Statistical Matching and Imputation of Survey Data with StatMatch*

Marcello D'Orazio
Italian National Institute of Statistics (Istat), Rome, Italy
E-mail: madorazi@istat.it

December 24, 2013

1 Introduction

Statistical matching techniques aim at integrating two or more data sources (usually data from sample surveys) referred to the same target population. In the basic statistical matching framework, there are two data sources A and B sharing a set of variables X while the variable Y is available only in A and the variable Z is observed just in B. The X variables are common to both the data sources, while the variables Y and Z are not jointly observed. The objective of statistical matching (hereafter denoted as SM) consists in investigating the relationship between Y and Z at "micro" or "macro" level (D'Orazio *et al.*, 2006b). In the micro case the SM aims at creating a "synthetic" data source in which all the variables, X, Y and Z, are available (usually $A \cup B$ with all the missing values filled in or simply A filled in with the values of Z). When the objective is macro, the data sources are integrated to derive an estimate of the parameter of interest, e.g. the correlation coefficient between Y and Z or the contingency table $Y \times Z$.

A parametric approach to SM requires the explicit adoption of a model for (X, Y, Z); obviously, if the model is misspecified the results will not be reliable. The nonparametric approach is more flexible in handling complex situations (mixed type variables). The two approaches can be mixed: first a parametric model is assumed and its parameters are estimated then a completed synthetic data set is derived through a nonparametric micro approach. In this manner the advantages of both parametric and nonparametric approaches are maintained: the model is parsimonious while nonparametric techniques offer protection against model misspecification. Table 1 provides a summary of the objectives and approaches to SM (D'Orazio *et al.*, 2008).

*This document is partly based on the work carried out in the framework of the ESSnet project on Data Integration, partly funded by Eurostat (December 2009–December 2011). For more information on the project visit http://www.essnet-portal.eu/di/data-integration

Table 1: Objectives and approaches to Statistical matching.

Objectives of Statistical matching	Approaches to statistical Matching		
	Parametric	Nonparametric	Mixed
MAcro	yes	yes	no
MIcro	yes	yes	yes

It is worth noting that in the traditional SM framework when only A and B are available, all the SM methods (parametric, nonparametric and mixed) that use the set of common variables X to match A and B, implicitly assume the *conditional independence* (CI) of Y and Z given X:

$$f(x,y,z) = f(y|x) \times f(z|x) \times f(x)$$

This assumption is particularly strong and seldom holds in practice. In order to avoid the CI assumption the SM should incorporate some auxiliary information concerning the relationship between Y and Z (see Chap. 3 in D'Orazio *et al.* 2006b). The auxiliary information can be at micro level (a new data source in which Y and Z or X, Y and Z are jointly observed) or at macro level (e.g. an estimate of the correlation coefficient ρ_{XY} or an estimate of the contingency table $Y \times Z$, etc.) or simply consist of some logic constraints about the relationship between Y and Z (structural zeros, etc.; for further details see D'Orazio *et al.*, 2006a).

An alternative approach to SM consists in evaluating the *uncertainty* concerning an estimate of the parameter of interest. This uncertainty is due to the lack of joint information concerning Y and Z. For instance, let us consider a SM application whose target consists in estimating the correlation matrix of the trivariate normal distribution holding for (X,Y,Z); in the basic SM framework the available data allow to estimate all the components of the correlation matrix with the exception of ρ_{YZ}; in this case, due to the properties of the correlation matrix (has to be semidefinite positive), it is possible to conclude that:

$$\rho_{XY}\rho_{XZ} - \sqrt{\left(1-\rho_{YX}^2\right)\left(1-\rho_{XZ}^2\right)} \leq \rho_{YZ} \leq \rho_{XY}\rho_{XZ} + \sqrt{\left(1-\rho_{YX}^2\right)\left(1-\rho_{XZ}^2\right)}$$

The higher is the correlation between X and Y and between X and Z, the shorter will be the interval and consequently the lower will be the uncertainty. In practical applications, by substituting the unknown correlation coefficient with the corresponding estimates it is possible to derive a "range" of admissible values of the unknown ρ_{YZ}. The topic of the uncertainty will be discussed in the Section 6.

Section 2 will be discuss some practical aspects concerning the preliminary steps, with emphasis on the choice of the marching variables; moreover some example data will be introduced. In Section 3 some nonparametric approaches to SM at micro will be shown. Section 4 is devoted to the mixed approaches to SM. Section 5 will discuss SM approaches to deal with data arising from complex sample surveys from finite populations.

2 Practical steps in an application of statistical matching

Before applying SM methods in order to integrate two or more data sources some decisions and preprocessing steps are required (Scanu, 2008). In practice, given two data sources A and B related to the same target population, the following steps are necessary:

1. Choice of the target variables Y and Z, i.e. of the variables observed distinctly in two sample surveys.
2. Identification of all the common variables X shared by A and B. In this step some harmonization procedures may be required because of different definitions and/or classifications. Obviously, if two similar variables can not be harmonized they have to be discarded. The common variables should not present missing values and the observed values should be accurate (low or absent measurement error). Note that, the common variable in the two data sources are expected to share the same marginal/joint distribution, if A and B are representative samples of the same population.
3. Potentially all the X variables can be used as matching variables but actually, not all them are used in the SM. Section 2.2 will provide more details concerning this topic.
4. The choice of the matching variables is strictly related to the matching framework (see Table 1).
5. Once decided the framework, a SM technique is used to match the samples.
6. Finally the results of the matching, whereas possible, should be evaluated.

2.1 Example data

The next Sections will provide simple examples of application of some SM techniques in the R environment (R Development Core Team, 2012) by using the functions in **StatMatch** (D'Orazio, 2012). These examples will refer to artificial data derived from the data set `eusilcS` contained in the package **simPopulation** (Alfons and Kraft, 2012). This is an artificial data set generated from real Austrian EU-SILC (European Union Statistics on Income and Living Conditions) data containing 11 725 observations on 18 variables (see `eusilcS` help pages for details):

```
> library(simPopulation, warn.conflicts=FALSE) #loads pkg simPopulation
> data(eusilcS)
> str(eusilcS)
```

```
'data.frame':        11725 obs. of  18 variables:
 $ db030    : int  1 1 2 3 4 4 4 5 5 5 ...
 $ hsize    : int  2 2 1 1 3 3 3 5 5 5 ...
 $ db040    : Factor w/ 9 levels "Burgenland","Carinthia",..: 4 4 7 5 7 7 7 4 4 4 ...
```

```
 $ age      : int  72 66 56 67 70 46 37 41 35 9 ...
 $ rb090    : Factor w/ 2 levels "male","female": 1 2 2 2 2 1 1 1 2 2 ...
 $ pl030    : Factor w/ 7 levels "1","2","3","4",..: 5 5 2 5 5 3 1 1 3 NA ...
 $ pb220a   : Factor w/ 3 levels "AT","EU","Other": 1 1 1 1 1 1 3 1 1 NA ...
 $ netIncome: num  22675 16999 19274 13319 14366 ...
 $ py010n   : num  0 0 19274 0 0 ...
 $ py050n   : num  0 0 0 0 0 ...
 $ py090n   : num  0 0 0 0 0 ...
 $ py100n   : num  22675 0 0 13319 14366 ...
 $ py110n   : num  0 0 0 0 0 0 0 0 0 NA ...
 $ py120n   : num  0 0 0 0 0 0 0 0 0 NA ...
 $ py130n   : num  0 16999 0 0 0 ...
 $ py140n   : num  0 0 0 0 0 0 0 0 0 NA ...
 $ db090    : num  7.82 7.82 8.79 8.11 7.51 ...
 $ rb050    : num  7.82 7.82 8.79 8.11 7.51 ...
```

In order to use these data for our purposes, some manipulations are needed to discard units not relevant (obs. with `age<16`, whose income and personal economic status are missing), to categorize some variables, etc.

```
> # discard units with age<16
> silc.16 <- subset(eusilcS, age>15) # units
> nrow(silc.16)

[1] 9522

> # categorize age
> silc.16$c.age <- cut(silc.16$age, c(16,24,49,64,100), include.lowest=T)
> #
> # truncate hsize
> aa <- as.numeric(silc.16$hsize)
> aa[aa>6] <- 6
> silc.16$hsize6 <- factor(aa, ordered=T)
> #
> # recode personal economic status
> aa <- as.numeric(silc.16$pl030)
> aa[aa<3] <- 1
> aa[aa>1] <- 2
> silc.16$work <- factor(aa, levels=1:2, labels=c("working","not working"))
> #
> # categorize personal net income
> silc.16$c.netI <- cut(silc.16$net/1000,
+                       breaks=c(-6,0,5,10,15,20,25,30,40,50,200))
```

In order to reproduce the basic SM framework, the data frame `silc.16` is split randomly in two data sets: `rec.A` consisting of 4 000 observations and `don.B` with the remaining 5 522 units. The two data frames `rec.A` and `don.B` share the variables `X.vars`; the person's economic status (`y.var`) is available only in `rec.A` while the net income (`z.var`) is available in `don.B`.

```
> # simulate samples
> set.seed(123456)
> obs.A <- sample(nrow(silc.16), 4000, replace=F)
> X.vars <- c("hsize","hsize6","db040","age","c.age",
+             "rb090","pb220a","rb050")
> y.var <- c("pl030","work")
> z.var <- c("netIncome", "c.netI")
> rec.A <- silc.16[obs.A, c(X.vars, y.var)]
> don.B <- silc.16[-obs.A, c(X.vars, z.var)]
> #
> # determine a rough weighting
> # compute N, the est. size of pop(age>16)
> N <- round(sum(silc.16$rb050))
> N
```

```
[1] 67803
```

```
> #rescale origin weights
> rec.A$wwA <- rec.A$rb050/sum(rec.A$rb050)*N
> don.B$wwB <- don.B$rb050/sum(don.B$rb050)*N
```

2.2 The choice of the matching variables

In SM A and B, may share many common variables. In practice, just the most relevant ones , usually called *matching variables*, are used in in the matching. The selection of these variables should be performed through opportune statistical methods (descriptive, inferential, etc.) and by consulting subject matter experts.

From a statistical point of view, the choice of the marching variables X_M ($X_M \subseteq X$) should be carried out in a "multivariate sense" in order to identify the subset of the X_M variables connected at the same time with Y and Z (Cohen, 1991); unfortunately this would require the availability of an auxiliary data source in which all the variables (X, Y, Z) are observed. In the basic SM framework the data in A permit to explore the relationship between Y and X, while the relationship between Z and X can be investigated in the file B. Then the results of the two separate analyses have to be combined in some manner; usually the subset of the matching variables is obtained as $X_M = X_Y \cup X_Z$, being X_Y ($X_Y \subseteq X$) the subset of the common variables that better explains Y, while X_Z is the subset of the common variables that better explain Z ($X_Z \subseteq X$). The risk in such a procedure is that of obtaining too many matching variables, and consequently increasing the complexity of the problem and potentially affect negatively the results of SM. In particular, in the micro approach this may introduce additional undesired variability and bias as far as the joint (marginal) distribution of X_M and Z is concerned. For this reason sometimes the set of the matching variables is obtained as a compromise among $X_Y \cap X_Z \subseteq X_M \subseteq X_Y \cup X_Z$.

