Package: StatMatch (via r-universe)

September 11, 2024

Version 1.4.2

Date 2024-05-13

Title Statistical Matching or Data Fusion

Author Marcello D'Orazio

Maintainer Marcello D'Orazio <mdo.statmatch@gmail.com>

Depends R (>= 2.7.0), proxy, survey, lpSolve, ggplot2, dplyr

Suggests Hmisc, MASS, mipfp, clue, RANN,

Description Integration of two data sources referred to the same
target population which share a number of 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 GPL (>= 2)

URL https://github.com/marcellodo/StatMatch,

https:
//github.com/marcellodo/StatMatch/tree/master/Tutorials_Vignette_OtherDocs

NeedsCompilation no

Repository CRAN

Date/Publication 2024-05-13 13:53:18 UTC

Contents

StatMatch-package   *Statistical Matching or Data Fusion*

Description

Functions to perform statistical matching (aka data fusion), i.e. the integration of two data sources. Some functions can also be used to impute missing values in data sets through hot deck imputation methods.

Details

Statistical matching (aka *data fusion*) aims at integrating two data sources (typically samples) referred to same target population and sharing a number of variables. The final objective is that of studying the relationship of variables not jointly observed in a single data sources. The integration can be performed at *micro* (a *synthetic* or *fused* file is the output) or *macro* level (goal is estimation of correlation coefficients, regression coefficients, contingency tables, etc.). Nonparametric hot deck imputation methods (random, rank and nearest neighbour donor) can be used to derive the synthetic data set. In alternative it is possible to use a mixed (parametric-nonparametric) procedure based on predictive mean matching. Methods to perform statistical matching when dealing with data from complex sample surveys are available too. Finally some functions can be used to explore the uncertainty due to the typical matching framework. For major details see D'Orazio et al. (2006) or the package vignette.

Author(s)

Marcello D'Orazio

Maintainer: Marcello D'Orazio <mdo.statmatch@gmail.com>

References

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

---

comb.samples   *Statistical Matching of data from complex sample surveys*

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).

It is possible also to use the function to derive the estimates that a unit falls in one of the categories of the target variable (estimation are based on Liner Probability Models and are obtained as a by-product of the Renssen's method).

Usage

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

Arguments

svy.A   A svydesign R object that stores the data collected in the survey A and all the information concerning the corresponding sampling design. This object can be created by using the function [svydesign](#) in the package **survey**. All the variables specified in form.x and by y.lab must be available in svy.A.

svy.B   A svydesign R object that stores the data collected in the survey B and all the information concerning the corresponding sampling design. This object can be created by using the function [svydesign](#) in the package **survey**. All the variables specified in form.x and by z.lab must be available in svy.B.

svy.C   A svydesign R object that stores the data collected in the the survey C and all the information concerning the corresponding sampling design. This object can be created by using the function [svydesign](#) in the package **survey**.

               When svy.C=NULL (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).

               When svy.C is available, if estimation="incomplete" then it must contain at least y.lab and z.lab variables. On the contrary, when estimation="synthetic" all the variables specified in form.x, y.lab and z.lab must be available in svy.C.

y.lab   A string providing the name of the Y variable, available in survey A and in
                 survey C (if available). The Y variable can be a categorical variable (factor in
                 R) or a continuous one (in this latter case z.lab should be categorical).

z.lab   A string providing the name of the Z variable available in survey B and in survey
                 C (if available). The Z variable can be a categorical variable (factor in R) or a
                 continuous one (in this latter case y.lab should be categorical).

form.x   A R formula specifying the matching variables (subset of the **X** variables) col-
                 lected in all the surveys, and how have to be considered in combining samples.
                 For instance, form.x=~x1+x2 means that the variables x1 and x2 have to be con-
                 sidered marginally 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 as form.x=~x1+x2-1.

                 When dealing with categorical variables, form.x=~x1:x2-1 means that the the
                 joint distribution of the two variables (table of x1 vs. x2) has to be considered.

                 To better understand the usage of form.x see [model.matrix](#) (see also
                 [formula](#)).

                 Due to weights calibration features, it is preferable to work with categorical **X**
                 variables. In some cases, procedure may be successful when a single continuous
                 variable is considered jointly with one or more categorical variables but, often,
                 it may be necessary to categorize the continuous variable (see Details).

estimation   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 **X** variables speci-
                 fied via form.x, as well as the Y and Z variables. See Details for further infor-
                 mation.

micro   Logical, when TRUE predictions of Z in A and of Y in B are provided. In par-
                 ticular when Y and Z are both categorical variables it is provided the estimated
                 probability that a unit falls in each of the categories of the given variable. These
                 probabilities are estimated as a by-product of the whole procedure by consider-
                 ing Linear Probability Models, as suggested in Renssen (1998) (see Details)

...   Further arguments that may be necessary for calibration. In particular, the argu-
                 ment calfun allows to specify the calibration function:
                 (i) calfun="linear" for linear calibration (default);
                 (ii) calfun="raking" to rake the survey weights; and
                 (iii) calfun="logit" for logit calibration. See [calibrate](#) for major details.

                 Note that when calfun="linear" calibration may return negative weights. Gen-
                 erally speaking, in sample surveys weights are expected to be greater than or
                 equal to 1, i.e. bounds=c(1, Inf).

                 The number of iterations used in calibration can be modified by using the argu-
                 ment maxit (by default maxit=50).

                 See [calibrate](#) for further details.

Details

This function estimates the contingency table of Y vs. Z by performing a series of calibrations 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. $p(Y,Z) = p(Y|\boldsymbol{X}) \times p(Z|\boldsymbol{X}) \times p(\boldsymbol{X})$.

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 estimated on survey A, and that of Z estimated on survey B. Note that the distribution of the $\boldsymbol{X}$ 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 $\boldsymbol{X}$ variables observed in C. This method consists in correcting the table of Y vs. Z estimated under the CIA according to the relationship between Y and Z observed in survey C (for major details see Renssen, 1998).

When the argument `micro` is set to `TRUE` the function provides also `Z.A` and `Y.B`. The first data.frame has the same rows as `svy.A` and the number of columns equals the number of categories of the Z variable specified via `z.lab`. Each row provides the estimated probabilities of assuming a value in the various categories. The same happens for `Y.B` which presents the estimated probabilities of assuming a category of `y.lab` for each unit in B. The estimated probabilities are obtained by applying the linear probability models (for major details see Renssen, 1998). Unfortunately, such models may provide estimated probabilities less than 0 or greater than 1. Much caution should be used in using such predictions for practical purposes.

Value

A R list with the results of the calibration procedure according to the input arguments.

yz.CIA	The table of Y (`y.lab`) vs. Z (`z.lab`) estimated under the Conditional Independence Assumption (CIA).
cal.C	The survey object `svy.C` after the calibration. Only when `svy.C` is provided.
yz.est	The table of Y (`y.lab`) vs. Z (`z.lab`) estimated under the method specified via `estimation` argument. Only when `svy.C` is provided.
Z.A	Only when `micro=TRUE`. It is a data frame with the same rows as in `svy.A` and the number of columns is equal to the number of categories of the variable `z.lab`. Each row provides the estimated probabilities for a unit being in the various categories of `z.lab`.
Y.B	Only when `micro=TRUE`. It is a data frame with the same rows as in `svy.B` and the number of columns is equal to the number of categories of `y.lab`. Each row provides the estimated probabilities for a unit being in the various categories of `y.lab`.
call	Stores the call to this function with all the values specified for the various arguments (call=match.call()).

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

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

suppressWarnings(RNGversion("3.5.0"))
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)

# Harmonizazion 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

suppressWarnings(RNGversion("3.5.0"))
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)

# call comb.samples
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
# only macro estimation

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)
```

```
# synthetic two-way stratification
# with micro predictions

out.4 <- 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",
            micro=TRUE, calfun="linear",bounds=c(.5,Inf),
            maxit=100)

head(out.4$Z.A)
head(out.4$Y.B)
```

---

comp.cont	*Empirical comparison of two estimated distributions of the same continuous variable*

Description

This function estimates the "closeness" of distributions of the same continuous variable(s) but estimated from different data sources.

Usage

```
comp.cont(data.A, data.B, xlab.A, xlab.B = NULL, w.A = NULL, w.B = NULL, ref = FALSE)
```

Arguments

data.A	A dataframe or matrix containing the variable of interest xlab.A and eventual survey weights w.A.
data.B	A dataframe or matrix containing the variable of interest xlab.B and eventual associated survey weights w.B.
xlab.A	Character string providing the name of the variable in data.A whose estimated distribution should be compared with that estimated from data.B.
xlab.B	Character string providing the name of the variable in data.B whose distribution should be compared with that estimated from data.A. If xlab.B=NULL (default) then it assumed xlab.B=xlab.A.
w.A	Character string providing the name of the optional weighting variable in data.A that, in case, should be used to estimate the distribution of xlab.A
w.B	Character string providing the name of the optional weighting variable in data.B that, in case, should be used to estimate the distribution of xlab.B
ref	Logical. When ref = TRUE, the distribution of xlab.B estimated from data.B is considered the reference distribution (true or reliable estimate of distribution). Affects some estimation procedures as explained in the Details.

Details

As a first output, the function returns some well–known summary measures (min, Q1, median, mean, Q3, max and sd) estimated from the available input data sources.

Secondly this function performs a comparison between the quantiles estimated from data.A and data.B; in particular, the average of the absolute value of the differences as well as the average of the squared differences are returned. The number of estimated percentiles depends on the minimum between the two sample sizes. Only quartiles are calculated when min(n.A, n.B)<=50; quintiles are estimated when min(n.A, n.B)>50 and min(n.A, n.B)<=150; deciles are estimated when min(n.A, n.B)>150 and min(n.A, n.B)<=250; finally quantiles for probs=seq(from = 0.05, to = 0.95, by = 0.05) are estimated when min(n.A, n.B)>250. When the survey weights are available (indicated with th arguments w.A and/or w.B) they are used in estimating the quantiles by calling the function [wtd.quantile](#) in the package **Hmisc**.

The function estimates also the dissimilarities between the estimated empirical distribution function. The measures considered are the maximum of the absolute differences, the sum between the maximum differences inverting the terms in the difference and the average of the absolute value of the differences. When the weights are provided they are used in estimating the empirical cumulative distribution function. Note that when ref=TRUE the estimation of the density and of the empirical cumulative distribution are guided by the data in data.B.

The final output is the total variation distance, the overlap and the Hellinger distance calculated considering the transformed categorized variable. The breaks to categorize the variable are decided according to the Freedman-Diaconis rule ([nclass](#)) and, in this case, when ref=TRUE the IQR is estimated solely on data.B, whereas with ref=FALSE it is estimated by joining the two data sources. When present, the weights are used in estimating the relative frequencies of the categorized variable. For additional details on these distances please see ([comp.prop](#))

Value

A list object with four components.

summary	A matrix with summaries of xlab.A estimated on data.A and summaries of xlab.B estimated on data.B
diff.Qs	Average of absolute and squared differences between the quantiles of xlab.A estimated on data.A and the corresponding ones of xlab.B estimated on data.B
dist.ecdf	Dissimilarity measures between the estimated empirical cumulative distribution functions.
dist.discr	Distance between the distributions after discretization of the target variable.

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

References

Bellhouse D.R. and J. E. Stafford (1999). "Density Estimation from Complex Surveys". *Statistica Sinica*, **9**, 407–424.

See Also

[plotCont](), [comp.prop]()

Examples

