Package 'nbpMatching'

July 2, 2014

Type Package

Title functions for non-bipartite optimal matching

Version 1.4.0

Date 2012-11-19

Author Bo Lu, Robert Greevy, Cole Beck

Maintainer Cole Beck <cole.beck@vanderbilt.edu>

Description solves Mahalanobis distance matrices

License GPL (>= 2)

LazyLoad yes

Depends methods, Hmisc

Imports MASS

Repository CRAN

Repository/R-Forge/Project nbpmatching

Repository/R-Forge/Revision 16

Repository/R-Forge/DateTimeStamp 2013-04-16 20:57:31

Date/Publication 2013-04-17 17:28:58

NeedsCompilation yes

R topics documented:

`nbpMatching-package` *Nonbipartite Matching*

Description

This package will take an input distance matrix and generate the set of pairwise matches that minimizes the sum of distances between the pairs by running nonbimatch.

The most current documentation is available at http://biostat.mc.vanderbilt.edu/wiki/Main/MatchedRandomization.

Author(s)

Bo Lu, Robert Greevy, Cole Beck

Maintainer: Cole Beck <cole.beck@vanderbilt.edu>

References

Lu B, Greevy R, Xu X, Beck C. Optimal Nonbipartite Matching and its Statistical Applications. The American Statistician. Vol. 65, no. 1. : 21-30. 2011.

Greevy RA Jr, Grijalva CG, Roumie CL, Beck C, Hung AM, Murff HJ, Liu X, Griffin MR. Reweighted Mahalanobis distance matching for cluster-randomized trials with missing data. Pharmacoepidemiol Drug Saf. 2012 May;21 Suppl 2:148-54. doi: 10.1002/pds.3260.

Examples

```
# create a covariate matrix
df <- data.frame(id=LETTERS[1:25], val1=rnorm(25), val2=rnorm(25))
# create distances
df.dist <- gendistance(df, idcol=1)
# create distancematrix object
df.mdm <- distancematrix(df.dist)
# create matches
```

```
df.match <- nonbimatch(df.mdm)
# review quality of matches
df.qom <- qom(df.dist$cov, df.match$matches)

# some helper functions are available
# runner -- start with the covariate, run through the entire process
df.1 <- runner(df, idcol=1)
# full.qom -- start with the covariate, generate a full quality of match report
df.2 <- full.qom(df)

## Not run:
try a large matrix
nonbimatch(distancematrix(as.matrix(dist(sample(1:10^8, 5000, replace=TRUE)))))

## End(Not run)
```

`assign.grp` *Random Group Assignment*

Description

Randomly assign each element into treatment group A or B.

Usage

```
assign.grp(matches, seed=68, ...)
```

Arguments

`matches`	A data.frame object. Contains information on how to match the covariate data set.
`seed`	Seed provided for random-number generation. Default value of 68.
`...`	Additional arguments, not used at the moment.

Details

This function takes the matched pairs generated by nonbimatch and randomly assigns each element to a group.

Value

original data.frame with treatment group column

Author(s)

Cole Beck

See Also

nonbimatch

Examples

```
df <- data.frame(id=LETTERS[1:25], val1=rnorm(25), val2=rnorm(25))
df.dist <- gendistance(df, idcol=1)
df.mdm <- distancematrix(df.dist)
df.match <- nonbimatch(df.mdm)
assign.grp(df.match$matches)
```

---

distancematrix *Distance matrix*

Description

The distancematrix function is used to reformat the input distance matrix into the format required by the nonbipartite matching Fortran code. The original matrix should have dimensions NxN, where N is the total number of elements to be matched. The matrix may be created in R and input into the distancematrix function. Alternately, the matrix may be read in from a CSV file, i.e. a text file where distances in a given row are delimited by commas. If a list element is given, it should have a data.frame element named "dist", preferably generated by the gendistance function.

Usage

```
distancematrix(x, ...)
```

Arguments

x	A matrix, data.frame, list or filename. This should be an NxN distance matrix for the N elements to be matched. The values in the diagonal are ignored because an element cannot be matched to itself. Using zeros in the diagonal is preferable, although other values are acceptable provided they are not so large that they distort the scaling of the other values.
...	Additional arguments, potentially used when reading in a filename and passed into read.csv.

