`micromatch` package:

Ines Garmendia

2014-10-14

About this document

This is the main vignette for `micromatch` package. This package provides a set of utilities to ease the task of statistically matching independent microdata files, with a focus to official statistics.

The main methods in `micromatch` are described in two books (see [1] and [2]), and are a result of two Eurostat projects in Data Integration and Statistical Matching (see [3] and [4]). `micromatch` heavily relies on `StatMatch`, an R package that implements the main methods in the mentioned sources.

This document has two main parts. In the first chapter the reader will find an overview of the main concepts in statistical matching methodology. The second chapter deals with the use of `micromatch` in practice.

`micromatch` package also provides vignettes for specific examples with real data. These additional documents can be found in the package documentation.

Fundamentals of Statistical Matching

> Statistical matching provides a methodology to explore ways for producing combined analyses or indicators for independent surveys referred to the same population of interest, from data with to distinct observations and stored in separate files, but sharing a common block of information

Statistical matching (also known as data fusion, data merging or synthetic matching) is set of techniques for providing joint information on variables or indicators collected through multiple sources, usually, surveys drawn from the same population of interest. The potential benefits of this approach lie in the possibility to enhance the complementary use and analytical potential of existing data sources. ([5] A. Leulescu & M. Agafitei, 2013).

Statistical matching has been widely used in market research, to link consumer behavior data and media consumption data.

In official statistics, it can be used to link different aspects that are usually studied separately for the same target population (i.e. the inhabitants in a country or a particular geographic area). A unique questionnaire covering all aspects such as population health, income, consumption, labour market, social capital… is seldom conceived: such a questionnaire would be too long, leading to a higher response burden, and to poor quality.

A separate survey is usually conducted to study each specific aspect of the population, the drawback being that the responses will eventually lie in separate files. Statistical matching tries to overcome this limitation, by making use of shared information in order to infer some type of "new" knowledge about aspects measured through independent surveys.

The starting point (the input)

> The basic assumption is that the number of individuals or units in both samples (i.e., the overlap) is negligible. In fact, the fundamental difference with respect to other methods such as "record linkage" is that in the latter, we have identical units and we wish to find a correspondence between them to link the files. In statistical matching, we "know" the units are different, but we "wish" to find similar ones.

Consider two independent surveys conducted on the same population of interest, each of which produces measures regarding a specific field (e.g. living styles and consumer behavior).

The surveys share a block of variables (sociodemographic variables such as the age, sex, or social status). When putting observations from distinct sources together, a particular missing data pattern emerges due to the non-observed values (i.e. answers we don't have in one survey just because they correspond to the other, and viceversa), see Fig 1.

Fig 1. The starting point: a block of common variables (Z) and two block of specific, non-jointly-observed, variables (X and Y)

The aim is to obtain integrated analyses or results relating the non-jointly-observed variables (blocks X and Y in the figure), and to achieve this we need to make use of the common information between the files (block Z) in some efficient and reliable way.

The results (the output)

After matching will typically obtain one of these results:

- a synthetic file containing full information on the variables and all units from both sources. This enhanced dataset can be used later to make combined statistical analyses.
- particular estimates regarding variables living in separate files. The user might wish to estimate a contingency table or correlation coefficient, or any parameter of interest regarding variables in separate files.

The former is named the **micro** approach. The latter is the **macro** approach.

The matching process

Regardless of the matching method itself —that is, the computational method by means of which we will produce a synthetic file (in the micro case) or estimations for certain parameters of the joint distributions (in the macro case)—, the matching task involves a series of pre-processing steps that have to be tackled in practice:

1. The choice of target variables (X and Y), i.e. the variables observed separately in distinct surveys.
2. Identification of the variables shared by the sources, and the study of their degree of coherence taking into account not only the wording of questions (which can be different, leading to non-agreeable measures), but also the marginal distributions observed separately in the data files. Variables that fail to show a minimum degree of coherence must be discarded. This step can be time-consuming but can also the key for a successful matching.
3. Possibly, discarding further variables that are not predictive (i.e. are not related to) for the target variables, o are redundant with others. This can be of a relevant step, depending on the computational method.
4. The choice of a matching framework (parametric, non-parametric, mixed) in a specific setting (micro or macro), and applying the corresponding matching/imputation/estimation algorithm. This algorithm will make use of the chosen subset of shared variables in steps 2 and 3 (namely, the *common matching variables*) to relate target variables fixed in step 1 (the *specific variables*).
5. A thorough validation of results.

Using micromatch

```
* The idea is that the user should start defining particular attributes of the
data files related to the statistical matching context, i.e. common and specific
variables, weights, and so on. In this way, each step in the matching process
has its implementation (or definition) by means of functions or methods that
"act" on these special objects.
```

This chapter deals with using `micromatch` in practice, and shows how functions in this package may be used to tackle a specific matching task with real data.

Well-known `R` packages such as `StatMatch` or `mice` provide sophisticated algorithms to solve different statistical matching tasks. `micromatch` does not offer new algorithms for matching; rather, it provides a "context" where the matching task is made easier, independently of the chosen methodology. In this way, alternative methodologies available through independent packages are integrated in a common "matching pipeline".

From the programming point of view, `micromatch` makes use of S4 classes to create special objects which are adapted to the statistical matching task.

A simple example

To illustrate the use of `micromatch` we will be using data frames data `samp.A` and `samp.B` included in `StatMatch` package. These examples provide some artificial data simulating typical variables present in the European Union Statistics on Income and Living Conditions Survey (EU-SILC).

```
library(StatMatch)
data(samp.A) #loads data into workspace
data(samp.B) #loads data to workspace
str(samp.A)
```

```
## 'data.frame':    3009 obs. of  13 variables:
##  $ HH.P.id : chr  "10149.01" "17154.02" "5628.01" "15319.01" ...
##  $ area5   : Factor w/ 5 levels "NE","NO","C",..: 1 1 2 4 4 3 4 5 4 3 ...
##  $ urb     : Factor w/ 3 levels "1","2","3": 1 1 2 1 2 2 2 2 2 2 ...
##  $ hsize   : int  1 2 1 2 5 2 4 3 4 4 ...
##  $ hsize5  : Factor w/ 5 levels "1","2","3","4",..: 1 2 1 2 5 2 4 3 4 4 ...
##  $ age     : num  85 78 48 78 17 28 26 51 60 21 ...
##  $ c.age   : Factor w/ 5 levels "[16,34]","(34,44]",..: 5 5 3 5 1 1 1 3 4 1 ...
##  $ sex     : Factor w/ 2 levels "1","2": 2 1 1 1 1 2 2 2 2 2 ...
##  $ marital : Factor w/ 3 levels "1","2","3": 3 2 3 2 1 2 1 2 2 1 ...
##  $ edu7    : Factor w/ 7 levels "0","1","2","3",..: 4 4 4 2 2 6 6 3 2 4 ...
##  $ n.income: num  1677 13520 20000 12428 0 ...
##  $ c.neti  : Factor w/ 7 levels "(-Inf,0]","(0,10]",..: 2 3 4 3 1 1 2 3 1 1 ...
##  $ ww      : num  3592 415 2735 1240 5363 ...
```

```
str(samp.B)
```

```
## 'data.frame':    6686 obs. of  12 variables:
##  $ HH.P.id: chr  "5.01" "5.02" "24.01" "24.02" ...
##  $ area5  : Factor w/ 5 levels "NE","NO","C",..: 5 5 3 3 1 1 2 2 2 2 ...
##  $ urb    : Factor w/ 3 levels "1","2","3": 3 3 2 2 1 1 2 2 2 2 ...
##  $ hsize  : int  2 2 2 2 2 2 3 3 3 3 ...
## $ hsize5 : Factor w/ 5 levels "1","2","3","4",..: 2 2 2 2 2 2 3 3 3 3 ...
## $ age    : num  45 18 76 74 47 46 53 55 21 53 ...
## $ c.age  : Factor w/ 5 levels "[16,34]","(34,44]",..: 3 1 5 5 3 3 3 4 1 3 ...
## $ sex    : Factor w/ 2 levels "1","2": 2 2 1 2 1 2 2 1 2 1 ...
## $ marital: Factor w/ 3 levels "1","2","3": 3 1 2 2 1 1 2 2 1 2 ...
## $ edu7   : Factor w/ 7 levels "0","1","2","3",..: 4 3 2 3 3 6 4 4 4 3 ...
## $ labour5: Factor w/ 5 levels "1","2","3","4",..: 3 5 4 4 1 5 5 1 5 1 ...
## $ ww     : num  179 179 330 330 1116 ...
```