```
data(samp.A)
data(samp.B)

comp.cont(data.A = samp.A, data.B = samp.B, xlab.A = "age")

comp.cont(data.A = samp.A, data.B = samp.B, xlab.A = "age",
          w.A = "ww", w.B = "ww")
```

---

comp.prop   *Compares two distributions of the same categorical variable*

Description

This function compares two (estimated) distributions of the same categorical variable(s).

Usage

```
comp.prop(p1, p2, n1, n2=NULL, ref=FALSE)
```

Arguments

p1   A vector or an array containing relative or absolute frequencies for one or more
                categorical variables. Usually it is the output of the function [xtabs]() or [table]().

p2   A vector or an array containing relative or absolute frequencies for one or more
                categorical variables. Usually it is the output of the function [xtabs]() or [table]().
                If ref = FALSE then p2 is a further estimate of the distribution of the categorical
                variable(s) being considered. On the contrary (ref = TRUE) it is the 'reference'
                distribution (the distribution considered true or a reliable estimate).

n1   The size of the sample on which p1 has been estimated.

n2   The size of the sample on which p2 has been estimated, required just when ref =
                FALSE (p2 is estimated on another sample and is not the reference distribution).

ref   Logical. When ref = TRUE, p2 is the reference distribution (true or reliable es-
                timate of distribution), on the contrary when ref = FALSE it an estimate of the
                distribution derived from another sample with sample size n2.

Details

This function computes some similarity or dissimilarity measures between marginal (joint) distribution of categorical variables(s). The following measures are considered:

Dissimilarity index or *total variation distance*:

$$\Delta_{12} = \frac{1}{2} \sum_{j=1}^{J} |p_{1,j} - p_{2,j}|$$

where $p_{s,j}$ are the relative frequencies ($0 \leq p_{s,j} \leq 1$). The dissimilarity index ranges from 0 (minimum dissimilarity) to 1. It can be interpreted as the smallest fraction of units that need to be reclassified in order to make the distributions equal. When p2 is the reference distribution (true or expected distribution under a given hypothesis) than, following the Agresti's rule of thumb (Agresti 2002, pp. 329–330) , values of $\Delta_{12} \leq 0.03$ denotes that the estimated distribution p1 follows the true or expected pattern quite closely.

Overlap between two distributions:

$$O_{12} = \sum_{j=1}^{J} min(p_{1,j}, p_{2,j})$$

It is a measure of similarity which ranges from 0 to 1 (the distributions are equal). It is worth noting that $O_{12} = 1 - \Delta_{12}$.

Bhattacharyya coefficient:

$$B_{12} = \sum_{j=1}^{J} \sqrt{p_{1,j} \times p_{2,j}}$$

It is a measure of similarity and ranges from 0 to 1 (the distributions are equal).

Hellinger's distance:

$$d_{H,12} = \sqrt{1 - B_{12}}$$

It is a dissimilarity measure ranging from 0 (distributions are equal) to 1 (max dissimilarity). It satisfies all the properties of a distance measure ($0 \leq d_{H,12} \leq 1$; symmetry and triangle inequality). Hellinger's distance is related to the dissimilarity index, and it is possible to show that:

$$d_{H,12}^2 \leq \Delta_{12} \leq d_{H,12}\sqrt{2}$$

Alongside with those similarity/dissimilarity measures the Pearson's Chi-squared is computed. Two formulas are considered. When p2 is the reference distribution (true or expected under some hypothesis, ref=TRUE):

$$\chi_P^2 = n_1 \sum_{j=1}^{J} \frac{(p_1, j - p_{2,j})^2}{p_{2,j}}$$

When `p2` is a distribution estimated on a second sample then:

$$\chi^2_P = \sum_{i=1}^{2} \sum_{j=1}^{J} n_i \frac{\left(p_{i,j} - p_{+,j}\right)^2}{p_{+,j}}$$

where $p_{+,j}$ is the expected frequency for category j, obtained as follows:

$$p_{+,j} = \frac{n_1 p_{1,j} + n_2 p_{2,j}}{n_1 + n_2}$$

being n_1 and n_2 the sizes of the samples.

The Chi-Square value can be used to test the hypothesis that two distributions are equal ($df = J-1$). Unfortunately such a test would not be useful when the distribution are estimated from samples selected from a finite population using complex selection schemes (stratification, clustering, etc.). In such a case different alternative corrected Chi-square tests are available (cf. Sarndal et al., 1992, Sec. 13.5). One possibility consist in dividing the Pearson's Chi-square test by the *generalised design effect* of both the surveys. Its estimation is not straightforward (sampling design variables need to be available). Generally speacking, the generalised design effect is smaller than 1 in the presence of stratified random sampling designs, while it exceeds 1 the presence of a two stage cluster sampling design. For the purposes of analysis it is reported the value of the generalised design effect g that would determine the acceptance of the null hypothesis (equality of distributions) in the case of $\alpha = 0.05$ ($df = J - 1$), i.e. values of g such that

$$\frac{\chi^2_P}{g} \leq \chi^2_{J-1,0.05}$$

Value

A `list` object with two or three components depending on the argument `ref`.

`meas`	A vector with the measures of similarity/dissimilarity between the distributions: dissimilarity index (`"tvd"`), overlap (`"overlap"`), Bhattacharyya coefficient (`"Bhatt"`) and Hellinger's distance (`"Hell"`).
`chi.sq`	A vector with the following values: Pearson's Chi-square (`"Pearson"`), the degrees of freedom (`"df"`), the percentile of a Chi-squared distribution (`"q0.05"`) and the largest admissible value of the generalised design effect that would determine the acceptance of H0 (equality of distributions).
`p.exp`	When `ref=FALSE` it is reported the value of the reference distribution $p_{+,j}$ estimated used in deriving the Chi-square statistic and also the dissimilarity index. On the contrary (`ref=FALSE`) it is set equal to the argument p2.

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

References

Agresti A (2002) *Categorical Data Analysis. Second Edition*. Wiley, new York.

Sarndal CE, Swensson B, Wretman JH (1992) *Model Assisted Survey Sampling*. Springer–Verlag, New York.

Examples

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

# split quine in two subsets
suppressWarnings(RNGversion("3.5.0"))
set.seed(124)
lab.A <- sample(nrow(quine), 70, replace=TRUE)
quine.A <- quine[lab.A, c("Eth","Sex","Age")]
quine.B <- quine[-lab.A, c("Eth","Sex","Age")]

# compare est. distributions from 2 samples
# 1 variable
tt.A <- xtabs(~Age, data=quine.A)
tt.B <- xtabs(~Age, data=quine.B)
comp.prop(p1=tt.A, p2=tt.B, n1=nrow(quine.A), n2=nrow(quine.B), ref=FALSE)

# joint distr. of more variables
tt.A <- xtabs(~Eth+Sex+Age, data=quine.A)
tt.B <- xtabs(~Eth+Sex+Age, data=quine.B)
comp.prop(p1=tt.A, p2=tt.B, n1=nrow(quine.A), n2=nrow(quine.B), ref=FALSE)

# compare est. distr. with a one considered as reference
tt.A <- xtabs(~Eth+Sex+Age, data=quine.A)
tt.all <- xtabs(~Eth+Sex+Age, data=quine)
comp.prop(p1=tt.A, p2=tt.all, n1=nrow(quine.A), n2=NULL, ref=TRUE)
```

---

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 *recipient* in the statistical matching application.

data.don   A matrix or data frame that plays the role of *donor* 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 NND.hotdeck, RANDwNND.hotdeck, rankNND.hotdeck, and mixed.mtc.

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 <mdo.statmatch@gmail.com>

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"))
```

---

create.imputed	*Fills-in missing values in the recipient dataset with values observed on the donors units*

Description

Imputes the missing values (NAs) in the recipient dataset with values observed on the donors units after search of donors with NND or random hotdeck.

Usage

```
create.imputed(data.rec, data.don, mtc.ids)
```

Arguments

data.rec	A matrix or data frame that has missing values.
data.don	A matrix or data frame that is used for donation (imputation).

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 corre-
             sponding donor record (row) in data.don. Note that this type of matrix is re-
             turned by the functions NND.hotdeck, RANDwNND.hotdeck, rankNND.hotdeck,
             and mixed.mtc.

Details

This function allows to fill in the missing values (NAs) in the recipient with values observed in the
donor data set after the search of donors via NND or random hotdeck with available functions in
the package, i.e. NND.hotdeck, RANDwNND.hotdeck, rankNND.hotdeck, and mixed.mtc. When
the same recorc in the recipient dataset presents 2 or more NAs they all will be replaced with values
observed on the chosen donor for that unit; this corresponds to joint hotdeck imputation.

Value

The data frame data.rec missing values (NAs) filled in.

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

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

```
# introduce missing values
# in Petal.Length of iris dataset

set.seed(13579)
pos <- sample(x = 1:nrow(iris), size = 15,
              replace = FALSE)


iris.rec <- iris[pos, ]  # recipient data.frame with missing values
iris.rec[, "Petal.Length"] <- NA

iris.don <- iris[-pos, ] # donor data.frame ALL observed

# find the closest donors using NND hot deck;
# distances are computed on "Petal.Width"
# donors only of the same Specie

out.NND <- NND.hotdeck(data.rec=iris.rec, data.don=iris.don,
```

```
                               match.vars=c("Petal.Width"),
                               don.class="Species")

# impute missing
iris.rec.imp <- create.imputed(data.rec=iris.rec, data.don=iris.don,
                            mtc.ids=out.NND$mtc.ids)
summary(iris.rec.imp$Petal.Length)
```

---

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, while numeric variables remain unchanged.

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 the corresponding 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. Note that if a factor includes a missing values (NA) then all the associated dummies will report an NA in correspondence of the missing observation (row).

Value

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

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

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)
head(ir.mat)
```

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 average width of conditional bounds.

Usage

```
Fbwidths.by.x(tab.x, tab.xy, tab.xz, deal.sparse="discard",
          nA=NULL, nB=NULL, ...)
```

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). 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.

deal.sparse
Text, how to estimate the cell relative frequencies when dealing with too sparse tables. When deal.sparse="discard" (default) no estimation is performed if tab.xy or tab.xz is too sparse. When deal.sparse="relfreq" the standard estimator (cell count divided by the sample size) is considered. Note that here sparseness is measured by number of cells with respect to the sample size; sparse table are those where the number of cells exceeds the sample size (see Details).

nA
Integer, sample size of file A used to estimate tab.xy. If NULL, it is obtained as sum of frequencies intab.xy.

nB
Integer, sample size of file B used to estimate tab.xz. If NULL, it is obtained as sum of frequencies intab.xz.

