cem: Coarsened Exact Matching

Version 1.0.2
August 6, 2008

Contents

1 Introduction

This program is designed to improve the estimation of causal effects via a powerful method of matching that is widely applicable in observational data and exceptionally easy to understand and use (if you understand how to draw a histogram, you will understand this method). The program implements the CEM (Coarsened Exact Matching) algorithm described in

> Stefano M. Iacus, Gary King, and Giuseppe Porro, "Matching for Causal Inference Without Balance Checking", copy at `http://gking.harvard.edu/files/abs/cem-abs.shtml`.

CEM is a monotonoic imbalance bounding (MIS) matching method — which means that the maximum imbalance between the treated and control groups is chosen by the user ex ante rather than discovered through the usual laborious process of checking after the fact and repeatdly reestimating, and so that adjusting the imbalance on one variable has no effect on the maximum imbalance of any other. CEM also strictly bounds through ex ante user choice both the degree of model dependence and the average treatment effect estimation error, eliminates the need for a separate procedure to restrict data to common empirical support, meets the congruence principle, is robust to measurement error, works well with multiple imputation methods for missing data, can be completely automated, and is extremely fast computationally even with very large data sets. After preprocessing data with CEM, the analyst may then use a simple difference in means or whatever statistical model they would have applied without matching. CEM also works well for multicategory treatments, determining blocks in experimental designs, and evaluating extreme counterfactuals.

2 Software Requirements

CEM works in conjunction with the R Project for Statistical Computing, and will run on any platform where R is installed (Windows, Unix, or Mac). R is available free for download at the Comprehensive R Archive Network (CRAN) at `http://cran.r-project.org/`. CEM has been tested on the most recent version of R.

CEM may be run by installing the program directly, as indicated below, or by using the alternative interface to CEM provided by MatchIt (`http://gking.harvard.edu/matchit`, (Ho et al., Forthcoming)). Using CEM directly is faster. The MatchIt interface is easier for some applications and also works seemlessly with Zelig (`http://gking.harvard.edu/zelig`) for estimating causal effects after matching. (A Stata verison of CEM is under development.)

3 Installation

To install cem, type at the R command prompt,

```
> install.packages("cem")
```

and CEM will install itself onto your system automatically from CRAN. (You may alternatively load the beta test version as

```
> install.packages("cem",repos="http://gking.harvard.edu/cem")
```

4 Loading CEM

You need to install CEM only once, but you must load it prior to each use. Do this at the R prompt:

```
> library(cem)
```

5 Updating CEM

We recommend that you periodically update CEM at the R prompt by typing:

```
> update.packages()
> library(cem)
```

which will update all the libraries including CEM and load the new version of CEM.

6 Highlights on CEM software usage

We discuss briefly the use of `cem` without discussing statistical properties which can be found in (Iacus, King and Porro, 2008). As a driving example, we use the National Supported Work (NSW) Demonstration data, also known as the Lalonde data set (Lalonde, 1986). The program provided training to the participants for 12-18 months and helped them in finding a job. The goal of the program was to increase participants' earnings, and so 1978 earnings (`re78`) is the key outcome variable. Pre-treatment variables were measured for both participants and controls, including age (`age`), years of education (`education`), marital status (`married`), lack of a high school diploma (`nodegree`), race (`black`, `hispanic`), indicator variables for unemployment in 1974 (`u74`) and 1975 (`u75`), and real earnings in 1974 (`re74`) and 1975 (`re75`). Some of these are dichotomous (`married`, `nodegree`, `black`, `hispanic`, `u74`, `u75`), some are categorical (`age` and `education`), and the earnings variables are continuous and highly skewed, with point masses at zero.

```
> require(cem)
```

```
How to use CEM? Type vignette("cem")
```

```
> data(LL)
> tr <- which(LL$treated == 1)
> ct <- which(LL$treated == 0)
> ntr <- length(tr)
> nct <- length(ct)
```

there a 297 treated units and 425 control units in the data. A simple but biased estimator of the treatment effect is given by the simple difference in means.

```
> mean(LL$re78[tr]) - mean(LL$re78[ct])
```

```
[1] 886.3038
```