The simplest procedure to identify X_Y consists in pairwise correlation/association measures among the Y and all the available predictors X. When response and predictors are all categorical, then Chi-square based association measures (Cramer's V) or *proportional reduction of the variance* measures can be considered. The function `pw.assoc` in **StatMatch** provides some of them.

```
> # analyses on A
> library(StatMatch) #loads StatMatch
> # response is pl030
> pw.assoc(pl030~db040+hsize6+c.age+rb090+pb220a, data=rec.A)

$V
 pl030.db040 pl030.hsize6  pl030.c.age  pl030.rb090 pl030.pb220a
  0.07369617   0.19172123   0.52701354   0.43451872   0.11761739

$lambda
 pl030.db040 pl030.hsize6  pl030.c.age  pl030.rb090 pl030.pb220a
  0.00000000   0.05476951   0.27339115   0.00000000   0.00000000

$tau
 pl030.db040 pl030.hsize6  pl030.c.age  pl030.rb090 pl030.pb220a
 0.005804228  0.053874437  0.245431041  0.054777396  0.004970513

$U
 pl030.db040 pl030.hsize6  pl030.c.age  pl030.rb090 pl030.pb220a
 0.010376238  0.065400904  0.272715553  0.073490286  0.009215848

> #response is work (aggregated pl030)
> pw.assoc(work~db040+hsize6+c.age+rb090+pb220a, data=rec.A)

$V
 work.db040 work.hsize6  work.c.age  work.rb090 work.pb220a
 0.06329734  0.20670621  0.55617833  0.20081742  0.02615234

$lambda
 work.db040 work.hsize6  work.c.age  work.rb090 work.pb220a
0.009325288 0.127811300 0.409215579 0.119583105 0.000000000

$tau
  work.db040   work.hsize6   work.c.age   work.rb090  work.pb220a
0.0040065534 0.0427274592 0.3093343374 0.0405296059 0.0006839447
```

```
$U
  work.db040  work.hsize6   work.c.age   work.rb090  work.pb220a
0.0029056489 0.0312998272 0.2689070320 0.0296360550 0.0004989826
```

In practice it comes out the best predictor of person's economic status (`pl030`) is the age conveniently categorized (`c.age`). If we consider as Y the aggregated person's economic status (variable `work`), then it can be observed that it is slightly associated also with gender (`rb090`) and household size (`hsize6`).

When the response variable is continuous one can look at correlation with the predictors. In order to identify eventual nonlinear relationship it may be convenient to consider the ranks (Spearman's rank correlation coefficient). An interesting suggestion from Harrell (2012) consists in looking at the adjusted R^2 related to the regression model rank(Y) vs. rank(X) (unadjusted R^2 corresponds to squared Spearman's rank correlation coefficient). When X is categorical nominal variable it is considered the adjusted R^2 of the regression model rank(Y) vs. dummies(X). The function `spearman2` in the package **Hmisc** (Harrell,2012) computes automatically the adjusted R^2 for each couple or response-predictor.

```
> # analyses on B
> require(Hmisc)
> spearman2(netIncome~db040+hsize+age+rb090+pb220a, data=don.B)

Spearman rho^2    Response variable:netIncome

        rho2      F df1  df2      P Adjusted rho2    n
db040  0.003   2.20   8 5513 0.0243         0.002 5522
hsize  0.030 170.97   1 5520 0.0000         0.030 5522
age    0.032 184.98   1 5520 0.0000         0.032 5522
rb090  0.147 952.42   1 5520 0.0000         0.147 5522
pb220a 0.018  50.98   2 5519 0.0000         0.018 5522
```

By looking at the adjusted R^2, it comes out that just the gender (`rb090`) has a certain predictive power on `netIncome`.

To summarize, in our case it come out that the set of the matching variables is composed by `age` and `rb090` ($X_M = X_Y \cup X_Z$).

When too many variables are available before computing pairwise association/correlation measures, it would be necessary to discard the redundant predictors (functions `redun` and `varclus` in **Hmisc** can be of help).

Sometimes the important predictors can be identified by fitting models and then running procedures for selecting the best predictor. The selection of the subset X_Y can also be demanded to nonparametric procedures such as *Classification And Regression Trees* (Breiman *et al.*, 1984). Instead of fitting a single tree, it would be better to fit a *random forest* (Breiman, 2001) by means of the function `randomForest` available in the package

randomForest (Liaw and Wiener, 2002) which provides a measure of importance for the predictors (to be used with caution).

The approach to SM based on the study of uncertainty offers the possibility of choosing the matching variable by selecting just those common variables with the highest contribution to the reduction of the uncertainty. The function `Fbwidths.by.x` in **StatMatch** permits to explore the reduction of uncertainty when all the variables (X, Y, Z) are categorical. In particular, assuming that X_D correspond to the complete crossing of the matching variables X_M, it is possible to show that in the basic SM framework

$$P_{j,k}^{(low)} \leq P_{Y=j,Z=k} \leq P_{j,k}^{(up)},$$

being

$$\begin{aligned} P_{j,k}^{(low)} &= \sum_i P_{X_D=i} \times \max\left\{0; P_{Y=j|X_D=i} + P_{Z=k|X_D=i} - 1\right\} \\ P_{j,k}^{(up)} &= \sum_i P_{X_D=i} \times \min\left\{P_{Y=j|X_D=i}; P_{Z=k|X_D=i}\right\} \end{aligned}$$

for $j = 1, \ldots, J$ and $k = 1, \ldots, K$, being J and K the categories of Y and Z respectively.

The function `Fbwidths.by.x` estimates $(P_{j,k}^{(low)}, P_{j,k}^{(up)})$ for each cell in the contingency table $Y \times Z$ in correspondence of all the possible combinations of the X variables; then the reduction of uncertainty is measured according to the proposal of Conti *et al.* (2012):

$$\hat{\Delta} = \sum_{i,j,k} \left(\hat{P}_{j,k}^{(up)} - \hat{P}_{j,k}^{(low)}\right) \times \hat{P}_{Y=j|X_D=i} \times \hat{P}_{Z=k|X_D=i} \times \hat{P}_{X_D=i}$$

An alternative naive measure refers to the average widths of the intervals:

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

```
> xx <- xtabs(~db040+hsize6+c.age+rb090+pb220a, data=rec.A)
> xy <- xtabs(~db040+hsize6+c.age+rb090+pb220a+work, data=rec.A)
> xz <- xtabs(~db040+hsize6+c.age+rb090+pb220a+c.netI, data=don.B)
> library(StatMatch) #loads StatMatch
> out.fbw <-  Fbwidths.by.x(tab.x=xx, tab.xy=xy, tab.xz=xz)
> # sort according to overall uncertainty
> sort.ov.unc <- out.fbw$sum.unc[order(out.fbw$sum.unc$ov.unc),]
> head(sort.ov.unc) # best 6 models

                                x.vars x.cells x.freq0
|c.age+rb090                         2       8       0
|c.age                               1       4       0
|hsize6+c.age                        2      24       0
|db040+hsize6+c.age+rb090+pb220a     5    1296     721
```

```
|c.age+rb090+pb220a                  3       24        1
|c.age+pb220a                        2       12        0
                                 av.width    ov.unc
|c.age+rb090                   0.07738444 0.1204430
|c.age                         0.08439346 0.1213526
|hsize6+c.age                  0.08282457 0.1236898
|db040+hsize6+c.age+rb090+pb220a 0.05775534 0.1238597
|c.age+rb090+pb220a            0.07642623 0.1246536
|c.age+pb220a                  0.08432518 0.1257270
```

The results in terms of overall uncertainty confirm the finding of the previous analysis: the highest reduction of the overall uncertainty is obtained by considering classes of age (`c.age`) and gender (`rb090`). It is worth noting that the age alone helps a lot in reducing the uncertainty in estimating the joint distribution of aggregated person's economic status (`work`) and classes of net income (`c.netI`).

3 Nonparametric micro techniques

Nonparametric approach is very popular in SM when the objective is the creation of a synthetic data set. Most of the nonparametric micro approaches consists in filling in the data set chosen as the *recipient* with the values of the variable which is available only in the other data set, the *donor* one. In this approach it is important to decide which data set plays the role of the recipient. Usually this is the data set to be used ad the basis for further statistical analysis, and a logic choice seems that of using the larger one because it would provide more accurate results. Unfortunately, such a way of working may provide inaccurate SM results, especially when the sizes of the two data sources are very different. The reason is quite simple, the larger is the recipient with respect to the donor, the more times a unit in the latter could be selected as a donor. In this manner, there is a high risk that the distribution of the imputed variable does not reflect the original one (estimated form the donor data set). In the following it will be assumed that A is the recipient while B is the donor, being $n_A \leq n_B$ (n_A and n_B are the sizes of A and B respectively). Hence the objective of SM will be that of filling in A with values of Y (variable available only in B).

In **StatMatch** the following nonparametric micro techniques are available: *random hot deck*, *nearest neighbor hot deck* and *rank hot deck* (see Section 2.4 in D'Orazio *et al.*, 2006b; Singh *et al.*, 1993).

3.1 Nearest neighbor distance hot deck

The nearest neighbor distance hot deck techniques are implemented in the function `NND.hotdeck`. This function searches in `data.don` the nearest neighbor of each unit in `data.rec` according to a distance computed on the matching variables X_M specified with the argument `match.vars`. By default the Manhattan (city block) distance is considered (`dist.fun="Manhattan"`). In order to reduce the effort to compute distances it is preferable to define some donation classes (argument `don.class`): for a record in given donation class it will be selected a donor in the same class (the distances are computed only between units belonging to the same class). Usually, the donation classes are defined according to one or more categorical common variables (geographic area, etc.). In the following, a simple example of usage of `NND.hotdeck` is reported; donation classes are formed using gender and region, while distances are computed on age

```
> group.v <- c("rb090","db040")
> X.mtc <- "age"
> out.nnd <- NND.hotdeck(data.rec=rec.A, data.don=don.B,
+                          match.vars=X.mtc, don.class=group.v)
```

```
Warning: The  Manhattan  distance is being used
All the categorical matching variables in rec and don
 data.frames, if present are recoded into dummies
```

The function `NND.hotdeck` does not create the synthetic data set; for each unit in A the corresponding closest donor in B is identified according to the imputation classes (when defined) and the chosen distance function; the recipient-donor units' identifiers are saved in the data.frame `mtc.ids` stored in the output list returned by `NND.hotdeck`. The output list provides also the distance between each couple recipient-donor (saved in the `dist.rd` component of the output list) and the number of available donors at the minimum distance for each recipient (component `noad`). Note that when there are more donors at the minimum distance, then one of them is picked up at random.

```
> summary(out.nnd$dist.rd) # summary distances rec-don
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.00    0.00    0.04    0.00    7.00
```

```
> summary(out.nnd$noad) # summary available donors at min. dist.
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    4.00    6.00    6.56    9.00   21.00
```

```
> table(out.nnd$noad)
```

```
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
234 375 389 426 361 370 378 375 220 180 175 198 117  72  90   4
 17  18  20  21
  5  12  10   9
```

