

# Package ‘StatMatch’

April 4, 2012

**Type** Package

**Title** Statistical Matching

**Version** 1.0.5

**Date** 2012-04-03

**Author** Marcello D’Orazio

**Maintainer** Marcello D’Orazio <madorazi@istat.it>

**Depends** R (>= 2.7.0), proxy, lpSolve, survey

**Suggests** optmatch, SDaA, simPopulation, MASS

**Description** The package StatMatch provides some R functions to perform statistical matching, i.e. the integration of two data sources referred to the same target population which share a number of common variables. Some functions can also be used to impute missing values in data sets through hot deck imputation methods. Methods to perform statistical matching when dealing with data from complex sample surveys are available too.

**License** EUPL

**Repository** CRAN

**Date/Publication** 2012-04-04 07:57:46

## R topics documented:

StatMatch-package	2
comb.samples	3
create.fused	7
fact2dummy	9
Fbwidths.by.x	10
Frechet.bounds.cat	12
gower.dist	16

harmonize.x . . . . .	18
mahalanobis.dist . . . . .	23
maximum.dist . . . . .	25
mixed.mtc . . . . .	26
NND.hotdeck . . . . .	31
RANDwNND.hotdeck . . . . .	36
rankNND.hotdeck . . . . .	40

<b>Index</b>	<b>44</b>
--------------	-----------

---

StatMatch-package	<i>Statistical Matching</i>
-------------------	-----------------------------

---

## Description

The package StatMatch provides some R functions to perform statistical matching, i.e. the integration of two data sources referred to the same target population which share a number of common variables. Some functions can also be used to impute missing values in data sets through hot deck imputation methods. Methods to perform statistical matching when dealing with data from complex sample surveys are available too.

## Details

Package:	StatMatch
Type:	Package
Version:	1.0.3
Date:	2011-10-06
License:	EUPL

## Author(s)

Marcello D'Orazio  
 Maintainer: Marcello D'Orazio <madorazi@istat.it>

## References

D'Orazio M., Di Zio M., Scanu M. (2006) *Statistical Matching, Theory and Practice*. Wiley, Chichester.

**Description**

This function permits to cross-tabulate two categorical variables, Y and Z, observed separately in two independent surveys (Y is collected in survey A and Z is collected in survey B) carried out on the same target population. The two surveys share a number of common variables X. When it is available a third survey C, carried on the same population, in which both Y and Z are collected, these data are used as a source of auxiliary information.

The statistical matching is performed by carrying out calibration of the survey weights, as suggested in Renssen (1998).

**Usage**

```
comb.samples(svy.A, svy.B, svy.C=NULL, y.lab, z.lab, form.x,
             estimation=NULL, ...)
```

**Arguments**

- |       |  |
|-------|--|
| svy.A | A svydesign R object that stores the data collected in the survey A and all the information concerning the sampling design. This type of object can be created by using the function <code>svydesign</code> in the package <b>survey</b> by Lumley. All the variables specified in <code>form.x</code> and by <code>y.lab</code> must be available in <code>svy.A</code> .   |
| svy.B | A svydesign R object that stores the data collected in the survey B and all the information concerning the sampling design. This type of object can be created by using the function <code>svydesign</code> in the package <b>survey</b> by Lumley. All the variables specified in <code>form.x</code> and by <code>z.lab</code> must be available in <code>svy.B</code> .   |
| svy.C | A svydesign R object that stores the data collected in the the survey C and all the information concerning the sampling design. This type of object can be created by using the function <code>svydesign</code> in the package <b>survey</b> by Lumley.<br>When <code>svy.C=NULL</code> (default), i.e. no auxiliary information is available, the function returns an estimate of the contingency table of Y vs. Z under the Conditional Independence assumption (CIA) (see Details for major information).<br>When additional auxiliary information ( <code>svy.C</code> ) is available, if <code>estimation="incomplete"</code> then <code>svy.C</code> must contain at least <code>y.lab</code> and <code>z.lab</code> variables. On the contrary, when <code>estimation="synthetic"</code> all the variables specified in <code>form.x</code> , by <code>y.lab</code> and by <code>z.lab</code> must be available in <code>svy.C</code> . |
| y.lab | A string providing the name of the Y variable, collected in survey A and in survey C (if available). The Y variable must be a categorical variable (factor or integer in R).   |
| z.lab | A string providing the name of the Z variable collected in survey B and in survey C (if available). The Z variable must be a categorical variable (factor or integer in R).  |

form.x	<p>A R formula specifying which of the <b>X</b> variables, collected in all the surveys, have to be considered, and how have to be considered in combining samples. For instance</p> <p>form.x=<math>\sim</math>x1+x2 means that the variables x1 and x2 have to be considered separately without taking into account their cross-tabulation; just their marginal distribution is considered. In order to skip the Intercept the formula has to be written in the following manner: =<math>\sim</math>x1+x2-1.</p> <p>When dealing with categorical variables, if the formula is written as form.x=<math>\sim</math>x1 : x2-1, it means that the the joint distribution of the two variables (table of x1 vs. x2) has to be considered.</p> <p>To better understand the usage of form.x see <a href="#">model.matrix</a> (see also <a href="#">formula</a>).</p> <p>Due to weights calibration features, it is preferable to work with categorical <b>X</b> variables. In some cases, procedure may be successful when a single continuous variable is considered jointly with one or more categorical variables, but in most of the cases, it may be necessary to categorize the continuous variable (see Details).</p>
estimation	<p>A character string that identifies the method to be used to estimate the table of Y vs. Z when data from survey C are available. As suggested in Renssen (1998), two alternative methods are available: (i) Incomplete Two-Way Stratification (estimation="incomplete", or estimation="ITWS" the default one) and (ii) Synthetic Two-Way Stratification (estimation="synthetic" or estimation="STWS"). In the first case (estimation="incomplete") only Y and Z variables must be available in svy.C. On the contrary, when estimation="synthetic" the survey C must contain all the <b>X</b> variables specified via form.x, the Y variable and the Z variable. See Details for further information.</p>
...	<p>Further arguments that may be necessary for calibration. In particular, the argument calfun allows to specify the calibration function. Three alternatives are available:</p> <ul style="list-style-type: none"> <li>(i) calfun="linear" for linear calibration (default);</li> <li>(ii) calfun="raking" to rake the survey weights; and</li> <li>(iii) calfun="logit" for logit calibration. See <a href="#">calibrate</a> for major details.</li> </ul> <p>Note that when calfun="linear" there is the chance of having negative calibrated weights. This drawback can be avoided by requiring that the final weights lie in a given interval specified via the argument bounds of the function <a href="#">calibrate</a>. Generally speaking, in sample surveys one expects that weights are greater than or equal to 1, i.e. bounds=c(1, Inf).</p> <p>The number of iterations used in calibration can be modified by using the argument maxit (by default maxit=50).</p> <p>See <a href="#">calibrate</a> for further details.</p>

## Details

This function estimates the contingency table of Y vs. Z by performing the calibration of the survey weights. In practice the estimation is carried out on data in survey C by exploiting all the information from surveys A and B. When survey C is not available the table of Y vs. Z is estimated

under the assumption of Conditional Independence (CIA), i.e.  $p(Y, Z) = p(Y|\mathbf{X}) \times p(Z|\mathbf{X}) \times p(\mathbf{X})$ .

When data from survey C are available (Renssen, 1998), the table of Y vs. Z can be estimated by: Incomplete Two-Way Stratification (ITWS) or Synthetic Two-Way Stratification (STWS). In the first case (ITWS) the weights of the units in survey C are calibrated so that the new weights allow to reproduce the marginal distributions of Y variable estimated on survey A, and that of Z estimated on survey B. Note that the distribution of the X variables in survey A and in survey B, must be harmonized before performing ITWS (see [harmonize.x](#)). The Synthetic Two-Way Stratification allows to estimate the table of Y vs. Z by considering also the X variables observed in C. This method consists in correcting the table of Y vs. Z estimated under the CIA by according to the relationship between Y and Z observed in survey C (for major details see Renssen, 1998).

### Value

A R list with the results of the calibration procedure according to the input arguments. When `svy.C=NULL` the list will contain just two components `yz.CIA` (Y vs. Z estimated under the CIA) and `call` (the call to the function). On the contrary, when data from `svy.C` are available, the following components will be available:

<code>yz.CIA</code>	The table of Y vs. Z estimated under the Conditional Independence Assumption (CIA).
<code>cal.C</code>	The survey object <code>svy.C</code> after the calibration.
<code>yz.est</code>	The table of Y vs. Z estimated under the method specified via estimation argument.
<code>call</code>	Stores the call to this function with all the values specified for the various arguments ( <code>call=match.call()</code> ).

### Author(s)

Marcello D’Orazio <[madorazi@istat.it](mailto:madorazi@istat.it)>

### References

- D’Orazio, M., Di Zio, M. and Scanu, M. (2006). *Statistical Matching: Theory and Practice*. Wiley, Chichester.
- Renssen, R.H. (1998) “Use of Statistical Matching Techniques in Calibration Estimation”. *Survey Methodology*, **24**, pp. 171–183.

### See Also

[calibrate](#), [svydesign](#), [harmonize.x](#)

### Examples

```
data(quine, package="MASS") #loads quine from MASS
str(quine)
quine$c.Days <- cut(quine$Days, c(-1, seq(0,20,10),100))
table(quine$c.Days)
```

```

# split quine in two subsets
set.seed(124)
lab.A <- sample(nrow(quine), 70, replace=TRUE)
quine.A <- quine[lab.A, c("Eth", "Sex", "Age", "Lrn")]
quine.B <- quine[-lab.A, c("Eth", "Sex", "Age", "c.Days")]

# create svydesign objects
require(survey)
quine.A$f <- 70/nrow(quine) # sampling fraction
quine.B$f <- (nrow(quine)-70)/nrow(quine)
svy.qA <- svydesign(~1, fpc=~f, data=quine.A)
svy.qB <- svydesign(~1, fpc=~f, data=quine.B)

# Harmonization wrt the joint distribution
# of ('Sex' x 'Age' x 'Eth')

# vector of population total known
# estimated from the full data set
# note the formula!
tot.m <- colSums(model.matrix(~Eth:Sex:Age-1, data=quine))
tot.m

out.hz <- harmonize.x(svy.A=svy.qA, svy.B=svy.qB, x.tot=tot.m,
                    form.x=~Eth:Sex:Age-1, cal.method="linear")

# estimation of 'Lrn' vs. 'c.Days' under the CIA

svy.qA.h <- out.hz$cal.A
svy.qB.h <- out.hz$cal.B

out.1 <- comb.samples(svy.A=svy.qA.h, svy.B=svy.qB.h,
                    svy.C=NULL, y.lab="Lrn", z.lab="c.Days",
                    form.x=~Eth:Sex:Age-1)

out.1$yz.CIA
addmargins(out.1$yz.CIA)

#
# incomplete two-way stratification

# select a sample C from quine
# and define a survey object

set.seed(4321)
lab.C <- sample(nrow(quine), 50, replace=TRUE)
quine.C <- quine[lab.C, c("Lrn", "c.Days")]
quine.C$f <- 50/nrow(quine) # sampling fraction
svy.qC <- svydesign(~1, fpc=~f, data=quine.C)

```

```

out.2 <- comb.samples(svy.A=svy.qA.h, svy.B=svy.qB.h,
  svy.C=svy.qC, y.lab="Lrn", z.lab="c.Days",
  form.x=~Eth:Sex:Age-1, estimation="incomplete",
  calfun="linear", maxit=100)

summary(weights(out.2$cal.C))
out.2$yz.est # estimated table of 'Lrn' vs. 'c.Days'
# difference wrt the table 'Lrn' vs. 'c.Days' under CIA
addmargins(out.2$yz.est)-addmargins(out.2$yz.CIA)

# synthetic two-way stratification

quine.C <- quine[lab.C, ]
quine.C$f <- 50/nrow(quine) # sampling fraction
svy.qC <- svydesign(~1, fpc=~f, data=quine.C)

out.3 <- comb.samples(svy.A=svy.qA.h, svy.B=svy.qB.h,
  svy.C=svy.qC, y.lab="Lrn", z.lab="c.Days",
  form.x=~Eth:Sex:Age-1, estimation="synthetic",
  calfun="linear", bounds=c(.5, Inf), maxit=100)

summary(weights(out.3$cal.C))

out.3$yz.est # estimated table of 'Lrn' vs. 'c.Days'
# difference wrt the table of 'Lrn' vs. 'c.Days' under CIA
addmargins(out.3$yz.est)-addmargins(out.3$yz.CIA)
# diff wrt the table of 'Lrn' vs. 'c.Days' under incomplete 2ws
addmargins(out.3$yz.est)-addmargins(out.2$yz.CIA)

```

---

```
create.fused
```

```
Creates a matched (synthetic) dataset
```

---

## Description

Creates a *synthetic* data frame after the statistical matching of two data sources at *micro* level.