This would be the real estimate of ATT (average treatment effect on the treated) if the treated and control units where perfectly or approximatively perfectly matched on all covariates. Looking at the summary of this data for treated and control untis, we can see that only the means are approximatively balanced, i.e., for the subsample of the treated units, we have

```
> summary(LL[tr, -c(1, 9)])
```

```
      age          education         black           married      
 Min.   :17.00   Min.   : 4.00   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:20.00   1st Qu.: 9.00   1st Qu.:1.0000   1st Qu.:0.0000  
 Median :23.00   Median :11.00   Median :1.0000   Median :0.0000  
 Mean   :24.63   Mean   :10.38   Mean   :0.8013   Mean   :0.1684  
 3rd Qu.:27.00   3rd Qu.:12.00   3rd Qu.:1.0000   3rd Qu.:0.0000  
 Max.   :49.00   Max.   :16.00   Max.   :1.0000   Max.   :1.0000  
    nodegree           re74               re75          hispanic      
 Min.   :0.0000   Min.   :    0.0   Min.   :    0   Min.   :0.00000  
 1st Qu.:0.0000   1st Qu.:    0.0   1st Qu.:    0   1st Qu.:0.00000  
 Median :1.0000   Median :  858.3   Median : 1117   Median :0.00000  
 Mean   :0.7306   Mean   : 3571.0   Mean   : 3066   Mean   :0.09428  
 3rd Qu.:1.0000   3rd Qu.: 5491.5   3rd Qu.: 4310   3rd Qu.:0.00000  
 Max.   :1.0000   Max.   :37431.7   Max.   :37432   Max.   :1.00000  
      u74              u75        
 Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.:0.0000  
 Median :0.0000   Median :0.0000  
 Mean   :0.4411   Mean   :0.3737  
 3rd Qu.:1.0000   3rd Qu.:1.0000  
 Max.   :1.0000   Max.   :1.0000  
```

and for the subsample of the control units

```
> summary(LL[ct, -c(1, 9)])
```

```
      age          education         black        married      
 Min.   :17.00   Min.   : 3.00   Min.   :0.0   Min.   :0.0000  
 1st Qu.:19.00   1st Qu.: 9.00   1st Qu.:1.0   1st Qu.:0.0000  
 Median :23.00   Median :10.00   Median :1.0   Median :0.0000  
 Mean   :24.45   Mean   :10.19   Mean   :0.8   Mean   :0.1576  
 3rd Qu.:28.00   3rd Qu.:11.00   3rd Qu.:1.0   3rd Qu.:0.0000  
 Max.   :55.00   Max.   :14.00   Max.   :1.0   Max.   :1.0000  
    nodegree           re74              re75             hispanic     
 Min.   :0.0000   Min.   :    0.0   Min.   :    0.0   Min.   :0.0000  
 1st Qu.:1.0000   1st Qu.:    0.0   1st Qu.:    0.0   1st Qu.:0.0000  
 Median :1.0000   Median :  788.5   Median :  823.3   Median :0.0000  
 Mean   :0.8141   Mean   : 3672.5   Mean   : 3026.7   Mean   :0.1129  
 3rd Qu.:1.0000   3rd Qu.: 4906.6   3rd Qu.: 3649.8   3rd Qu.:0.0000  
 Max.   :1.0000   Max.   :39570.7   Max.   :36941.3   Max.   :1.0000  
      u74              u75        
 Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.:0.0000  
 Median :0.0000   Median :0.0000  
 Mean   :0.4612   Mean   :0.4188  
 3rd Qu.:1.0000   3rd Qu.:1.0000  
 Max.   :1.0000   Max.   :1.0000  
```