In order to derive the synthetic data set it is necessary to run the function `create.fused`:

```
> head(out.nnd$mtc.ids)
```

```
     rec.id don.id
[1,] "401"  "376"
[2,] "71"   "118"
[3,] "92"   "106"
[4,] "225"  "350"
[5,] "364"  "408"
[6,] "370"  "350"
```

```
> fA.nnd <- create.fused(data.rec=rec.A, data.don=don.B,
+                        mtc.ids=out.nnd$mtc.ids,
+                        z.vars=c("netIncome","c.netI"))
> head(fA.nnd) #first 6 obs.
```

```
    hsize hsize6      db040 age    c.age rb090 pb220a    rb050
401     5      5 Burgenland  45  (24,49]  male     AT 4.545916
71      2      2 Burgenland  65 (64,100]  male     AT 6.151409
92      2      2 Burgenland  81 (64,100]  male     AT 6.151409
225     3      3 Burgenland  51  (49,64]  male     AT 5.860364
364     4      4 Burgenland  18  [16,24]  male     AT 6.316554
370     5      5 Burgenland  50  (49,64]  male     AT 4.545916
    pl030        work      wwA netIncome  c.netI
401     1     working 10.85782  47159.21 (40,50]
71      5 not working 14.69250  21316.32 (20,25]
92      5 not working 14.69250  21667.53 (20,25]
225     1     working 13.99734  20667.61 (20,25]
364     1     working 15.08694   9461.48  (5,10]
370     1     working 10.85782  20667.61 (20,25]
```

As far as distances are concerned (argument `dist.fun`), all the distance functions in the package **proxy** (Meyer and Butchta, 2012) are available. Anyway, for some particular distances it was decided to write specific R functions. In particular, when dealing with continuous matching variables it is possible to use the *maximum distance* (L^{∞} norm) implemented in `maximum.dist`; this function works on the true observed values (continuous variables) or on transformed ranked values (argument `rank=TRUE`) as suggested in Kovar *et al.* (1988); the transformation (ranks divided by the number of units) removes the effect of different scales and the new values are uniformly distributed in the interval $[0,1]$. The Mahalanobis distance can be computed by using `mahalanobis.dist` which allows an external estimate of the covariance matrix (argument `vc`). When dealing with mixed type matching variables, the *Gowers's dissimilarity* (Gower, 1981) can be computed (function `gower.dist`): it is an average of the distances computed on the single variables according to different rules, depending on the type of the variable. All the distances are scaled to range from 0 to 1, hence the overall distance cat take a value in $[0,1]$. When dealing with mixed types matching variables it is still possible to use the distance functions for continuous variables but `NND.hotdeck` transforms factors into dummies (by means of the function `fact2dummy`).

By default `NND.hotdeck` does not pose constraints on the "usage" of donors: a record in the donor data set can be selected many times as a donor. The multiple usage of a donor can be avoided by resorting to a *constrained hot deck* (argument `constrained=TRUE` in `NND.hotdeck`); in such a case, a donor can be used just once and all the donors are selected in order to minimize the overall matching distance. In practice, the donors are identified by solving a traveling salesperson problem; two alternatives are available: the Hungarian algorithm (argument `constr.alg="Hungarian"` implemented in the function `solve_LSAP` in the package **clue** (Hornik, 2012) and the algorithm provided by the package **lpSolve** (Berkelaar *et al.*, 2012) (argument `constr.alg="lPsolve"`). Setting `constr.alg="Hungarian"` (default) is more efficient and faster.

```
> group.v <- c("rb090","db040")
> X.mtc <- "age"
> out.nnd.c <- NND.hotdeck(data.rec=rec.A, data.don=don.B,
+                             match.vars=X.mtc, don.class=group.v,
+                             dist.fun="Manhattan", constrained=TRUE,
+                             constr.alg="Hungarian")

Warning: The  Manhattan  distance is being used
All the categorical matching variables in rec and don
 data.frames, if present are recoded into dummies

> fA.nnd.c <- create.fused(data.rec=rec.A, data.don=don.B,
+                       mtc.ids=out.nnd.c$mtc.ids,
+                       z.vars=c("netIncome","c.netI"))
```

The constrained matching returns an overall matching distance greater than the one in the unconstrained case, but it tends to better preserve the marginal distribution of the variable imputed in the synthetic data set.

```
> #comparing distances
> sum(out.nnd$dist.rd) # unconstrained

[1] 160

> sum(out.nnd.c$dist.rd) # constrained

[1] 1189
```

To compare the marginal joint distributions of a set of categorical variables it is possible to resort to the function `comp.prop` in **StatMatch** which provides some similarity measure among distributions of categorical variables and performs also the Chi-square test (for details see `comp.prop` the help pages).

```
> # estimating marginal distribution of C.netI
> tt0 <- xtabs(~c.netI, data=don.B) # reference distr.
> tt <- xtabs(~c.netI, data=fA.nnd)  # synt unconstr.
> ttc <- xtabs(~c.netI, data=fA.nnd.c) #synt. constr.
> #
> # comparing marginal distributions
> comp.prop(p1=tt, p2=tt0, n1=nrow(fA.nnd), n2=NULL, ref=TRUE)

$meas
       tvd    overlap      Bhatt       Hell
0.01993173 0.98006827 0.99971600 0.01685236

$chi.sq
   Pearson         df      q0.05   delta.h0
 9.3242717  9.0000000 16.9189776  0.5511132

$p.exp
c.netI
    (-6,0]      (0,5]     (5,10]    (10,15]    (15,20]    (20,25]
0.12893879 0.09253894 0.14034770 0.17348787 0.18598334 0.13274176
   (25,30]    (30,40]    (40,50]   (50,200]
0.06609924 0.05106845 0.01321985 0.01557407

> comp.prop(p1=ttc, p2=tt0, n1=nrow(fA.nnd), n2=NULL, ref=TRUE)

$meas
        tvd     overlap       Bhatt        Hell
0.006615628 0.993384372 0.999967141 0.005732269

$chi.sq
    Pearson          df       q0.05    delta.h0
 1.05225865  9.00000000 16.91897760  0.06219399

$p.exp
c.netI
    (-6,0]      (0,5]     (5,10]    (10,15]    (15,20]    (20,25]
0.12893879 0.09253894 0.14034770 0.17348787 0.18598334 0.13274176
   (25,30]    (30,40]    (40,50]   (50,200]
0.06609924 0.05106845 0.01321985 0.01557407
```

By looking at `comp.prop` output it comes out that, as expected, the marginal distribution of `c.netI` in the synthetic file obtained after constrained NND is closer to the reference distribution (estimated on the donor dataset) than the one estimated from the synthetic file after the unconstrained NND.

3.2 Random hot deck

The function RANDwNND.hotdeck carries out the random selection of each donor from a suitable subset of all the available donors. This subset can be formed in different ways, e.g. by considering all the donors sharing the same characteristics of the recipient (defined according to some X_M variables, such as geographic region, etc.). The traditional *random hot deck* (Singh *et al.*, 1993) within imputation classes is performed by simply specifying the donation classes via the argument don.class (the classes are formed by crossing the categories of the categorical variables being considered). For each record in the recipient data set in a given donation class, a donor is picked up completely at random within the same donation class.

```
> group.v <- c("db040","rb090")
> rnd.1 <- RANDwNND.hotdeck(data.rec=rec.A, data.don=don.B,
+                                match.vars=NULL, don.class=group.v)
> fA.rnd <- create.fused(data.rec=rec.A, data.don=don.B,
+                            mtc.ids=rnd.1$mtc.ids,
+                            z.vars=c("netIncome", "c.netI"))
```

As for NND.hotdeck, the function RANDwNND.hotdeck does not create the synthetic data set; the recipient-donor units' identifiers are saved in the component mtc.ids of the list returned in output. The number of donors available in each donation class are saved in the component noad.

RANDwNND.hotdeck implements various alternative methods to restrict the subset of the potential donors. These methods are based essentially on a distance measure computed on the matching variables provided via the argument match.vars. In practice, when cut.don="k.dist" only the donors whose distance from the recipient is less or equal to threshold k are considered (see Andridge and Little, 2010). By setting cut.don="exact" the k ($0 < k \leq n_D$) closest donors are retained (n_D is the number of available donors for a given recipient). With cut.don="span" a proportion k ($0 < k \leq 1$) of the closest available donors it is considered while; setting cut.don="rot" and k=NULL the subset reduces to the $\left[\sqrt{n_D}\right]$ closest donors; finally, when cut.don="min" only the donors at the minimum distance from the recipient are retained.

```
> # random choiches of a donor among the closest k=20 wrt age
> group.v <- c("db040","rb090")
> X.mtc <- "age"
> rnd.2 <- RANDwNND.hotdeck(data.rec=rec.A, data.don=don.B,
+                                match.vars=X.mtc, don.class=group.v,
+                                dist.fun="Manhattan",
+                                 cut.don="exact", k=20)

Warning: The  Manhattan  distance is being used
All the categorical matching variables in rec and don data.frames,
 if present, are recoded into dummies
```

```
> fA.knnd <- create.fused(data.rec=rec.A, data.don=don.B,
+                         mtc.ids=rnd.2$mtc.ids,
+                         z.vars=c("netIncome", "c.netI"))
```

When distances are computed on some matching variables, then the output of `RANDwNND.hotdeck` provides some information concerning the distances of the possible available donors for each recipient observation.

```
> head(rnd.2$sum.dist)
```

```
     min max       sd cut dist.rd
[1,]   0  47 11.02087   5       2
[2,]   0  49 14.54555   4       1
[3,]   0  65 19.01027   9       4
[4,]   1  41 10.09283   6       3
[5,]   1  74 19.53088  11       7
[6,]   0  42 10.16749   5       2
```

In particular, `"min"`, `"max"` and `"sd"` columns report respectively the minimum, the maximum and the standard deviation of the distances (all the available donors are considered), while `"cut"` refers to the distance of the `k`th closest donor; `"dist.rd"` is distance existing among the recipient and the randomly chosen donor.

When selecting a donor among those available in the subset identified by the arguments `cut.don` and `k`, it is possible to use a weighted selection by specifying a weighting variable via `weight.don` argument. This issue will be tackled in Section 5.