Details

- The distancematrix function is used to reformat the input distance matrix into the format required by the nonbipartite matching Fortran code.
- If an extra column or row is present, it will be converted into row names. In other words, if the matrix has dimensions $(N+1)$xN, or Nx$(N+1)$, then the function will take the first row, or column, as an ID column. If both row and column names are present, i.e. a $(N+1)$x$(N+1)$ matrix, the function cannot identify the names.

- If an odd number of elements exist, a ghost element, or sink, will be created whose distance is zero to all of the other elements. For example, when matching 17 elements, the function will create an 18th element that matches every element perfectly. This sink may or not be appropriate for your application. Naturally, you may create sinks as needed in the distance matrix you input to the distancematrix function.
- The elements of distancematrix may not be re-assigned once created. In other words, you cannot edit the formatted distance matrix. You need to edit the matrix being input into the distancematrix function.

Value

distancematrix S4 object

Author(s)

Cole Beck

See Also

nonbimatch gendistance

Examples

```
plainmatrix<-as.matrix(dist(sample(1:25, 8, replace=TRUE)))
diag(plainmatrix) <- 99999  # setting diagonal to an infinite distance for
                            # pedagogical reasons (the diagonal may be left
                            # as zero)
mdm<-distancematrix(plainmatrix)
df <- data.frame(id=LETTERS[1:25], val1=rnorm(25), val2=rnorm(25))
df[sample(seq_len(nrow(df)), ceiling(nrow(df)*0.1)), 2] <- NA
df.dist <- gendistance(df, idcol=1, ndiscard=2)
mdm2 <- distancematrix(df.dist)
```

---

fill.missing *Data Imputation*

Description

The fill.missing function uses the 'transcan' function from the **Hmisc** package to impute values for the given data.frame.

Usage

```
fill.missing(x, seed = 101, simplify=TRUE, idcol="id", ...)
```

Arguments

x	A data.frame object. It should have missing values.
seed	Seed provided for random-number generation. Default value of 101.
simplify	A boolean value. When TRUE, duplicate missingness columns will be removed.
idcol	An integer value or character string. Indicates the column containing IDs, specified as column index or column name. Defaults to "id", or NA, when not found.
...	Additional arguments, potentially passed to transcan.

Details

The fill.missing function with fill the missing values within a data.frame with the values imputed with the transcan function. An idcol may be specified to prevent including the use of IDs in the imputation. In addition for every column that contains missing data, a new column will be attached to the data.frame containing an indicator of missingness. A "1" indicates that the value was missing and has been imputed.

Value

data.frame with imputed values

Author(s)

Cole Beck

See Also

transcan

Examples

```
set.seed(1)
df <- data.frame(id=LETTERS[1:25], val1=rnorm(25), val2=rnorm(25))
df[sample(seq_len(nrow(df)), ceiling(nrow(df)*0.1)), 2] <- NA
df <- fill.missing(df)
```

---

gendistance *Generate a Distance Matrix*

Description

The gendistance function creates a distance matrix from a covariates matrix.

Usage

```
gendistance(covariate, idcol=NULL, weights=NULL, prevent=NULL, force=NULL, rankcols=NULL, missing.we
```

Arguments

covariate	A data.frame object, containing the covariates of the data set.
idcol	An integer, providing the index of the column containing row ID's.
weights	A numeric vector, the length should match the number of columns. This value determines how much weight is given to each column when generating the distance matrix.
prevent	An integer vector, providing the index of columns that should be used to prevent matches. When generating the distance matrix, elements that match on these columns are given a maximum distance.
force	A integer, providing the index of the column contain information used to force pairs to match.
rankcols	An integer vector, providing the index of columns that should have the rank function applied to them before generating the distance matrix.
missing.weight	A numeric value, or vector, used to generate the weight of missingness indicator columns. Missingness indicator columns are created if there is missing data within the data set. Defaults to 0.1. If a single value is supplied, weights are generating by multiplying this by the original columns' weight. If a vector is supplied, it's length should match the number of columns with missing data, and the weight is used as is.
ndiscard	An integer, providing the number of elements that should be allowed to match phantom values. The default value is 0.
singular.method	A character string, indicating the function to use when encountering a singular matrix. By default, 'solve' is called. The alternative is to call 'ginv' from the **MASS** package.
...	Additional arguments, not used at this time.