In (Iacus, King and Porro, 2008), it was introduced a global measure of imbalance among multidimensional distributions of treated and control units. It is based on the L_1 distance between the multidimensional histograms of the two subpopulations of treated and control units. More formally, for each of the k covariates, say X_j, a number of bins (or levels for categorical variables) ℓ_j is chosen. Denote by $f_{\ell_1 \cdots \ell_k}$ (resp. $g_{\ell_1 \cdots \ell_k}$) the frequency of treated (rep. control) units belonging to on of the cells defined by the cross-tabulation of $X_1 \times \cdots \times X_k$. The measure of global imbalance is defined as

$$\mathcal{L}_1(f, g) = \sum_{\ell_1 \cdots \ell_k} |f_{\ell_1 \cdots \ell_k} - g_{\ell_1 \cdots \ell_k}|$$

where the summation is over all cells of the multivariate histogram. An important property is that the typically numerous empty cells do not affect $\mathcal{L}_1(f, g)$. We now measure the global imbalance in the original data. To this end, we choose some bins for the numerical variables and we keep these bins all over the text to ensure comparability[1].

```
> L1breaks = list(re74 = hist(LL$re74, plot = FALSE)$breaks, re75 = hist(LL$re75,
+     plot = FALSE)$breaks, age = hist(LL$age, plot = FALSE)$breaks,
+     education = hist(LL$education, plot = FALSE)$breaks)
> L1 <- L1.meas(LL$treated, LL[, -c(1, 9)], breaks = L1breaks)
> L1
```

```
[1] 1.149392
```

[1]Of course, like in histograms drawing, the choice of bins affects the final result. The important thing is to choose one and keep it the same to allow for fair comparisons.

so the $\mathcal{L}_1$ measure equals 1.14939195880373. The $\mathcal{L}_1$ measure, can also be calculated variable by variable along with other one-dimensional measures useing the function `eval.match` as follows (we suppress variable 8 which is `re78` because this is the outcome variable in the experiment)

```
> eval.match(LL$treated, LL[, -c(1, 9)])[, -8]

              statistics          L1 min 25%       50%       75%        max
age         1.792038e-01 0.184701921   0   1   0.00000   -1.0000    -6.0000
education   1.922361e-01 0.153788869   1   0   1.00000    1.0000     2.0000
black       1.346801e-03 0.002693603   0   0   0.00000    0.0000     0.0000
married     1.070311e-02 0.021406219   0   0   0.00000    0.0000     0.0000
nodegree   -8.347792e-02 0.166955833   0  -1   0.00000    0.0000     0.0000
re74       -1.014862e+02 0.071792434   0   0  69.73096  584.9160 -2139.0195
re75        3.941545e+01 0.114874233   0   0 294.18457  660.6865   490.3945
hispanic   -1.866508e-02 0.037330164   0   0   0.00000    0.0000     0.0000
u74        -2.009903e-02 0.040198059   0   0   0.00000    0.0000     0.0000
u75        -4.508616e-02 0.090172311   0   0   0.00000    0.0000     0.0000
```

from which it is seen that the distribution of the numerical variables are quite imbalanced. The column `statistics` calculates the χ^2 or the t test statistics depending on the nature of the variable. This is only intended for comparison with other matching software, but never considered in the real analysis of data via CEM.

6.1 An example of use of `cem`

We now apply `cem`. CEM is an algorithm which applies exact matching on pre-coarsened data. The use may choose the coarsening level according to real knowledge of the data or applying one of the standard automatic methods. Some classic measures of bin size are based on the range of the data, an underlying normal distribution, or the inter-quartile range. These are, respectively, known as Sturges, $\Delta_{\text{st}} = (x_{(n)} - x_{(1)})/(\log_2 n + 1)$, Scott, $\Delta_{\text{sc}} = 3.5\sqrt{s_n^2}n^{-1/3}$ (Scott, 1992), and Freedman and Diaconis (1981) $\Delta_{\text{fd}} = 2(Q_3 - Q_1)n^{-1/3}$. More recently, Shimazaki and Shinomoto (2007) developed an approach based on Poisson sampling in time series analysis (in the attempt to recover spikes), which we find works well. Although, `cem` has an interface similar to the `hist` function in base R to specify break points, we suggest the reader to refer to `cem` man page for detailed explanation. In our example we use automatic choice, i.e. we do not specify anything to `cem`, the minimal requires is the specification of the data set to be matched and, when available, the name of the treatment variable which, in our example, is `treated`. If the treatment variable is not specified, the software produces a match in which the observations are collected in strata according to their position in the multidimensional grid defined by the coarsening. When the treatment variable is specified, then `cem` rejects all the strata in which only one group of observations is present. The treatment variable does not need to be dichotomic, i.e. `cem` works on multitreatment experiments. In the latter case, only strata with at least one observation per group (as defined by the treatment variable) are retained. In our example we have a treatment variable, hence we proceed as follows

```
> mat <- cem(treatment = "treated", data = LL[, -9])
```