3.3 Rank hot deck

The *rank hot deck distance* method has been introduced by Singh *et al.* (1993). It searches for the donor at a minimum distance from the given recipient record but, in this case, the distance is computed on the percentage points of the empirical cumulative distribution function of the unique (continuous) common variable X_M being considered. The empirical cumulative distribution function is estimated by:

$$\hat{F}(x) = \frac{1}{n} \sum_{i=1}^{n} I\left(x_i \leq x\right)$$

being $I() = 1$ if $x_i \leq x$ and 0 otherwise. This transformation provides values uniformly distributed in the interval $[0, 1]$; moreover, it can be useful when the values of X_M can not be directly compared because of measurement errors which however do not affect the "position" of a unit in the whole distribution (D'Orazio *et al.*, 2006b). This method is implemented in the function `rankNND.hotdeck`. The following simple example shows how to call it.

```
> rnk.1 <- rankNND.hotdeck(data.rec=rec.A, data.don=don.B,
+                          var.rec="age", var.don="age")
```

```
> #create the synthetic data set
> fA.rnk <- create.fused(data.rec=rec.A, data.don=don.B,
+                        mtc.ids=rnk.1$mtc.ids,
+                        z.vars=c("netIncome", "c.netI"),
+                        dup.x=TRUE, match.vars="age")
> head(fA.rnk)
```

```
      hsize hsize6         db040 age    c.age  rb090 pb220a
4547      2      2     Carinthia  45  (24,49]   male     AT
9819      4      4      Salzburg  35  (24,49] female     AT
4461      2      2     Carinthia  57  (49,64]   male     AT
10222     2      2         Tyrol  69 (64,100] female     AT
8228      4      4 Upper Austria  25  (24,49] female     AT
3361      3      3        Vienna  22  [16,24]   male  Other
         rb050 pl030        work      wwA age.don netIncome
4547  6.863162     1     working 16.39250      45  17424.96
9819  6.089967     1     working 14.54575      35   8803.81
4461  6.863162     1     working 16.39250      58  43339.47
10222 6.857877     5 not working 16.37988      70   2820.05
8228  6.945309     4 not working 16.58871      25      0.00
3361  8.374000     1     working 20.00110      22   3016.03
        c.netI
4547   (15,20]
9819    (5,10]
4461   (40,50]
10222    (0,5]
8228    (-6,0]
3361     (0,5]
```

The function `rankNND.hotdeck` allows for constrained and unconstrained matching in the same manner as in `NND.hotdeck`. It is also possible to define some donation classes (argument `don.class`), in this case the empirical cumulative distribution is estimated separately class by class.

```
> rnk.2 <- rankNND.hotdeck(data.rec=rec.A, data.don=don.B, var.rec="age",
+                        var.don="age", don.class="rb090",
+                         constrained=TRUE, constr.alg="Hungarian")
> fA.grnk <- create.fused(data.rec=rec.A, data.don=don.B,
+                        mtc.ids=rnk.2$mtc.ids,
+                        z.vars=c("netIncome", "c.netI"),
+                        dup.x=TRUE, match.vars="age")
> head(fA.grnk)
```

```
     hsize hsize6         db040 age   c.age rb090 pb220a    rb050
4547     2      2     Carinthia  45 (24,49]  male     AT 6.863162
```

```
4461      2       2     Carinthia  57 (49,64]  male     AT 6.863162
3361      3       3        Vienna  22 [16,24]  male  Other 8.374000
827       2       2 Lower Austria  57 (49,64]  male     AT 6.913897
8061      3       3 Upper Austria  31 (24,49]  male     AT 7.509383
1925      4       4 Lower Austria  49 (24,49]  male     AT 7.757150
     pl030    work      wwA age.don netIncome   c.netI
4547     1 working 16.39250      46  23149.70  (20,25]
4461     1 working 16.39250      59  45463.71  (40,50]
3361     1 working 20.00110      22  30458.38  (30,40]
827      1 working 16.51368      59  53567.80 (50,200]
8061     1 working 17.93599      31  15863.65  (15,20]
1925     1 working 18.52777      51  56824.81 (50,200]
```

In estimating the empirical cumulative distribution it is possible to consider the units' weights (arguments `weight.rec` and `weight.don`). This topic will be tackled in Section 5.

3.4 Using functions in StatMatch to impute missing values in a survey

All the functions in **StatMatch** that implement the hot deck imputation techniques can be used to impute missing values in a single data set. In this case it is necessary to:

1. separate the observations in two data sets: the file A plays the role of recipient and will contain the units with missing values on the target variable, while the file B is the donor and will contain all the available donors (units with non missing values for the target variable).
2. Fill in the missing values in the recipient, e.g. by using a nonparametric imputation
3. Join recipient and donor file

In the following a simple example with the `iris` data.frame is reported. Distance hot deck is used to fill missing values in the recipient.

```
> # step 0) introduce missing values in iris
> set.seed(1324)
> miss <- rbinom(150, 1, 0.30) #generates randomly missing
> data(iris, package="datasets")
> iris.miss <- iris
> iris.miss$Petal.Length[miss==1] <- NA
> summary(iris.miss$Petal.L)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
    1.1     1.6     4.3     3.8     5.1     6.9      46
```

```
> #
> # step 1) separate units in two data sets
> rec <- subset(iris.miss, is.na(Petal.Length), select=-Petal.Length)
> don <- subset(iris.miss, !is.na(Petal.Length))
> #
> # step 2) search for closest donors
> X.mtc <- c("Sepal.Length", "Sepal.Width", "Petal.Width")
> nnd <- NND.hotdeck(data.rec=rec, data.don=don,
+                          match.vars=X.mtc, don.class="Species",
+                          dist.fun="Manhattan")

Warning: The  Manhattan  distance is being used
All the categorical matching variables in rec and don
 data.frames, if present are recoded into dummies

> # fills rec
> imp.rec <- create.fused(data.rec=rec, data.don=don,
+                         mtc.ids=nnd$mtc.ids, z.vars="Petal.Length")
> imp.rec$imp.PL <- 1 # flag for imputed
> #
> # step 3) re-aggregate data sets
> don$imp.PL <- 0
> imp.iris <- rbind(imp.rec, don)
> #summary stat of imputed and non imputed Petal.Length
> tapply(imp.iris$Petal.Length, imp.iris$imp.PL, summary)

$`0`
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    1.1     1.6     4.3     3.8     5.1     6.9 

$`1`
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.300   1.425   4.200   3.591   5.100   6.700 
```

4 Mixed methods

A SM mixed method consists of two steps: (1) a model is fitted and all its parameters are estimated, then (2) a nonparametric approach is used to create the synthetic data set. The model is more parsimonious while the nonparametric approach offers "protection" against model misspecification. The proposed mixed approaches for SM are based essentially on *predictive mean matching* imputation methods (see D'Orazio *et al.* 2006b, Section 2.5 and 3.6). The function `mixed.mtc` in **StatMatch** implements two similar mixed methods that deal with variables (X_M, Y, Z) following the the multivariate normal distribution. The main difference is in step (1) when estimating the parameters of the two regressions Y vs. X_M and Z vs. X_M. By default the parameters are estimated through maximum likelihood (argument `method="ML"` in `mixed.mtc`); in alternative a method proposed by Moriarity and Scheuren (2001, 2003) (argument `method="MS"`) is available. At the end of the step (1), the data set A is filled in with the "intermediate" values $\tilde{z}_a = \hat{z}_a + e_a$ ($a = 1, \ldots, n_A$) obtained by adding a random residual term e_a to the predicted values $\hat{z}_a$. The same happens in B which is filled in with the values $\tilde{y}_b = \hat{y}_b + e_b$ ($b = 1, \ldots, n_B$).

In the step (2) each record in A is filled in with the value of Z observed on the donor found in B according to a constrained distance hot deck; the Mahalanobis distance is computed by considering the intermediate and live values: couples $(y_a, \tilde{z}_a)$ in A and $(\tilde{y}_b, z_b)$ in B.

Such a two steps procedure presents various advantages: it offers protection against model misspecification and at the same time reduces the risk of bias in the marginal distribution of the imputed variable because the distances are computed on intermediate and truly observed values of the target value instead of the matching variables X_M. In fact when computing the distances by considering many matching variables, the variables with low predictive power on the target variable may influence negatively the distances.

D'Orazio *et al.* (2005) compared the two alternative methods based in an extensive simulation study: in general ML tends to perform better, moreover it permits to avoid some incoherencies in the estimation of the parameters that can happen with the Moriarity and Scheuren approach.

In the following example the `iris` data set is used just to show how `mixed.mtc` works.

```
> # uses iris data set
> iris.A <- iris[101:150, 1:3]
> iris.B <- iris[1:100, c(1:2,4)]
> X.mtc <- c("Sepal.Length","Sepal.Width") # matching variables
> # parameters estimated using ML
> mix.1 <- mixed.mtc(data.rec=iris.A, data.don=iris.B, match.vars=X.mtc,
+                      y.rec="Petal.Length", z.don="Petal.Width",
+                      method="ML", rho.yz=0,
+                      micro=TRUE, constr.alg="Hungarian")
> mix.1$mu #estimated means

Sepal.Length  Sepal.Width Petal.Length  Petal.Width
    5.843333     3.057333     4.996706     1.037109

> mix.1$cor #estimated cor. matrix

             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.9131794   0.8490516
Sepal.Width    -0.1175698   1.0000000   -0.0992586  -0.4415012
Petal.Length    0.9131794  -0.0992586    1.0000000   0.7725288
Petal.Width     0.8490516  -0.4415012    0.7725288   1.0000000
```

```
> head(mix.1$filled.rec) # A filled in with Z

    Sepal.Length Sepal.Width Petal.Length Petal.Width
101          6.3         3.3          6.0         0.2
102          5.8         2.7          5.1         1.3
103          7.1         3.0          5.9         1.7
104          6.3         2.9          5.6         1.4
105          6.5         3.0          5.8         1.5
106          7.6         3.0          6.6         1.8

> cor(mix.1$filled.rec)

             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  0.45722782    0.8642247  0.47606997
Sepal.Width     0.4572278  1.00000000    0.4010446 -0.01582276
Petal.Length    0.8642247  0.40104458    1.0000000  0.34391854
Petal.Width     0.4760700 -0.01582276    0.3439185  1.00000000
```

When using `mixed.mtc` the synthetic data set is provided in output as the component `filled.rec` of the list returned by calling it with the argument `micro=TRUE`. When `micro=FALSE` the function `mixed.mtc` returns just the estimates of the parameters (parametric macro approach).

The function `mixed.mtc` by default performs mixed SM under the CI assumption ($\rho_{YZ|X_M} = 0$ argument `rho.yz=0`). When some additional auxiliary information about the correlation between Y and Z is available (estimates from previous surveys or form external sources) then it can be exploited in SM by specifying a value ($\neq 0$) for the argument `rho.yz`; it represents a guess for $\rho_{YZ|X_M}$ when using the ML estimation, or a guess for ρ_{YZ} when estimating the parameters via the Moriarity and Scheuren approach.

```
> # parameters estimated using ML and rho_YZ|X=0.85
> mix.2 <- mixed.mtc(data.rec=iris.A, data.don=iris.B, match.vars=X.mtc,
+                    y.rec="Petal.Length", z.don="Petal.Width",
+                    method="ML", rho.yz=0.85,
+                    micro=TRUE, constr.alg="Hungarian")
> mix.2$cor

             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.9131794   0.8490516
Sepal.Width    -0.1175698   1.0000000   -0.0992586  -0.4415012
Petal.Length    0.9131794  -0.0992586    1.0000000   0.9113867
Petal.Width     0.8490516  -0.4415012    0.9113867   1.0000000

> head(mix.2$filled.rec)
```

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width
101          6.3         3.3          6.0         1.5
102          5.8         2.7          5.1         1.2
103          7.1         3.0          5.9         1.6
104          6.3         2.9          5.6         1.4
105          6.5         3.0          5.8         1.5
106          7.6         3.0          6.6         1.5
```

Special attention is required when specifying a guess for ρ_{YZ} under the Moriarity and Scheuren estimation approach (`method="MS"`); in particular it may happen that the specified value for ρ_{YZ} is not compatible with the given SM framework (the correlation matrix must be positive semidefinite). If this is the case, then `mixed.mtc` substitutes the input value of `rho.yz` by its closest admissible value, as shown in the following example.