...
Additional arguments that may be required when deriving an estimate of uncertainty by calling [Frechet.bounds.cat](#).

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(Y|**X**) and P(Z|**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 uncertainty is measured by the average of the widths of the bounds for the cells in the table Y vs. Z:

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

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

Provided that uncertainty, measured in terms of $\bar{d}$, tends to reduce when conditioning on a higher number of **X** variables. Two penalties are introduced to account for the additional number of cells to be estimated when adding a X variable. The first penalty, introduced in D'Orazio et al. (2017), is:

$$g_1 = log\left(1 + \frac{H_{D_m}}{H_{D_Q}}\right)$$

Where H_{D_m} is the number of cell in the table obtained by crossing the given subset of **X** variables and the H_{D_Q} is the number of cell in the table achieved by crossing all the available **X** variables. A second penalty takes into account the number of cells to estimate with respect to the sample size (D'Orazio et al., 2019). It is obtained as:

$$g_2 = max \left[\frac{1}{n_A - H_{D_m} \times J}, \frac{1}{n_B - H_{D_m} \times K} \right]$$

with $n_A > H_{D_m} \times J$ and $n_B > H_{D_m} \times K$. In practice, it is considered the number of cells to estimate compared to the sample size. This criterion is considered to measure sparseness too. In particular, for the purposes of this function, tables are NOT considered sparse when:

$$min \left[\frac{n_A}{H_{D_m} \times J}, \frac{n_B}{H_{D_m} \times K} \right] > 1$$

This rule is applied when deciding how to proceed with estimation in case of sparse table (argument `deal.sparse`). Note that sparseness can be measured in different manners. The outputs include also the empty cells in each table (due to statistical zeros or structural zeros) and the Cohen's effect size with respect to the case of uniform distribution of frequencies across cells (the value 1/no.of.cells in every cell):

$$\omega_{eq} = \sqrt{H \sum_{h=1}^{H} (\hat{p}_h - 1/H)^2}$$

values of ω_{eq} jointly with $n/H \leq 1$ usually indicate severe sparseness.

Value

A list with the estimated bounds for the cells in the table of Y vs. Z for each possible subset of the **X** variables. The final component in the list, `sum.unc`, is a data.frame that summarizes the main results. In particular, it reports the number of **X** variables (`"x.vars"`), the number of cells in each of the input tables and the cells with frequency equal to 0 (columns ending with `freq0`). Moreover, it reported the value (`"av.n"`) of the rule used to decide whether we are dealing with a sparse case (see Details) and the Cohen's effect size measured for the table crossing the considered combination of the X variables. Finally, it is provided the average width of the uncertainty intervals (`"av.width"`), the penalty terms g1 and g2 (`"penalty1"` and `"penalty2"` respectively), and the penalized average widths (`"av.width.pen1"` and `"av.width.pen2"`, where av.width.pen1=av.width+pen1 and av.width.pen2=av.width+pen2).

Author(s)

Marcello D'Orazio <`mdo.statmatch@gmail.com`>

References

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

D'Orazio, M., Di Zio, M. and Scanu, M. (2017). "The use of uncertainty to choose matching variables in statistical matching". *International Journal of Approximate Reasoning* , 90, pp. 433-440.

D'Orazio, M., Di Zio, M. and Scanu, M. (2019). "Auxiliary variable selection in a a statistical matching problem". In Zhang, L.-C. and Chambers, R. L. (eds.) *Analysis of Integrated Data*, Chapman & Hall/CRC (Forthcoming).

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
suppressWarnings(RNGversion("3.5.0"))
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 Fbwidths.by.x()
freq.xA <- xtabs(~Eth+Sex+Age, data=quine.A)
freq.xB <- xtabs(~Eth+Sex+Age, data=quine.B)

freq.xy <- xtabs(~Eth+Sex+Age+Lrn, data=quine.A)
freq.xz <- xtabs(~Eth+Sex+Age+c.Days, data=quine.B)

# apply Fbwidths.by.x()
bounds.yz <- Fbwidths.by.x(tab.x=freq.xA+freq.xB, tab.xy=freq.xy,
        tab.xz=freq.xz)

bounds.yz$sum.unc
```

| 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", align.margins = FALSE,
                        tol= 0.001, warn = TRUE)
```

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).
                 When tab.x = NULL then only tab.xy and tab.xz must be supplied.

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 variable 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. Usually, it is assumed that the
                 joint distribution of the **X** variables computed from tab.xy is equal to tab.x (a
                 warning appears if any absolute difference is greater than tol). Note that when
                 marginal distribution of **X** in tab.xy is not equal to that of tab.x it is possible
                 to ask their alignment (see argument align.margins).

                 When tab.x = NULL then tab.xy should be a one–dimensional table providing
                 the marginal distribution of the Y variable.

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. Usually, it is assumed that the joint distri-
                 bution of the **X** variables computed from tab.xz is equal to tab.x (a warning
                 appears if any absolute difference is greater than tol). Note that when marginal
                 distribution of **X** in tab.xz is not equal to that of tab.x it is possible to ask their
                 alignment (see argument align.margins).

                 When tab.x = NULL then tab.xz should be a one–dimensional table providing
                 the marginal distribution of the Z variable.

print.f   A string: when print.f="tables" (default) all the cells' estimates will be
                 saved as tables in a list. On the contrary, if print.f="data.frame", they will
                 be saved as columns of a data.frame.

align.margins   Logical (default FALSE). When when TRUE the distribution of **X** variables in
                 tab.xy is aligned with the distribution resulting from tab.x, without affect-
                 ing the marginal distribution of Y. Similarly, the distribution of **X** variables in
                 tab.xz is aligned with the distribution resulting from tab.x without affecting
                 the marginal distribution of Z. The alignment is performed by running IPF algo-
                 rithm as implemented in the function [Estimate](#) in the package **mipfp**. Note that
                 to avoid lack of convergence due to combinations of Xs encountered in one table
                 but not in the other (statistical 0s), before running IPF a small constant (1e-06)
                 is added to empty cells in tab.xy and tab.xz.

tol	Tolerance used in comparing joint distributions as far as **X** variables are considered (default `tol= 0.001`); estimation of cells bounds would require that distribution of **X** variables computed from `tab.xy` and `tab.xz` should be approximately equal to that in `tab.x`, on contrary incoherences in estimated cells' bounds could happen. In case of not-coherent marginal distributions it is suggested to get them aligned by setting `align.margins=TRUE`.
warn	Logical, when `TRUE` (default) return warnings when marginal distributions of **X** variables show differences grater than `tol`.

Details

This function permits to compute the expected conditional 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 expected conditional bounds for the relative frequencies $p_{j,k}$ in the table Y vs. Z are:

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

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

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

Estimation requires that all the starting tables share the same marginal distribution of the **X** variables.

This function returns also the unconditional bounds for the relative frequencies in the contingency table of Y vs. Z, i.e. computed also without considering the **X** variables:

$$\max\{0; p_{Y=j} + p_{Z=k} - 1\} \le p_{Y=j,Z=k} \le \min\{p_{Y=j}; p_{Z=k}\}$$

These bounds represent the unique output when `tab.x = NULL`.

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

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

When `tab.x = NULL` then it is also provided the expected table under the assumption of independence between Y and Z:

$$p_{Y=j,Z=k} = p_{Y=j} \times p_{Z=k}.$$

The presence of too many cells with 0s in the input contingency tables is an indication of sparseness; this is an unappealing situation when estimating the cells' relative frequencies needed to derive the bounds; in such cases the corresponding results may be unreliable. A possible alternative way of working consists in estimating the required parameters by considering a pseudo-Bayes estimator (see pBayes); in practice the input `tab.x`, `tab.xy` and `tab.xz` should be the ones provided by the pBayes function.

Value

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

low.u	The estimated lower bounds for the relative frequencies in the table Y vs. Z without conditioning on the **X** variables.
up.u	The estimated upper bounds for the relative frequencies in the table Y vs. Z without conditioning on the **X** variables.
CIA	The estimated relative frequencies in the table Y vs. Z under the Conditional Independence Assumption (CIA).
low.cx	The estimated lower bounds for the relative frequencies in the table Y vs. Z when conditioning on the **X** variables.
up.cx	The estimated upper bounds for the relative frequencies in the table Y vs. Z when conditioning on the **X** variables.
uncertainty	The uncertainty associated to input data, measured in terms of average width of uncertainty bounds with and without conditioning on the **X** variables.

When `print.f="data.frame"` the output list contains just two components:

bounds	A data.frame whose columns reports the estimated uncertainty bounds.
uncertainty	The uncertainty associated to input data, measured in terms of average width of uncertainty bounds with and without conditioning on the **X** variables.

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

References

D'Orazio, M., Di Zio, M. and Scanu, M. (2006) "Statistical Matching for Categorical Data: Displaying Uncertainty and Using Logical Constraints", *Journal of Official Statistics*, 22, pp. 137–157.

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
suppressWarnings(RNGversion("3.5.0"))
set.seed(7654)
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.xA <- xtabs(~Sex+Age, data=quine.A)
freq.xB <- xtabs(~Sex+Age, data=quine.B)

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.xA+freq.xB, tab.xy=freq.xy,
        tab.xz=freq.xz, print.f="data.frame")
bounds.yz

# harmonize distr. of Sex vs. Age during computations
# in Frechet.bounds.cat()

#compare marg. distribution of Xs in A and B vs. pooled estimate
comp.prop(p1=margin.table(freq.xy,c(1,2)), p2=freq.xA+freq.xB,
          n1=nrow(quine.A), n2=nrow(quine.A)+nrow(quine.B), ref=TRUE)

comp.prop(p1=margin.table(freq.xz,c(1,2)), p2=freq.xA+freq.xB,
          n1=nrow(quine.A), n2=nrow(quine.A)+nrow(quine.B), ref=TRUE)

bounds.yz <- Frechet.bounds.cat(tab.x=freq.xA+freq.xB, tab.xy=freq.xy,
        tab.xz=freq.xz, print.f="data.frame", align.margins=TRUE)
bounds.yz
```

---

gower.dist   *Computes the Gower's Distance*

Description

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

Usage

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

Arguments

data.x   A matrix or a data frame containing variables that should be used in the compu-
                tation of the distance.

                Columns of mode numeric will be considered as interval scaled variables; columns
                of mode character or class factor will be considered as categorical nominal
                variables; columns of class ordered will be considered as categorical ordinal

variables and, columns of mode `logical` will be considered as binary asymmetric variables (see Details for further information).

Missing values (NA) are allowed.

If only `data.x` is supplied, the dissimilarities between rows of `data.x` will be computed.

data.y | A numeric matrix or data frame with the same variables, of the same type, as those in `data.x`. Dissimilarities between rows of `data.x` and rows of `data.y` will be computed. If not provided, by default it is assumed equal to `data.x` and only dissimilarities between rows of `data.x` will be computed.

rngs | A vector with the ranges to scale the variables. Its length must be equal to number of variables in `data.x`. In correspondence of non-numeric variables, just put 1 or NA. When `rngs=NULL` (default) the range of a numeric variable is estimated by jointly considering the values for the variable in `data.x` and those in `data.y`. Therefore, assuming `rngs=NULL`, if a variable "X1" is considered:

```
rngs["X1"] <- max(data.x[,"X1"], data.y[,"X1"]) -
                min(data.x[,"X1"], data.y[,"X1"])
```

.

KR.corr | When `TRUE` (default) the extension of the Gower's dissimilarity measure proposed by Kaufman and Rousseeuw (1990) is used. Otherwise, when `KR.corr=FALSE`, the Gower's (1971) formula is considered.

var.weights | By default (NULL) each variable has the same weight (value 1) when calculating the overall distance (weighted average of distances on single variables; see Details). User can specify different weights for the different variables by providing a numeric value for each of the variables contributing to the distance. In other words, `var.weights` should be set equal to a numeric vector having length equal to the number of variables considered in calculating distance. Entered weights are scales to sum up to 1.

robcb | By default is (NULL). If `robcb="boxp"` the scaling of the Manhattan distance is done by using the difference between upper and lower fences of the Boxplot with k=3. In alternative, `robcb="asyboxp"` the scaling of the Manhattan distance is done by the difference between upper and lower fences of the modified Boxplot to accocunt for slight skewness. In this case scaled distances greater than 1 are set equal to 1. This option is suggested in the presence of outliers in the continuous variables.

Details

This function computes distances between 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:

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

In particular, d_{ijk} represents the distance between the *i*th and *j*th unit computed considering the *k*th variable, while w_k is the weight assigned to variable *k* (by default 1 for all the variables, unless different weights are provided by user with argument var.weights). Distance 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} = $ 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 δ_{ijk} is concerned:

- $\delta_{ijk} = 0$ if $x_{ik} = $ NA or $x_{jk} = $ NA;

- $\delta_{ijk} = 0$ if the variable is asymmetric binary and $x_{ik} = x_{jk} = 0$ or $x_{ik} = x_{jk} = $ FALSE;

- $\delta_{ijk} = 1$ in all the other cases.

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

Value

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

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

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 between observations in xx
dx <- gower.dist(xx)
head(dx)

# matrix of distances between first obs. in xx
# and the remaining ones
gower.dist(data.x=xx[1:6,], data.y=xx[7:10,], var.weights = c(1,2,5,2))
```

---

harmonize.x	*Harmonizes the marginal (joint) distribution of a set of variables observed independently in two sample surveys referred to the same target population*

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 suggested 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 corresponding sampling design. This object can
               be created by using the function [svydesign]() in the package **survey**.

svy.B   A svydesign R object that stores the data collected in the the survey B and all
               the information concerning the corresponding sampling design. This object can
               be created by using the function [svydesign]() in the package **survey**.

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, the formula form.x=~x1:x2-1 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). 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 es-
                timated 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 linear calibration is performed (cal.method="linear").
                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 ad-
                justments are carried out by considering the function
                postStratify in the package **survey**.

...   Further arguments that may be necessary for calibration or post-stratification.
                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 **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 harmo-
nization is carried out according to a procedure suggested 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 **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 **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 **X** variables.

Post-stratification is a special case of calibration; all the weights of the units in a given post-stratum are modified so as to 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 with list the results of calibration procedures carried out on survey A and survey B, respectively. In particular the following components will be provided:

cal.A   The survey object svy.A after the calibration; in particular, the weights now are calibrated with respect to the totals of the **X** variables.

cal.B   The survey object svy.B after the calibration; in particular, the weights now are calibrated with respect to the totals of the **X** variables.

weights.A   The new calibrated weights associated to the the units in svy.A.

weights.B   The new calibrated weights associated to the the units in svy.B.

call   Stores the call to this function with all the values specified for the various arguments (call=match.call()).

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

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
suppressWarnings(RNGversion("3.5.0"))
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)
# Harmonizazion of the distr. of Sex vs. Age
# usign poststratification

# (1.a) known population totals
# the population toatal 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)
```

```
# Harmonizazion wrt the maginal 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)
# Harmonizazion 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)
```

---

mahalanobis.dist   *Computes the Mahalanobis Distance*

Description

This function computes the Mahalanobis distance among units in a dataset or between 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 between units. 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 (i.e. vc = 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. vc = var(rbind(data.x, data.y))).

Details

The Mahalanobis distance is calculated by means of:

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

The covariance matrix *S* is estimated from the available data when vc=NULL, 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 <mdo.statmatch@gmail.com>

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   *Computes the Maximum Distance*

Description

This function computes the Maximum distance (or L^∞ norm) between units in a dataset or between 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 com-
                putation 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 are com-
                puted.

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 be-
                tween 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 rank func-
                tion).

Details

This function computes the L^∞ distance also know as *minimax* distance. In practice the distance
between two records is the maximum of the absolute differences on the available variables:

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

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 between rows of data.x and those of data.y.

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

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=NULL, micro=FALSE, constr.alg="Hungarian")
```

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 re-coded into dummies. Missing values (NA) are not allowed.
data.don	A matrix or data frame that plays the role of *donor* 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 match.vars and z.don. Note that continuous variables are expected, if there are some categorical variables they are re-coded 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 ($\mathbf{X}$).
y.rec	A character vector with the name of the target variable Y that is observed only for units in data.rec. 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 data.don. 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. $\mathbf{X}$ and Z vs. $\mathbf{X}$. Maximum Likelihood method is used when method="ML" (default); on the contrary, when method="MS" 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 between the Y (y.rec) and the Z variable (z.don) that are not jointly observed. When method="MS" then the argument cor.yz must specify the value of the correlation coefficient ρ_{YZ}; on the contrary, when method="ML", it must specify the *partial correlation coefficient* between Y and Z given $\mathbf{X}$ ($\rho_{YZ|\mathbf{X}}$).

By default (rho.yz=NULL). 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 between Y and Z given $\mathbf{X}$ (Conditional Independence Assumption, CIA), i.e. $\rho_{YZ|\mathbf{X}} = 0$.

micro   Logical. When micro=FALSE (default) only the parameters' estimates are returned. On the contrary, when micro=TRUE the function returns also data.rec filled in with the values for the variable Z. The donors for filling in Z in data.rec are identified using a constrained distance hot deck method. In this case, the number of units (rows) in data.don must be grater or equal to the number of units (rows) in data.rec. See next argument and Details for further information.

constr.alg   A string that has to be specified when micro=TRUE, in order to solve the transportation problem involved by the constrained distance hot deck method. Two choices are available: "lpSolve" and "Hungarian". 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="Hungarian" (default) the transportation problem is solved using the Hungarian method implemented in the function solve_LSAP available in the package **clue** (Hornik, 2012). Note that Hungarian algorithm is more efficient and requires less processing time.

Details

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

(1) adoption of a parametric model for the joint distribution of $(\mathbf{X}, Y, Z)$ and estimation of its parameters;

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

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

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

step 1b) *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 $a = 1, \ldots, n_A$, 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 $b = 1, \ldots, n_B$, 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 2) *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 1a) 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 1a). 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.

mu	The estimated mean vector.
vc	The estimated variance–covariance matrix.
cor	The estimated correlation matrix.
res.var	A vector with estimates of the residual variances $\sigma_{Y|Z\mathbf{X}}$ and $\sigma_{Z|Y\mathbf{X}}$.
start.prho.yz	It is the initial guess for the partial correlation coefficient $\rho_{YZ|\mathbf{X}}$ passed in input via the rho.yz argument when method="ML".
rho.yz	Returned in output only when method="MS". It is a vector with four values: the initial guess for ρ_{YZ}; the lower and upper bounds for $\hat{\rho}_{YZ}$ in the statistical matching framework given the correlation coefficients between Y and $\mathbf{X}$ and the correlation coefficients between Z and $\mathbf{X}$ estimated from the available data; and, finally, the closest admissible value used in computations instead of the initial rho.yz that resulted not coherent with the others correlation coefficients estimated from the available data.
phi	When method="MS". Estimates of the ϕ terms introduced by Moriarity and Scheuren (2001 and 2003).

filled.rec	The `data.rec` filled in with the values of Z. It is returned only when `micro=TRUE`.
mtc.ids	when `micro=TRUE`. This is 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 between each recipient unit and the corresponding donor, returned only in case `micro=TRUE`.
call	How the function has been called.

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

References

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.

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

Hornik K. (2012). clue: Cluster ensembles. R package version 0.3-45. https://CRAN.R-project.org/package=clue.

Moriarity, C., and Scheuren, F. (2001). "Statistical matching: a paradigm for assessing the uncertainty in the procedure". *Journal of Official Statistics*, **17**, 407–422.

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
suppressWarnings(RNGversion("3.5.0"))
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="Hungarian")

# 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)
```

NND.hotdeck   *Distance Hot Deck method.*

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="Hungarian",
            k=1, keep.t=FALSE, ...)
```

Arguments

data.rec   A matrix or data frame that plays the role of *recipient*. This data frame must
              contain the variables (columns) that should be used, directly or indirectly, in the
              matching application (specified via match.vars and eventually don.class).
              Missing values (NA) are allowed.

data.don   A matrix or data frame that plays the role of *donor*. 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 (specified via match.vars and even-
              tually don.class).

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 between records
                    (rows) in data.rec and those in data.don. The variables used in computing
                    distances may contain missing values but only a limited number of distance
                    functions can handle them (see Details for clarifications).

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.

                    The variables chosen for the creation of the donation classes should NOT contain
                    missing values (NAs).

                    When not specified (default), no donation classes are used. This choice may re-
                    quire more memory to store a larger distance matrix and a higher computational
                    effort.

dist.fun   A string with the name of the distance function that has to be used. The fol-
                    lowing distances are allowed: "Manhattan" (aka "City block"; default), "Eu-
                    clidean", "Mahalanobis","exact" or "exact matching", "Gower", "minimax" or
                    one of the distance functions available in the package **proxy**. Note that the
                    distance is computed using the function [dist](#) of the package **proxy** with the ex-
                    ception of the "Gower" (see function [gower.dist](#) for details), "Mahalanobis"
                    (function [mahalanobis.dist](#)) and "minimax" (see [maximum.dist](#)) cases.