The independent sources `samp.A` and `samp.B`, separately contain:

- a shared block of variables:
  - `HH.P.id`: unit identifier
  - `area5` and `urb`: geographic variables
  - `hsize` and `hsize5`: family size (numeric and categorized)
  - `age` and `c.age`: age (numeric and categorized)
  - `sex`: gender
  - `marital`: marital status
  - `edu7`: education level
- one specific variable in each of the files:
  - in file `samp.A`: `n.income` and `c.neti`, net personal income (numeric and categorized, thousand of euros)
  - in file `samp.B`: `labour5`, the person's self-defined economic status.
- a weight variable, `ww`, with the same name in both files

For more information on these data files please refer to `StatMatch` package documentation.

Now we will illustrate how the matching task can be tackled with `micromatch`, step by step.

Step 1: The specific (target) variables are the income and the labour status, and it is advisable to store their name in the `R` session.

For this example we will use the categorical version of variable income, `c.neti`:

```
varesp_A <- "c.neti" # specific variable in file samp.A
varesp_B <- "labour5" # specific variable in file samp.B
```

Step 2: The shared variables are the remaining variables (excluding the identifier, `HH.P.id`, and the weight variable, `ww`). For this example we will use the categorical versions of the variables and one geographic area: `urb`.

```
varshared <- c("urb", "c.age", "hsize5", "sex", "marital", "edu7") # shared variables
```

There is also a weight variable, with the same name in both files. Note that naming the same variables equally is in general a good practice.

```r
weights <- "ww" # weight variable (same name in samp.A and samp.B)
```

Now that we have all the information, the purpose of matching can be made concrete:

```
* We want to relate variables `c.neti` and `labour5` by applying some matching
method that will use a subset of `varshared` variables to produce a synthetic,
complete file. Specifically, we will fill `samp.A` -the receptor file-, by adding
variable `labour5` from `samp.B` -the donor file-.
```

Important Note

In general, the file with less observations will be used as receptor. Otherwise, donor observations would have to be used many times to “fill”" the bigger file.

In `micromatch`, we have a way to assign these roles to the original data frames thanks to the `receptor` and `donor` constructor functions.

We may also want to fill both files. In this case we will not assign any specific role to the files, and we may say they have a `symmetric` role.

In either case, the first thing in `micromatch` will be to create `receptor` and `donor` pairs (or two `symmetric` objects, not shown here):

```r
library(micromatch)
# create the receptor object
rec <- receptor(data = samp.A, matchvars = varshared, specvars = varesp_A, weights=weights)
#
# create the donor object
don <- donor(data = samp.B, matchvars = varshared, specvars = varesp_B, weights=weights)
```

Parameter (slot) values can be checked by using `str` function:

```r
str(rec)
```

```
## Formal class 'filetomatch' [package "micromatch"] with 6 slots
##   ..@ role      : chr "receptor"
##   ..@ data      :'data.frame':   3009 obs. of  13 variables:
##   .. ..$ HH.P.id : chr [1:3009] "10149.01" "17154.02" "5628.01" "15319.01" ...
##   .. ..$ area5   : Factor w/ 5 levels "NE","NO","C",..: 1 1 2 4 4 3 4 5 4 3 ...
##   .. ..$ urb     : Factor w/ 3 levels "1","2","3": 1 1 2 1 2 2 2 2 2 2 ...
##   .. ..$ hsize   : int [1:3009] 1 2 1 2 5 2 4 3 4 4 ...
##   .. ..$ hsize5  : Factor w/ 5 levels "1","2","3","4",..: 1 2 1 2 5 2 4 3 4 4 ...
##   .. ..$ age     : num [1:3009] 85 78 48 78 17 28 26 51 60 21 ...
##   .. ..$ c.age   : Factor w/ 5 levels "[16,34]","(34,44]",..: 5 5 3 5 1 1 1 3 4 1 ...
##   .. ..$ sex     : Factor w/ 2 levels "1","2": 2 1 1 1 1 2 2 2 2 2 ...
##   .. ..$ marital : Factor w/ 3 levels "1","2","3": 3 2 3 2 1 2 1 2 2 1 ...
##   .. ..$ edu7    : Factor w/ 7 levels "0","1","2","3",..: 4 4 4 2 2 6 6 3 2 4 ...
##   .. ..$ n.income: num [1:3009] 1677 13520 20000 12428 0 ...
##   .. ..$ c.neti  : Factor w/ 7 levels "(-Inf,0]","(0,10]",..: 2 3 4 3 1 1 2 3 1 1 ...
##   .. ..$ ww      : num [1:3009] 3592 415 2735 1240 5363 ...
##   ..@ matchvars : chr [1:6] "urb" "c.age" "hsize5" "sex" ...
##   ..@ specvars  : chr "c.neti"
##   ..@ stratavars: NULL
##   ..@ weights   : chr "ww"
```

```r
str(don)
```

```
## Formal class 'filetomatch' [package "micromatch"] with 6 slots
##   ..@ role      : chr "donor"
##   ..@ data      :'data.frame':   6686 obs. of  12 variables:
##   .. ..$ HH.P.id: chr [1:6686] "5.01" "5.02" "24.01" "24.02" ...
##   .. ..$ area5  : Factor w/ 5 levels "NE","NO","C",..: 5 5 3 3 1 1 2 2 2 2 ...
##   .. ..$ urb    : Factor w/ 3 levels "1","2","3": 3 3 2 2 1 1 2 2 2 2 ...
##   .. ..$ hsize  : int [1:6686] 2 2 2 2 2 2 3 3 3 3 ...
##   .. ..$ hsize5 : Factor w/ 5 levels "1","2","3","4",..: 2 2 2 2 2 2 3 3 3 3 ...
##   .. ..$ age    : num [1:6686] 45 18 76 74 47 46 53 55 21 53 ...
##   .. ..$ c.age  : Factor w/ 5 levels "[16,34]","(34,44]",..: 3 1 5 5 3 3 3 4 1 3 ...
##   .. ..$ sex    : Factor w/ 2 levels "1","2": 2 2 1 2 1 2 2 1 2 1 ...
##   .. ..$ marital: Factor w/ 3 levels "1","2","3": 3 1 2 2 1 1 2 2 1 2 ...
##   .. ..$ edu7   : Factor w/ 7 levels "0","1","2","3",..: 4 3 2 3 3 6 4 4 4 3 ...
##   .. ..$ labour5: Factor w/ 5 levels "1","2","3","4",..: 3 5 4 4 1 5 5 1 5 1 ...
##   .. ..$ ww     : num [1:6686] 179 179 330 330 1116 ...
##   ..@ matchvars : chr [1:6] "urb" "c.age" "hsize5" "sex" ...
##   ..@ specvars  : chr "labour5"
##   ..@ stratavars: NULL
##   ..@ weights   : chr "ww"
```