```
> mix.3 <- mixed.mtc(data.rec=iris.A, data.don=iris.B, match.vars=X.mtc,
+                    y.rec="Petal.Length", z.don="Petal.Width",
+                    method="MS", rho.yz=0.75,
+                    micro=TRUE, constr.alg="Hungarian")

input value for rho.yz is 0.75
low(rho.yz)= -0.1662
up(rho.yz)= 0.5565
Warning: value for rho.yz is not admissible: a new value is chosen for it
The new value for rho.yz is  0.5465

> mix.3$rho.yz

  start low.lim  up.lim    used
 0.7500 -0.1662  0.5565  0.5465
```

5 Statistical matching of data from complex sample surveys

The SM techniques presented in the previous Sections implicitly or explicitly assume that the observed values in A and B are i.i.d. Unfortunately, when dealing with samples selected from a finite population by means of complex sampling designs (with stratification, clustering, etc.) it is difficult to maintain the i.i.d. assumption: it would mean that the sampling design can be ignored. If this is not the case, inferences have to account for sampling design and the weights assigned to the units (usually design weights corrected for unit nonresponse, frame errors, etc.) (see Särndal *et al.*, 1992, Section 13.6).

5.1 Naive micro approaches

A naive approach to SM of data from complex sample surveys consists in applying nonparametric micro methods (NND, random or rank hot deck) without considering the design nor the units weights. Once obtained the synthetic dataset (recipient filled in with the missing variables) the successive statistical analyses are carried out by considering the sampling design underlying the recipient data set and the corresponding survey weights. In the following a simple example of nearest neighbor hot deck is reported.

```
> # summary info on the weights
> sum(rec.A$wwA) # estimated pop size from A

[1] 67803

> sum(don.B$wwB) # estimated pop size from B

[1] 67803

> summary(rec.A$wwA)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  8.538  14.470  16.510  16.950  19.370  29.920

> summary(don.B$wwB)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  6.149  10.580  11.890  12.280  13.950  21.550

> # NND constrained hot deck
> group.v <- c("rb090","db040")
> out.nnd <- NND.hotdeck(data.rec=rec.A, data.don=don.B,
+                        match.vars="age", don.class=group.v,
+                        dist.fun="Manhattan",
+                        constrained=TRUE, constr.alg="Hungarian")

Warning: The  Manhattan  distance is being used
All the categorical matching variables in rec and don
 data.frames, if present are recoded into dummies

> fA.nnd.m <- create.fused(data.rec=rec.A, data.don=don.B,
+                     mtc.ids=out.nnd$mtc.ids,
+                     z.vars=c("netIncome","c.netI"))
> # estimating average net income
> weighted.mean(fA.nnd.m$netIncome, fA.nnd.m$wwA) # imputed in A

[1] 14940.63

> weighted.mean(don.B$netIncome, don.B$wwB) # ref. estimate in B

[1] 15073.95
```

```
> # comparing marginal distribution of C.netI using weights
> tt.0w <- xtabs(wwB~c.netI, data=don.B)
> tt.fw <- xtabs(wwA~c.netI, data=fA.nnd.m)
> comp.prop(p1=tt.fw, p2=tt.0w, n1=nrow(fA.nnd.m), ref=TRUE)

$meas
        tvd     overlap       Bhatt        Hell
0.009945058 0.990054942 0.999926091 0.008597056

$chi.sq
   Pearson          df      q0.05   delta.h0
 2.3762837  9.0000000 16.9189776  0.1404508

$p.exp
c.netI
    (-6,0]      (0,5]     (5,10]    (10,15]    (15,20]    (20,25]
0.11953297 0.09037855 0.14101622 0.17515499 0.18965562 0.13543995
   (25,30]    (30,40]    (40,50]   (50,200]
0.06746229 0.05171910 0.01334022 0.01630008
```

As far as imputation of missing values is concerned, a way of taking into account the sampling design can consist in forming the donation classes by using the design variables (stratification and/or clustering variables) jointly with the most relevant common variables (Andridge and Little, 2010). Unfortunately in SM this can increase the complexity or may be unfeasible because the design variables may not be available or may be partly available. Moreover, the two sample surveys may have quite different designs and the design variables used in one survey maybe not available in the other one and vice versa.

When imputing missing values in a survey, another possibility, consists in using sampling weights (design weights) to form the donation classes (Andridge and Little, 2010). But again, in SM applications the problem can be slightly more complex even because the sets of weights can be quite different from one survey to the other (usually the available weights are the design weights corrected to compensate for unit nonresponse, to satisfy some given constraints etc.). The same Authors (Andridge and Little, 2010) indicate that when imputing the missing values, the selection of the donors can be carried out with probability proportional to weights associated to the donors (*weighted random hot deck*). This feature is implemented in `RANDwNDD.hotdeck` which permits to select the donors with probability proportional to weights specified via the `weight.don` argument.

```
> group.v <- c("rb090","db040")
> X.mtc <- "age"
> rnd.2 <- RANDwNND.hotdeck(data.rec=rec.A, data.don=don.B,
+                            match.vars=NULL, don.class=group.v,
+                            weight.don="wwB")
> fA.wrnd <- create.fused(data.rec=rec.A, data.don=don.B,
```

```
+                          mtc.ids=rnd.2$mtc.ids,
+                          z.vars=c("netIncome","c.netI"))
> weighted.mean(fA.wrnd$netIncome, fA.wrnd$wwA) # imputed in A

[1] 14905.48

> weighted.mean(don.B$netIncome, don.B$wwB) # ref. estimate in B

[1] 15073.95

> # comparing marginal distribution of C.netI using weights
> tt.0w <- xtabs(wwB~c.netI, data=don.B)
> tt.fw <- xtabs(wwA~c.netI, data=fA.wrnd)
> comp.prop(p1=tt.fw, p2=tt.0w, n1=nrow(fA.nnd.m), ref=TRUE)

$meas
       tvd    overlap      Bhatt       Hell
0.01447498 0.98552502 0.99970755 0.01710121

$chi.sq
   Pearson         df      q0.05   delta.h0
 8.6084180  9.0000000 16.9189776  0.5088025

$p.exp
c.netI
    (-6,0]      (0,5]     (5,10]    (10,15]    (15,20]    (20,25]
0.11953297 0.09037855 0.14101622 0.17515499 0.18965562 0.13543995
   (25,30]    (30,40]    (40,50]   (50,200]
0.06746229 0.05171910 0.01334022 0.01630008
```

The function `rankNND.hotdeck` can use the units' weights (w_i) in estimating the percentage points of the the empirical cumulative distribution function:

$$\hat{F}(x) = \frac{\sum_{i=1}^{n} w_i I\left(x_i \leq x\right)}{\sum_{i=1}^{n} w_i}$$

In the following it is reported an very simple example with constrained rank hot deck.

```
> rnk.w <- rankNND.hotdeck(data.rec=rec.A, data.don=don.B,
+                           don.class="db040", var.rec="age",
+                           var.don="age", weight.rec="wwA",
+                           weight.don="wwB", constrained=TRUE,
+                           constr.alg="Hungarian")
> #
> #create the synthetic data set
> fA.wrnk <- create.fused(data.rec=rec.A, data.don=don.B,
```

```
+                          mtc.ids=rnk.w$mtc.ids,
+                          z.vars=c("netIncome", "c.netI"),
+                          dup.x=TRUE, match.vars="age")
> #
> weighted.mean(fA.wrnk$netIncome, fA.wrnk$wwA) # imputed in A

[1] 14656.13

> weighted.mean(don.B$netIncome, don.B$wwB) # ref. estimate in B

[1] 15073.95

> # comparing marginal distribution of C.netI using weights
> tt.0w <- xtabs(wwB~c.netI, data=don.B)
> tt.fw <- xtabs(wwA~c.netI, data=fA.wrnk)
> comp.prop(p1=tt.fw, p2=tt.0w, n1=nrow(fA.nnd.m), ref=TRUE)

$meas
       tvd    overlap      Bhatt       Hell
0.01360393 0.98639607 0.99978278 0.01473855

$chi.sq
   Pearson         df      q0.05   delta.h0
 6.9805051  9.0000000 16.9189776  0.4125843

$p.exp
c.netI
    (-6,0]      (0,5]     (5,10]    (10,15]    (15,20]    (20,25]
0.11953297 0.09037855 0.14101622 0.17515499 0.18965562 0.13543995
   (25,30]    (30,40]    (40,50]   (50,200]
0.06746229 0.05171910 0.01334022 0.01630008
```

D'Orazio *et al.* (2012) compared several naive procedures. In general, when rank and random hot deck procedures use the weights, as shown before, they tend to perform quite well in terms of preservation in the synthetic data set of the marginal distribution of the imputed variable Z and of the joint distribution $X \times Z$. The nearest neighbour donor, performs well only when constrained matching is used and a design variable (used in stratification) is considered in forming donation classes.

5.2 Statistical matching method that account explicitly for the sampling weights

In literature there are few SM methods that explicitly take into account the sampling design and the corresponding sampling weights: Renssen's approach based on weights' *calibrations* (Renssen, 1998); Rubin's *file concatenation* (Rubin, 1986) and the approach based on the empirical likelihood proposed by Wu (2004). A comparison among these approaches can be found in D'Orazio (2010).

The package **StatMatch** provides functions to apply the procedures suggested by Renssen (1998). Renssen's approach consists in a series of calibration steps of the survey weights in A and B in order to achieve consistency between estimates (mainly totals) computed separately from the two data sources. Calibration is a technique very common in sample surveys for deriving new weights, as close as possible to the starting ones, which fulfill a series of constraints concerning totals for a set of auxiliary variables (for further details on calibration see Särndal, 2005). The Renssen's approach works well when dealing with categorical variables or in a mixed case in which the number of continuous variables is very limited. In the following it will be assumed that all the variables (X_D, Y, Z) are categorical, being X_D a complete or an incomplete crossing of the matching variables X_M. The procedure and the functions developed in **StatMatch** permits to have one or more continuous variables (better just one) in the subset of the matching variables X_M, while Y and Z must be categorical. Obviously, when this is not the case, in order to apply the following procedure it is necessary to categorize the variables.

The first step in the Renssen's procedure consists in calibrating weights in A and in B such that the new weights when applied to the set of the X_D variables allow to reproduce some known (or estimated) population totals. In **StatMatch** the harmonization step can be performed by using `harmonize.x`. This function performs weights calibration (or post-stratification) by means of functions available in the R package **survey** (Lumley, 2012). When the population totals are already known then they have to be passed to `harmonize.x` via the argument `x.tot`; on the contrary, when they are unknown (`x.tot=NULL`) they are estimated by a weighted average of the totals estimated on the two surveys before the harmonization step:

$$\tilde{t}_{X_D} = \lambda \hat{t}^{(A)}_{X_D} + (1-\lambda)\,\hat{t}^{(B)}_{X_D}$$

being $\lambda = n_A/(n_A + n_B)$ (n_A and n_B are the sample sizes of A and B respectively) (Korn and Graubard, 1999, pp. 281–284).

The following example shows how to harmonize the joint distribution of the gender and classes of age with the data from the previous example, assuming that the joint distribution of age and gender is not known.