                    When dist.fun="Manhattan", "Euclidean", "Mahalanobis" or "minimax"
                    all the matching variables in data.rec and data.don must be numeric. When
                    dist.fun="exact" or dist.fun="exact matching", all the variables in data.rec
                    and data.don will be converted to character and, as far as the distance compu-
                    tation is concerned, they will be treated as categorical nominal variables, i.e.
                    distance is 0 if a couple of units presents the same response category and 1
                    otherwise.

constrained   Logical. When constrained=FALSE (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 k
                    times. In this case, the set of donors is selected by solving an optimization prob-
                    lem, whose goal is 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 "hungarian". In the first case, constr.alg="lpSolve",
                    the optimization problem is solved by means of the function [lp.transport](#)
                    available in the package **lpSolve**. When constr.alg="hungarian" (default)
                    the problem is solved using the Hungarian method, implemented in function
                    [solve_LSAP](#) available in the package **clue**. Note that Hungarian algorithm is
                    faster and more efficient if compared to constr.alg="lpSolve" but it allows
                    selecting a donor just once, i.e. k = 1 .

k   The number of times that a unit in data.don can be selected as a donor when
                    constrained=TRUE (default k = 1). When k>1 then optimization problem can

be solved by setting constr.alg="lpSolve". Hungarian algorithm (constr.alg="hungarian") can be used only when k = 1.

keep.t   Logical, when donation classes are used by setting keep.t=TRUE prints infor-
               mation on the donation classes being processed (by default keep.t=FALSE).

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

Details

This function finds a donor record in data.don for each record in data.rec. In the unconstrained case, it searches for the closest donor record 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 a donor can be used just a fixed number of times, as specified by the k argument, but the whole set of donors is chosen in order to minimize the overall matching distance. When k=1 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 target variable; on the contrary, data.don is the part of the data set without missing values on it. See R code in the Examples for details.

Please note that only "Gower" and "minimax" distance functions allow for the presence of missing values (NAs) in the variables used in computing distances. In both the cases when one of the of the observations presents a variable showing an NA, then this variable is excluded from the computation of distance between them.

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 between 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 <mdo.statmatch@gmail.com>

References

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

Hornik K. (2012). clue: Cluster ensembles. R package version 0.3-45. https://CRAN.R-project.org/package=clue.

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

```
# create 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")
head(fused.1)

# 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, "Hungarian" algorithm

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

# 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)
head(final)
```

---

pBayes   *Pseudo-Bayes estimates of cell probabilities*

Description

Estimation of cells counts in contingency tables by means of the pseudo-Bayes estimator.

Usage

```
pBayes(x, method="m.ind", const=NULL)
```

Arguments

x   A contingency table with observed cell counts. Typically the output of [table](#) or
            [xtabs](#). More in general an R `array` with the counts.

method   The method for estimating the final cell frequencies. The following options are
            available:

            method = "Jeffreys", consists in adding 0.5 to each cell before estimation of
            the relative frequencies.

            method = "minimax", consists in adding $\sqrt{(n)}/c$ to each cell before estimation
            of the relative frequencies, being n the sum of all the counts and c the number
            of cells in the table.

            method = "invcat", consists in adding $1/c$ to each cell before estimation of the
            relative frequencies.

            method = "user", consists in adding a used defined constant a ($a > 0$) to each
            cell before estimation of the relative frequencies. The constant a should be
            passed via the argument const.

            method = "m.ind", the prior guess for the unknown cell probabilities is obtained
            by considering estimated probabilities under the mutual independence hypoth-
            esis. This option is available when dealing with at least two-way contingency
            tables
            (length(dim(x))>=2).

            method = "h.assoc", the prior guess for the unknown cell probabilities is ob-
            tained by considering estimated probabilities under the homogeneous associa-
            tion hypothesis. This option is available when dealing with at least two-way
            contingency tables (length(dim(x))>=2).

const   Numeric value, a user defined constant a ($a > 0$) to be added to each cell before
            estimation of the relative frequencies when method = "user". As a general rule
            of thumb, it is preferable to avoid that the sum of constant over all the cells is
            greater than $0.20 \times n$.

Details

This function estimates the frequencies in a contingency table by using the pseudo-Bayes approach.
In practice the estimator being considered is a weighted average of the input (observed) cells counts
n_h and a suitable prior guess, γ_h, for cells probabilities :

$$\tilde{p}_h = \frac{n}{n+K}\hat{p}_h + \frac{K}{n+K}\gamma_h$$

K depends on the parameters of Dirichlet prior distribution being considered (for major details see
Chapter 12 in Bishop et al., 1974). It is worth noting that with a constant prior guess $\gamma_h = 1/c$
($h = 1, 2, \cdots, c$), then $K = 1$ and in practice corresponds to adding $1/c$ to each cell before
estimation of the relative frequencies (method = "invcat"); $K = c/2$ when the constant 0.5 is
added to each cell (method = "Jeffreys"); finally $K = \sqrt{n}$ when the quantity $\sqrt{n}/c$ is added to

each cell (`method = "minimax"`). All these cases corresponds to adding a flattening constant; the higher is the value of K the more the estimates will be shrinked towards $\gamma_h = 1/c$ (flattening).

When `method = "m.ind"` the prior guess γ_h is estimated under the hypothesis of mutual independence between the variables crossed in the initial contingency table x, supposed to be at least a two-way table. In this case the value of K is estimated via a data-driven approach by considering

$$\hat{K} = \frac{1 - \sum_h \hat{p}_h^2}{\sum_h \left(\hat{\gamma}_h - \hat{p}_h\right)^2}$$

On the contrary, when `method = "h.assoc"` the prior guess γ_h is estimated under the hypothesis of homogeneous association between the variables crossed in the initial contingency table x.

Please note that when the input table is estimated from sample data where a weight is assigned to each unit, the weights should be used in estimating the input table, but it is suggested to rescale them so that their sum is equal to *n*, the sample size.

Value

A `list` object with three components.

`info`	A vector with the sample size `"n"`, the number of cells (`"no.cells"`) in x, the average cell frequency (`"av.cfr"`), the number of cells showing frequencies equal to zero (`"no.0s"`), the `const` input argument, the chosen/estimated K (`"K"`) and the relative size of K, i.e. $K/(n + K)$ (`"rel.K"`).
`prior`	A table having the same dimension as x with the considered prior values for the cell frequencies.
`pseudoB`	A table with having the same dimension as x providing the pseudo-Bayes estimates for the cell frequencies in x.

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

References

Bishop Y.M.M., Fienberg, S.E., Holland, P.W. (1974) *Discrete Multivariate Analysis: Theory and Practice.* The Massachusetts Institute of Technology

Examples

```
data(samp.A, package="StatMatch")
tab <- xtabs(~ area5 + urb + c.age + sex + edu7, data = samp.A)
out.pb <- pBayes(x=tab, method="m.ind")
out.pb$info

out.pb <- pBayes(x=tab, method="h.assoc")
out.pb$info

out.pb <- pBayes(x=tab, method="Jeffreys")
out.pb$info
```

```
# usage of weights in estimating the input table
n <- nrow(samp.A)
r.w <- samp.A$ww / sum(samp.A$ww) * n   # rescale weights to sum up to n
tab.w <- xtabs(r.w ~ area5 + urb + c.age + sex + edu7, data = samp.A)
out.pbw <- pBayes(x=tab.w, method="m.ind")
out.pbw$info
```

plotBounds	*Graphical representation of the uncertainty bounds estimated through the* `Frechet.bounds.cat` *function*

Description

The function uses the output of the function `Frechet.bounds.cat` to produce a basic graphical representation of the uncertainty bounds related to the contingency table of Y vs. Z.

Usage

```
plotBounds(outFB)
```

Arguments

outFB   the list provided in output from `Frechet.bounds.cat`.

Details

This function represents graphically the uncertainty bounds estimated by the function `Frechet.bounds.cat` for each relative frequency in the contingency table of Y vs. Z. the dotted line indicates the width of the bounds estimated without conditioning on the Xs (the size is reported in parenthesis below the line). The full line indicates the width of the estimated bounds conditional on the Xs (expected conditional Frechet bounds for the relative frequencies in the contingency table of Y vs. Z (size reported below the line, not in the the parenthesis). Not that when the X are not used it is drawn only the width of the unconditional bounds and the size is shown below the line.

The figure on the top od the line indicated the estimated relative frequency under the assumption of independence between Y and Z conditional on one or more X variables (Conditional Independence Assumption, CIA; for details see help pages of `Frechet.bounds.cat`), otherwise it corresponds to the estimated relative frequency under the assumption of independence between Y and Z.

Value

The required graphical representation is drawn using standard **graphics** facilities.

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

See Also

Frechet.bounds.cat

Examples

```
# # compute the tables required by Frechet.bounds.cat()
# freq.xA <- xtabs(~sex+c.age, data=samp.A)
# freq.xB <- xtabs(~sex+c.age, data=samp.B)
# freq.xy <- xtabs(~sex+c.age+c.neti, data=samp.A)
# freq.xz <- xtabs(~sex+c.age+labour5, data=samp.B)
#
# # apply Frechet.bounds.cat()
# bounds.yz <- Frechet.bounds.cat(tab.x=freq.xA+freq.xB, tab.xy=freq.xy,
#                                 tab.xz=freq.xz, print.f="data.frame")
#
#
# plot.bounds(bounds.yz)
```

---

plotCont	*graphical comparison of the estimated distributions for the same continuous variable.*

Description

Compares graphically the estimated distributions for the same continuous variable using data coming from two different data sources.

Usage

```
plotCont(data.A, data.B, xlab.A, xlab.B=NULL, w.A=NULL, w.B=NULL,
         type="density", ref=FALSE)
```

Arguments

data.A	A dataframe or matrix containing the variable of interest xlab.A and eventual associated survey weights w.A.
data.B	A dataframe or matrix containing the variable of interest xlab.B and eventual associated survey weights w.B.
xlab.A	Character string providing the name of the variable in data.A whose distribution should be represented graphically and compared with that estimated from data.B.
xlab.B	Character string providing the name of the variable in data.B whose distribution should be represented graphically and compared with that estimated from data.A. If xlab.B=NULL (default) then it assumed xlab.B=xlab.A.

w.A	Character string providing the name of the optional weighting variable in data.A that, in case, should be used to estimate the distribution of xlab.A
w.B	Character string providing the name of the optional weighting variable in data.B that, in case, should be used to estimate the distribution of xlab.B
type	A character string indicating the type of graphical representation that should be used to compare the estimated distributions of xlab.A and xlab.B. By default (type="density") density plots are used. Other possible options are "ecdf", "qqplot", "qqshift" and "hist". See Details for more information.
ref	Logical, indicating whether the distribution estimated from data.B should be considered the reference or not. Default ref=FALSE. when Default ref=TRUE the estimation of the histograms, the density and the empirical cumulative distribution function are guided by data in data.B

Details

This function compares graphically the distribution of the same variable but estimated from data coming from two different data sources. The graphical comparison con be done in different manners. When type="hist" the continuous variable is categorized and the corresponding histograms, estimated from data.A and data.B, are compared. When present, the weights are used in estimating the relative frequencies. Note that the breaks to categorize the variable are decided according to the Freedman-Diaconis rule ([nclass](#)) and, in this case, with ref=TRUE the IQR is estimated solely on data.B, whereas with ref=FALSE it is estimated by joining the two data sources.