Now the object `mat` contains several informations about the match. One is the summary of the matching solution

```
> mat$tab

           G0  G1
All       425 297
Matched   222 163
Unmatched 203 134
```

from which is emerges that even after coarsening, the treated and control units are not all really comparable in terms of their covariates. Let us see the summary for the matched data

```
> cem.idx <- which(mat$matched)
> L1.cem <- L1.meas(LL$treated[cem.idx], LL[cem.idx, -c(1, 9)],
+     breaks = L1breaks)
> L1.cem

[1] 0.6160946
```

and the one-dimensional statistics

```
> eval.match(LL$treated[cem.idx], LL[cem.idx, -c(1, 9)])[, -8]

             statistics         L1 min 25%      50%       75%      max
age         -0.42486044 0.25125739   0  -1  -2.0000    0.0000    1.000
education   -0.10855027 0.21804013   0   0  -1.0000    0.0000    0.000
black       -0.01771403 0.03542807   0   0   0.0000    0.0000    0.000
married     -0.01630465 0.03260930   0   0   0.0000    0.0000    0.000
nodegree     0.09022827 0.18045653   0   0   0.0000    0.0000    0.000
re74      -119.33548135 0.10418394   0   0   0.0000 -492.9500  416.416
re75       -50.01527694 0.07411706   0   0 -49.3559 -136.4500 -852.252
hispanic     0.01561377 0.03122755   0   0   0.0000    0.0000    0.000
u74          0.01619411 0.03238822   0   0   0.0000    0.0000    0.000
u75          0.02310286 0.04620571   0   0   0.0000    0.0000    0.000
```

To really take advantage of CEM properties, `cem` also produces weights to be used to evaluated imbalance measures and statistical estimates of the treatment effect. These are contained in element `w` of the `cem` output. We now see at the same statistics using weights

```
> L1.cem.w <- L1.meas(LL$treated[cem.idx], LL[cem.idx, -c(1, 9)],
+     breaks = L1breaks, weights = mat$w[cem.idx])
> L1.cem.w

[1] 0.3402863

> eval.match(LL$treated[cem.idx], LL[cem.idx, -c(1, 9)], weights = mat$w[cem.idx])[,
+     -8]

             statistics           L1 min 25%      50%       75%      max
age        1.862046e-01 1.820041e-01   0   0   0.0000   1.00000    1.000
education  1.022495e-02 2.044990e-02   0   0   0.0000   0.00000    0.000
black     -1.110223e-16 1.249001e-16   0   0   0.0000   0.00000    0.000
married    0.000000e+00 1.110223e-16   0   0   0.0000   0.00000    0.000
nodegree  -1.110223e-16 1.110223e-16   0   0   0.0000   0.00000    0.000
re74       7.197514e+00 8.005612e-02   0   0   0.0000 -70.85522  416.416
re75       1.220698e+01 1.426976e-01   0   0 234.4843 140.79126 -852.252
hispanic   0.000000e+00 1.110223e-16   0   0   0.0000   0.00000    0.000
u74        0.000000e+00 5.551115e-17   0   0   0.0000   0.00000    0.000
u75        0.000000e+00 0.000000e+00   0   0   0.0000   0.00000    0.000
```