## Usage

```
create.fused(data.rec, data.don, mtc.ids,
  z.vars, dup.x=FALSE, match.vars=NULL)
```

## Arguments

data.rec	A matrix or data frame that plays the role of <i>recipient</i> in the statistical matching application.
data.don	A matrix or data frame that that plays the role of <i>donor</i> in the statistical matching application.

mtc.ids	A matrix with two columns. Each row must contain the name or the index of the recipient record (row) in data.don and the name or the index of the corresponding donor record (row) in data.don. Note that this type of matrix is returned by the functions <a href="#">NND.hotdeck</a> , <a href="#">RANDwNND.hotdeck</a> , <a href="#">rankNND.hotdeck</a> , and <a href="#">mixed.mtc</a> .
z.vars	A character vector with the names of the variables available only in data.don that should be “donated” to data.rec.
dup.x	Logical. When TRUE the values of the matching variables in data.don are also “donated” to data.rec. The names of the matching variables have to be specified with the argument match.vars. To avoid confusion, the matching variables added to data.rec are renamed by adding the suffix “don”. By default dup.x=FALSE.
match.vars	A character vector with the names of the matching variables. It has to be specified only when dup.x=TRUE.

### Details

This function allows to create the synthetic (or fused) data set after the application of a statistical matching in a *micro* framework. For details see D’Orazio *et al.* (2006).

### Value

The data frame data.rec with the z.vars filled in and, when dup.x=TRUE, with the values of the matching variables match.vars observed on the donor records.

### Author(s)

Marcello D’Orazio <madorazi@istat.it>

### References

D’Orazio, M., Di Zio, M. and Scanu, M. (2006). *Statistical Matching: Theory and Practice*. Wiley, Chichester.

### See Also

[NND.hotdeck](#) [RANDwNND.hotdeck](#) [rankNND.hotdeck](#)

### Examples

```
lab <- c(1:15, 51:65, 101:115)
iris.rec <- iris[lab, c(1:3,5)] # recipient data.frame
iris.don <- iris[-lab, c(1:2,4:5)] # donor data.frame

# Now iris.rec and iris.don have the variables
# "Sepal.Length", "Sepal.Width" and "Species"
# in common.
# "Petal.Length" is available only in iris.rec
# "Petal.Width" is available only in iris.don
```

```
# find the closest donors using NND hot deck;
# distances are computed on "Sepal.Length" and "Sepal.Width"

out.NND <- NND.hotdeck(data.rec=iris.rec, data.don=iris.don,
  match.vars=c("Sepal.Length", "Sepal.Width"),
  don.class="Species")

# create synthetic data.set, without the
# duplication of the matching variables

fused.0 <- create.fused(data.rec=iris.rec, data.don=iris.don,
  mtc.ids=out.NND$mtc.ids, z.vars="Petal.Width")

# create synthetic data.set, with the "duplication"
# of the matching variables

fused.1 <- create.fused(data.rec=iris.rec, data.don=iris.don,
  mtc.ids=out.NND$mtc.ids, z.vars="Petal.Width",
  dup.x=TRUE, match.vars=c("Sepal.Length", "Sepal.Width"))
```

---

fact2dummy

*Transforms a categorical variable in a set of dummy variables*

---

## Description

Transforms a factor or more factors contained in a data frame in a set of dummy variables.

## Usage

```
fact2dummy(data, all=TRUE, lab="x")
```

## Arguments

data	A factor or a data frame that contains one or more factors (columns whose class is “factor” or “ordered”) that have to be substituted by dummy variables.
all	Logical. When all=TRUE (default) the output matrix will contain as many dummy variables as the number of the levels of the factor variable. On the contrary, when all=FALSE, the dummy variable related to the last level of the factor is dropped.
lab	A character string with the name of the variable to be pasted with its levels. This is used only when data is a factor. By default it is set to “x”.

## Details

This function substitutes categorical variables in the input data frame (columns whose class is “factor” or “ordered”) with the corresponding dummy variables.

**Value**

A matrix with the dummy variables instead of initial factor variables.

**Author(s)**

Marcello D'Orazio <madorazi@istat.it>

**See Also**

[gower.dist](#)

**Examples**

```
x <- runif(5)
y <- factor(c(1,2,1,2,2))
z <- ordered(c(1,2,3,2,2))
xyz <- data.frame(x,y,z)
fact2dummy(xyz)

fact2dummy(xyz, all=FALSE)

#example with iris data frame
str(iris)
ir.mat <- fact2dummy(iris)
```

---

Fbwidths.by.x

*Computes the Frechet bounds of cells in a contingency table by considering all the possible subsets of the common variables.*

---

**Description**

This function permits to compute the bounds for cell probabilities in the contingency table Y vs. Z starting from the marginal tables (X vs. Y), (X vs. Z) and the joint distribution of the X variables, by considering all the possible subsets of the X variables. In this manner it is possible to identify which subset of the X variables produces the major reduction of the uncertainty measured in terms of the width of the bounds.

**Usage**

```
Fbwidths.by.x(tab.x, tab.xy, tab.xz)
```

**Arguments**

- `tab.x` A R table crossing the **X** variables. This table must be obtained by using the function `xtabs` or `table`, e.g.  
`tab.x <- xtabs(~x1+x2+x3, data=data.all).`
- `tab.xy` A R table of **X** vs. **Y** variable. This table must be obtained by using the function `xtabs` or `table`, e.g.  
`table.xy <- xtabs(~x1+x2+x3+y, data=data.A).`  
 A single categorical **Y** variables is allowed. One or more categorical variables can be considered as **X** variables (common variables). Obviously, the same **X** variables in `tab.x` must be available in `tab.xy`. Moreover, it is assumed that the joint distribution of the **X** variables computed from `tab.xy` is equal to `tab.x`; a warning is produced if this is not true.
- `tab.xz` A R table of **X** vs. **Z** variable. This table must be obtained by using the function `xtabs` or `table`, e.g.  
`tab.xz <- xtabs(~x1+x2+x3+z, data=data.B).`  
 A single categorical **Z** variable is allowed. One or more categorical variables can be considered as **X** variables (common variables). The same **X** variables in `tab.x` must be available in `tab.xz`. Moreover, it is assumed that the joint distribution of the **X** variables computed from `tab.xz` is equal to `tab.x`; a warning is produced if this is not true.

**Details**

This function permits to compute the Frechet bounds for the frequencies in the contingency table of **Y** vs. **Z**, starting from the conditional distributions  $P(\mathbf{Y}|\mathbf{X})$  and  $P(\mathbf{Z}|\mathbf{X})$  (for details see [Frechet.bounds.cat](#)), by considering all the possible subsets of the **X** variables. In this manner it is possible to identify the subset of the **X** variables, with highest association with both **Y** and **Z**, that permits to reduce the uncertainty concerning the distribution of **Y** vs. **Z**. The reduction of the uncertainty is measured in terms of the average of the widths of the bounds for the cells in the table of **Y** vs. **Z**:

$$\bar{d} = \frac{1}{J \times K} \sum_{j,k} (p_{j,k}^{up} - p_{j,k}^{low})$$

For details see [Frechet.bounds.cat](#).

**Value**

A list with the estimated estimated bounds for the cells in the table of **Y** vs. **Z** for each possible subset of the **X** variables. The final component of the list summarizes the average width of the bounds for each subset of the **X** variables.

**Author(s)**

Marcello D'Orazio <madorazi@istat.it>

## References

Ballin, M., D’Orazio, M., Di Zio, M., Scanu, M. and Torelli, N. (2009) “Statistical Matching of Two Surveys with a Common Subset”. *Working Paper*, **124**. Dip. Scienze Economiche e Statistiche, Univ. di Trieste, Trieste.

D’Orazio, M., Di Zio, M. and Scanu, M. (2006). *Statistical Matching: Theory and Practice*. Wiley, Chichester.

## See Also

[Frechet.bounds.cat](#), [harmonize.x](#)

## Examples

```
data(quine, package="MASS") #loads quine from MASS
str(quine)
quine$c.Days <- cut(quine$Days, c(-1, seq(0,50,10),100))
table(quine$c.Days)

# split quine in two subsets
set.seed(4567)
lab.A <- sample(nrow(quine), 70, replace=TRUE)
quine.A <- quine[lab.A, 1:4]
quine.B <- quine[-lab.A, c(1:3,6)]

# compute the tables required by Fwidths.by.x()
freq.x <- xtabs(~Eth+Sex+Age, data=quine.A)
freq.xy <- xtabs(~Eth+Sex+Age+Lrn, data=quine.A)
freq.xz <- xtabs(~Eth+Sex+Age+c.Days, data=quine.B)

# apply Fwidths.by.x()
bounds.yz <- Fwidths.by.x(tab.x=freq.x, tab.xy=freq.xy,
  tab.xz=freq.xz)
bounds.yz$av.widths
#
# to view a plot of the av.widths run also the
#following code
#barplot(bounds.yz$av.widths$av.width,
#        names.arg=row.names(bounds.yz$av.widths), las=2,
#        cex.names=0.75)
```

---

Frechet.bounds.cat      *Frechet bounds of cells in a contingency table*

---

## Description

This function permits to derive the bounds for cell probabilities of the table Y vs. Z starting from the marginal tables (X vs. Y), (X vs. Z) and the joint distribution of the X variables.