Step 3-1 (assess coherence) First we must inspect the concordance of marginal distributions of the shared variables. In `micromatch` three kind of tools are implemented: frequency tables, plots and empirical measures (as computed by `comp.prop` function in `StatMatch`).

Because we have previously stored information about each type of variable in `receptor` and `donor` objects, all we need is to choose some options in the method `compare_matchvars`:

- `type`: equal to `table`, `plot` or `measures`;
- `cell_values`: `abs` (absolute numbers) `rel` (relative, i.e. percents) for type `table` or `plot`
- `weights`: `TRUE` or `FALSE`;
- `strata`: `TRUE` or `FALSE`: to be used when we want to study distributions separately for specific groups in the population (male and female, etc)

```r
# tables
compare_matchvars(x = rec, y = don, type = "table", cell_values = 'abs', weights = TRUE)
```

```
## $`Table for data:  slot(x, "data")`
## x_vector
##       1       2       3
## 2192673 2060570  841709
##
## $`Table for data:  slot(y, "data")`
## x_vector
##       1       2       3
## 2264015 2080967  812600
##
## $`Table for data:  slot(x, "data")`
## x_vector
##  [16,34]  (34,44]  (44,54]  (54,64] (64,104]
##  1221890   987127   936503   727844  1221589
```

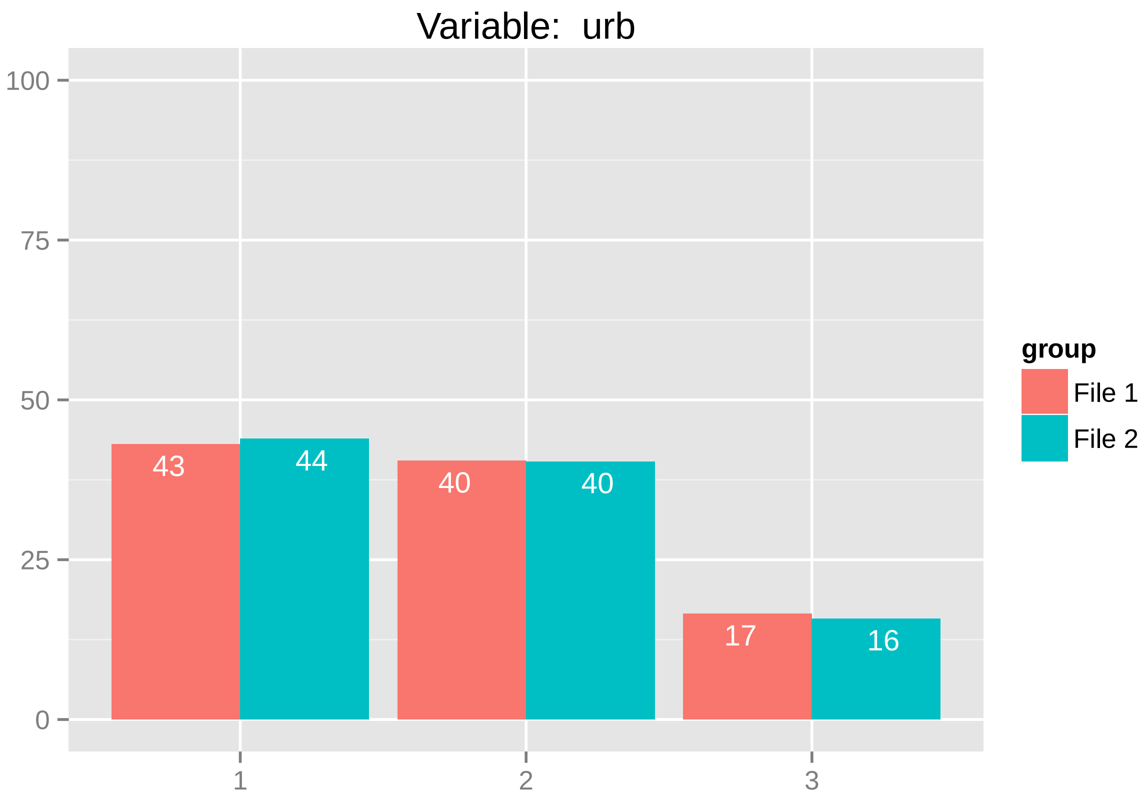
```
##
## $`Table for data:  slot(y, "data")`
## x_vector
##  [16,34]  (34,44]  (44,54]  (54,64] (64,104]
##  1210076  1039953   890703   759901  1256950
##
## $`Table for data:  slot(x, "data")`
## x_vector
##       1       2       3       4     >=5
##  889539 2093652 1194842  726198  190722
##
## $`Table for data:  slot(y, "data")`
## x_vector
##       1       2       3       4     >=5
##  854453 2213844 1182355  740957  165973
##
## $`Table for data:  slot(x, "data")`
## x_vector
##       1       2
## 2454480 2640472
##
## $`Table for data:  slot(y, "data")`
## x_vector
##       1       2
## 2512704 2644878
##
## $`Table for data:  slot(x, "data")`
## x_vector
##       1       2       3
## 1525656 2838298  730998
##
## $`Table for data:  slot(y, "data")`
## x_vector
##       1       2       3
## 1511497 2858124  787961
##
## $`Table for data:  slot(x, "data")`
## x_vector
##       0       1       2       3       4       5       6
##  135596  943911 1586638 1709141  143383  558240   18042
##
## $`Table for data:  slot(y, "data")`
## x_vector
##       0       1       2       3       4       5       6
##  149580  997272 1604171 1687398  141107  564486   13568

## $urb
## NULL
##
## $c.age
## NULL
##
## $hsize5
## NULL
```