6.2 Progressive coarsening

In case the user is not satisfied by the matching solution, it is possible to relax the `cem` solution selectively by changing the coarsening on each variable individually. Next example shows the effect on the matching solution when one variable is relaxed

```
> cem("treated", LL[, -9], cutpoints = list(age = 10))$tab

           G0  G1
All       425 297
Matched   228 161
Unmatched 197 136

> cem("treated", LL[, -9], cutpoints = list(age = 6))$tab

           G0  G1
All       425 297
Matched   261 186
Unmatched 164 111

> cem("treated", LL[, -9], cutpoints = list(age = 3))$tab
```

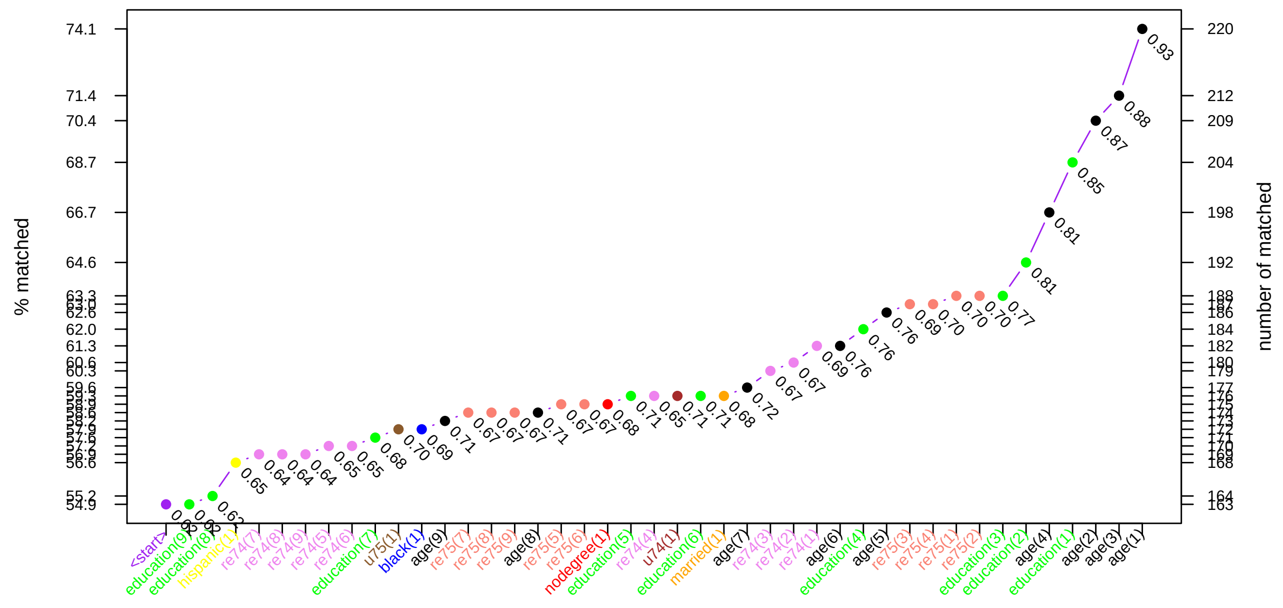

Figure 1: Example of `relax.plot`.

```
          G0  G1
All      425 297
Matched  307 209
Unmatched 118  88
```

But it is also possible to explore different solutions using the `relax.cem` function. This function, starts for the output of `cem` and relax variables one at times (`depth=1`), couple of variables (`depth=2`), triplets (`depth=3`), etc. eventually keeping unchanged some subset of the variables (`fixed`). It is also possible to specify the minimal number of breaks of each variable (the limit being 1). We start with an example

```
> tab <- relax.cem(mat, LL, depth = 1, L1.breaks = L1breaks, plot = FALSE)
```

```
Executing 42 different relaxations
...........30...40...50...........80...90...100.
```