**Usage**

```
Frechet.bounds.cat(tab.x, tab.xy, tab.xz, print.f="tables")
```

**Arguments**

tab.x	A R table crossing the <b>X</b> variables. This table must be obtained by using the function <code>xtabs</code> or <code>table</code> , e.g. <code>tab.x &lt;- xtabs(~x1+x2+x3, data=data.all).</code>
tab.xy	A R table of <b>X</b> vs. <b>Y</b> variable. This table must be obtained by using the function <code>xtabs</code> or <code>table</code> , e.g. <code>table.xy &lt;- xtabs(~x1+x2+x3+y, data=data.A).</code> A single categorical <b>Y</b> variable is allowed. One or more categorical variables can be considered as <b>X</b> variables (common variables). Obviously, the same <b>X</b> variables in <code>tab.x</code> must be available in <code>tab.xy</code> . Moreover, it is assumed that the joint distribution of the <b>X</b> variables computed from <code>tab.xy</code> is equal to <code>tab.x</code> ; a warning appears if this is not true.
tab.xz	A R table of <b>X</b> vs. <b>Z</b> variable. This table must be obtained by using the function <code>xtabs</code> or <code>table</code> , e.g. <code>tab.xz &lt;- xtabs(~x1+x2+x3+z, data=data.B).</code> A single categorical <b>Z</b> variable is allowed. One or more categorical variables can be considered as <b>X</b> variables (common variables). The same <b>X</b> variables in <code>tab.x</code> must be available in <code>tab.xz</code> . Moreover, it is assumed that the joint distribution of the <b>X</b> variables computed from <code>tab.xz</code> is equal to <code>tab.x</code> ; a warning appears if this is not true.
print.f	A string specifying the data structure of the output. When <code>print.f="tables"</code> (default) all the results will be saved as tables in a list. On the contrary, if <code>print.f="data.frame"</code> , all results will be saved as columns of a <code>data.frame</code> .

**Details**

This function permits to compute the Frechet bounds for the relative frequencies in the contingency table of **Y** vs.**Z**, starting from the distributions  $P(Y|X)$ ,  $P(Z|X)$  and  $P(X)$ . The bounds for the relative frequencies  $p_{j,k}$  in the table **Y** vs. **Z** are:

$$p_{YZ}^{low}(j, k) = \sum_i p_X(i) \max(0; p_{Y|X}(j|i) + p_{Z|X}(k|i) - 1)$$

$$p_{YZ}^{up}(j, k) = \sum_i p_X(i) \min(p_{Y|X}(j|i); p_{Z|X}(k|i))$$

The relative frequencies  $p_X(i) = n_i/n$  are computed from the frequencies in `tab.x`; the relative frequencies  $p_{Y|X}(j|i) = n_{ij}/n_{i\bullet}$  are computed from the `tab.xy`, finally,  $p_{Z|X}(k|i) = n_{ik}/n_{i\bullet}$  are derived from `tab.xy`.

It is assumed that the marginal distribution of the **X** variables is the same in all the input tables: `tab.x`, `tab.xy` and `tab.xz`. If this is not true a warning message will appear.

Note that the cells bounds for the relative frequencies in the contingency table of  $Y$  vs.  $Z$  are computed also without considering the  $X$  variables:

$$\max\{0; p_Y(j) + p_Z(k) - 1\} \leq p_{YZ}(j, k) \leq \min\{p_Y(j); p_Z(k)\}$$

Finally, the contingency table of  $Y$  vs.  $Z$  estimated under the Conditional Independence Assumption (CIA) is obtained by considering:

$$p_{YZ}(j, k) = p_{Y|X}(j|i) \times p_{Z|X}(k|i) \times p_X(i).$$

### Value

When `print.f="tables"` (default) a list with the following tables:

<code>low.u</code>	The estimated lower bounds for the relative frequencies in the table $Y$ vs. $Z$ without conditioning on the $X$ variables.
<code>up.u</code>	The estimated upper bounds for the relative frequencies in the table $Y$ vs. $Z$ without conditioning on the $X$ variables.
<code>CIA</code>	The estimated relative frequencies in the table $Y$ vs. $Z$ under the Conditional Independence Assumption (CIA).
<code>low.cx</code>	The estimated lower bounds for the relative frequencies in the table $Y$ vs. $Z$ when conditioning on the $X$ variables.
<code>up.cx</code>	The estimated upper bounds for the relative frequencies in the table $Y$ vs. $Z$ when conditioning on the $X$ variables.

When `print.f="data.frame"` the estimated tables are saved as columns of a data.frame.

### Author(s)

Marcello D'Orazio <madorazi@istat.it>

### References

Ballin, M., D'Orazio, M., Di Zio, M., Scanu, M. and Torelli, N. (2009) "Statistical Matching of Two Surveys with a Common Subset". *Working Paper*, **124**. Dip. Scienze Economiche e Statistiche, Univ. di Trieste, Trieste.

D'Orazio, M., Di Zio, M. and Scanu, M. (2006). *Statistical Matching: Theory and Practice*. Wiley, Chichester.

### See Also

[Fbwidths.by.x](#), [harmonize.x](#)

**Examples**

```

data(quine, package="MASS") #loads quine from MASS
str(quine)

# split quine in two subsets
set.seed(765)
lab.A <- sample(nrow(quine), 70, replace=TRUE)
quine.A <- quine[lab.A, 1:3]
quine.B <- quine[-lab.A, 2:4]

# compute the tables required by Frechet.bounds.cat()
freq.x <- xtabs(~Sex+Age, data=quine.A)
freq.xy <- xtabs(~Sex+Age+Eth, data=quine.A)
freq.xz <- xtabs(~Sex+Age+Lrn, data=quine.B)

# apply Frechet.bounds.cat()
bounds.yz <- Frechet.bounds.cat(tab.x=freq.x, tab.xy=freq.xy,
                               tab.xz=freq.xz, print.f="data.frame")
bounds.yz

# harmonize distr. of Sex vs. Age before applying
# Frechet.bounds.cat()

quine.A$f <- 70/nrow(quine) # sampling fraction
quine.B$f <- (nrow(quine)-70)/nrow(quine)

# derive the table of Sex vs. Age related to the whole data set
tot.sex.age <- xtabs(~Sex+Age, data=quine)
tot.sex.age

# use hamonize.x() to harmonize the Sex vs. Age between
# quine.A and quine.B

# create svydesign objects
svy.qA <- svydesign(~1, fpc=~f, data=quine.A)
svy.qB <- svydesign(~1, fpc=~f, data=quine.B)

# apply harmonize.x using poststratification
out.hz <- harmonize.x(svy.A=svy.qA, svy.B=svy.qB, form.x=~Sex+Age,
                     cal.method="poststratify")

# compute the new tables required by Frechet.bounds.cat()
freq.x <- xtabs(out.hz$weights.A~Sex+Age, data=quine.A)
freq.xy <- xtabs(out.hz$weights.A~Sex+Age+Eth, data=quine.A)
freq.xz <- xtabs(out.hz$weights.B~Sex+Age+Lrn, data=quine.B)

# apply Frechet.bounds.cat()
bounds.yz <- Frechet.bounds.cat(tab.x=freq.x, tab.xy=freq.xy,
                               tab.xz=freq.xz, print.f="data.frame")
bounds.yz

```

---

`gower.dist`                      *Computes the Gower's Distance*

---

### Description

This function computes the Gower's distance (dissimilarity) among units in a dataset or among observations in two distinct datasets.

### Usage

```
gower.dist(data.x, data.y=data.x, rngs=NULL, KR.corr=TRUE)
```

### Arguments

<code>data.x</code>	<p>A matrix or a data frame containing variables that should be used in the computation of the distance.</p> <p>Columns of mode <code>numeric</code> will be considered as interval scaled variables; columns of mode <code>character</code> or class <code>factor</code> will be considered as categorical nominal variables; columns of class <code>ordered</code> will be considered as categorical ordinal variables and, columns of mode <code>logical</code> will be considered as binary asymmetric variables (see <a href="#">Details</a> for further information).</p> <p>Missing values (NA) are allowed.</p> <p>If only <code>data.x</code> is supplied, the dissimilarities between rows of <code>data.x</code> will be computed.</p>
<code>data.y</code>	<p>A numeric matrix or data frame with the same variables, of the same type, as those in <code>data.x</code>. Dissimilarities between rows of <code>data.x</code> and rows of <code>data.y</code> will be computed. If not provided, by default it is assumed equal to <code>data.x</code> and only dissimilarities between rows of <code>data.x</code> will be computed.</p>
<code>rngs</code>	<p>A vector with the ranges to scale the variables. Its length must be equal to number of variables in <code>data.x</code>. In correspondence of nonnumeric variables, just put 1 or NA. When <code>rngs=NULL</code> (default) the range of a numeric variable is estimated by jointly considering the values for the variable in <code>data.x</code> and those in <code>data.y</code>. Therefore, assuming <code>rngs=NULL</code>, if a variable "X1" is considered:</p> <pre>rngs["X1"] &lt;- max(data.x[, "X1"], data.y[, "X1"]) -                min(data.x[, "X1"], data.y[, "X1"])</pre>
<code>KR.corr</code>	<p>When TRUE (default) the extension of the Gower's dissimilarity measure proposed by Kaufman and Rousseeuw (1990) is used. Otherwise, when <code>KR.corr=FALSE</code>, the Gower's (1971) formula is considered.</p>

### Details

This function computes distances among records when variables of different type (categorical and continuous) have been observed. In order to handle different types of variables, the Gower's dissimilarity coefficient (Gower, 1971) is used. By default (`KR.corr=TRUE`) the Kaufman and Rousseeuw (1990) extension of the Gower's dissimilarity coefficient is used.

The final dissimilarity between the  $i$ th and  $j$ th unit is obtained as a weighted sum of dissimilarities for each variable:

$$d(i, j) = \frac{\sum_k \delta_{ijk} d_{ijk}}{\sum_k \delta_{ijk}}$$

In particular,  $d_{ijk}$  represents the distance between the  $i$ th and  $j$ th unit computed considering the  $k$ th variable. It depends on the nature of the variable:

- logical columns are considered as asymmetric binary variables, for such case  $d_{ijk} = 0$  if  $x_{ik} = x_{jk} = \text{TRUE}$ , 1 otherwise;
- factor or character columns are considered as categorical nominal variables and  $d_{ijk} = 0$  if  $x_{ik} = x_{jk}$ , 1 otherwise;
- numeric columns are considered as interval-scaled variables and

$$d_{ijk} = \frac{|x_{ik} - x_{jk}|}{R_k}$$

being  $R_k$  the range of the  $k$ th variable. The range is the one supplied with the argument `rngs` (`rngs[k]`) or the one computed on available data (when `rngs=NULL`);

- ordered columns are considered as categorical ordinal variables and the values are substituted with the corresponding position index,  $r_{ik}$  in the factor levels. When `KR.corr=FALSE` these position indexes (that are different from the output of the R function `rank`) are transformed in the following manner

$$z_{ik} = \frac{(r_{ik} - 1)}{\max(r_{ik}) - 1}$$

These new values,  $z_{ik}$ , are treated as observations of an interval scaled variable.

As far as the weight  $\delta_{ijk}$  is concerned:

- $\delta_{ijk} = 0$  if  $x_{ik} = \text{NA}$  or  $x_{jk} = \text{NA}$ ;
- $\delta_{ijk} = 0$  if the variable is asymmetric binary and  $x_{ik} = x_{jk} = 0$  or  $x_{ik} = x_{jk} = \text{FALSE}$ ;
- $\delta_{ijk} = 1$  in all the other cases.

In practice, NAs and couple of cases with  $x_{ik} = x_{jk} = \text{FALSE}$  do not contribute to distance computation.

## Value

A matrix object with distances among rows of `data.x` and those of `data.y`.

## Author(s)

Marcello D'Orazio <madorazi@istat.it>

## References

Gower, J. C. (1971), "A general coefficient of similarity and some of its properties". *Biometrics*, **27**, 623–637.

Kaufman, L. and Rousseeuw, P.J. (1990), *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York.