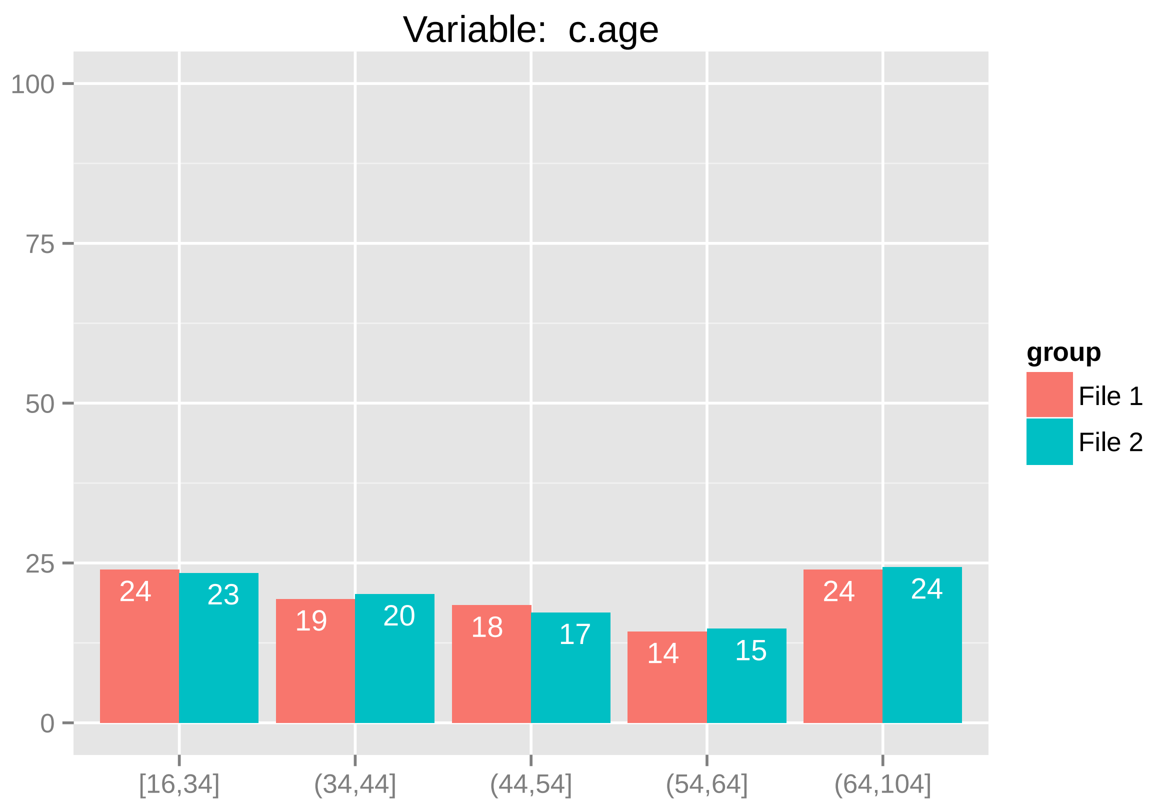
```
##
## $sex
## NULL
##
## $marital
## NULL
##
## $edu7
## NULL
```

```
# plots
compare_matchvars(x = rec, y = don, type = "plot", cell_values = 'rel', weights = TRUE)
```

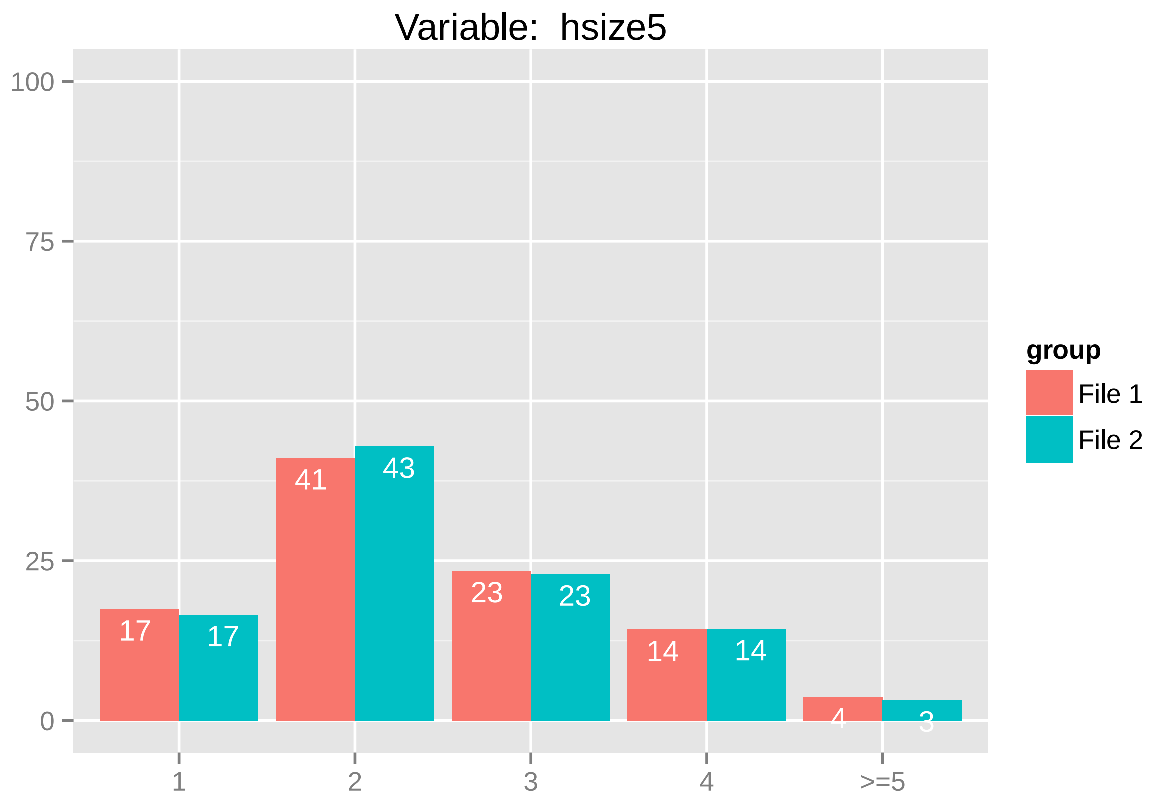
```
## ymax not defined: adjusting position using y instead
```