```
> tt.A <- xtabs(wwA~rb090+c.age, data=rec.A)
> tt.B <- xtabs(wwB~rb090+c.age, data=don.B)
> (prop.table(tt.A)-prop.table(tt.B))*100

        c.age
rb090       [16,24]    (24,49]    (49,64]   (64,100]
  male    0.3661141  1.0995148 -0.9456418 -0.6383618
  female  0.0891681 -0.4970682  1.0772175 -0.5509426

> comp.prop(p1=tt.A, p2=tt.B, n1=nrow(rec.A),
+           n2=nrow(don.B), ref=FALSE)
```

```
$meas
       tvd    overlap      Bhatt       Hell
0.02632014 0.97367986 0.99956825 0.02077856

$chi.sq
   Pearson         df      q0.05   delta.h0
 8.0082627  7.0000000 14.0671404  0.5692886

$p.exp
        c.age
rb090       [16,24]    (24,49]    (49,64]   (64,100]
  male   0.07010041 0.22316357 0.11129434 0.07671609
  female 0.06578194 0.22773205 0.11842264 0.10678897

> library(survey, warn.conflicts=FALSE) # loads survey
> # creates svydesign objects
> svy.rec.A <- svydesign(~1, weights=~wwA, data=rec.A)
> svy.don.B <- svydesign(~1, weights=~wwB, data=don.B)
> #
> # harmonizes wrt to joint distr. of gender vs. c.age
> out.hz <- harmonize.x(svy.A=svy.rec.A, svy.B=svy.don.B,
+                  form.x=~c.age:rb090-1)
> #
> summary(out.hz$weights.A) # new calibrated weights for A

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  8.647  14.390  16.570  16.950  19.030  31.470

> summary(out.hz$weights.B) # new calibrated weights for B

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  6.279  10.540  11.840  12.280  13.910  22.400

> tt.A <- xtabs(out.hz$weights.A~rb090+c.age, data=rec.A)
> tt.B <- xtabs(out.hz$weights.B~rb090+c.age, data=don.B)
> comp.prop(p1=tt.A, p2=tt.B, n1=nrow(rec.A),
+           n2=nrow(don.B), ref=FALSE)

$meas
         tvd      overlap        Bhatt         Hell
4.163336e-17 1.000000e+00 1.000000e+00 0.000000e+00

$chi.sq
     Pearson           df        q0.05     delta.h0
8.940923e-29 7.000000e+00 1.406714e+01 6.355892e-30
```

```
$p.exp
        c.age
rb090        [16,24]    (24,49]    (49,64]   (64,100]
  male   0.07010041 0.22316357 0.11129434 0.07671609
  female 0.06578194 0.22773205 0.11842264 0.10678897
```

The second step in the Renssen's procedure consists in estimating the two-way contingency table $Y \times Z$. In absence of auxiliary information it is estimated under the CI assumption by means of:

$$\hat{P}^{(CIA)}_{(Y=j,Z=k)} = \hat{P}^{(A)}_{Y=j|X_D=i} \times \hat{P}^{(B)}_{Z=k|X_D=i} \times \hat{P}_{X_D=i}$$

for $i = 1, \ldots, I;\ j = 1, \ldots, J;\ K = 1, \ldots, K;$

In practice, $\hat{P}^{(A)}_{Y=j|X_D=i}$ is computed from A; $\hat{P}^{(B)}_{Z=k|X_D=i}$ is computed from data in B while $P_{X_D=i}$ can be estimated indifferently from A or B (the data set are harmonized with respect to the X_D distribution).

In **StatMatch** an estimate of the table $Y \times Z$ under the CIA is provided by the function `comb.samples`.

```
> # estimating c.pl030 vs. c.netI under the CI assumption
> out <- comb.samples(svy.A=out.hz$cal.A, svy.B=out.hz$cal.B,
+             svy.C=NULL, y.lab="work", z.lab="c.netI",
+             form.x=~c.age:rb090-1)
> #
> addmargins(t(out$yz.CIA))  # table estimated under the CIA

           working not working        Sum
(-6,0]    4203.9273   3929.4698  8133.3971
(0,5]     3212.7539   2941.5722  6154.3261
(5,10]    4436.4472   5108.0075  9544.4547
(10,15]   5648.5383   6199.2373 11847.7756
(15,20]   7129.6193   5716.1572 12845.7765
(20,25]   5391.3879   3802.7509  9194.1388
(25,30]   2877.6585   1696.1470  4573.8055
(30,40]   2249.5066   1256.9719  3506.4786
(40,50]    555.7829    345.2169   900.9998
(50,200]   688.8992    412.9481  1101.8473
Sum      36394.5210  31408.4790 67803.0000
```

When some auxiliary information is available, e.g. a third data source C, containing all the variables (X_M, Y, Z) or just (Y, Z), the Renssen's approach permits to exploit it in estimating $Y \times Z$. Two alternative methods are available: (a) *incomplete two-way stratification*; and (b) *synthetic two-way stratification*. In practice, both the methods estimate $Y \times Z$ from C after some further calibration steps (for further details see Renssen, 1998). The function `comb.samples` implements both the methods. In practice, the synthetic two-way stratification (argument `estimation="synthetic"`) can be applied only when C contains all the variables of interest (X_M, Y, Z); on the contrary, when the data source C observes just Y and Z, only the incomplete two-way stratification method can be applied (argument `estimation="incomplete"`). In the following a simple example is reported based on the artificial EU-SILC data introduced in Section 2.1; here a small sample C ($n_C = 200$) with all the variables of interest (X_M, Y, Z) is artificially created.

```
> # generating artificial sample C
> set.seed(43210)
> obs.C <- sample(nrow(silc.16), 200, replace=F)
> #
> X.vars <- c("hsize","hsize6","db040","age","c.age",
+             "rb090","pb220a", "rb050")
> y.var <- c("pl030","work")
> z.var <- c("netIncome","c.netI")
> #
> aux.C <- silc.16[obs.C, c(X.vars, y.var, z.var)]
> aux.C$wwC <- aux.C$rb050/sum(aux.C$rb050)*round(sum(silc.16$rb050)) # rough w
> svy.aux.C <- svydesign(~1, weights=~wwC, data=aux.C)
> #
> # incomplete two-way estimation
> out.inc <- comb.samples(svy.A=out.hz$cal.A, svy.B=out.hz$cal.B,
+                  svy.C=svy.aux.C, y.lab="work", z.lab="c.netI",
+                  form.x=~c.age:rb090-1, estimation="incomplete")
> addmargins(t(out.inc$yz.est))

            working not working        Sum
(-6,0]     318.3646   7815.0325  8133.3971
(0,5]     3155.6684   2998.6577  6154.3261
(5,10]    3960.8064   5583.6483  9544.4547
(10,15]   4736.0014   7111.7742 11847.7756
(15,20]   9302.3226   3543.4539 12845.7765
(20,25]   6318.9931   2875.1457  9194.1388
(25,30]   4011.6435    562.1620  4573.8055
(30,40]   2587.8739    918.6047  3506.4786
(40,50]    900.9998      0.0000   900.9998
(50,200]  1101.8473      0.0000  1101.8473
Sum      36394.5210  31408.4790 67803.0000
```

The incomplete two-way stratification method estimates the table $Y \times Z$ from C by preserving the marginal distribution of Y and of Z estimated respectively from A and from B after the initial harmonization step; on the contrary, the joint distribution of the matching variables (which is the basis of the harmonization step) is not preserved.

```
> new.wwC <- weights(out.inc$cal.C) #new cal. weights for C
> #
> # marginal distributions of work
> m.work.cA <- xtabs(out.hz$weights.A~work, data=rec.A)
> m.work.cC <- xtabs(new.wwC~work, data=aux.C)
> m.work.cA-m.work.cC

work
    working not working
          0           0

> #
> # marginal distributions of c.netI
> m.cnetI.cB <- xtabs(out.hz$weights.B~c.netI, data=don.B)
> m.cnetI.cC <- xtabs(new.wwC~c.netI, data=aux.C)
> m.cnetI.cB-m.cnetI.cC

c.netI
       (-6,0]         (0,5]        (5,10]       (10,15]
 6.366463e-12 -9.094947e-13  0.000000e+00  0.000000e+00
      (15,20]       (20,25]       (25,30]       (30,40]
 0.000000e+00  0.000000e+00  9.094947e-13  0.000000e+00
      (40,50]      (50,200]
 2.273737e-13  2.273737e-13

> # joint distribution of the matching variables
> tt.A <- xtabs(out.hz$weights.A~rb090+c.age, data=rec.A)
> tt.B <- xtabs(out.hz$weights.B~rb090+c.age, data=don.B)
> tt.C <- xtabs(new.wwC~rb090+c.age, data=aux.C)
> comp.prop(p1=tt.A, p2=tt.B, n1=nrow(rec.A),
+           n2=nrow(don.B), ref=FALSE)

$meas
         tvd      overlap        Bhatt         Hell
4.163336e-17 1.000000e+00 1.000000e+00 0.000000e+00

$chi.sq
     Pearson           df        q0.05     delta.h0
8.940923e-29 7.000000e+00 1.406714e+01 6.355892e-30

$p.exp
        c.age
rb090       [16,24]    (24,49]    (49,64]   (64,100]
  male   0.07010041 0.22316357 0.11129434 0.07671609
  female 0.06578194 0.22773205 0.11842264 0.10678897
```

```
> comp.prop(p1=tt.C, p2=tt.A, n1=nrow(aux.C),
+           n2=nrow(rec.A), ref=FALSE)

$meas
       tvd    overlap      Bhatt       Hell
0.05326808 0.94673192 0.99727747 0.05217785

$chi.sq
   Pearson         df      q0.05   delta.h0
 4.6813708  7.0000000 14.0671404  0.3327877

$p.exp
        c.age
rb090       [16,24]    (24,49]    (49,64]   (64,100]
  male   0.07179371 0.22345791 0.11060518 0.07710921
  female 0.06558083 0.22671249 0.11857843 0.10616222
```

As said before, the synthetic two-way stratification (argument `estimation="synthetic"`) requires that the auxiliary data source C contains the matching variables X_M and the target variables Y and Z.