Details

Given a data.frame of covariates, generate a distance matrix. Missing values are imputed with fill.missing. For each column with missing data, a missingness indicator column will be added. Phantoms are fake elements that perfectly match all elements. They can be used to discard a certain number of elements.

Value

a list object with several elements

dist	generated distance matrix
cov	covariate matrix used to generate distances
ignored	ignored columns from original covariate matrix
weights	weights applied to each column in covariate matrix
prevent	columns used to prevent matches
mates	index of rows that should be forced to match

rankcols	index of columns that should use rank
missing.weight	weight to apply to missingness indicator columns
ndiscard	number of elements that will match phantoms

Author(s)

Cole Beck

See Also

distancematrix

Examples

```
set.seed(1)
df <- data.frame(id=LETTERS[1:25], val1=rnorm(25), val2=rnorm(25))
# add some missing data
df[sample(seq_len(nrow(df)), ceiling(nrow(df)*0.1)), 2] <- NA
df.dist <- gendistance(df, idcol=1, ndiscard=2)
# up-weight the second column
df.weighted <- gendistance(df, idcol=1, weights=c(1,2,1), ndiscard=2, missing.weight=0.25)
df[,3] <- df[,2]*2
df.sing.solve <- gendistance(df, idcol=1, ndiscard=2)
df.sing.ginv <- gendistance(df, idcol=1, ndiscard=2, singular.method="ginv")
```

---

get.sets *Get named sets of matches*

Description

Create a factor variable using the names from a matched data set.

Usage

```
get.sets(matches, remove.unpaired=TRUE, ...)
```

Arguments

matches	A data.frame object. Contains information on how to match the covariate data set.
remove.unpaired	A boolean value. The default is to remove elements matched to phantom elements.
...	Additional arguments, not used at this time.

Details

Calculate a name for each pair by using the ID columns from the matched data set. Return a factor of these named pairs.

Value

a factor vector

Author(s)

Cole Beck

Examples

```
df <- data.frame(id=LETTERS[1:25], val1=rnorm(25), val2=rnorm(25))
df.dist <- gendistance(df, idcol=1)
df.mdm <- distancematrix(df.dist)
df.match <- nonbimatch(df.mdm)
get.sets(df.match$matches)
# include the phantom match
get.sets(df.match$matches, FALSE)
```

---

make.phantoms *Add Phantom Rows and Columns*

Description

The make.phantoms function will take an NxN matrix and add NP phantom elements, thus creating a matrix with $N + NP$x$N + NP$ dimensions.

Usage

```
make.phantoms(x, nphantoms, name="phantom", maxval=2*10^8, ...)
```

Arguments

x	A matrix or data.frame object, with NxN dimensions.
nphantoms	An integer, providing the number of phantom elements to add.
name	A character string, indicating the name attribute for new elements. Defaults to "phantom".
maxval	An integer value, the default value to give the pairs of phantoms (indeces [N+1:N+NP, N+1:N+NP]), assumed to be a maximum distance. Defaults to 2*10^8.
...	Additional arguments, not used at this time.

Details

This function is internal to the 'gendistance' function, but may be useful in manufacturing personalized distance matrices. Phantoms are fake elements that perfectly match all elements. They can be used to discard a certain number of elements.

Value

a matrix or data.frame object

Author(s)

Cole Beck

See Also

gendistance distancematrix

Examples

```
# 5x5 distance matrix
dist.mat <- matrix(c(0,5,10,15,20,5,0,15,25,35,10,15,0,25,40,15,25,25,0,15,20,35,40,15,0), nrow=5)
# add one phantom element
dm.ph <- make.phantoms(dist.mat, 1)
# create distancematrix object
distancematrix(dm.ph)
# add three phantoms
make.phantoms(dist.mat, 3)
```

---

nonbimatch *Nonbipartite Matching*

Description

The nonbinmatch function creates the set of pairwise matches that minimizes the sum of distances between the pairs.