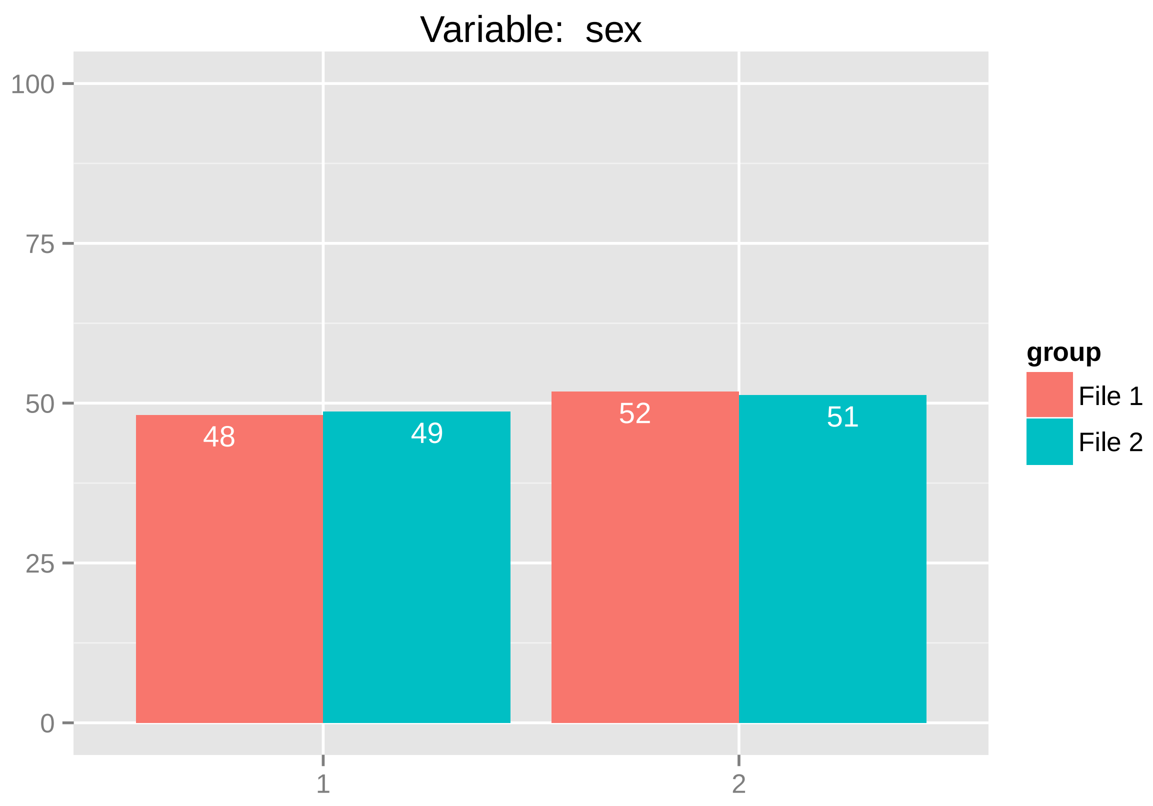
```
## ymax not defined: adjusting position using y instead
```



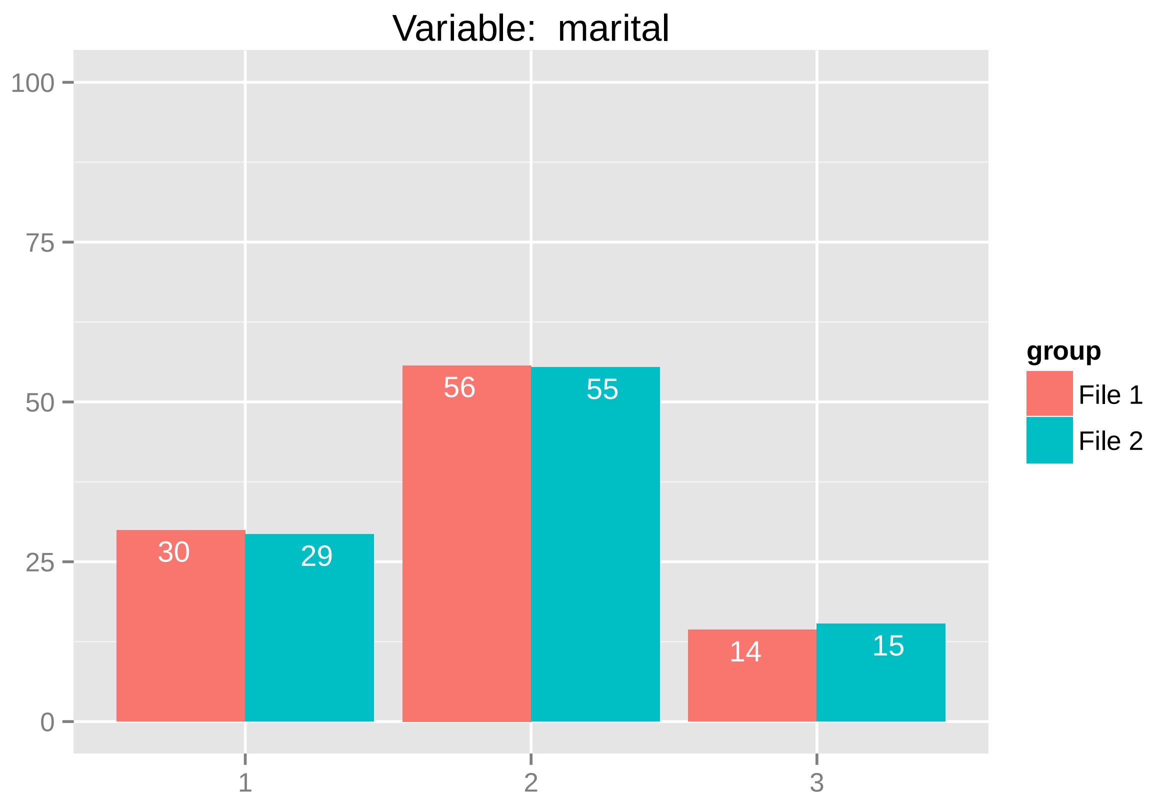
```
## ymax not defined: adjusting position using y instead
```