**See Also**[daisy](#), [dist](#)**Examples**

```
x1 <- as.logical(rbinom(10,1,0.5))
x2 <- sample(letters, 10, replace=TRUE)
x3 <- rnorm(10)
x4 <- ordered(cut(x3, -4:4, include.lowest=TRUE))
xx <- data.frame(x1, x2, x3, x4, stringsAsFactors = FALSE)

# matrix of distances among observations in xx
gower.dist(xx)

# matrix of distances among first obs. in xx
# and the remaining ones
gower.dist(data.x=xx[1:3,], data.y=xx[4:10,])
```

---

harmonize.x	<i>Harmonizes the marginal (joint) distribution of a set of variables observed independently in two sample surveys on the same target population</i>
-------------	--

---

**Description**

This function permits to harmonize the marginal or the joint distribution of a set of variables observed independently in two sample surveys carried out on the same target population. This harmonization is carried out by using the calibration of the survey weights of the sample units in both the surveys according to the procedure introduced by Renssen (1998).

**Usage**

```
harmonize.x(svy.A, svy.B, form.x, x.tot=NULL,
            cal.method="linear", ...)
```

**Arguments**

svy.A	A svydesign R object that stores the data collected in the the survey A and all the information concerning the sampling design. This type of object can be created by using the function <a href="#">svydesign</a> in the package <b>survey</b> by Lumley.
svy.B	A svydesign R object that stores the data collected in the the survey B and all the information concerning the sampling design. This type of object can be created by using the function <a href="#">svydesign</a> in the package <b>survey</b> by Lumley.

- `form.x` A R formula specifying which of the variables, common to both the surveys, have to be considered, and how have to be considered. For instance `form.x=~x1+x2` means that the marginal distribution of the variables `x1` and `x2` have to be harmonized and there is also an 'Intercept'. In order to skip the intercept the formula has to be written in the following manner `form.x=~x1+x2-1`.
- When dealing with categorical variables, if the formula is written in the following manner `form.x=~x1:x2-1` it means that the harmonization has to be carried out by considering the joint distribution of the two variables (`x1` vs. `x2`). To better understand how `form.x` works see [model.matrix](#) (see also [formula](#)).
- Due to weights calibration features, it is preferable to work with categorical **X** variables. In some cases, the procedure may be successful when a single continuous variable is considered jointly with one or more categorical variables. When dealing with several continuous variable it may be preferable to categorize them.
- `x.tot` A vector or table with known population totals for the **X** variables. A vector is required when `cal.method="linear"` or `cal.method="raking"`. The names and the length of the vector depends on the way it is specified the argument `form.x` (see [model.matrix](#)). On the contrary a contingency table is required when `cal.method="poststratify"` (for details see [postStratify](#)).
- When `x.tot` is not provided (i.e. `x.tot=NULL`) then the vector of totals is estimated as a weighted average of the totals estimated on the two surveys. The weight assigned to the totals estimated from A is  $\lambda = n_A / (n_A + n_B)$ ;  $1 - \lambda$  is the weight assigned to **X** totals estimated from survey B ( $n_A$  and  $n_B$  are the number of units in A and B respectively).
- `cal.method` A string that specifies how the calibration of the weights in `svy.A` and `svy.B` has to be carried out. By default `cal.method="linear"` linear calibration is performed. In particular, the calibration is carried out by mean of the function [calibrate](#) in the package **survey**.
- Alternatively, it is possible to rake the origin survey weights by specifying `cal.method="raking"`. Finally, it is possible to perform a simple post-stratification by setting `cal.method="poststratify"`. Note that in this case the weights adjustments are carried out by considering the function [postStratify](#) in the package **survey**.
- ... Further arguments that may be necessary for calibration or post-stratification. In particular, when `cal.method="linear"` there is the chance of having negative weights. This drawback can be avoided by requiring that the final weights lie in a given interval specified via `bounds` argument (an argument of [calibrate](#)). In sample surveys one expects that weights are greater than or equal to 1, i.e. `bounds=c(1, Inf)`.
- The number of iterations used in calibration can be modified too by using the argument `maxit` (by default `maxit=50`).
- See [calibrate](#) for further details.

## Details

This function harmonizes the totals of the  $\mathbf{X}$  variables, observed in both survey A and survey B, to be equal to given known totals specified via `x.tot`. When these totals are not known (`x.tot=NULL`) they are estimated by combining the estimates derived from the two separate surveys. The harmonization is carried out according to a procedure introduced by Renssen (1998) based on calibration of survey weights (for major details on calibration see Sarndal and Lundstrom, 2005). The procedure is particularly suited to deal with categorical  $\mathbf{X}$  variables, in this case it permits to harmonize the joint or the marginal distribution of the categorical variables being considered. Note that an incomplete crossing of the  $\mathbf{X}$  variables can be considered: i.e. harmonisation wrt to the joint distribution of  $X_1 \times X_2$  and the marginal distribution of  $X_3$ .

The calibration procedure may not produce the final result due to convergence problems. In this case an error message appears. In order to reach convergence it may be necessary to launch the procedure with less constraints (i.e a reduced number of population totals) by joining adjacent categories or by discarding some variables.

In some limited cases it could be possible to consider both categorical and continuous variables. In this situation it may happen that calibration is not successful. In order to reach convergence it may be necessary to categorize the continuous  $\mathbf{X}$  variables.

Post-stratification is a special case of calibration; all the weights of the units in a given post-stratum are modified so the reproduce the known total for that post-stratum. Post-stratification avoids problems of convergence but, on the other hand it may produce final weights with a higher variability than those derived from the calibration.

## Value

A R list with the results of calibration procedures carried out on survey A and survey B, respectively. In particular the following components will be provided:

<code>cal.A</code>	The survey object <code>svy.A</code> after the calibration; in particular, the weights now are calibrated with respect to the totals of the $\mathbf{X}$ variables.
<code>cal.B</code>	The survey object <code>svy.B</code> after the calibration; in particular, the weights now are calibrated with respect to the totals of the $\mathbf{X}$ variables.
<code>weights.A</code>	The new calibrated weights associated to the the units in <code>svy.A</code> .
<code>weights.B</code>	The new calibrated weights associated to the the units in <code>svy.B</code> .
<code>call</code>	Stores the call to this function with all the values specified for the various arguments ( <code>call=match.call()</code> ).

## Author(s)

Marcello D'Orazio <madorazi@istat.it>

## References

- D'Orazio, M., Di Zio, M. and Scanu, M. (2006). *Statistical Matching: Theory and Practice*. Wiley, Chichester.
- Renssen, R.H. (1998) "Use of Statistical Matching Techniques in Calibration Estimation". *Survey Methodology*, N. 24, pp. 171–183.
- Sarndal, C.E. and Lundstrom, S. (2005) *Estimation in Surveys with Nonresponse*. Wiley, Chichester.

**See Also**

[comb.samples](#), [calibrate](#), [svydesign](#), [postStratify](#),

**Examples**

```

data(quine, package="MASS") #loads quine from MASS
str(quine)

# split quine in two subsets
set.seed(7654)
lab.A <- sample(nrow(quine), 70, replace=TRUE)
quine.A <- quine[lab.A, c("Eth", "Sex", "Age", "Lrn")]
quine.B <- quine[-lab.A, c("Eth", "Sex", "Age", "Days")]

# create svydesign objects
require(survey)
quine.A$f <- 70/nrow(quine) # sampling fraction
quine.B$f <- (nrow(quine)-70)/nrow(quine)
svy.qA <- svydesign(~1, fpc=~f, data=quine.A)
svy.qB <- svydesign(~1, fpc=~f, data=quine.B)

#-----
# example (1)
# Harmonization of the distr. of Sex vs. Age
# usign poststratification

# (1.a) known population totals
# the population total are computed on the full data frame
tot.sex.age <- xtabs(~Sex+Age, data=quine)
tot.sex.age

out.hz <- harmonize.x(svy.A=svy.qA, svy.B=svy.qB, form.x=~Sex+Age,
  x.tot=tot.sex.age, cal.method="poststratify")

tot.A <- xtabs(out.hz$weights.A~Sex+Age, data=quine.A)
tot.B <- xtabs(out.hz$weights.B~Sex+Age, data=quine.B)

tot.sex.age-tot.A
tot.sex.age-tot.B

# (1.b) unknown population totals (x.tot=NULL)
# the population total is estimated by combining totals from the
# two surveys

out.hz <- harmonize.x(svy.A=svy.qA, svy.B=svy.qB, form.x=~Sex+Age,
  x.tot=NULL, cal.method="poststratify")

tot.A <- xtabs(out.hz$weights.A~Sex+Age, data=quine.A)
tot.B <- xtabs(out.hz$weights.B~Sex+Age, data=quine.B)

tot.A

```

```

tot.A-tot.B

#-----
# example (2)
# Harmonization wrt the marginal distribution
# of 'Eth', 'Sex' and 'Age'
# using linear calibration

# (2.a) vector of population total known
# estimated from the full data set
# note the formula! only marginal distribution of the
# variables are considered
tot.m <- colSums(model.matrix(~Eth+Sex+Age-1, data=quine))
tot.m

out.hz <- harmonize.x(svy.A=svy.qA, svy.B=svy.qB, x.tot=tot.m,
                    form.x=~Eth+Sex+Age-1, cal.method="linear")

summary(out.hz$weights.A) #check for negative weights
summary(out.hz$weights.B) #check for negative weights

tot.m
svytable(formula=~Eth, design=out.hz$cal.A)
svytable(formula=~Eth, design=out.hz$cal.B)

svytable(formula=~Sex, design=out.hz$cal.A)
svytable(formula=~Sex, design=out.hz$cal.B)

# Note: margins are equal but joint distributions are not!
svytable(formula=~Sex+Age, design=out.hz$cal.A)
svytable(formula=~Sex+Age, design=out.hz$cal.B)

# (2.b) vector of population total unknown
out.hz <- harmonize.x(svy.A=svy.qA, svy.B=svy.qB, x.tot=NULL,
                    form.x=~Eth+Sex+Age-1, cal.method="linear")
svytable(formula=~Eth, design=out.hz$cal.A)
svytable(formula=~Eth, design=out.hz$cal.B)

svytable(formula=~Sex, design=out.hz$cal.A)
svytable(formula=~Sex, design=out.hz$cal.B)

#-----
# example (3)
# Harmonization wrt the joint distribution of 'Sex' vs. 'Age'
# and the marginal distribution of 'Eth'
# using raking

# vector of population total known
# estimated from the full data set
# note the formula!
tot.m <- colSums(model.matrix(~Eth+(Sex:Age-1)-1, data=quine))
tot.m

```

```

out.hz <- harmonize.x(svy.A=svy.qA, svy.B=svy.qB, x.tot=tot.m,
  form.x=~Eth+(Sex:Age)-1, cal.method="raking")

summary(out.hz$weights.A) #check for negative weights
summary(out.hz$weights.B) #check for negative weights

tot.m
svytable(formula=~Eth, design=out.hz$cal.A)
svytable(formula=~Eth, design=out.hz$cal.B)

svytable(formula=~Sex+Age, design=out.hz$cal.A)
svytable(formula=~Sex+Age, design=out.hz$cal.B)

#-----
# example (4)
# Harmonization wrt the joint distribution
# of ('Sex' x 'Age' x 'Eth')

# vector of population total known
# estimated from the full data set
# note the formula!
tot.m <- colSums(model.matrix(~Eth:Sex:Age-1, data=quine))
tot.m

out.hz <- harmonize.x(svy.A=svy.qA, svy.B=svy.qB, x.tot=tot.m,
  form.x=~Eth:Sex:Age-1, cal.method="linear")

tot.m
svytable(formula=~Eth+Sex+Age, design=out.hz$cal.A)
svytable(formula=~Eth+Sex+Age, design=out.hz$cal.B)

```

---

mahalanobis.dist

*Computes the Mahalanobis Distance*


---

### Description

This function computes the Mahalanobis distance among units in a dataset or among observations in two distinct datasets.