With type="density" the density plots are drawn; when available the weights are used in the estimation of the density that are based on the histograms (as suggested by Bellhouse and Stafford, 1999). Whentype="ecdf" the comparison relies on the empirical cumulative distribution function, that can be estimated considering the weights. Note that when ref=TRUE the estimation of the density and the empirical cumulative distribution are guided by the data in data.B.

The comparison is based on percentiles with type="qqplot" and type="qqshift". In the first case, the function draws a scatterplot (red dots) of the estimated percentiles of xlab.A vs. those of xlab.B; the dashed line indicated the ideal situation of equality of percentiles (points lying on the line). When type="qqshift" the scatterplot refers to (percentiles.A - percentiles.B) vs. percentiles.B; in this case the points lying on horizontal line passing through 0 indicate equality (difference equal to 0). Note that the number of estimated percentiles depends on the minimum between the two sample sizes. Only quartiles are calculated when min(n.A, n.B)<=50; quintiles are estimated when min(n.A, n.B)>50 and min(n.A, n.B)<=150; deciles are estimated when min(n.A, n.B)>150 and min(n.A, n.B)<=250; finally quantiles for probs=seq(from = 0.05,to = 0.95,by = 0.05) are estimated when min(n.A, n.B)>250. When survey weights are available (indicated through w.A and/or w.B) they are used in estimating the quantiles by calling the function [wtd.quantile](#) in the package **Hmisc**.

Value

The required graphical representation is drawn using the **ggplot2** facilities.

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

References

Bellhouse D.R. and J. E. Stafford (1999). "Density Estimation from Complex Surveys". *Statistica Sinica*, **9**, 407–424.

See Also

comp.cont

Examples

```
# plotCont(data.A = samp.A, data.B = samp.B, xlab.A="age")
# plotCont(data.A = samp.A, data.B = samp.B, xlab.A="age", w.A = "ww")
```

---

plotTab	*Graphical comparison of the estimated distributions for the same categorical variable.*

Description

Compares graphically the estimated distributions for the same categorical variable(s) using data coming from two different data sources.

Usage

```
plotTab(data.A, data.B, xlab.A, xlab.B=NULL, w.A=NULL, w.B=NULL)
```

Arguments

data.A	A dataframe or matrix containing the variable of interest xlab.A and eventual associated survey weights w.A.
data.B	A dataframe or matrix containing the variable of interest xlab.B and eventual associated survey weights w.B.
xlab.A	Character string providing the name(s) of one or more variables in data.A whose (joint) distribution should be represented graphically and compared with that estimated from data.B.
xlab.B	Character string providing the name(s) of one or more variables in data.A whose (joint) distribution should be represented graphically and compared with that estimated from data.A. If xlab.B=NULL (default) then it assumed xlab.B=xlab.A.
w.A	Character string providing the name of the optional weighting variable in data.A that, in case, should be used to estimate the distribution of xlab.A
w.B	Character string providing the name of the optional weighting variable in data.B that, in case, should be used to estimate the distribution of xlab.B

Details

This function compares graphically the (joint) distribution of the same variables but estimated from data coming from two different data sources. The graphical comparison is done using barcharts.

Value

The required graphical representation is drawn using the **ggplot2** facilities.

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

See Also

comp.prop

Examples

```
# plotTab(data.A = samp.A, data.B = samp.B, xlab.A="edu7", w.A = "ww")
# plotTab(data.A = samp.A, data.B = samp.B, xlab.A=c("urb", "sex"), w.A = "ww", w.B="ww")
```

---

pw.assoc	*Pairwise measures between categorical variables*

Description

This function computes some association and *Proportional Reduction in Error* (PRE) measures between a categorical nominal variable and each of the other available predictors (being also categorical variables).

Usage

```
pw.assoc(formula, data, weights=NULL, out.df=FALSE)
```

Arguments

formula   A formula of the type y~x1+x2 where y denotes the name of the categorical variable (a factor in **R**) which plays the role of the dependent variable, while x1 and x2 are the name of the predictors (both categorical variables). Numeric variables are not allowed; eventual numerical variables should be categorized (see function [cut](cut)) before being passed to pw.assoc.

data   The data frame which contains the variables called by formula.

weights
: The name of the variable in `data` which provides the units' weights. Weights are used to estimate frequencies (a cell frequency is estimated by summing the weights of the units which present the given characteristic). Default is `NULL` (no weights available) and each unit counts 1. When case weight are provided, then they are scales so that their sum equals *n*, the sample size (assumed to be `nrow(data)`).

out.df
: Logical. If `NULL` measures will be organized in a data frame (a column for each measure).

Details

This function computes some association, PRE measures, AIC and BIC for each couple response-predictor that can be created starting from argument `formula`. In particular, a two-way contingency table $X \times Y$ is built for each available X variable (X in rows and Y in columns); then the following measures are considered.

Cramer's *V*:

$$V = \sqrt{\frac{\chi^2}{n \times min\left[I - 1, J - 1\right]}}$$

n is the sample size, *I* is the number of rows (categories of X) and *J* is the number of columns (categories of Y). Cramer's *V* ranges from 0 to 1.

Bias-corrected Cramer's *V* (V_c) proposed by Bergsma (2013).

Mutual information:

$$I(X;Y) = \sum_{i,j} p_{ij} \, log\left(\frac{p_{ij}}{p_{i+}p_{+j}}\right)$$

equal to 0 in case of independence but with infinite upper bound, i.e. $0 \leq I(X;Y) < \infty$. In it $p_{ij} = n_{ij}/n$.

A normalized version of $I(X;Y)$, ranging from 0 (independence) to 1 and not affected by number of categories (*I* and *J*):

$$I(X;Y)^* = \frac{I(X;Y)}{min(H_X, H_Y)}$$

being H_X and H_Y the entropy of the variable X and Y, respectively.

Goodman-Kruskal $\lambda(Y|X)$ (i.e. response conditional on the given predictor):

$$\lambda(Y|X) = \frac{\sum_{i=1}^{I} max_j(p_{ij}) - max_j(p_{+j})}{1 - max_j(p_{+j})}$$

It ranges from 0 to 1, and denotes how much the knowledge of the row variable X (predictor) helps in reducing the prediction error of the values of the column variable Y (response).

Goodman-Kruskal $\tau(Y|X)$:

$$\tau(Y|X) = \frac{\sum_{i=1}^{I} \sum_{j=1}^{J} p_{ij}^2/p_{i+} - \sum_{j=1}^{J} p_{+j}^2}{1 - \sum_{j=1}^{J} p_{+j}^2}$$

It takes values in the interval [0,1] and has the same PRE meaning of the lambda.

Theil's uncertainty coefficient:

$$U(Y|X) = \frac{\sum_{i=1}^{I} \sum_{j=1}^{J} p_{ij} log(p_{ij}/p_{i+}) - \sum_{j=1}^{J} p_{+j} log p_{+j}}{-\sum_{j=1}^{J} p_{+j} log p_{+j}}$$

It takes values in the interval [0,1] and measures the reduction of uncertainty in the column variable Y due to knowing the row variable X. Note that the numerator of U(Y|X) is the mutual information I(X;Y)

It is worth noting that λ, τ and U can be viewed as measures of the proportional reduction of the variance of the Y variable when passing from its marginal distribution to its conditional distribution given the predictor X, derived from the general expression (cf. Agresti, 2002, p. 56):

$$\frac{V(Y) - E[V(Y|X)]}{V(Y)}$$

They differ in the way of measuring variance, in fact it does not exist a general accepted definition of the variance for a categorical variable.

Finally, AIC (and BIC) is calculated, as suggested in Sakamoto and Akaike (1977). In particular:

$$AIC(Y|X) = -2 \sum_{i,j} n_{ij} log\left(\frac{n_{ij}}{n_{i+}}\right) + 2I(J-1)$$

$$BIC(Y|X) = -2 \sum_{i,j} n_{ij} log\left(\frac{n_{ij}}{n_{i+}}\right) + I(J-1)log(n)$$

being $I(J-1)$ the parameters (conditional probabilities) to estimate. Note that the R package **catdap** provides functions to identify the best subset of predictors based on AIC.

Please note that the missing values are excluded from the tables and therefore excluded from the estimation of the various measures.

Value

When out.df=FALSE (default) a list object with four components:

V	A vector with the estimated Cramer's V for each couple response-predictor.
bcV	A vector with the estimated bias-corrected Cramer's V for each couple response-predictor.
mi	A vector with the estimated mutual information I(X;Y) for each couple response-predictor.
norm.mi	A vector with the normalized mutual information I(X;Y)* for each couple response-predictor.

lambda	A vector with the values of Goodman-Kruscal $\lambda(Y|X)$ for each couple response-predictor.
tau	A vector with the values of Goodman-Kruscal $\tau(Y|X)$ for each couple response-predictor.
U	A vector with the values of Theil's uncertainty coefficient U(Y|X) for each couple response-predictor.
AIC	A vector with the values of AIC(Y|X) for each couple response-predictor.
BIC	A vector with the values of BIC(Y|X) for each couple response-predictor.
npar	A vector with the number of parameters (conditional probabilities) estimated to calculate AIC and BIC for each couple response-predictor.

When out.df=TRUE the output will be a data.frame with a column for each measure.

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

References

Agresti A (2002) *Categorical Data Analysis. Second Edition*. Wiley, new York.

Bergsma W (2013) A bias-correction for Cramer's V and Tschuprow's T. *Journal of the Korean Statistical Society*, 42, 323–328.

The Institute of Statistical Mathematics (2018). catdap: Categorical Data Analysis Program Package. R package version 1.3.4. https://CRAN.R-project.org/package=catdap

Sakamoto Y and Akaike, H (1977) Analysis of Cross-Classified Data by AIC. *Ann. Inst. Statist. Math.*, 30, 185-197.