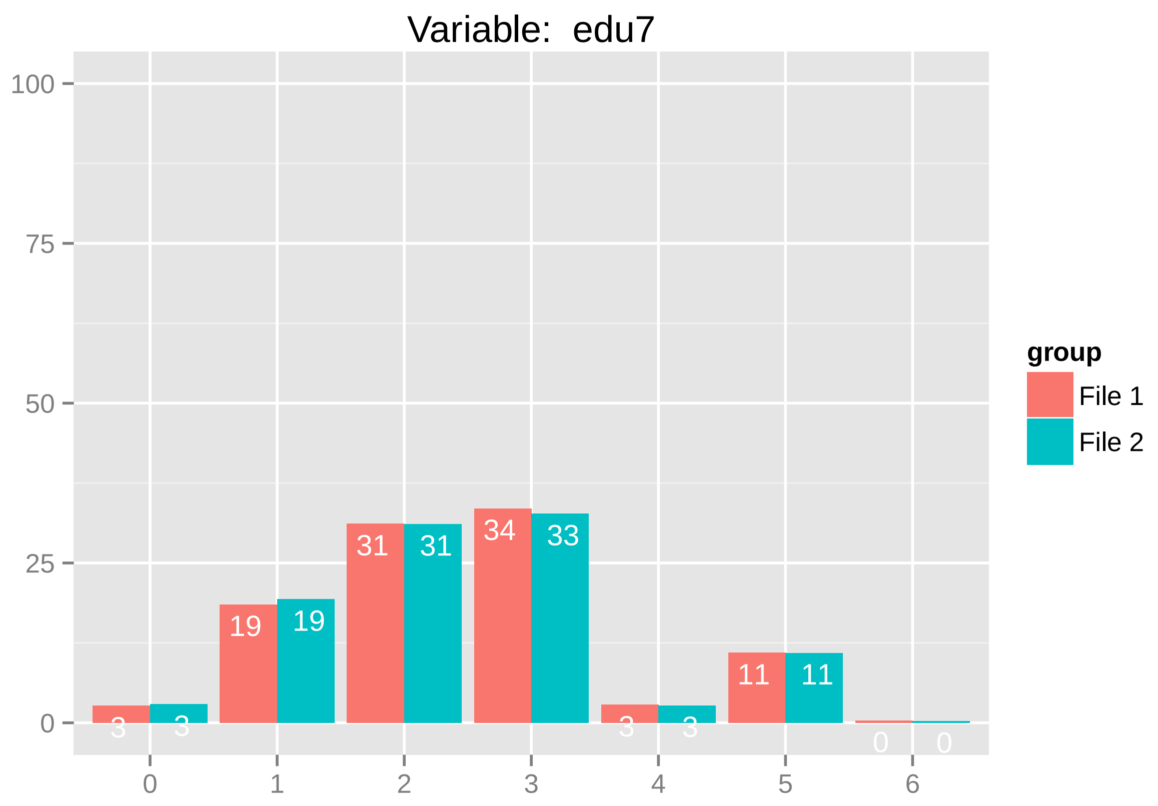
```
## ymax not defined: adjusting position using y instead
```



```
## ymax not defined: adjusting position using y instead
```



```
## ymax not defined: adjusting position using y instead
```



```
## $urb
## NULL
##
## $c.age
## NULL
##
## $hsize5
## NULL
##
## $sex
## NULL
##
## $marital
## NULL
##
## $edu7
## NULL
```

```
# disimilarity measures
compare_matchvars(x = rec, y = don, type = "measures", weights = TRUE)
```

```
## [1] "Measures for variable: urb"
## [1] "Measures for variable: c.age"
## [1] "Measures for variable: hsize5"
## [1] "Measures for variable: sex"
## [1] "Measures for variable: marital"
## [1] "Measures for variable: edu7"
##              urb   c.age  hsize5      sex  marital    edu7
## tvd     0.008606 0.01632 0.01944 0.005439 0.009302 0.01048
## overlap 0.991394 0.98368 0.98056 0.994561 0.990698 0.98952
## Bhatt   0.999933 0.99983 0.99973 0.999985 0.999908 0.99987
## Hell    0.008181 0.01287 0.01654 0.003848 0.009599 0.01147
```

Overall, for the these marginal distributions, the previous results indicate that the shared variables are highly concordant between the data frames.

Note that four types of empirical measures are used:

- Dissimilarity index or total variation distance, `D`
- Overlap between two distributions, `O`
- Bhattacharyya coefficient, `B`
- Hellinger's distance, `d_H`

For more information on these measures please refer to `StatMatch` or Agresti's book ([6]).

In the example, Hellinger's distance (Hell) is below 0.05 in all cases (an usual rule of thumb in statistical matching, see reference [5]).

Step 3-2 (assess predictive value) Now we should assess the predictive value of the common variables with respect to the specific ones, in order to discard unnecessary information (i.e. variables that are not predictive).

In `micromatch`, we can use `predictvalue` which relies on `StatMatch` function `pw.assoc`. This function returns four well-known statistical association measures for all the combinations of variables, based on Chi-Square and others. (Note that currently `predictvalue` only accepts categorical variables)

- Cramer's `V`
- Goodman-Kruskal `lambda`
- Goodman-Kruskal `tau`
- Theil's uncertainty coefficient `U`

For more information on these measures please refer to `StatMatch` or Agresti's book ([6]).

```r
predictvalue(x = rec) # predictive value in file samp.A
```

```
## Warning: Chi-squared approximation may be incorrect
```

```
## [[1]]
## [[1]]$V
##     c.neti.urb   c.neti.c.age  c.neti.hsize5     c.neti.sex c.neti.marital
##        0.06198        0.22634        0.12084        0.35188        0.18533
##    c.neti.edu7
##        0.16891
##
## [[1]]$lambda
##     c.neti.urb   c.neti.c.age  c.neti.hsize5     c.neti.sex c.neti.marital
##        0.00000        0.08683        0.02994        0.04448        0.03636
##    c.neti.edu7
##        0.04534
##
## [[1]]$tau
##     c.neti.urb   c.neti.c.age  c.neti.hsize5     c.neti.sex c.neti.marital
##       0.001131       0.040176       0.011965       0.022123       0.013850
##    c.neti.edu7
##       0.027589
##
## [[1]]$U
##     c.neti.urb   c.neti.c.age  c.neti.hsize5     c.neti.sex c.neti.marital
##       0.002091       0.053322       0.017561       0.034456       0.019466
##    c.neti.edu7
##       0.041839
```

```
predictvalue(x = don) # predictive value in file samp.B
```

```
## Warning: Chi-squared approximation may be incorrect
```

```
## [[1]]
## [[1]]$V
##     labour5.urb   labour5.c.age  labour5.hsize5     labour5.sex
##         0.03222         0.39412         0.11126         0.32181
## labour5.marital    labour5.edu7
##         0.23630         0.23968
##
## [[1]]$lambda
##     labour5.urb   labour5.c.age  labour5.hsize5     labour5.sex
##         0.00000         0.29530         0.02598         0.08789
## labour5.marital    labour5.edu7
##         0.04587         0.15386
##
## [[1]]$tau
##     labour5.urb   labour5.c.age  labour5.hsize5     labour5.sex
##       0.0004783       0.1988406       0.0142621       0.0329951
## labour5.marital    labour5.edu7
##       0.0299137       0.0776071
##
## [[1]]$U
##     labour5.urb   labour5.c.age  labour5.hsize5     labour5.sex
##       0.0007065       0.2485754       0.0166305       0.0371158
## labour5.marital    labour5.edu7
##       0.0420075       0.0803915
```

A simple, temptative choice would be to keep varibles `c.age`, `sex` and `edu7`. Also, it can be a good idea to introduce `sex` as a group or strata variable.

Note

This variable selection is also backed by the reduction of uncertainty approach illustrated in the `StatMatch` package vignette. This functionality has not been implemented yet in `micromatch`.

We now proceed to update the information in the `receptor` and `donor` objects by using `update` function in `micromatch`. This function allows to re-define the objects by just updating the needed information (and keeping the rest of values unchanged).

Note that the new objects must be stored in the session:

```r
# update variables for file A (receptor)
rec1 <- update(x = rec, matchvars = c("c.age", "edu7"), stratavars = "sex")
#
# update variables for file B (donor)
don1 <- update(x = don, matchvars = c("c.age", "edu7"), stratavars = "sex")
```

Step 4: In this example distance hot-deck imputation will be used to fill the non-observed values (variable `labour5` from `samp.B`) in file `samp.A`.

In `micromatch` we can use the `match.hotdek` function, which in turn calls to `NND.hotdeck` function in `StatMatch`. This function finds the closest donor record in `donor` for each record in `receptor`, based on the chosen matching variables.