```
> # synthetic two-way estimation
> out.synt <- comb.samples(svy.A=out.hz$cal.A, svy.B=out.hz$cal.B,
+                 svy.C=svy.aux.C, y.lab="work", z.lab="c.netI",
+                 form.x=~c.age:rb090-1, estimation="synthetic")
> #
> addmargins(t(out.synt$yz.est))

           working not working        Sum
(-6,0]    351.6488   7781.7483  8133.3971
(0,5]    3610.2537   2544.0724  6154.3261
(5,10]   4052.7261   5491.7286  9544.4547
(10,15]  5384.8795   6462.8961 11847.7756
(15,20]  8542.0337   4303.7428 12845.7765
(20,25]  5971.5562   3222.5826  9194.1388
(25,30]  3781.3214    792.4840  4573.8055
(30,40]  2697.2545    809.2241  3506.4786
(40,50]   900.9998      0.0000   900.9998
(50,200] 1101.8473      0.0000  1101.8473
Sum     36394.5210  31408.4790 67803.0000
```

As in the case of incomplete two-way stratification, also the synthetic two-way stratification derives the table $Y \times Z$ from C by preserving the marginal distribution of Y and of Z estimated respectively from A and from B after the initial harmonization step; on the contrary, the joint distribution of the matching variables (which is the basis of the harmonization step) is still not preserved.

```
> new.wwC <- weights(out.synt$cal.C) #new cal. weights for C
> #
> # marginal distributions of work
> m.work.cA <- xtabs(out.hz$weights.A~work, data=rec.A)
> m.work.cC <- xtabs(new.wwC~work, data=aux.C)
> m.work.cA-m.work.cC

work
     working  not working
2.910383e-11 6.912160e-11

> # marginal distributions of c.netI
> m.cnetI.cB <- xtabs(out.hz$weights.B~c.netI, data=don.B)
> m.cnetI.cC <- xtabs(new.wwC~c.netI, data=aux.C)
> m.cnetI.cB-m.cnetI.cC

c.netI
       (-6,0]          (0,5]         (5,10]        (10,15]
 2.819434e-11  1.637090e-11  1.637090e-11  2.364686e-11
      (15,20]        (20,25]        (25,30]        (30,40]
 9.094947e-12  7.275958e-12  4.547474e-12 -3.637979e-12
      (40,50]       (50,200]
-1.705303e-12 -2.501110e-12

> # joint distribution of the matching variables
> tt.A <- xtabs(out.hz$weights.A~rb090+c.age, data=rec.A)
> tt.B <- xtabs(out.hz$weights.B~rb090+c.age, data=don.B)
> tt.C <- xtabs(new.wwC~rb090+c.age, data=aux.C)
> comp.prop(p1=tt.A, p2=tt.B, n1=nrow(rec.A),
+           n2=nrow(don.B), ref=FALSE)

$meas
         tvd      overlap        Bhatt         Hell
4.163336e-17 1.000000e+00 1.000000e+00 0.000000e+00

$chi.sq
     Pearson           df        q0.05     delta.h0
8.940923e-29 7.000000e+00 1.406714e+01 6.355892e-30

$p.exp
        c.age
rb090       [16,24]    (24,49]    (49,64]   (64,100]
  male   0.07010041 0.22316357 0.11129434 0.07671609
  female 0.06578194 0.22773205 0.11842264 0.10678897
```

```
> comp.prop(p1=tt.C, p2=tt.A, n1=nrow(aux.C),
+           n2=nrow(rec.A), ref=FALSE)

$meas
       tvd    overlap      Bhatt       Hell
0.04533274 0.95466726 0.99721476 0.05277541

$chi.sq
  Pearson        df     q0.05  delta.h0
 4.750685  7.000000 14.067140  0.337715

$p.exp
       c.age
rb090       [16,24]    (24,49]    (49,64]   (64,100]
  male   0.07181280 0.22295324 0.11042887 0.07692488
  female 0.06555624 0.22773120 0.11866015 0.10593262
```

It is worth noting that `comb.samples` can also be used for micro imputation. In particular, when the argument `micro` is set to `TRUE` the function returns also the two data frames `Z.A` and `Y.B`. The first ones 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 for a unit 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 probabilities are obtained as a by-product of the whole procedure which is based on the usage of the *linear probability models* (for major details see Renssen, 1998). The procedure corresponds to a regression imputation that when dealing with all categorical variables (X_D, Y, Z), provides a synthetic data set (A filled in with Z) which preserves the marginal distribution of the Z variable and the joint distribution $X \times Z$. Unfortunately, linear probability models have some well known drawbacks and may provide estimated probabilities less than 0 or greater than 1. For this reason, such predictions should be used carefully.

D'orazio *et al.* (2012) suggest using a randomization mechanism to derive the predicted category starting from the estimated probabilities.

```
> # predicting prob of c.netI in A under the CI assumption
> out <- comb.samples(svy.A=out.hz$cal.A, svy.B=out.hz$cal.B,
+                     svy.C=NULL, y.lab="work", z.lab="c.netI",
+                     form.x=~c.age:rb090-1, micro=TRUE)
> head(out$Z.A)

         c.netI1    c.netI2    c.netI3   c.netI4   c.netI5
4547  0.02431737 0.03461536 0.07333853 0.1260644 0.2441140
9819  0.18449296 0.11651122 0.17192894 0.1828479 0.1536327
4461  0.01360657 0.02121963 0.08784363 0.1647151 0.2455691
10222 0.12862694 0.08280089 0.24704563 0.2624476 0.1531360
```

```
8228  0.18449296 0.11651122 0.17192894 0.1828479 0.1536327
3361  0.23596552 0.20079618 0.13191092 0.1593456 0.1707472
         c.netI6    c.netI7     c.netI8     c.netI9    c.netI10
4547  0.22507260 0.12213224 0.099898952 0.020044253 0.030402300
9819  0.10775094 0.04282870 0.024714585 0.009347124 0.005944963
4461  0.14258653 0.12907572 0.116517140 0.036077503 0.042789098
10222 0.09119902 0.01759877 0.011944017 0.003473668 0.001727448
8228  0.10775094 0.04282870 0.024714585 0.009347124 0.005944963
3361  0.07521334 0.02080676 0.005214446 0.000000000 0.000000000

> # predicting prob of c.netI in A under the CI assumption
> out <- comb.samples(svy.A=out.hz$cal.A, svy.B=out.hz$cal.B,
+                     svy.C=NULL, y.lab="work", z.lab="c.netI",
+                     form.x=~c.age:rb090-1, micro=TRUE)
> head(out$Z.A)

         c.netI1    c.netI2    c.netI3   c.netI4   c.netI5
4547  0.02431737 0.03461536 0.07333853 0.1260644 0.2441140
9819  0.18449296 0.11651122 0.17192894 0.1828479 0.1536327
4461  0.01360657 0.02121963 0.08784363 0.1647151 0.2455691
10222 0.12862694 0.08280089 0.24704563 0.2624476 0.1531360
8228  0.18449296 0.11651122 0.17192894 0.1828479 0.1536327
3361  0.23596552 0.20079618 0.13191092 0.1593456 0.1707472
         c.netI6    c.netI7     c.netI8     c.netI9    c.netI10
4547  0.22507260 0.12213224 0.099898952 0.020044253 0.030402300
9819  0.10775094 0.04282870 0.024714585 0.009347124 0.005944963
4461  0.14258653 0.12907572 0.116517140 0.036077503 0.042789098
10222 0.09119902 0.01759877 0.011944017 0.003473668 0.001727448
8228  0.10775094 0.04282870 0.024714585 0.009347124 0.005944963
3361  0.07521334 0.02080676 0.005214446 0.000000000 0.000000000

> sum(out$Z.A<0) # negative est. prob.

[1] 0

> sum(out$Z.A>1) # est. prob. >1

[1] 0

> # compare marginal distributions of Z
> t.zA <- colSums(out$Z.A*out.hz$weights.A)
> t.zB <- xtabs(out.hz$weights.B~don.B$c.netI)
> comp.prop(p1=t.zA, p2=t.zB, n1=nrow(rec.A), ref=TRUE)
```

```
$meas
         tvd      overlap        Bhatt         Hell
1.951564e-16 1.000000e+00 1.000000e+00 0.000000e+00

$chi.sq
     Pearson           df        q0.05     delta.h0
7.816082e-28 9.000000e+00 1.691898e+01 4.619713e-29

$p.exp
don.B$c.netI
    (-6,0]      (0,5]     (5,10]    (10,15]    (15,20]    (20,25]
0.11995630 0.09076776 0.14076744 0.17473822 0.18945735 0.13560077
   (25,30]    (30,40]    (40,50]   (50,200]
0.06745727 0.05171568 0.01328849 0.01625072

> # predicting class of netIncome in A
> # randomized prediction with prob proportional to estimated prob.
> pred.zA <- apply(out$Z.A,1,sample,x=1:ncol(out$Z.A), size=1,replace=F)
> rec.A$c.netI <- factor(pred.zA, levels=1:nlevels(don.B$c.netI),
+                    labels=as.character(levels(don.B$c.netI)), ordered=T)
> # comparing marginal distributions of Z
> t.zA <- xtabs(out.hz$weights.A~rec.A$c.netI)
> comp.prop(p1=t.zA, p2=t.zB, n1=nrow(rec.A), ref=TRUE)

$meas
       tvd    overlap      Bhatt       Hell
0.02609102 0.97390898 0.99950055 0.02234844

$chi.sq
   Pearson         df      q0.05   delta.h0
15.7141897  9.0000000 16.9189776  0.9287907

$p.exp
don.B$c.netI
    (-6,0]      (0,5]     (5,10]    (10,15]    (15,20]    (20,25]
0.11995630 0.09076776 0.14076744 0.17473822 0.18945735 0.13560077
   (25,30]    (30,40]    (40,50]   (50,200]
0.06745727 0.05171568 0.01328849 0.01625072

> # comparing joint distributions of X vs. Z
> t.xzA <- xtabs(out.hz$weights.A~c.age+rb090+c.netI, data=rec.A)
> t.xzB <- xtabs(out.hz$weights.B~c.age+rb090+c.netI, data=don.B)
> out.comp <- comp.prop(p1=t.xzA, p2=t.xzB, n1=nrow(rec.A), ref=TRUE)
> out.comp$meas
```

```
      tvd    overlap      Bhatt       Hell
0.05162525 0.94837475 0.99707673 0.05406729

> out.comp$chi.sq

   Pearson         df      q0.05   delta.h0
74.5161159 75.0000000 96.2166708  0.7744616
```

6 Exploring uncertainty due to the statistical matching framework

When the objective of SM consists in estimating a parameter (macro approach) it is possible to tackle SM in an alternative way consisting in the "exploration" of the uncertainty on the model chosen for (X_M, Y, Z), due to the lack of knowledge typical of the basic SM framework (no auxiliary information is available). This approach does not end with a unique estimate of the unknown parameter characterizing the joint p.d.f. for (X_D, Y, Z); on the contrary it identifies an interval of plausible values for it. When dealing with categorical variables, the estimation of the intervals of plausible values for the probabilities in the table $Y \times Z$ are provided by the Fréchet bounds:

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

for $j = 1, \dots, J$ and $k = 1, \dots, K$, being J and K the categories of Y and Z respectively.

Let consider the matching variables X_M, for sake of simplicity let assume that X_D is the variable obtained by the crossproduct of the chosen X_M variables; by conditioning on X_D, it is possible to derive the following result (D'Orazio *et al.*, 2006a):

$$P^{(low)}_{j,k} \;\leq\; P_{Y=j,Z=k} \;\leq\; P^{(up)}_{j,k}$$

with

$$\begin{aligned} P^{(low)}_{j,k} &= \sum_i P_{X_D=i} \times \max\left\{0; P_{Y=j|X_D=i} + P_{Z=k|X_D=i} - 1\right\} \\ P^{(up)}_{j,k} &= \sum_i P_{X_D=i} \times \min\left\{P_{Y=j|X_D=i}; P_{Z=k|X_D=i}\right\} \end{aligned}$$

for $j = 1, \dots, J$ and $k = 1, \dots, K$. It is interesting to observe that the CIA estimate of $P_{Y=j,Z=k}$ is always included in the interval identified by such bounds:

$$P^{(low)}_{j,k} \;\leq\; P^{(CIA)}_{Y=j,Z=k} \;\leq\; P^{(up)}_{j,k}$$

In the SM basic framework, the probabilities $P_{Y=j|X_D=i}$ are estimated from A, the $P_{Z=k|X_D=i}$ are estimated from B, while the marginal distribution $P_{X_D=i}$ can be estimated indifferently on A or on B, assuming that both the samples, being representative samples of the same population, provide not significantly different estimates of $P(X_M = i)$. If this is not the case, before computing the bounds it would be preferable to harmonize the distribution of X_D in A and in B by using the function `harmonize.x`.

In **StatMatch** the Fréchet bounds for $P_{Y=j,Z=k}$ ($j = 1, \ldots, J$ and $k = 1, \ldots, K$), conditioned or not on X_D, are provided by `Frechet.bounds.cat`.