Examples

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

# how Lrn is response variable
pw.assoc(Lrn~Age+Sex+Eth, data=quine)

# usage of units' weights
quine$ww <- runif(nrow(quine), 1,4) #random gen  1<=weights<=4
pw.assoc(Lrn~Age+Sex+Eth, data=quine, weights="ww")
```

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 at random.

Usage

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

Arguments

data.rec   A numeric matrix or data frame that plays the role of *recipient*. This data frame
              must contain the variables (columns), specified via match.vars and don.class,
              that should be used in the matching.

              Missing values (NA) are allowed.

data.don   A matrix or data frame that plays the role of *donor*. This data frame must contain
              the variables (columns), specified via match.vars and don.class, that should
              be used in the matching.

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 between records (rows)
              in data.rec and those in data.don. When no matching variables are con-
              sidered (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 com-
              putation 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.

              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 fol-
              lowing distances can be used: "Manhattan" (aka "City block"; default), "Eu-
              clidean", "Mahalanobis","exact" or "exact matching", "Gower", "minimax",
              "difference", or one of the distance functions available in the package **proxy**.
              Note that the distances are computed using the function [dist](#) of the package

proxy with the exception of the "Gower" (see function `gower.dist` for details), "Mahalanobis" (function `mahalanobis.dist`), "minimax" (see `maximum.dist`) "difference" case. Note that `dist.fun="difference"` computes just the difference between the values of the unique numeric matching variable considered; in practice, it should be used when the subset of the donation classes should be formed by comparing the values of the unique matching variable (for further details see the argument `cut.don`.

By setting `dist.fun="ANN"` or `dist.fun="RANN"` it is possible to search for the k nearest neighbours for each recipient record by using the the Approximate Nearest Neighbor (ANN) search as implemented in the function `nn2` provided by the package **RANN**.

When `dist.fun="Manhattan"`, `"Euclidean"`, `"Mahalanobis"` or `"minimax"` all the variables in `data.rec` and `data.don` must be numeric. On the contrary, when `dist.fun="exact"` or `dist.fun="exact matching"`, 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.

cut.don   A character string that, jointly with the argument k, identifies the rule to be used to form the subset of the closest donor records.

- `cut.don="rot"`: (default) then the number of the closest donors to retain is given by $\left[\sqrt{n_D}\right] + 1$; being n_D the total number of available donors. In this case k must not to be specified.
- `cut.don="span"`: the number of closest donors is determined as the proportion k of all the available donors, i.e. $[n_D \times k]$. Note that, in this case, $0 < k \le 1$.
- `cut.don="exact"`: the kth closest donors out of the n_D are retained. In this case, $0 < k \le n_D$.
- `cut.don="min"`: the donors at the minimum distance from the recipient are retained.
- `cut.don="k.dist"`: only the donors whose distance from the recipient is less or equal to the value specified with the argument k. Note that in this case it is not possible to use `dist.fun="ANN"`.
- `cut.don="lt"` or `cut.don="<"`: only the donors whose value of the matching variable is smaller than the value of the recipient are retained. Note that in this case it is has to be set `dist.fun="difference"`.
- `cut.don="le"` or `cut.don="<="`: only the donors whose value of the matching variable is smaller or equal to the value of the recipient are retained. Note that in this case it is has to be set `dist.fun="difference"`.
- `cut.don="ge"` or `cut.don=">="`: only the donors whose value of the matching variable is greater or equal to the value of the recipient are retained. Note that in this case it is has to be set `dist.fun="difference"`.
- `cut.don="gt"` or `cut.don=">"`: only the donors whose value of the matching variable is greater than the value of the recipient are retained. Note that in this case it is has to be set `dist.fun="difference"`.

k   Depends on the `cut.don` argument.

weight.don   A character string providing the name of the variable with the weights associ-
                  ated to the donor units in `data.don`. 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 `weight.don=NULL` (default) all the units in the
                  subset of the closest donors will have the same probability of being selected.

keep.t   Logical, when donation classes are used by setting `keep.t=TRUE` prints infor-
                  mation on the donation classes being processed (by default `keep.t=FALSE`).

...   Additional arguments that may be required by `gower.dist`, by
                  `maximum.dist`, by `dist` or by `nn2`.

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 `RANDwNND.hotdeck`, 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 is known as *weighted random hot deck* (cf. Andridge and Little, 2010).

The search for the subset of the closest donors can be speed up by using the Approximate Nearest
Neighbor search as implemented in the function `nn2` provided by the package **RANN**. Note that this
search can be used in all the cases with the exception of `cut.don="k.dist"`.

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:

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.

sum.dist   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.

noad   For each recipient unit, reports the number of donor records in the subset of
                  closest donors.

call   How the function has been called.

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

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

```
data(samp.A, samp.B, package="StatMatch") #loads data sets
?samp.A
?samp.B


# samp.A plays the role of recipient
# samp.B plays the role of donor
# find a donor in the in the same region ("area5") and with the same
# gender ("sex"), then only the closest k=20 donors in terms of
# "age" are cnsidered and one of them is picked up at random

out.RND.1 <- RANDwNND.hotdeck(data.rec=samp.A, data.don=samp.B,
                              don.class=c("area5", "sex"), dist.fun="ANN",
                              match.vars="age", cut.don="exact", k=20)

# create the synthetic (or fused) data.frame:
# fill in "labour5" in A
fused.1 <- create.fused(data.rec=samp.A, data.don=samp.B,
                        mtc.ids=out.RND.1$mtc.ids, z.vars="labour5")
head(fused.1)

# weights ("ww") are used in selecting the donor in the final step

out.RND.2 <- RANDwNND.hotdeck(data.rec=samp.A, data.don=samp.B,
                              don.class=c("area5", "sex"), dist.fun="ANN",
                              match.vars="age", cut.don="exact",
                              k=20, weight.don="ww")
fused.2 <- create.fused(data.rec=samp.A, data.don=samp.B,
                        mtc.ids=out.RND.2$mtc.ids, z.vars="labour5")
```

```
head(fused.2)

# find a donor in the in the same region ("area5") and with the same
# gender ("sex"), then only the donors with "age" <= to the age of the
# recipient are considered,
# then one of them is picked up at random

out.RND.3 <- RANDwNND.hotdeck(data.rec=samp.A, data.don=samp.B,
                              don.class=c("area5", "sex"), dist.fun="diff",
                              match.vars="age", cut.don="<=")

# create the synthetic (or fused) data.frame:
# fill in "labour5" in A
fused.3 <- create.fused(data.rec=samp.A, data.don=samp.B,
                        mtc.ids=out.RND.3$mtc.ids, z.vars="labour5")
head(fused.3)

# 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.RND <- 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.RND$mtc.ids, z.vars="Sepal.Length")

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

---

rankNND.hotdeck   *Rank distance hot deck method.*

Description

This function implements rank hot deck distance method. For each recipient record the closest donors is chosen by considering the distance between 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,
                constrained=FALSE, constr.alg="Hungarian",
                keep.t=FALSE)
```

Arguments

data.rec   A numeric matrix or data frame that plays the role of *recipient*. This data
                 frame must contain the variable var.rec to be used in computing the percent-
                 age points of the empirical cumulative distribution function and eventually the
                 variables that identify the donation classes (see argument don.class) and the
                 case weights (see argument weight.rec).

                 Missing values (NA) are not allowed.

data.don   A matrix or data frame that plays the role of *donor*. This data frame must contain
                 the variable var.don to be used in computing percentage points of the the em-
                 pirical cumulative distribution function and eventually the variables that identify
                 the donation classes (see argument don.class) and the case weights (see argu-
                 ment weight.don).

var.rec   A character vector with the name of the variable in data.rec that should be
                 ranked.

var.don   A character vector with the name of the variable data.don that should be ranked.
                 If not specified, by default var.don=var.rec.

don.class   A character vector with the names of the variables (columns in both the data
                 frames) that identify donation classes. In each donation class the computation
                 of percentage points is carried out independently. Then only distances between
                 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 that
                 the variables used to form donation classes are defined as factor.

                 When not specified (default), no donation classes are used.

weight.rec   Eventual name of the variable in data.rec that provides the weights that should
                 be used in computing the the empirical cumulative distribution function for
                 var.rec (see Details).

weight.don   Eventual name of the variable in data.don that provides the weights that should
                 be used in computing the the empirical cumulative distribution function for
                 var.don (see Details).

constrained   Logical. When constrained=FALSE (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 "Hungarian". 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="Hungarian"` (default) the transportation problem is solved using the Hungarian method, implemented in function [`solve_LSAP`](solve_LSAP) available in the package **clue**. Note that `constr.alg="Hungarian"` is faster and more efficient.

keep.t   Logical, when donation classes are used by setting `keep.t=TRUE` prints information on the donation classes being processed (by default `keep.t=FALSE`).

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:

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

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

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

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

In the unconstrained case, when there are more donors at the same distance, one of them is chosen at random.

When the donation class are introduced, then the 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. To avoid it set `constrained=TRUE`. In such a case a donor can be chosen just once and the selection of the donors is carried out by solving a transportation problem with the objective of minimizing the overall matching distance (sum of the distances recipient-donor).

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 between each recipient unit and the corresponding donor.

noad   The number of available donors at the minimum distance for each recipient unit (only in unconstrained case)

call   How the function has been called.

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

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

```
data(samp.A, samp.B, package="StatMatch") #loads data sets

# samp.A plays the role of recipient
?samp.A

# samp.B plays the role of donor
?samp.B


# rankNND.hotdeck()
# donation classes formed using "area5"
# ecdf conputed on "age"
# UNCONSTRAINED case
out.1 <- rankNND.hotdeck(data.rec=samp.A, data.don=samp.B, var.rec="age",
                         don.class="area5")
fused.1 <- create.fused(data.rec=samp.A, data.don=samp.B,
                        mtc.ids=out.1$mtc.ids, z.vars="labour5")
head(fused.1)

#  as before but ecdf estimated  using weights
# UNCONSTRAINED case
out.2 <- rankNND.hotdeck(data.rec=samp.A, data.don=samp.B, var.rec="age",
                         don.class="area5",
                         weight.rec="ww", weight.don="ww")
fused.2 <- create.fused(data.rec=samp.A, data.don=samp.B,
                        mtc.ids=out.2$mtc.ids, z.vars="labour5")
head(fused.2)
```

samp.A	*Artificial data set resembling EU–SILC survey*

Description

This data set provides a limited number of variables observed at persons levels among those usually collected in the European Union Statistics on Income and Living Conditions Survey (EU–SILC). The data are artificially generated, just to show the application of the statistical matching techniques implemented in **StatMatch**.

Usage

```
data(samp.A)
```

Format

A data frame with 3009 observations and the following variables:

HH.P.id unique unit identifier of the type aa.bb where aa identifies the Household while bb identifies the household member

area5 large geographic area, factor with 5 levels: 'NE'=North–East, 'NO'=North–West, 'C'=center, 'S'=South, 'I'=islands

urb Degree of urbanization, factor with 3 levels: '1'=densely populated area, '2'=intermediate area, '3'=thinly populated area

hsize integer, size of the household in which the person lives

hsize5 factor with 5 levels derived from hsize, where the 5th level '>=5' denotes 5 and more people in the household

age integer, the person's age

c.age factor, age categorized in 5 classes

sex factor, the person's gender: '1'=male, '2'=female

marital factor, the person's marital status: '1'=never married, '2'=married, '3'=other (separated, widowed, divorced)

edu7 factor, the person's highest education level attained, follows the ISCED-97 categories: '0'=pre–primary education, '1'=primary education, '2'=lower secondary education, '3'= (upper) secondary education, '4'= post–secondary non tertiary education, '5'=first stage of tertiary education (not leading directly to an advanced research qualification), '6'=second stage of tertiary education (leading to an advanced research qualification)

n.income numeric, the person's net income in Euros

c.neti factor, the person's net income categorized in 7 classes of thousand of Euros

ww numeric, the unit's weight

Details

Please note that this data set is just for illustrative purposes. The unit's weight do not reflect the Italian population size. The variables included are derived starting from the those usually observed in the EU–SILC survey.

Source

This data set is artificially created starting from the EU–SILC survey structure.

References

<https://ec.europa.eu/eurostat/web/income-and-living-conditions/overview>

---

samp.B   *Artificial data set resembling EU–SILC survey*

Description

This data set provides a limited number of variables observed at persons levels among those usually collected in the European Union Statistics on Income and Living Conditions Survey (EU–SILC). The data are artificially generated, just to show the application of the statistical matching techniques implemented in **StatMatch**.

Usage