### Usage

```
mahalanobis.dist(data.x, data.y=NULL, vc=NULL)
```

### Arguments

**data.x** A matrix or a data frame containing variables that should be used in the computation of the distance. Only continuous variables are allowed. Missing values (NA) are not allowed.

When only **data.x** is supplied, the distances between rows of **data.x** is computed.

data.y	A numeric matrix or data frame with the same variables, of the same type, as those in data.x (only continuous variables are allowed). Dissimilarities between rows of data.x and rows of data.y will be computed. If not provided, by default it is assumed data.y=data.x and only dissimilarities between rows of data.x will be computed.
vc	Covariance matrix that should be used in distance computation. If it is not supplied (vc=NULL) it is estimated from the input data. In particular, when vc=NULL and only data.x is supplied then the covariance matrix is estimated from data.x (var(data.x)). On the contrary when vc=NULL and both data.x and data.y are available then the covariance matrix is estimated on the joined data sets (i.e. var(rbind(data.x, data.y))).

### Details

This function computes the Mahalanobis distance:

$$d(i, j) = \sqrt{(x_i - x_j)^T S^{-1} (x_i - x_j)}$$

When vc=NULL the covariance matrix S is estimated from the available data (see argument vc for details) otherwise the one supplied via the argument vc is used.

### Value

A matrix object with distances among rows of data.x and those of data.y.

### Author(s)

Marcello D’Orazio <madorazi@istat.it>

### References

Mahalanobis, P C (1936) “On the generalised distance in statistics”. Proceedings of the National Institute of Sciences of India 2, pp. 49-55.

### See Also

[mahalanobis](#)

### Examples

```
md1 <- mahalanobis.dist(iris[1:6,1:4])
md2 <- mahalanobis.dist(data.x=iris[1:6,1:4], data.y=iris[51:60, 1:4])

vv <- var(iris[,1:4])
md1a <- mahalanobis.dist(data.x=iris[1:6,1:4], vc=vv)
md2a <- mahalanobis.dist(data.x=iris[1:6,1:4], data.y=iris[51:60, 1:4], vc=vv)
```

---

maximum.dist	<i>Computes the Maximum Distance</i>
--------------	--------------------------------------

---

### Description

This function computes the Maximum distance (or  $L^\infty$  norm) among units in a dataset or among observations in two distinct datasets.

### Usage

```
maximum.dist(data.x, data.y=data.x, rank=FALSE)
```

### Arguments

data.x	A matrix or a data frame containing variables that should be used in the computation of the distance. Only continuous variables are allowed. Missing values (NA) are not allowed. When only data.x is supplied, the distances between rows of data.x is computed.
data.y	A numeric matrix or data frame with the same variables, of the same type, as those in data.x (only continuous variables are allowed). Dissimilarities between rows of data.x and rows of data.y will be computed. If not provided, by default it is assumed data.y=data.x and only dissimilarities between rows of data.x will be computed.
rank	Logical, when TRUE the original values are substituted by their ranks divided by the number of values plus one (following suggestion in Kovar et al. 1988). This rank transformation permits to remove the effect of different scales on the distance computation. When computing ranks the tied observations assume the average of their position (ties.method = "average" in calling the <a href="#">rank</a> function).

### Details

This function computes the  $L^\infty$  distance also know as minimax distance. In practice the distance among two records is the maximum of the absolute differences among the observed variables:

$$d(i, j) = \max(|x_{1i} - x_{1j}|, |x_{2i} - x_{2j}|, \dots, |x_{Ki} - x_{Kj}|)$$

When rank=TRUE the original values are substituted by their ranks divided by the number of values plus one (following suggestion in Kovar et al. 1988).

### Value

A matrix object with distances among rows of data.x and those of data.y.

**Author(s)**

Marcello D’Orazio <madorazi@istat.it>

**References**

Kovar, J.G., MacMillan, J. and Whitridge, P. (1988). “Overview and strategy for the Generalized Edit and Imputation System”. Statistics Canada, Methodology Branch Working Paper No. BSMD 88-007 E/F.

**See Also**

[rank](#),

**Examples**

```
md1 <- maximum.dist(iris[1:10,1:4])
md2 <- maximum.dist(iris[1:10,1:4], rank=TRUE)

md3 <- maximum.dist(data.x=iris[1:50,1:4], data.y=iris[51:100,1:4])
md4 <- maximum.dist(data.x=iris[1:50,1:4], data.y=iris[51:100,1:4], rank=TRUE)
```

---

mixed.mtc

*Statistical Matching via Mixed Methods*

---

**Description**

This function implements some mixed methods to perform statistical matching between two data sources.

**Usage**

```
mixed.mtc(data.rec, data.don, match.vars, y.rec, z.don, method="ML",
          rho.yz=0, micro=FALSE, constr.alg="lpSolve")
```

**Arguments**

`data.rec` A matrix or data frame that plays the role of *recipient* in the statistical matching application. This data set must contain all variables (columns) that should be used in statistical matching, i.e. the variables called by the arguments `match.vars` and `y.rec`. Note that continuous variables are expected, if there are some categorical variables they are recoded into dummies. Missing values (NA) are not allowed.

data.don	A matrix or data frame that plays the role of <i>donor</i> in the statistical matching application. This data set must contain all the numeric variables (columns) that should be used in statistical matching, i.e. the variables called by the arguments <code>match.vars</code> and <code>z.don</code> . Note that continuous variables are expected, if there are some categorical variables they are recoded into dummies. Missing values (NA) are not allowed.
match.vars	A character vector with the names of the common variables (the columns in both the data frames) to be used as matching variables ( <b>X</b> ).
y.rec	A character vector with the name of the target variable Y that is observed only for units in <code>data.rec</code> . Only one continuous variable is allowed.
z.don	A character vector with the name of the target variable Z that is observed only for units in <code>data.don</code> . Only one continuous variable is allowed.
method	A character vector that identifies the method that should be used to estimate the parameters of the regression models: Y vs. <b>X</b> and Z vs. <b>X</b> . Maximum Likelihood method is used when <code>method="ML"</code> (default); on the contrary, when <code>method="MS"</code> the parameters are estimated according to approach proposed by Moriarity and Scheuren (2001 and 2003). See Details for further information.
rho.yz	A numeric value representing a guess for the correlation among the Y ( <code>y.rec</code> ) and the Z variable ( <code>z.don</code> ) that are not jointly observed. When <code>method="MS"</code> then the argument <code>cor.yz</code> must specify the value of the correlation coefficient $\rho_{YZ}$ ; on the contrary, when <code>method="ML"</code> , it must specify the <i>partial correlation coefficient</i> among Y and Z given <b>X</b> ( $\rho_{YZ X}$ ). By default ( <code>rho.yz=0</code> ). In practice, in absence of auxiliary information concerning the correlation coefficient or the partial correlation coefficient, the statistical matching is carried out under the assumption of independence among Y and Z given <b>X</b> (Conditional Independence Assumption, CIA), i.e. $\rho_{YZ X} = 0$ .
micro	Logical. When <code>micro=FALSE</code> (default) only the parameter estimates are returned. On the contrary, when <code>micro=TRUE</code> the function returns also <code>data.rec</code> filled in with the values for the variable Z. The donors for filling in Z in <code>data.rec</code> are identified using a constrained distance hot deck method. In this case, the number of units (rows) in <code>data.don</code> must be greater or equal to the number of units (rows) in <code>data.rec</code> . See next argument and Details for further information.
constr.alg	A string that has to be specified when <code>micro=TRUE</code> , in order to solve the transportation problem involved by the constrained distance hot deck method. Two choices are available: "lpSolve" and "relax". In the first case, <code>constr.alg="lpSolve"</code> , the transportation problem is solved by means of the function <code>lp.transport</code> available in the package <b>lpSolve</b> . When <code>constr.alg="relax"</code> the transportation problem is solved using RELAX-IV algorithm from Bertsekas and Tseng (1994), implemented in function <code>pairmatch</code> available in the package <b>optmatch</b> . Note that <code>constr.alg="relax"</code> is faster and requires less computational effort, but the usage of this algorithm is allowed only for research purposes (for details see function <code>relaxinfo()</code> in the package <b>optmatch</b> ).

## Details

This function implements some mixed methods to perform statistical matching. A mixed method consists of two steps:

- (i) adoption of a parametric model for the joint distribution of  $(\mathbf{X}, Y, Z)$  and estimation of its parameters;
- (ii) derivation of a complete “synthetic” data set (recipient data set filled in with values for the  $Z$  variable) using a nonparametric approach.

In this case, as far as (i) is concerned, it is assumed that  $(\mathbf{X}, Y, Z)$  follows a multivariate normal distribution. Please note that if some of  $\mathbf{X}$  are categorical, then they are recoded into dummies before starting with the estimation. In such a case the assumption of multivariate normal distribution maybe questionable.

The whole procedure is based on the imputation method known as *predictive mean matching*. The procedure consists of three steps:

step 1) – Regression step: the two linear regression models  $Y$  vs.  $\mathbf{X}$  and  $Z$  vs.  $\mathbf{X}$  are considered and their parameters are estimated.

step 2) – Computation of intermediate values. For the units in `data.rec` the following intermediate values are derived:

$$\tilde{z}_a = \hat{\alpha}_Z + \hat{\beta}_{Z\mathbf{X}}\mathbf{x}_a + e_a$$

for each  $a = 1, \dots, n_A$ , being  $n_A$  the number of units in `data.rec` (rows of `data.rec`). Note that,  $e_a$  is a random draw from the multivariate normal distribution with zero mean and estimated residual variance  $\hat{\sigma}_{Z|\mathbf{X}}$ .

Similarly, for the units in `data.don` the following intermediate values are derived:

$$\tilde{y}_b = \hat{\alpha}_Y + \hat{\beta}_{Y\mathbf{X}}\mathbf{x}_b + e_b$$

for each  $b = 1, \dots, n_B$ , being  $n_B$  the number of units in `data.don` (rows of `data.don`).  $e_b$  is a random draw from the multivariate normal distribution with zero mean and estimated residual variance  $\hat{\sigma}_{Y|\mathbf{X}}$ .

step 3) – Matching step. For each observation (row) in `data.rec` a donor is chosen in `data.don` through a nearest neighbor constrained distance hot deck procedure. The distances are computed between  $(y_a, \tilde{z}_a)$  and  $(\tilde{y}_b, z_b)$  using Mahalanobis distance.

For further details see Sections 2.5.1 and 3.6.1 in D’Orazio *et al.* (2006).

In step 1) the parameters of the regression model can be estimated by means of the Maximum Likelihood method (`method="ML"`) (see D’Orazio *et al.*, 2006, pp. 19–23,73–75) or, using the Moriarity and Scheuren (2001 and 2003) approach (`method="MS"`) (see also D’Orazio *et al.*, 2006, pp. 75–76). The two estimation methods are compared in D’Orazio *et al.* (2005).

