the same length as z. The best caliper for p is found. Often p is the propensity score.
`exact`	If exact is NULL, then there is no exact matching, and the caliper refers to p alone. Otherwise, exact is a vector of the same length as z for exact matching, such that two individuals, i and j, can be matched only if they have the same value of exact, exact[i]=exact[j]. In this case, the caliper is best among calipers that permit exact matching for exact. Typically, exact has a moderate number of possible values, far fewer than length(z).
`ncontrol`	A positive integer giving the number of controls to be matched to each treated subject. If ncontrol is too large, the match will be infeasible.
`tol`	The tolerance. The optimal caliper is determined with an error of at most tol. tol=0.01 might be used for the propensity score, as it takes values between 0 and 1, whereas tol=1/2 for p=age would mean that the caliper for age errs by at most half a year.
`rank`	An indicator of whether we want a caliper on rank of p or p.
`subX`	If a subset matching is required, the variable that the subset matching is based on. That is, for each level of subX, extra treated will be discarded in order to have the number of matched treated subjects being the minimum size of treated and control groups. If exact matching on a variable x is desired and discarding extra treated is fine if there are more treated than controls for a certain level k, set exact=x, subX=x.

Details

The method uses binary search to find the optimal caliper. At each step in the search, it applies Glover's algorithm to determine whether a proposed caliper is feasible.

Often, we need a small and feasible caliper, but we do not need to determine the optimal caliper very precisely. Making tol larger will reduce the number of steps in the binary search.

Value

`caliper`	The optimal caliper, with an error of at most tol. This caliper is a little too large, at most tol too large, but because its error is on the high side, a match with this caliper does exist.
`interval`	An interval that contains the best caliper. The upper bound of the interval was returned as caliper above.
`interval.length`	The length of interval. By definition, length.interval<=tol.

References

Glover, F. (1967). Maximum Matching In Convex Bipartite Graphs. Naval Research Logistics Quarterly, 14, pp 313-316.

Lipski, W., Jr, and Preparata, F. P. (1981). Efficient Algorithms For Finding Maximum Matchings In Convex Bipartite Graphs And Related Problems. Journal Acta Informatica, 15, 4, pp 329-346.

Examples

```
data(nh0506)
attach(nh0506)

#optimal caliper using the propensity score alone
optcal(z,propens,tol=0.1,rank=FALSE)

#optimal caliper using the rank of propensity score
#and match each treated subject with two controls

optcal(z,propens,ncontrol=2,rank=TRUE)


#optimal caliper for the propensity score while requiring
#an exact match for female
optcal(z,propens,exact=female,tol=0.1,rank=FALSE)

detach(nh0506)
```

`optconstant` *Finds the optimal constant.*

Description

Finds the smallest constant k on variable p or rank of p such that a treated-control matching with that constant exists. If exact is not NULL, then finds the smallest constant on p or rank of p such that a treated-control matching with that constant exists while also matching exact for the variable exact. If caliper is NULL, we only consider match treated i to k controls with smallest difference of p or rank of p. If caliper is not NULL and there are more than k controls within that caliper for treated i, then we only consider match it to k controls with smallest difference of p or rank of p.

Usage

```
optconstant(z, p, caliper=NULL, exact=NULL, ncontrol=1, tol=1, rank=TRUE,
subX=NULL, ties.all=TRUE, seed=1)
```

Arguments

`z`	A vector whose ith coordinate is 1 for a treated unit and is 0 for a control.
`p`	A vector with the same length as z. The best constant for p is found. Often p is the propensity score.
`caliper`	If two individuals differ on p by more than caliper, there is no edge between this pair. If caliper is too small, the match may be infeasible.
`exact`	If exact is NULL, then there is no exact matching, and the constant refers to p alone. Otherwise, exact is a vector of the same length as z for exact matching, such that two individuals, i and j, can be matched only if they have the same value of exact, exact[i]=exact[j]. In this case, the constant is best among calipers that permit exact matching for exact. Typically, exact has a moderate number of possible values, far fewer than length(z).
`ncontrol`	A positive integer giving the number of controls to be matched to each treated subject. If ncontrol is too large, the match will be infeasible.
`tol`	The tolerance. The optimal constant is determined with an error of at most tol.
`rank`	An indicator of whether we want a constant on rank of p or p.
`subX`	If a subset matching is required, the variable that the subset matching is based on. That is, for each level of subX, extra treated will be discarded in order to have the number of matched treated subjects being the minimum size of treated and control groups. If exact matching on a variable x is desired and discarding extra treated is fine if there are more treated than controls for a certain level k, set exact=x, subX=x.
`ties.all`	If ties.all is True, include all ties while choosing nearest neighbors. In this case, some treated may have more than constant controls. Otherwise, randomly select one or several controls to make sure there are not more than constant controls for each treated.
`seed`	When ties.all is False, seed for randomly select one or several controls when there are ties.

Details

The method uses binary search to find the optimal constant. At each step in the search, it applies Glover's algorithm to determine whether a proposed constant is feasible.