```
data(samp.B)
```

Format

A data frame with 6686 observations and the following variables:

HH.P.id unique unit identifier of the type aa.bb where aa identifies the Household while bb identifies the household member

area5 large geographic area, factor with 5 levels: 'NE'=North–East, 'NO'=North–West, 'C'=center, 'S'=South, 'I'=islands

urb Degree of urbanization, factor with 3 levels: '1'=densely populated area, '2'=intermediate area, '3'=thinly populated area

hsize integer, size of the household in which the person lives

hsize5 factor with 5 levels derived from hsize, where the 5th level '>=5' denotes 5 and more people in the household

age integer, the person's age

c.age factor, age categorized in 5 classes

sex factor, the person's gender: '1'=male, '2'=female

marital factor, the person's marital status: '1'=never married, '2'=married, '3'=other (separated, widowed, divorced)

edu7 factor, the person's highest education level attained, follows the ISCED-97 categories: '0'=pre–
primary education, '1'=primary education, '2'=lower secondary education, '3'= (upper) sec-
ondary education, '4'= post–secondary non tertiary education, '5'=first stage of tertiary edu-
cation (not leading directly to an advanced research qualification), '6'=second stage of tertiary
education (leading to an advanced research qualification)

labour5 the person's self–defined economic status, factor with 5 levels: '1'=employee working
full–time or part–time, '2'=self–employed working full–time or part–time, '3'=unemployed,
'4'=In retirement or in early retirement or has given up business, '5'=other status (student,
permanent disabled, in compulsory military service, fulfilling domestic tasks, etc.)

ww numeric, the unit's weight

Details

Please note that this data set is just for illustrative purposes. The unit's weight do not reflect the
Italian population size. The variables included are derived starting from the those usually observed
in the EU–SILC survey.

Source

This data set is artificially created starting from the EU–SILC survey structure.

References

https://ec.europa.eu/eurostat/web/income-and-living-conditions/overview

---

samp.C   *Artificial data set resembling EU–SILC survey*

Description

This data set provides a limited number of variables observed at persons levels among those usually
collected in the European Union Statistics on Income and Living Conditions Survey (EU–SILC).
The data are artificially generated, just to show the application of the statistical matching techniques
implemented in **StatMatch**.

Usage

```
data(samp.C)
```

Format

A data frame with 980 observations and the following variables:

HH.P.id unique unit identifier of the type aa.bb where aa identifies the Household while bb iden-
tifies the household member

area5 large geographic area, factor with 5 levels: 'NE'=North–East, 'NO'=North–West, 'C'=center,
'S'=South, 'I'=islands

urb Degree of urbanization, factor with 3 levels: '1'=densely populated area, '2'=intermediate area, '3'=thinly populated area

hsize integer, size of the household in which the person lives

hsize5 factor with 5 levels derived from hsize, where the 5th level '>=5' denotes 5 and more people in the household

age integer, the person's age

c.age factor, age categorized in 5 classes

sex factor, the person's gender: '1'=male, '2'=female

marital factor, the person's marital status: '1'=never married, '2'=married, '3'=other (separated, widowed, divorced)

edu7 factor, the person's highest education level attained, follows the ISCED-97 categories: '0'=pre–primary education, '1'=primary education, '2'=lower secondary education, '3'= (upper) secondary education, '4'= post–secondary non tertiary education, '5'=first stage of tertiary education (not leading directly to an advanced research qualification), '6'=second stage of tertiary education (leading to an advanced research qualification)

labour5 the person's self–defined economic status, factor with 5 levels: '1'=employee working full–time or part–time, '2'=self–employed working full–time or part–time, '3'=unemployed, '4'=In retirement or in early retirement or has given up business, '5'=other status (student, permanent disabled, in compulsory military service, fulfilling domestic tasks, etc.)

n.income numeric, the person's net income in Euros

c.neti factor, the person's net income categorized in 7 classes of thousand of Euros

ww numeric, the unit's weight

Details

Please note that this data set is just for illustrative purposes. The unit's weight do not reflect the Italian population size. The variables included are derived starting from the those usually observed in the EU–SILC survey.

Source

This data set is artificially created starting from the EU–SILC survey structure.

References

https://ec.europa.eu/eurostat/web/income-and-living-conditions/overview

---

selMtc.by.unc   *Identifies the best combination if matching variables in reducing un-*
                       certainty in estimation the contingency table Y vs. Z.

Description

This function identifies the "best" subset of matching variables in terms of reduction of uncertainty
when estimating relative frequencies in the contingency table Y vs. Z. The sequential procedure
presented in D'Orazio *et al.* (2017 and 2019) is implemented. This procedure avoids exploring all
the possible combinations of the available X variables as in [Fbwidths.by.x](#).

Usage

```
selMtc.by.unc(tab.x, tab.xy, tab.xz, corr.d=2,
                 nA=NULL, nB=NULL, align.margins=FALSE)
```

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). **A minimum number of 3 vari-
               ables is needed**.

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. At least **three** categorical variables
               should be considered as **X** variables (common variables). The same **X** vari-
               ables in tab.x must be available in tab.xy. Usually, it is assumed that the joint
               distribution of the **X** variables computed from tab.xy is equal to tab.x (a warn-
               ing appears if any absolute difference is greater than tol). Note that when the
               marginal distribution of **X** in tab.xy is not equal to that of tab.x it is possible
               to align them before computations (see argument align.margins).

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. At least **three** categorical variables
               should be considered as **X** variables (common variables). The same **X** vari-
               ables in tab.x must be available in tab.xz. Usually, it is assumed that the joint
               distribution of the **X** variables computed from tab.xz is equal to tab.x (a warn-
               ing appears if any absolute difference is greater than tol). Note that when the
               marginal distribution of **X** in tab.xz is not equal to that of tab.x it is possible
               to align them before computations (see argument align.margins).

corr.d   Integer, indicates the penalty that should be introduced in estimating the uncer-
               tainty by means of the average width of cell bounds. When corr.d=1 the penalty
               being considered is the one introduced in D'Orazio *et al.* (2017) (i.e. penalty1 in
               [Fbwidths.by.x](#)). When corr.d=2 (default) it is considered a penalty suggested

in D'Orazio *et al.* (2019) (indicated as "penalty2" in `Fbwidths.by.x`). Finally, no penalties are considered when `corr.d=0`.

nA   Integer, sample size of file A used to estimate `tab.xy`. If NULL is obtained as sum of frequencies in `tab.xy`.

nB   Integer, sample size of file B used to estimate `tab.xz`. If NULL is obtained as sum of frequencies in `tab.xz`.

align.margins   Logical (default FALSE). When when TRUE the distribution of **X** variables in `tab.xy` is aligned with the distribution resulting from `tab.x`, without affecting the marginal distribution of Y. Similarly the distribution of **X** variables in `tab.xz` is aligned with the distribution resulting from `tab.x`, without affecting the marginal distribution of Z. The alignment is performed by running IPF algorithm as implemented in the function `Estimate` in the package **mipfp**. To avoid lack of convergence due to combinations of Xs encountered in one table but not in the other (statistical 0s), before running IPF a small constant (1e-06) is added to empty cells in `tab.xy` and `tab.xz`.

Details

This function follows the sequential procedure described in D'Orazio *et al.* (2017, 2019) to identify the combination of common variables most effective in reducing uncertainty when estimating the contingency table Y vs. Z. Initially, the available Xs are ordered according to the reduction of average width of uncertainty bounds when conditioning on each of them. Then in each step one the remaining X variables is added until the table became too sparse; in practice the procedure stops when:

$$min\left[\frac{n_A}{H_{D_m} \times J}, \frac{n_B}{H_{D_m} \times K}\right] \le 1$$

For major details see also `Fbwidths.by.x`.

Value

A list with the main outcomes of the procedure.

ini.ord   Average width of uncertainty bounds when conditioning on each of the available X variables. Variable most effective in reducing uncertainty comes first. The ordering determines the order in which they are entered in the sequential procedure.

list.xs   List with the various combinations of the matching variables being considered in each step.

av.df   Data.frame with all the relevant information for each of combination of X variables. The last row corresponds to the combination of the X variables identified as the best in reducing average width of uncertainty bounds (penalized or not depending on the input argument `corr.d`). For each combination of X variables the following additional information are reported: the number of cells (name starts with "nc"); the number of empty cells (name starts with "nc0"; the average relative frequency (name starts with "av.crf"); sparseness measured as Cohen's

effect size with respect to equiprobability (uniform distribution across cells). Finally there are the value of the stopping criterion ("`min.av`"), the unconditioned average width of uncertainty bounds ("`avw`"), the penalty term ("`penalty`") and the penalized width ("`avw.pen`"; avw.pen=avw+penalty).

Author(s)

Marcello D'Orazio <mdo.statmatch@gmail.com>

References

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

D'Orazio, M., Di Zio, M. and Scanu, M. (2017). "The use of uncertainty to choose matching variables in statistical matching". *International Journal of Approximate Reasoning*, 90, pp. 433-440.

D'Orazio, M., Di Zio, M. and Scanu, M. (2019). "Auxiliary variable selection in a a statistical matching problem". In Zhang, L.-C. and Chambers, R. L. (eds.) *Analysis of Integrated Data*, Chapman & Hall/CRC (forthcoming).

See Also

Fbwidths.by.x, Frechet.bounds.cat

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
suppressWarnings(RNGversion("3.5.0"))
set.seed(1111)
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 Fbwidths.by.x()
freq.xA <- xtabs(~Eth+Sex+Age, data=quine.A)
freq.xB <- xtabs(~Eth+Sex+Age, data=quine.B)

freq.xy <- xtabs(~Eth+Sex+Age+Lrn, data=quine.A)
freq.xz <- xtabs(~Eth+Sex+Age+c.Days, data=quine.B)

# apply Fbwidths.by.x()
bb <- Fbwidths.by.x(tab.x=freq.xA+freq.xB,
                             tab.xy=freq.xy,  tab.xz=freq.xz,
                             warn=FALSE)
bb$sum.unc
```

```
cc <- selMtc.by.unc(tab.x=freq.xA+freq.xB,
                       tab.xy=freq.xy,  tab.xz=freq.xz, corr.d=0)
cc$ini.ord
cc$av.df
```

Index