When `method="MS"`, if the value specified for the argument `rho.yz` is not compatible with the other correlation coefficients estimated from the data, then it is substituted with the closest value compatible with the other estimated coefficients.

When `micro=FALSE` only the estimation of the parameters is performed (step 1). Otherwise, (`micro=TRUE`) the whole procedure is carried out.

**Value**

A list with a varying number of components depending on the values of the arguments `method` and `rho.yz`.

<code>mu</code>	The estimated mean vector.
<code>vc</code>	The estimated variance–covariance matrix.
<code>cor</code>	The estimated correlation matrix.
<code>res.var</code>	A vector with estimates of the residual variances $\sigma_{Y Z\mathbf{X}}$ and $\sigma_{Z Y\mathbf{X}}$ .
<code>start.prho.yz</code>	It is the initial guess for the partial correlation coefficient $\rho_{YZ \mathbf{X}}$ passed in input via the <code>rho.yz</code> argument when <code>method="ML"</code> .
<code>rho.yz</code>	Returned in output only when <code>method="MS"</code> . It is a vector with four values: the initial guess for $\rho_{YZ}$ ; the lower and upper bounds for $\hat{\rho}_{YZ}$ in the statistical matching framework given the correlation coefficients among $Y$ and $\mathbf{X}$ and the correlation coefficients among $Z$ and $\mathbf{X}$ estimated from the available data; and, finally, the closest admissible value used in computations instead of the initial <code>rho.yz</code> that resulted not coherent with the others correlation coefficients estimated from the available data.
<code>phi</code>	When <code>method="MS"</code> . Estimates of the $\phi$ terms introduced by Moriarity and Scheuren (2001 and 2003).
<code>filled.rec</code>	The <code>data.rec</code> filled in with the values of $Z$ . It is returned only when <code>micro=TRUE</code> .
<code>mtc.ids</code>	when <code>micro=TRUE</code> . This is a matrix with the same number of rows of <code>data.rec</code> and two columns. The first column contains the row names of the <code>data.rec</code> and the second column contains the row names of the corresponding donors selected from the <code>data.don</code> . When the input matrices do not contain row names, a numeric matrix with the indexes of the rows is provided.
<code>dist.rd</code>	A vector with the distances among each recipient unit and the corresponding donor, returned only in case <code>micro=TRUE</code> .
<code>call</code>	How the function has been called.

**Author(s)**

Marcello D’Orazio <madorazi@istat.it>

**References**

- Bertsekas, D.P. and Tseng, P. (1994). “RELAX–IV: A Faster Version of the RELAX Code for Solving Minimum Cost Flow Problems”. *Technical Report*, LIDS-P-2276, Massachusetts Institute of Technology, Cambridge. [http://web.mit.edu/dimitrib/www/RELAX4\\_doc.pdf](http://web.mit.edu/dimitrib/www/RELAX4_doc.pdf)
- D’Orazio, M., Di Zio, M. and Scanu, M. (2005). “A comparison among different estimators of regression parameters on statistically matched files through an extensive simulation study”, *Contributi*, **2005/10**, Istituto Nazionale di Statistica, Rome. [http://www.istat.it/dati/pubbsci/contributi/Contributi/contr\\_2005/2005\\_10.pdf](http://www.istat.it/dati/pubbsci/contributi/Contributi/contr_2005/2005_10.pdf)
- D’Orazio, M., Di Zio, M. and Scanu, M. (2006). *Statistical Matching: Theory and Practice*. Wiley, Chichester.

Moriarity, C., and Scheuren, F. (2001). “Statistical matching: a paradigm for assessing the uncertainty in the procedure”. *Journal of Official Statistics*, **17**, 407–422. <http://www.jos.nu/Articles/abstract.asp?article=173407>

Moriarity, C., and Scheuren, F. (2003). “A note on Rubin’s statistical matching using file concatenation with adjusted weights and multiple imputation”, *Journal of Business and Economic Statistics*, **21**, 65–73.

### See Also

[NND.hotdeck](#), [mahalanobis.dist](#)

### Examples

```
# reproduce the statistical matching framework
# starting from the iris data.frame
set.seed(98765)
pos <- sample(1:150, 50, replace=FALSE)
ir.A <- iris[pos,c(1,3:5)]
ir.B <- iris[-pos, 2:5]

xx <- intersect(colnames(ir.A), colnames(ir.B))
xx # common variables

# ML estimation method under CIA ((rho_YZ|X=0));
# only parameter estimates (micro=FALSE)
# only continuous matching variables
xx.mtc <- c("Petal.Length", "Petal.Width")
mtc.1 <- mixed.mtc(data.rec=ir.A, data.don=ir.B, match.vars=xx.mtc,
                  y.rec="Sepal.Length", z.don="Sepal.Width")

# estimated correlation matrix
mtc.1$cor

# ML estimation method under CIA ((rho_YZ|X=0));
# only parameter estimates (micro=FALSE)
# categorical variable 'Species' used as matching variable

xx.mtc <- xx
mtc.2 <- mixed.mtc(data.rec=ir.A, data.don=ir.B, match.vars=xx.mtc,
                  y.rec="Sepal.Length", z.don="Sepal.Width")

# estimated correlation matrix
mtc.2$cor

# ML estimation method with partial correlation coefficient
# set equal to 0.5 (rho_YZ|X=0.5)
# only parameter estimates (micro=FALSE)

mtc.3 <- mixed.mtc(data.rec=ir.A, data.don=ir.B, match.vars=xx.mtc,
                  y.rec="Sepal.Length", z.don="Sepal.Width",
```

```

        rho.yz=0.5)

# estimated correlation matrix
mtc.3$cor

# ML estimation method with partial correlation coefficient
# set equal to 0.5 (rho_YZ|X=0.5)
# with imputation step (micro=TRUE)

mtc.4 <- mixed.mtc(data.rec=ir.A, data.don=ir.B, match.vars=xx.mtc,
                  y.rec="Sepal.Length", z.don="Sepal.Width",
                  rho.yz=0.5, micro=TRUE, constr.alg="lpSolve")

# first rows of data.rec filled in with z
head(mtc.4$filled.rec)

#
# Moriarity and Scheuren estimation method under CIA;
# only with parameter estimates (micro=FALSE)
mtc.5 <- mixed.mtc(data.rec=ir.A, data.don=ir.B, match.vars=xx.mtc,
                  y.rec="Sepal.Length", z.don="Sepal.Width",
                  method="MS")

# the starting value of rho.yz and the value used
# in computations
mtc.5$rho.yz

# estimated correlation matrix
mtc.5$cor

# Moriarity and Scheuren estimation method
# with correlation coefficient set equal to -0.15 (rho_YZ=-0.15)
# with imputation step (micro=TRUE)

mtc.6 <- mixed.mtc(data.rec=ir.A, data.don=ir.B, match.vars=xx.mtc,
                  y.rec="Sepal.Length", z.don="Sepal.Width",
                  method="MS", rho.yz=-0.15,
                  micro=TRUE, constr.alg="lpSolve")

# the starting value of rho.yz and the value used
# in computations
mtc.6$rho.yz

# estimated correlation matrix
mtc.6$cor

# first rows of data.rec filled in with z imputed values
head(mtc.6$filled.rec)

```

## Description

This function implements the distance hot deck method to match the records of two data sources that share some variables.

## Usage

```
NND.hotdeck(data.rec, data.don, match.vars,
            don.class=NULL, dist.fun="Manhattan",
            constrained=FALSE, constr.alg=NULL, ...)
```

## Arguments

- |             |  |
|-------------|--|
| data.rec    | A matrix or data frame that plays the role of <i>recipient</i> . This data frame must contain the variables (columns) that should be used, directly or indirectly, in the computation of distances between its observations (rows) and those of data.don. Missing values (NA) are allowed.   |
| data.don    | A matrix or data frame that plays the role of <i>donor</i> . The variables (columns) involved, directly or indirectly, in the computation of distance must be the same and of the same type as those in data.rec.  |
| match.vars  | A character vector with the names of the matching variables (the columns in both the data frames) that have to be used to compute distances among records (rows) in data.rec and those in data.don.  |
| don.class   | A character vector with the names of the variables (columns in both the data frames) that have to be used to identify the donation classes. In this case the computation of distances is limited to those units of data.rec and data.doc that belong to the same donation class. The case of empty donation classes should be avoided. It would be preferable that variables used to form donation classes are defined as factor.<br>When not specified (default) no donation classes are used. This may result in a heavy computational effort.   |
| dist.fun    | A string with the name of the distance function that has to be used. The following distances are allowed: "Manhattan" (aka "City block"; default), "Euclidean", "Mahalanobis", "exact" or "exact matching", "Gower", "minimax" or one of the distance functions available in the package <b>proxy</b> . Note that the distance is computed using the function <code>dist</code> of the package <b>proxy</b> with the exception of the "Gower" (see function <code>gower.dist</code> for details), "Mahalanobis" (function <code>mahalanobis.dist</code> ) and "minimax" (see <code>maximum.dist</code> ) cases.<br>When <code>dist.fun="Manhattan", "Euclidean", "Mahalanobis" or "minimax"</code> all the non numeric variables in data.rec and data.don will be converted to numeric. On the contrary, when <code>dist.fun="exact" or "exact matching"</code> , all the variables in data.rec and data.don will be converted to character and, as far as the distance computation is concerned, they will be treated as categorical nominal variables, i.e. distance is 0 if a couple of units shows the same response category and 1 otherwise. |
| constrained | Logical. When <code>constrained=FALSE</code> (default) each record in data.don can be used as a donor more than once. On the contrary, when  |

constrained=TRUE each record in `data.don` can be used as a donor only once. In this case, the set of donors is selected by solving a transportation problem, in order to minimize the overall matching distance. See description of the argument `constr.alg` for details.

`constr.alg` A string that has to be specified when `constrained=TRUE`. Two choices are available: “lpSolve” and “relax”. In the first case, `constr.alg="lpSolve"`, the transportation problem is solved by means of the function `lp.transport` available in the package `lpSolve`. When `constr.alg="relax"` the transportation problem is solved using RELAX-IV algorithm from Bertsekas and Tseng (1994), implemented in function `pairmatch` available in the package `optmatch`. Note that `constr.alg="relax"` is faster but the usage of the algorithm is allowed only for research purposes (for details see function `relaxinfo()` in the package `optmatch`).

... Additional arguments that may be required by `gower.dist`, or by `maximum.dist`, or by `dist`.

## Details

This function finds a donor record in `data.don` for each record in the recipient data frame `data.rec`. In the unconstrained case, it searches for the closest donor record in `data.don` for each record in the recipient data set, according to the chosen distance function. When for a given recipient record there are more donors available at the minimum distance, one of them is picked at random.

In the constrained case the set of donors is chosen in order to minimize the overall matching distance. In this case the number of units (rows) in the donor data set has to be larger or equal to the number of units of the recipient data set. When the donation classes are used, this condition must be satisfied in each donation class. For further details on nearest neighbor distance hot deck refer to Chapter 2 in D’Orazio *et al.* (2006).

This function can also be used to impute missing values in a data set using the nearest neighbor distance hot deck. In this case `data.rec` is the part of the initial data set that contains missing values; on the contrary, `data.don` is the part of the data set without missing values. See R code in the Examples for details.

## Value

A R list with the following components:

`mtc.ids` A matrix with the same number of rows of `data.rec` and two columns. The first column contains the row names of the `data.rec` and the second column contains the row names of the corresponding donors selected from the `data.don`. When the input matrices do not contain row names, a numeric matrix with the indexes of the rows is provided.