In this case, `c.age` and `edu7` will be used to find similar donors, these being the variables stored as `matchvars`. We also need to indicate that the strata variable in `stratavars`, i.e. `sex` should be used as such. That is, we want the search to be made within levels of `sex`, i.e., separately for male and female:

```r
# hot-deck distance matching
result <- match.hotdeck(x = rec1, y = don1, strata = TRUE)
```

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

This functions inherits other options available in the original function (`NND.hotdeck` in `StatMatch`). The most important are `dist.fun` and `constr.alg`:

- dist.fun: Choice of distance function. Available options are “Manhattan” (aka “City block”; default), “Euclidean”, “Mahalanobis”,“exact” or “exact matching”, “Gower”, “minimax” or one of the distance functions available in `proxy` package. For more information check `?NND.hotdeck`
- constr.alg: `TRUE` or `FALSE`. Indicates if the algorithm should be constrined, i.e. donor records should be used only once to fill the receptor records.

The procedure has two main steps. First (receptor, donor) pairs are formed which are similar in terms of `matchvars`. Second, value observed in the donor record to is picked to fill the receptor record in each pair. In this way, the `receptor` file is ‘completed’. The function returns an object of type `fusedfile`, in which `receptor` data are stored with additional (imputed) columns (in the example, an unique column, `labour5`).

The completed data can be re-used for further computations by extracting and storing the data frame in the session, as follows. (In the example our case we store the new data contains the additional column `labour5` and is stored with the name `A.imputed`):

```r
# Extract the new, 'complete' data and store it under the name 'A.imputed'
samp.A.imp <- slot(result, "data")
#
# First 6 records.
# The last column contains the imputed values for variable 'labour5'.
head(samp.A.imp)
```

```
##        HH.P.id area5 urb hsize hsize5 age    c.age sex marital edu7
## 35973 17154.02    NE   1     2      2  78 (64,104]   1       2    3
## 11774  5628.01    NO   2     1      1  48  (44,54]   1       3    3
## 32127 15319.01     S   1     2      2  78 (64,104]   1       2    1
## 6301   2973.05     S   2     5    >=5  17  [16,34]   1       1    1
## 27740 13206.01    NO   1     3      3  61  (54,64]   1       2    3
## 21483 10198.01    NE   2     1      1  54  (44,54]   1       2    3
##       n.income   c.neti     ww labour5
## 35973    13520  (10,15]  415.2       4
## 11774    20000  (15,20] 2735.4       1
## 32127    12428  (10,15] 1239.5       4
## 6301         0 (-Inf,0] 5362.8       1
## 27740    22823  (20,25]  277.9       1
## 21483        0 (-Inf,0]  985.7       2
```

TODO. Details about the receptor and donor pairs can be obtained by means of the `details` function.

Step 5: Now we should assess the validity of the resulting data frame, in terms of its usefulness to perform statistical analyses relating not-jointly observed variables (in our case, person's net income, `c.neti` and self-defined labour status, `labour5`).

The first, reasonable validation should be to check the similarity of imputed versus observed marginal distributions. For this purpose, we can use `tabulate2cat`, `plot2cat` and `similarity2cat` functions in `micromatch`, which essentially provide the same functionality as `compare_matchvars` (see Step 3-1 above).

In our example the distribution for variable `labour5` in the original file `samp.B` whould be compared to the imputed variable in `samp.A.imp` file. In `tabulate2cat`, `plot2cat` and `similarity2cat` functions, data frames have to be introduced directly as parameter values: in the example, `samp.B` and `samp.A.imp`.

- TODO. create `validate1` method that will act on rec.fused, don pairs with options type=table, plot or measures.

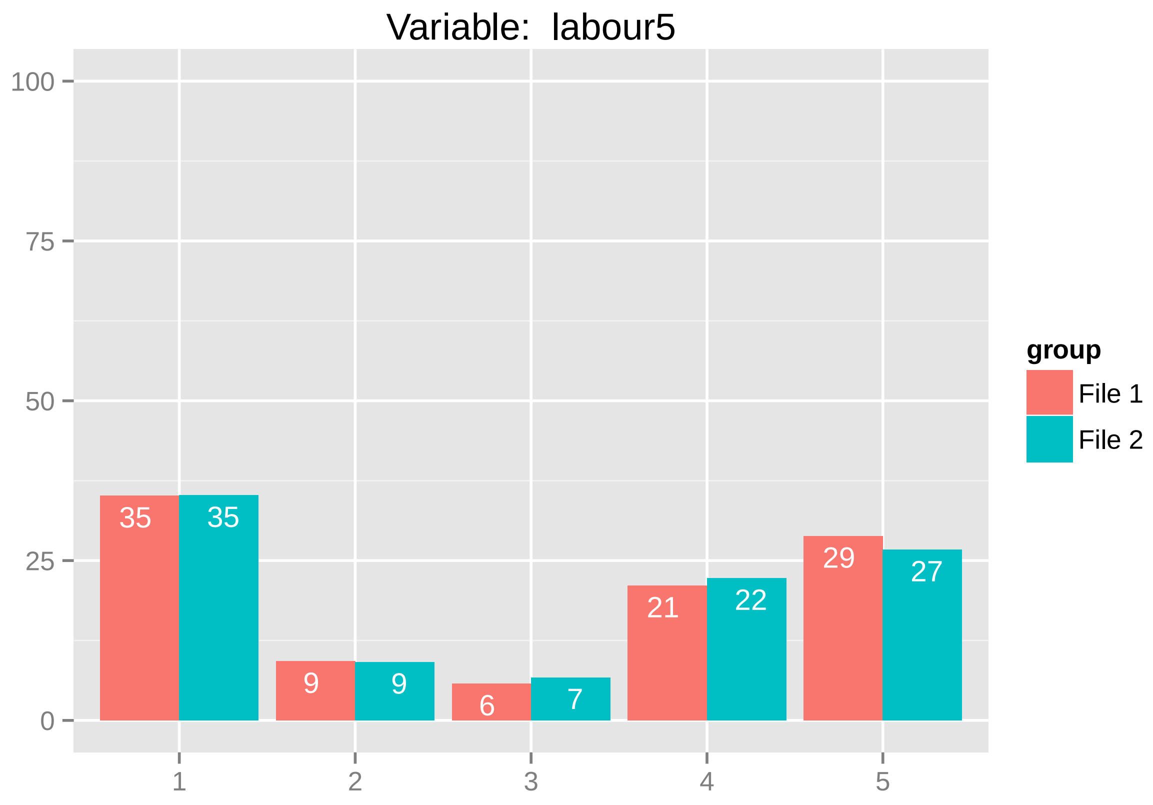
The variable to be compared is `labour5` in both files. The distributions are based on weighted data i.e. using the weights variable `ww`:

```
# Comparison of imputed vs observed distribution for variable 'labour5'
#
# store names in the session (for convenience)
var <- "labour5"
weights <- "ww"
#
# Compute raw tables
tabulate2cat(data_A = samp.B, data_B = samp.A.imp, var_A = var, var_B = var, weights_A = weights, weigh
```

```
## $`Table for data:  samp.B`
## x_vector
##       1       2       3       4       5
## 0.35125 0.09243 0.05730 0.21069 0.28833
##
## $`Table for data:  samp.A.imp`
## x_vector
##       1       2       3       4       5
## 0.35205 0.09135 0.06722 0.22239 0.26700
```

```
#
# Plots with percents
plot2cat(data_A = samp.B, data_B = samp.A.imp, var_A = var, var_B = var, weights_A = weights, weights_B
```

```
## ymax not defined: adjusting position using y instead
```



```
#
# Empirical measures
similarity2cat(data_A = samp.B, data_B = samp.A.imp, var_A = var, var_B = var, weights_A = weights, wei
```

```
## [1] "Measures for variable: labour5"
```

```
##     tvd overlap   Bhatt    Hell
## 0.02241 0.97759 0.99952 0.02199
```

The results are quite acceptable, but we should also compare distributions conditioned on other variables common to both files.