Usage

```
nonbimatch(mdm, threshold = NA, precision = 6, ...)
```

Arguments

mdm	A distancematrix object. See the distancematrix function.
threshold	An integer value, indicating the distance needed to create chameleon matches.
precision	The largest value in the matrix will have at most this many digits. The default value is six.
...	Additional arguments, these are not used.

Details

The nonbinmatch function calls the Fortran code (Derigs) and set of pairwise matches that minimizes the sum of distances between the pairs.

Value

matches	data.frame containing matches
halves	data.frame containing each match
total	sum of the distances across all pairs
mean	mean distance for each pair

Author(s)

Cole Beck

See Also

distancematrix

Examples

```
plainmatrix<-as.matrix(dist(sample(1:25, 8, replace=TRUE)))
diag(plainmatrix) <- 99999  # setting diagonal to an infinite distance for
                            # pedagogical reasons (the diagonal may be left
                            # as zero)
mdm<-distancematrix(plainmatrix)
res<-nonbimatch(mdm)
```

---

qom *Quality of Match*

Description

Quality of matches show how well matched pairs differ. For each variable the average distance is generated. Each item in a pair is assigned a group and after several iterations the quantile of these average distances is returned.

Usage

```
qom(covariate, matches, iterations=10000, probs=NA, use.se=FALSE, all.vals=FALSE, seed=101, ...)
```

Arguments

covariate	A data.frame object.
matches	A data.frame object. Contains information on how to match the covariate data set.
iterations	An integer. Number of iterations to run, defaults to 10,000.
probs	A numeric vector. Probabilities to pass to the quantile function.
use.se	A logical value. Determines if the standard error should be computed. Default value of FALSE.
all.vals	A logical value. Determines if false matches should be included in comparison. Default value of FALSE.
seed	Seed provided for random-number generation. Default value of 101.
...	Additional arguments, not used at the moment.

Details

This fuction is useful for determining the effectiveness of your weights (when generating a distance matrix). Weighting a variable more will lower the average distance, but it could penalize the distance of the other variables. Calculating the standard error requires calling 'hdquantile' from **Hmisc**. The quantiles may be slighly different when using 'hdquantile'.

Value

a list object containing elements with quality of match information

q	data.frame with quantiles for each covariate
se	data.frame with standard error for each covariate

Author(s)

Cole Beck

Examples

```
df <- data.frame(id=LETTERS[1:25], val1=rnorm(25), val2=rnorm(25))
df.dist <- gendistance(df, idcol=1)
df.mdm <- distancematrix(df.dist)
df.match <- nonbimatch(df.mdm)
qom(df.dist$cov, df.match$matches)
```

quantile *Quantile for upper-triangular values in distance matrix*

Description

Extend the stats quantile function for handling distancematrix objects.

Usage

```
quantile(x, ...)
```

Arguments

x	A distancematrix object.
...	Additional arguments, passed to stats:::quantile.

Details

The upper.triangular values of the distance matrix object are passed to the quantile funtion.

Value

numeric vector of quantiles corresponding to the given probabilities

Author(s)

Cole Beck

Examples

```
plainmatrix<-as.matrix(dist(sample(1:25, 8, replace=TRUE)))
mdm<-distancematrix(plainmatrix)
quantile(mdm, probs=c(0.0, 0.25, 0.50, 0.75, 1.00))
```

scalar.dist *Calculate scalar distance*

Description

Calculate the scalar distance between elements of a matrix.

Usage

```
scalar.dist(x, ...)
```

Arguments

x	A vector of numeric values.
...	Additional arguments, not used at this time.

Details

Take the absolute difference between all elements in a vector, and return a matrix of the distances.

Value

a matrix object

Author(s)

Cole Beck

Examples

```
scalar.dist(1:10)
```

Index