`dist.rd` A vector with the distances among each recipient unit and the corresponding donor.

`noad` When `constrained=FALSE`, it reports the number of available donors at the minimum distance for each recipient unit.

`call` How the function has been called.

**Author(s)**

Marcello D’Orazio <madorazi@istat.it>

**References**

Bertsekas, D.P. and Tseng, P. (1994). “RELAX–IV: A Faster Version of the RELAX Code for Solving Minimum Cost Flow Problems”. *Technical Report*, LIDS-P-2276, Massachusetts Institute of Technology, Cambridge. [http://web.mit.edu/dimitrib/www/RELAX4\\_doc.pdf](http://web.mit.edu/dimitrib/www/RELAX4_doc.pdf)

D’Orazio, M., Di Zio, M. and Scanu, M. (2006). *Statistical Matching: Theory and Practice*. Wiley, Chichester.

Rodgers, W.L. (1984). “An evaluation of statistical matching”. *Journal of Business and Economic Statistics*, **2**, 91–102.

Singh, A.C., Mantel, H., Kinack, M. and Rowe, G. (1993). “Statistical matching: use of auxiliary information as an alternative to the conditional independence assumption”. *Survey Methodology*, **19**, 59–79.

**See Also**

[RANDwNND.hotdeck](#)

**Examples**

```
# reproduce the classical matching framework
lab <- c(1:15, 51:65, 101:115)
iris.rec <- iris[lab, c(1:3,5)] # recipient data.frame
iris.don <- iris[-lab, c(1:2,4:5)] #donor data.frame

# Now iris.rec and iris.don have the variables
# "Sepal.Length", "Sepal.Width" and "Species"
# in common.
# "Petal.Length" is available only in iris.rec
# "Petal.Width" is available only in iris.don

# Find the closest donors donors computing distance
# on "Sepal.Length" and "Sepal.Width"
# unconstrained case, Euclidean distance

out.NND.1 <- NND.hotdeck(data.rec=iris.rec, data.don=iris.don,
                        match.vars=c("Sepal.Length", "Sepal.Width") )

# create the synthetic data.set:
# fill in "Petal.Width" in iris.rec

fused.1 <- create.fused(data.rec=iris.rec, data.don=iris.don,
                       mtc.ids=out.NND.1$mtc.ids, z.vars="Petal.Width")

# Find the closest donors computing distance
# on "Sepal.Length", "Sepal.Width" and Species;
```

```

# unconstrained case, Gower's distance

out.NND.2 <- NND.hotdeck(data.rec=iris.rec, data.don=iris.don,
                        match.vars=c("Sepal.Length", "Sepal.Width", "Species"),
                        dist.fun="Gower")

# find the closest donors using "Species" to form donation classes
# and "Sepal.Length" and "Sepal.Width" to compute distance;
# unconstrained case.

out.NND.3 <- NND.hotdeck(data.rec=iris.rec, data.don=iris.don,
                        match.vars=c("Sepal.Length", "Sepal.Width"),
                        don.class="Species")

# find the donors using "Species" to form donation classes
# and "Sepal.Length" and "Sepal.Width" to compute distance;
# constrained case, "RELAX" algorithm

library(optmatch)
out.NND.4 <- NND.hotdeck(data.rec=iris.rec, data.don=iris.don,
                        match.vars=c("Sepal.Length", "Sepal.Width"),
                        don.class="Species", constr=TRUE, constr.alg="relax")

# find the donors using "Species" to form donation classes
# and "Sepal.Length" and "Sepal.Width" to compute distance;
# constrained case, transportation problem solved by functions
# in package "lpSolve"

library(lpSolve)
out.NND.5 <- NND.hotdeck(data.rec=iris.rec, data.don=iris.don,
                        match.vars=c("Sepal.Length", "Sepal.Width"),
                        don.class="Species", constr=TRUE, constr.alg="lpSolve")

# Example of Imputation of missing values.
# Introducing missing values in iris
ir.mat <- iris
miss <- rbinom(nrow(iris), 1, 0.3)
ir.mat[miss==1,"Sepal.Length"] <- NA
iris.rec <- ir.mat[miss==1,-1]
iris.don <- ir.mat[miss==0,]

#search for NND donors
imp.NND <- NND.hotdeck(data.rec=iris.rec, data.don=iris.don,
                      match.vars=c("Sepal.Width", "Petal.Length", "Petal.Width"),
                      don.class="Species")

# imputing missing values
iris.rec.imp <- create.fused(data.rec=iris.rec, data.don=iris.don,
                           mtc.ids=imp.NND$mtc.ids, z.vars="Sepal.Length")

```

```
# rebuild the imputed data.frame
final <- rbind(iris.rec.imp, iris.don)
```

---

RANDwNND.hotdeck      *Random Distance hot deck.*

---

## Description

This function implements a variant of the distance hot deck method. For each recipient record a subset of of the closest donors is retained and then a donor is selected.

## Usage

```
RANDwNND.hotdeck(data.rec, data.don, match.vars=NULL,
                  don.class=NULL, dist.fun="Manhattan",
                  cut.don="rot", k=NULL, weight.don=NULL, ...)
```

## Arguments

- |            |   |
|------------|---|
| data.rec   | A numeric matrix or data frame that plays the role of <i>recipient</i> . This data frame must contain the variables (columns) that should be used, directly or indirectly, in the computation of distances between its observations (rows) and those in data.don.<br>Missing values (NA) are allowed.   |
| data.don   | A matrix or data frame that plays the role of <i>donor</i> . The variables (columns) involved, directly or indirectly, in the computation of distances must be the same and of the same type as those in data.rec.  |
| match.vars | A character vector with the names of the variables (the columns in both the data frames) that have to be used to compute distances among records (rows) in data.rec and those in data.don. When no matching variables are considered (match.vars=NULL) then all the units in the same donation class are considered as possible donors. Hence one of them is selected at random or with probability proportional to its weight (see argument weight.don). When match.vars=NULL and the donation classes are not created (don.class=NULL) then all the available records in the data.don are considered as potential donors. |
| don.class  | A character vector with the names of the variables (columns in both the data frames) that have to be used to identify donation classes. In this case the computation of distances is limited to those units in data.rec and data.doc that belong to the same donation class. The case of empty donation classes should be avoided. It would be preferable that variables used to form donation classes are defined as factor.<br>When not specified (default), no donation classes are used. This may result in a heavy computational effort.   |

<code>dist.fun</code>	<p>A string with the name of the distance function that has to be used. The following distances are allowed: “Manhattan” (aka “City block”; default), “Euclidean”, “Mahalanobis”, “exact” or “exact matching”, “Gower”, “minimax” or one of the distance functions available in the package <b>proxy</b>. Note that the distance is computed using the function <code>dist</code> of the package <b>proxy</b> with the exception of the “Gower” (see function <code>gower.dist</code> for details), “Mahalanobis” (function <code>mahalanobis.dist</code>) and “minimax” (see <code>maximum.dist</code>) cases.</p> <p>When <code>dist.fun="Manhattan"</code>, “Euclidean”, “Mahalanobis” or “minimax” all the non numeric variables in <code>data.rec</code> and <code>data.don</code> will be converted to numeric. On the contrary, when <code>dist.fun="exact"</code> or “exact matching”, all the variables in <code>data.rec</code> and <code>data.don</code> will be converted to character and, as far as the distance computation is concerned, they will be treated as categorical nominal variables, i.e. distance is 0 if a couple of units shows the same response category and 1 otherwise.</p>
<code>cut.don</code>	<p>A character string that, jointly with the argument <code>k</code>, identifies the rule to be used to form the subset of the closest donor records.</p> <ul style="list-style-type: none"> <li>• <code>cut.don="rot"</code>: (default) then the number of the closest donors to retain is given by <math>\lceil \sqrt{n_D} \rceil + 1</math>; being <math>n_D</math> the total number of available donors. In this case <code>k</code> must not to be specified.</li> <li>• <code>cut.don="span"</code>: the number of closest donors is determined as the proportion <code>k</code> of all the available donors, i.e. <math>\lceil n_D \times k \rceil</math>. Note that, in this case, <math>0 &lt; k \leq 1</math>.</li> <li>• <code>cut.don="exact"</code>: the <code>k</code>th closest donors out of the <math>n_D</math> are retained. In this case, <math>0 &lt; k \leq n_D</math>.</li> <li>• <code>cut.don="min"</code>: the donors at the minimum distance from the recipient are retained.</li> <li>• <code>cut.don="k.dist"</code>: only the donors whose distance from the recipient is less or equal to the value specified with the argument <code>k</code>.</li> </ul>
<code>k</code>	Depends on the <code>cut.don</code> argument.
<code>weight.don</code>	<p>A character string providing the name of the variable with the weights associated to the donor units in <code>data.don</code>. When this variable is specified, then the selection of a donor among those in the subset of the closest donors is done with probability proportional to its weight (units with larger weight will have a higher chance of being selected). When <code>weight.don=NULL</code> (default) all the units in the subset of the closest donors will have the same probability of being selected.</p>
...	<p>Additional arguments that may be required by <code>gower.dist</code>, or by <code>maximum.dist</code>, <code>dist</code>.</p>

## Details

This function finds a donor record for each record in the recipient data set. The donor is chosen at random in the subset of available donors. This procedure is known as *random hot deck* (cf. Andridge and Little, 2010). In this case, the number of closest donors retained to form the subset is determined according to criterion specified with the argument `cut.don`. The selection of the donor among those in the subset is carried out with equal probability (`weight.don=NULL`) or with probability proportional to a weight associated to the donors (specified via the `weight.don` argument). This procedure is known as *weighted random hot deck* (cf. Andridge and Little, 2010).

Note that the same donor can be used more than once.

This function can also be used to impute missing values in a data set. In this case `data.rec` is the part of the initial data set that contains missing values; on the contrary, `data.don` is the part of the data set without missing values. See R code in the Examples for details.

### Value

A R list with the following components:

<code>mtc.ids</code>	A matrix with the same number of rows of <code>data.rec</code> and two columns. The first column contains the row names of the <code>data.rec</code> and the second column contains the row names of the corresponding donors selected from the <code>data.don</code> . When the input matrices do not contain row names, then a numeric matrix with the indexes of the rows is provided.
<code>sum.dist</code>	A matrix with summary statistics concerning the subset of the closest donors. The first three columns report the minimum, the maximum and the standard deviation of the distances among the recipient record and the donors in the subset of the closest donors, respectively. The 4th column reports the cutting distance, i.e. the value of the distance such that donors at a higher distance are discarded. The 5th column reports the distance between the recipient and the donor chosen at random in the subset of the donors.
<code>noad</code>	For each recipient unit, reports the number of donor records in the subset of closest donors.
<code>call</code>	How the function has been called.

### Author(s)

Marcello D’Orazio <madorazi@istat.it>

### References

- Andridge, R.R., and Little, R.J.A. (2010) “A Review of Hot Deck Imputation for Survey Non-response”. *International Statistical Review*, **78**, 40–64.
- D’Orazio, M., Di Zio, M. and Scanu, M. (2006). *Statistical Matching: Theory and Practice*. Wiley, Chichester.
- Rodgers, W.L. (1984). “An evaluation of statistical matching”. *Journal of Business and Economic Statistics*, **2**, 91–102.
- Singh, A.C., Mantel, H., Kinack, M. and Rowe, G. (1993). “Statistical matching: use of auxiliary information as an alternative to the conditional independence assumption”. *Survey Methodology*, **19**, 59–79.