Often, we need a small and feasible constant, but we do not need to determine the optimal constant very precisely. Making tol larger will reduce the number of steps in the binary search.

Value

`constant`	The optimal constant, with an error of at most tol. This constant is a little too large, at most tol too large, but because its error is on the high side, a match with this constant does exist.

`interval`	An interval that contains the best constant. The upper bound of the interval was returned as constant above.
`interval.length`	The length of interval. By definition, length.interval<=tol.

References

Glover, F. (1967). Maximum Matching In Convex Bipartite Graphs. Naval Research Logistics Quarterly, 14, pp 313-316.

Lipski, W., Jr, and Preparata, F. P. (1981). Efficient Algorithms For Finding Maximum Matchings In Convex Bipartite Graphs And Related Problems. Journal Acta Informatica, 15, 4, pp 329-346.

Examples

```
data(nh0506)
attach(nh0506)

#optimal constant using the propensity score alone
optconstant(z,propens,rank=FALSE)

#optimal constant for the propensity score while requiring
#an exact match for female
optconstant(z,propens,exact=female,rank=FALSE)

#optimal constant for the propensity score given a caliper
oc=optcal(z,propens,tol=0.1,rank=FALSE)
optconstant(z,propens,caliper=oc$caliper,rank=FALSE)
detach(nh0506)
```

`smahal` *Creates a robust Mahalanobis distance for matching.*

Description

Computes a robust Mahalanobis distance list for use in matching.

This function and its use are discussed in Rosenbaum (2010). The robust Mahalanobis distance in described in Chapter 8 of Rosenbaum (2010).

Usage

```
smahal(z, p, X, caliper, constant=NULL, ncontrol=1, exact=NULL,
nearexact=NULL, nearexPenalty=100, Xextra=NULL, weight=NULL, subX=NULL, ties.all=TRUE)
```

Arguments

`z`	A vector whose ith coordinate is 1 for a treated unit and is 0 for a control.
`p`	A vector of length(z)=length(p) giving the variable used to define the caliper. Typically, p is the propensity score or its rank.
`X`	A matrix with length(z) rows giving the covariates. X should be of full column rank.
`caliper`	If two individuals differ on p by more than caliper, we will not calculate the distance for this pair.
`constant`	If the number of pairs within a caliper is greater than constant, we will select the constant closest ones.
`ncontrol`	A positive integer giving the number of controls to be matched to each treated subject. If ncontrol is too large, the match will be infeasible.
`exact`	If not NULL, then a vector of length(z)=length(p) giving variable that need to be exactly matched.
`nearexact`	If not NULL, then a vector of length length(z) or matrix with length(z) rows giving variables that need to be exactly matched. If it is not possible to exactly match all variables, we will exactly match as many variables as we can.
`nearexPenalty`	Penalty for mismatch on nearexact if nearexact is not NULL.
`Xextra`	If not NULL, another robust Mahalanobis distance based on Xextra is calculated. The distance between treated-control pair is a weighted sum of the two distances.
`weight`	The weight for Mahalanobis distance of Xextra.
`subX`	If a subset matching is required, the variable that the subset matching is based on. That is, for each level of subX, extra treated will be discarded in order to have the number of matched treated subjects being the minimum size of treated and control groups. If exact matching on a variable x is desired and discarding extra treated is fine if there are more treated than controls for a certain level k, set exact=x, subX=x.
`ties.all`	If ties.all is True, include all ties while choosing nearest neighbors. In this case, some treated may have more than constant controls. Otherwise, randomly select one or several controls to make sure there are not more than constant controls for each treated.

Details

The usual Mahalanobis distance works well for multivariate Normal covariates, but can exhibit odd behavior with typical covariates. Long tails or an outlier in a covariate can yield a large estimated variance, so the usual Mahalanobis distance pays little attention to large differences in this covariate. Rare binary covariates have a small variance, so a mismatch on a rare binary covariate is viewed by the usual Mahalanobis distance as extremely important. If you were matching for binary covariates indicating US state of residence, the usual Mahalanobis distance would regard a mismatch for Wyoming as much worse than a mismatch for California.

The robust Mahalanobis distance uses ranks of covariates rather than the covariates themselves, but the variances of the ranks are not adjusted for ties, so ties do not make a variable more important. Binary covariates are, of course, heavily tied.

Value

`d`	A distance list for each pair within the caliper distance and constant constraint.
`start`	The treated subject for each distance.
`end`	The control subject for each distance.
`nearex`	A vector or matrix with the same dimesion as nearexact. Its entry is TRUE for a connected pair whose nearexact variable values are different and is FALSE otherwise.

References

Rosenbaum, P. R. (2010) Design of Observational Studies. New York: Springer.

Examples

```
data(nh0506)
attach(nh0506)
X<-cbind(female,age,black,hispanic,education,povertyr)
dist<-smahal(z,propens,X,0.2)
dist$d[1:10]

dist2<-smahal(z,propens,X,0.2,exact=female,Xextra=hispanic,nearexact=bmi)
dist2$nearex[1:10]

detach(nh0506)
```

Index