```
> #comparing joint distribution of the X_M variables in A and in B
> t.xA <- xtabs(wwA~c.age+rb090, data=rec.A)
> t.xB <- xtabs(wwB~c.age+rb090, data=don.B)
> comp.prop(p1=t.xA, p2=t.xB, n1=nrow(rec.A), n2=nrow(don.B), ref=FALSE)

$meas
       tvd    overlap      Bhatt       Hell
0.02632014 0.97367986 0.99956825 0.02077856

$chi.sq
   Pearson         df      q0.05   delta.h0
 8.0082627  7.0000000 14.0671404  0.5692886

$p.exp
         rb090
c.age           male     female
  [16,24]  0.07010041 0.06578194
  (24,49]  0.22316357 0.22773205
  (49,64]  0.11129434 0.11842264
  (64,100] 0.07671609 0.10678897

> #
> #computing tables needed by Frechet.bounds.cat
> t.xy <- xtabs(wwA~c.age+rb090+work, data=rec.A)
> t.xz <- xtabs(wwB~c.age+rb090+c.netI, data=don.B)
> out.fb <- Frechet.bounds.cat(tab.x=t.xA, tab.xy=t.xy, tab.xz=t.xz,
+                               print.f="data.frame")
> out.fb

$bounds
         work   c.netI low.u        low.cx         CIA      up.cx
1     working   (-6,0]     0 0.0000000000 0.062451939 0.10745732
2 not working   (-6,0]     0 0.0130833912 0.058088772 0.12054071
3     working    (0,5]     0 0.0000000000 0.047854165 0.08127010
4 not working    (0,5]     0 0.0100349443 0.043450884 0.09130505
5     working   (5,10]     0 0.0000000000 0.065841680 0.10790942
6 not working   (5,10]     0 0.0325145796 0.074582323 0.14042400
7     working  (10,15]     0 0.0044505872 0.083877816 0.13317699
8 not working  (10,15]     0 0.0409858756 0.090285053 0.16971228
9     working  (15,20]     0 0.0315476801 0.106111106 0.15614074
```

```
10 not working  (15,20]     0 0.0330428837 0.083072522 0.15763595
11     working  (20,25]     0 0.0271769197 0.080524320 0.11362462
12 not working  (20,25]     0 0.0221981534 0.055298451 0.10864585
13     working  (25,30]     0 0.0035480015 0.042818593 0.06158708
14 not working  (25,30]     0 0.0058632580 0.024631748 0.06390234
15     working  (30,40]     0 0.0000000000 0.033456492 0.04850157
16 not working  (30,40]     0 0.0032094037 0.018254479 0.05171097
17     working  (40,50]     0 0.0000000000 0.008213067 0.01309882
18 not working  (40,50]     0 0.0001182705 0.005004024 0.01321709
19     working (50,200]     0 0.0000000000 0.010221237 0.01592268
20 not working (50,200]     0 0.0002598896 0.005961328 0.01618257
         up.u
1  0.11953297
2  0.11953297
3  0.09037855
4  0.09037855
5  0.14101622
6  0.14101622
7  0.17515499
8  0.17515499
9  0.18965562
10 0.18965562
11 0.13543995
12 0.13543995
13 0.06746229
14 0.06746229
15 0.05171910
16 0.05171910
17 0.01334022
18 0.01334022
19 0.01630008
20 0.01630008

$uncertainty
      av.u      av.cx    overall
0.10000000 0.07719662 0.11853164
```

The final component of the output list provided by `Frechet.bounds.cat` summarizes the uncertainty by means of the average width of the unconditioned bounds, the average width of the bounds obtained by conditioning on X_D and the overall uncertainty measured as suggested by Conti *et al.* (2012) (see Section 2.2).

When dealing with continuous variables, if it is assumed that their joint distribution is multivariate normal, the uncertainty bounds for the correlation coefficient ρ_{YZ} can be obtained by using the function `mixed.mtc` with argument `method="MS"`. The following example assumes multivariate normal distribution holding for joint distribution for age, gender (the matching variables), aggregated personal economic status (binary variable "work" which plays the role of Y) and log-transformed personal net income (log of "netIncome" which plays the role of Z).

```
> # continuous variables
> don.B$log.netI <- log( ifelse(don.B$netIncome>0, don.B$netIncome, 0)+1 )
> X.mtc <- c("age","rb090")
> mix.3 <- mixed.mtc(data.rec=rec.A, data.don=don.B, match.vars=X.mtc,
+                    y.rec="work", z.don="log.netI",
+                    method="MS")

input value for rho.yz is 0.03095
low(rho.yz)= -0.8048
up(rho.yz)= 0.8667
The input value for rho.yz is admissible
```

When a single X variable it is considered the bounds can be obtained explicitly by using formula in Section 1.

References

Alfons A., Kraft S. (2012) "simPopulation: Simulation of synthetic populations for surveys based on sample data". R package version 0.4.0 http://CRAN.R-project.org/package=simPopulation

Andridge R.R., Little R.J.A. (2009) "The Use of Sample Weights in Hot Deck Imputation". *Journal of Official Statistics*, **25**(1), 21–36.

Andridge R.R., Little R.J.A. (2010) "A Review of Hot Deck Imputation for Survey Nonresponse". *International Statistical Review*, **78**, 40–64.

Berkelaar M. and others (2011) "lpSolve: Interface to Lpsolve v. 5.5 to solve linear–integer programs". R package version 5.6.6. http://CRAN.R-project.org/package=lpSolve

Breiman, L. (2001) "Random Forests", *Machine Learning*, **45**(1), 5–32.

Breiman L., Friedman J. H., Olshen R. A., and Stone, C. J. (1984) *Classification and Regression Trees*. Wadsworth.

Cohen M.L. (1991) "Statistical matching and microsimulation models", in Citro and Hanushek (eds) *Improving Information for Social Policy Decisions: The Uses of Microsimulation Modeling. Vol II Technical papers*. Washington D.C.

Conti, P.L., Marella, D., and Scanu, M. (2012) "Uncertainty Analysis in Statistical Matching", *Journal of Official Statistics*, **28**, 69–88.

D'Orazio M. (2010) "Statistical matching when dealing with data from complex survey sampling", in *Report of WP1. State of the art on statistical methodologies for data integration*, ESSnet project on Data Integration, 33–37, http://www.essnet-portal.eu/sites/default/files/131/ESSnetDI_WP1_v1.32.pdf

D'Orazio M. (2012) "StatMatch: Statistical Matching". R package version 1.2.0. http://CRAN.R-project.org/package=StatMatch

D'Orazio M., Di Zio M., Scanu, M. (2005) "A comparison among different estimators of regression parameters on statistically matched files trough an extensive simulation study". *Contributi Istat*, **2005/10**

D'Orazio M., Di Zio M., Scanu M. (2006a) "Statistical matching for categorical data: Displaying uncertainty and using logical constraints". *Journal of Official Statistics* **2**2, 137–157.

D'Orazio M., Di Zio M., Scanu M. (2006b) *Statistical matching: Theory and practice*. Wiley, Chichester.

D'Orazio M., Di Zio M., Scanu M. (2008) "The statistical matching workflow", in: *Report of WP1: State of the art on statistical methodologies for integration of surveys and administrative data*, "ESSnet Statistical Methodology Project on Integration of Survey and Administrative Data", 25–26. http://cenex-isad.istat.it/

D'Orazio M., Di Zio M., Scanu M. (2010) "Old and new approaches in statistical matching when samples are drawn with complex survey designs". *Proceedings of the 45th "Riunione Scientifica della Societa' Italiana di Statistica"*, Padova 16–18 June 2010.

D'Orazio M., Di Zio M., Scanu M. (2012) "Statistical Matching of Data from Complex Sample Surveys". *Proceedings of the European Conference on Quality in Official Statistics - Q2012*, 29 May–1 June 2012, Athens, Greece.

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

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

Korn E.L., Graubard B.I. (1999) *Analysis of Health Surveys*. Wiley, New York

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

Liaw A., Wiener M. (2002) "Classification and Regression by randomForest". *R News*, **2**(3), 18–22.

Lumley T. (2012) "survey: analysis of complex survey samples". R package version 3.28-2. http://CRAN.R-project.org/package=survey

Meyer D., Buchta C. (2012) "proxy: Distance and Similarity Measures". R package version 0.4-9. http://CRAN.R-project.org/package=proxy

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

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

R Development Core Team (2012) *R: A language and environment for statistical computing. R Foundation for Statistical Computing*, Vienna, Austria. ISBN 3-900051-07-0, http://www.R-project.org/.

Rässler S. (2002) *Statistical matching: a frequentist theory, practical applications and alternative Bayesian approaches*. Springer Verlag, New York.

Renssen R.H.(1998) "Use of statistical matching techniques in calibration estimation". *Survey Methodology* **2**4, 171–183.

Rubin D.B. (1986) "Statistical matching using file concatenation with adjusted weights and multiple imputations". *Journal of Business and Economic Statistics*, **4**, 87–94.

Särndal C.E., Swensson B., Wretman J. (1992) *Model Assisted Survey Sampling.* Springer-Verlag, New York.

Särndal C.E., Lundström S. (2005) *Estimation in Surveys with Nonresponse.* Wiley, New York.

Scanu M. (2008) "The practical aspects to be considered for statistical matching". in: *Report of WP2: Recommendations on the use of methodologies for the integration of surveys and administrative data*, "ESSnet Statistical Methodology Project on Integration of Survey and Administrative Data", 34–35. http://cenex-isad.istat.it/

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

Wu C. (2004) "Combining information from multiple surveys through the empirical likelihood method". *The Canadian Journal of Statistics*, **32**, 15–26.