### See Also

[NND.hotdeck](#)

**Examples**

```

require(SDaA)
data(agpop, agsrs, agstrat, package="SDaA") #loads ag data from SDaA
str(agpop)
str(agsrs)
str(agstrat)

# adds variable "region" to agsrs
state.region <- data.frame(xtabs(weight~state+region, data=agstrat))
state.region <- subset(state.region, Freq>0)
agsrs <- merge(agsrs, state.region[,1:2], by="state", all.x=TRUE)

# simulate a statistical matching framework
A <- agsrs[, c("region", "acres82", "farms82", "acres87", "farms87")]
B <- agstrat[, c("region", "acres82", "farms82", "acres92", "farms92",
               "weight")]

# find a donor in the subset of closest donors using cut.don="rot";
# the distance is computed using "acres82" and "farms82"

out.NND.1 <- RANDwNND.hotdeck(data.rec=A, data.don=B,
                             match.vars=c("acres82", "farms82") )

# create the synthetic (or fused) data.frame:
# fill in "acres92" and "farms92" in A
fused.1 <- create.fused(data.rec=A, data.don=B,
                      mtc.ids=out.NND.1$mtc.ids, z.vars=c("acres92", "farms92"))
head(fused.1)

# find a donor in the subset of closest donors using cut.don="rot";
# the distance is computed using "acres82" and "farms82"
# weights are used in selecting the donor

out.NND.2 <- RANDwNND.hotdeck(data.rec=A, data.don=B,
                             match.vars=c("acres82", "farms82"), weight.don="weight" )
fused.2 <- create.fused(data.rec=A, data.don=B,
                      mtc.ids=out.NND.2$mtc.ids, z.vars=c("acres92", "farms92"))
head(fused.2)

# as before, but with a different criteria to reduce the no. of donors:
# the first half (k=0.5) of the closest available donors is retained,
# then a donor is chosen with prob proportional to its weight

out.NND.3 <- RANDwNND.hotdeck(data.rec=A, data.don=B,
                             match.vars=c("acres82", "farms82"),
                             cut.don="span", k=0.5, weight.don="weight")
fused.3 <- create.fused(data.rec=A, data.don=B,
                      mtc.ids=out.NND.3$mtc.ids, z.vars=c("acres92", "farms92"))
head(fused.3)

```

```

# as before, but the subset of closest donors is formed by considering
# only the first k=5 closest donors

out.NND.4 <- RANDwNND.hotdeck(data.rec=A, data.don=B,
                             match.vars=c("acres82", "farms82"),
                             cut.don="exact", k=5, weight.don="weight")
fused.4 <- create.fused(data.rec=A, data.don=B,
                       mtc.ids=out.NND.4$mtc.ids, z.vars=c("acres92","farms92"))
head(fused.4)

# find a donor in the subset of closest donors using cut.don="rot";
# the distance is computed using "acres82" and "farms82"
# only donors in the same "region" are considered

out.NND.5 <- RANDwNND.hotdeck(data.rec=A, data.don=B, don.class="region",
                              match.vars=c("acres82", "farms82") )
fused.5 <- create.fused(data.rec=A, data.don=B,
                       mtc.ids=out.NND.5$mtc.ids, z.vars=c("acres92","farms92"),
                       dup.x=TRUE, match.vars="region")
head(fused.5)

# Example of Imputation of missing values
# introducing missing vales in iris
ir.mat <- iris
miss <- rbinom(nrow(iris), 1, 0.3)
ir.mat[miss==1,"Sepal.Length"] <- NA
iris.rec <- ir.mat[miss==1,-1]
iris.don <- ir.mat[miss==0,]

#search for NND donors
imp.NND <- RANDwNND.hotdeck(data.rec=iris.rec, data.don=iris.don,
                           match.vars=c("Sepal.Width","Petal.Length", "Petal.Width"),
                           don.class="Species")

# imputing missing values
iris.rec.imp <- create.fused(data.rec=iris.rec, data.don=iris.don,
                            mtc.ids=imp.NND$mtc.ids, z.vars="Sepal.Length")

# rebuild the imputed data.frame
final <- rbind(iris.rec.imp, iris.don)

```

**Description**

This function implements rank hot deck distance method. For each recipient record the closest donors is chosen by considering the distance among the percentage points of the empirical cumulative distribution function.

**Usage**

```
rankNND.hotdeck(data.rec, data.don, var.rec, var.don=var.rec,
                don.class=NULL, weight.rec=NULL, weight.don=NULL)
```

**Arguments**

<code>data.rec</code>	A numeric matrix or data frame that plays the role of <i>recipient</i> . This data frame must contain the variable <code>var.rec</code> to be used in computing the percentage points of the empirical cumulative distribution function and eventually the variables that should be used to identify the donation classes (see argument <code>don.class</code> ) and the case weights (see argument <code>weight.rec</code> ). Missing values (NA) are not allowed.
<code>data.don</code>	A matrix or data frame that plays the role of <i>donor</i> . This data frame must contain the variable <code>var.don</code> to be used in computing percentage points of the the empirical cumulative distribution function and eventually the variables that should be used to identify the donation classes (see argument <code>don.class</code> ) and the case weights (see argument <code>weight.don</code> ).
<code>var.rec</code>	A character vector with the name of the variable that should be ranked.
<code>var.don</code>	A character vector with the name of the variable that should be ranked. If not specified, by default <code>var.don=var.rec</code> .
<code>don.class</code>	A character vector with the names of the variables (columns in both the data frames) that have to be used to identify donation classes. In each donation class the computation of percentage points is carried out independently. Then only distances among percentage points of the units in the same donation class are computed. The case of empty donation classes should be avoided. It would be preferable the variables used to form donation classes are defined as factor. When not specified (default), no donation classes are used.
<code>weight.rec</code>	Eventual name of the variable in <code>data.rec</code> that provides the weights that should be used in computing the the empirical cumulative distribution function for <code>var.rec</code> (see Details).
<code>weight.don</code>	Eventual name of the variable in <code>data.don</code> that provides the weights that should be used in computing the the empirical cumulative distribution function for <code>var.don</code> (see Details).

**Details**

This function finds a donor record for each record in the recipient data set. The chosen donor is the one at the closest distance in terms of empirical cumulative distribution (Singh et al., 1990). In practice the distance is computed by considering the estimated empirical cumulative distribution for the reference variable (`var.rec` and `var.don`) in `data.rec` and `data.don`. The empirical cumulative distribution function is estimated by:

$$\hat{F}(y_k) = \frac{1}{n} \sum_{i=1}^n I(y_i \leq y_k)$$

being  $I() = 1$  if  $y_i \leq y_k$  and 0 otherwise.

In the presence of weights the empirical cumulative distribution function is estimated by:

$$\hat{F}(y_k) = \frac{\sum_{i=1}^n w_i I(y_i \leq y_k)}{\sum_{i=1}^n w_i}$$

When there are more donors at the same distance then one is chosen at random.

Note that when the donation classes are introduced then empirical cumulative distribution function is estimated independently in each donation classes and the search of a recipient is restricted to donors in the same donation class.

A donor can be chosen more than once.

### Value

A R list with the following components:

mtc.ids	A matrix with the same number of rows of data.rec and two columns. The first column contains the row names of the data.rec and the second column contains the row names of the corresponding donors selected from the data.don. When the input matrices do not contain row names, then a numeric matrix with the indexes of the rows is provided.
dist.rd	A vector with the distances among each recipient unit and the corresponding donor.
noad	The number of available donors at the minimum distance for each recipient unit.
call	How the function has been called.

### Author(s)

Marcello D'Orazio <madorazi@istat.it>

### References

- D'Orazio, M., Di Zio, M. and Scanu, M. (2006). *Statistical Matching: Theory and Practice*. Wiley, Chichester.
- Singh, A.C., Mantel, H., Kinack, M. and Rowe, G. (1993). "Statistical matching: use of auxiliary information as an alternative to the conditional independence assumption". *Survey Methodology*, **19**, 59–79.

### See Also

[NND.hotdeck](#)

**Examples**

```
require(SDaA)
data(agpop, agsrs, agstrat, package="SDaA") #loads ag datasets from SDaA
str(agpop)
str(agsrs)
str(agstrat)

agsrs$w.srs <- nrow(agpop)/nrow(agsrs) # add weights

# adds region to agsrs
state.region <- data.frame(xtabs(weight~state+region, data=agstrat))
state.region <- subset(state.region, Freq>0)
agsrs <- merge(agsrs, state.region[,1:2], by="state", all.x=TRUE)

# simulate statistical matching framework
A <- agsrs[, c("region", "acres82", "acres87", "w.srs")]
B <- agstrat[, c("region", "acres82", "acres92", "weight")]

# simplest call to rankNND.hotdeck()
out.1 <- rankNND.hotdeck(data.rec=A, data.don=B, var.rec="acres82")
fused.1 <- create.fused(data.rec=A, data.don=B,
  mtc.ids=out.1$mtc.ids, z.vars="acres92")
head(fused.1)

# call to rankNND.hotdeck() with usage of weights
out.2 <- rankNND.hotdeck(data.rec=A, data.don=B, var.rec="acres82",
  weight.rec="w.srs", weight.don="weight")
fused.2 <- create.fused(data.rec=A, data.don=B,
  mtc.ids=out.2$mtc.ids, z.vars="acres92")
head(fused.2)

# call to rankNND.hotdeck() with usage of weights and don classes
out.3 <- rankNND.hotdeck(data.rec=A, data.don=B, var.rec="acres82",
  don.class="region", weight.rec="w.srs", weight.don="weight")
fused.3 <- create.fused(data.rec=A, data.don=B,
  mtc.ids=out.3$mtc.ids, z.vars="acres92")
head(fused.3)
```

# Index

- \*Topic **cluster**
  - fact2dummy, 9
  - gower.dist, 16
- \*Topic **manip**
  - create.fused, 7
- \*Topic **multivariate**
  - fact2dummy, 9
  - Fbwidths.by.x, 10
  - Frechet.bounds.cat, 12
  - gower.dist, 16
  - mahalanobis.dist, 23
  - maximum.dist, 25
- \*Topic **nonparametric**
  - mixed.mtc, 26
  - NND.hotdeck, 32
  - RANDwNND.hotdeck, 36
  - rankNND.hotdeck, 40
- \*Topic **regression**
  - mixed.mtc, 26
- \*Topic **survey**
  - comb.samples, 3
  - harmonize.x, 18

calibrate, 4, 5, 19, 21

comb.samples, 3, 21

create.fused, 7

daisy, 18

dist, 18, 32, 33, 37

fact2dummy, 9

Fbwidths.by.x, 10, 14

formula, 4, 19

Frechet.bounds.cat, 11, 12, 12

gower.dist, 10, 16, 32, 33, 37

harmonize.x, 5, 12, 14, 18

lp.transport, 27, 33

mahalanobis, 24

mahalanobis.dist, 23, 30, 32, 37

maximum.dist, 25, 32, 33, 37

mixed.mtc, 8, 26

model.matrix, 4, 19

NND.hotdeck, 8, 30, 31, 38, 42

pairmatch, 27, 33

postStratify, 19, 21

RANDwNND.hotdeck, 8, 34, 36

rank, 17, 25, 26

rankNND.hotdeck, 8, 40

StatMatch (StatMatch-package), 2

StatMatch-package, 2

svydesign, 3, 5, 18, 21

table, 11, 13

xtabs, 11, 13