For example, a natural comparison would be to check distributions conditioned on `sex`, which was in fact used as strata variable. This can be done in with the same functions, by subseting over strata values, as follows:

However, in statistical matching the validation should imply a bit more effort. The reason is that most matching algorithms assume what is known as the conditional independence assumption, which amounts to saying that the common variables (Z) explain (or “mediate between”) all the (non-observed) relation between specific variables (X and Y).

Such an assumption is particularly strong and seldom holds in practice. What is worse, in abscence of complete observations —possibly in the form of a third independent file `C`, that may contain observatins for all variables, maybe from a previous wave of the same surveys, and not too distant in time so that it refers to almost the same population of interest—, we will lack of the necessary information to check how far we are from the ideal situation.

One recommended approach is to perform an uncertainty analysis. In the case of categorical variables, the ultimate aim will is to estimate contingency tables between variables observed in separate files, and Frèchet bounds can be computed to compute a range of possible vaues, i.e. numeric results that are coherent with the independently observed marginal distributions (i.e. `X` versus `Z` and `Y` versus `Z`). For more information Frèchet bounds please refer to [1] or to the `StatMatch` package vignette.

- TODO. Here, an illustration of uncertainty in cell values in contingency table between net income and labour status

```
# Uncertainty if we use all shared variables
slot(rec, "matchvars") #"urb"     "c.age"   "hsize5"  "sex"     "marital" "edu7"
```

```
## [1] "urb"     "c.age"   "hsize5"  "sex"     "marital" "edu7"
```

```
xx <- xtabs(~urb + c.age + hsize5 + sex + marital + edu7, data = samp.B)
xy <- xtabs(~urb + c.age + hsize5 + sex + marital + edu7 + c.neti, data = samp.A)
xz <- xtabs(~urb + c.age + hsize5 + sex + marital + edu7 + labour5, data = samp.B)
#todas las posibles combinaciones de vars comunes
#Fbwidths.by.x(tab.x = xx, tab.xy = xy, tab.xz = xz)
Frechet.bounds.cat(tab.x = xx, tab.xy = xy, tab.xz = xz, print.f = "data.frame")
```

```
## Warning: The marginal distr. of the X variables
##  in tab.xy is not equal to tab.x
## Warning: The marginal distr. of the X variables
##  in tab.xy and in tab.xz are not equal
```

```
## $bounds
##       c.neti labour5 low.u    low.cx      CIA    up.cx    up.u
## 1   (-Inf,0]       1     0 0.0101598 0.051297 0.092309 0.18744
## 2     (0,10]       1     0 0.0089131 0.042499 0.082317 0.22300
## 3    (10,15]       1     0 0.0131615 0.045643 0.083148 0.17979
## 4    (15,20]       1     0 0.0183774 0.053945 0.092110 0.16617
## 5    (20,25]       1     0 0.0142931 0.038704 0.062918 0.10203
## 6    (25,35]       1     0 0.0124055 0.036317 0.057836 0.09040
## 7  (35, Inf]       1     0 0.0045539 0.016968 0.028723 0.05118
## 8   (-Inf,0]       2     0 0.0023931 0.011601 0.030445 0.10036
## 9     (0,10]       2     0 0.0008226 0.010430 0.034115 0.10036
## 10   (10,15]       2     0 0.0015704 0.012854 0.038402 0.10036
## 11   (15,20]       2     0 0.0031658 0.017161 0.046047 0.10036
## 12   (20,25]       2     0 0.0026922 0.012398 0.036206 0.10036
## 13   (25,35]       2     0 0.0026673 0.013119 0.036431 0.09040
## 14 (35, Inf]       2     0 0.0023532 0.008588 0.023068 0.05118
## 15  (-Inf,0]       3     0 0.0035255 0.015227 0.032296 0.06087
## 16    (0,10]       3     0 0.0014957 0.009696 0.030168 0.06087
## 17   (10,15]       3     0 0.0013461 0.007923 0.026388 0.06087
## 18   (15,20]       3     0 0.0017200 0.007873 0.022957 0.06087
## 19   (20,25]       3     0 0.0004487 0.003373 0.012559 0.06087
## 20   (25,35]       3     0 0.0006730 0.004021 0.013784 0.06087
## 21 (35, Inf]       3     0 0.0000000 0.001095 0.005382 0.05118
## 22  (-Inf,0]       4     0 0.0015517 0.012621 0.023400 0.18744
## 23    (0,10]       4     0 0.0307526 0.058284 0.082794 0.22300
## 24   (10,15]       4     0 0.0287910 0.052624 0.074987 0.17979
## 25   (15,20]       4     0 0.0183876 0.036740 0.051501 0.16617
## 26   (20,25]       4     0 0.0071455 0.018299 0.027249 0.10203
## 27   (25,35]       4     0 0.0062282 0.014640 0.021543 0.09040
## 28 (35, Inf]       4     0 0.0033092 0.010201 0.016090 0.05118
## 29  (-Inf,0]       5     0 0.0401325 0.082918 0.136839 0.18744
## 30    (0,10]       5     0 0.0207716 0.069930 0.127249 0.22300
## 31   (10,15]       5     0 0.0052369 0.043109 0.087779 0.17979
## 32   (15,20]       5     0 0.0023477 0.026898 0.061573 0.16617
## 33   (20,25]       5     0 0.0026174 0.012968 0.031681 0.10203
## 34   (25,35]       5     0 0.0017449 0.009250 0.023930 0.09040
## 35 (35, Inf]       5     0 0.0004986 0.004552 0.011995 0.05118
##
## $uncertainty
##    av.u   av.cx overall
## 0.11599 0.03971 0.17286
```

Additional features

References

[1] D'Orazio, M., Di Zio, M., & Scanu, M. (2006). *Statistical matching: Theory and practice.* John Wiley & Sons.

[2] Rässler, S. (2002). *Statistical matching.* Springer.

[3] *Data Integration* ESSnet project. (http://www.cros-portal.eu/content/data-integration-finished)

[4] *ISAD* ESSnet project (http://www.cros-portal.eu/content/isad-finished)

[5] Leulescu A. & Agafitei, M. *Statistical matching: a model based approach for data integration*, Eurostat methodologies and working papers, 2013. (http://epp.eurostat.ec.europa.eu/cache/ITY_OFFPUB/KS-RA-13-020/EN/KS-RA-13-020-EN.PDF)