after all possible coarsening relaxations are attempted, the function returns a list of tables. There is one table per group (i.e. treated and control). Each row of the tables contain the information about the number of treated and control units matched, the value of the $\mathcal{L}_1$ measure, and the type of relaxation made. Each table is the sorted according to the number of treated (or control) units matched. The user may want to see the output of `tab$G1` or `tab$G0` but these tables may be very long, so we provide a function `relax.plot` to plot these tables to visually get an idea of which matching solution is acceptable or, simply, which variable is more difficult to match. The output of `relax.plot(tab)` is given in Figure 1 from which it is seen that the most difficult variables to match are `age` and `education`. On the x-axis of the plot the variable and the number of equally sized bins used for the coarsening are used. On the y-axis on the right the absolute number of treated units matched is given, while the left-hand side y-axis reports the same number in percentage. The numbers below the dots in the graphs represents the $\mathcal{L}_1$ measure for that matching solution. This graph also gives a feeling of the monotonic behaviour of `cem`. When the tables produced by `relax.cem` are too large, the `relax.plot` function, allows for some reduction like printing only the best matching solutions (in the terms of number of treated units matched), removing duplicates (i.e. different coarsenings may lead to the same matching solution), or printing only solution where at least some percentage of treated units has been matched, or a combination of these. For more information refer to the man page of `relax.plot`.

6.3 An example of estimation of ATT from `cem` output

Now we pass to the estimation of the treatment effect. The package allow for an easy way to produce such estimates via the `att` function. It is as easy as follows

```
> est <- att(mat, re78 ~ treated, data = LL)
> est

             (Intercept)     treated
Estimate    4.686104e+03 550.9625644
Std. Error  3.979609e+02 611.6134147
t value     1.177529e+01   0.9008347
Pr(>|t|)    1.577548e-27   0.3682423
```

The att estimate is the coefficient of the `treated` variable, in our case 550.962564412043. The function `att` allows for the typical `formula` interface and, by default, it uses `lm` to estimate the model and it uses the weights as calculated by `cem`. In the estimation of the model, the real data are used and not the coarsened ones. The user can also specify `glm` modeling in the case of binary outcome. For more information, see the man page of the function `att`.

6.4 Working with multiply imputed data

It is not very uncommon that data comes with missing data. As an example we create a version of the Lalonde data with missing data as follows

```
> n <- dim(LL)[1]
> k <- dim(LL)[2]
> set.seed(123)
> LL1 <- LL
> idx <- sample(1:n, 0.3 * n)
> invisible(sapply(idx, function(x) LL1[x, sample(2:k, 1)] <<- NA))
```

Now LL1 contain several missing data

```
> summary(LL1)

    treated             age          education         black       
 Min.   :0.0000   Min.   :17.00   Min.   : 3.00   Min.   : 0.0000  
 1st Qu.:0.0000   1st Qu.:19.00   1st Qu.: 9.00   1st Qu.: 1.0000  
 Median :0.0000   Median :23.00   Median :10.00   Median : 1.0000  
 Mean   :0.4114   Mean   :24.49   Mean   :10.27   Mean   : 0.8037  
 3rd Qu.:1.0000   3rd Qu.:27.00   3rd Qu.:11.00   3rd Qu.: 1.0000  
 Max.   :1.0000   Max.   :54.00   Max.   :16.00   Max.   : 1.0000  
                  NA's   :18.00   NA's   :21.00   NA's   :19.0000  
    married            nodegree            re74              re75        
 Min.   : 0.0000   Min.   : 0.0000   Min.   :    0.0   Min.   :    0.0  
 1st Qu.: 0.0000   1st Qu.: 1.0000   1st Qu.:    0.0   1st Qu.:    0.0  
 Median : 0.0000   Median : 1.0000   Median :  824.4   Median :  935.3  
 Mean   : 0.1593   Mean   : 0.7778   Mean   : 3686.7   Mean   : 3056.7  
 3rd Qu.: 0.0000   3rd Qu.: 1.0000   3rd Qu.: 5272.0   3rd Qu.: 4064.4  
 Max.   : 1.0000   Max.   : 1.0000   Max.   :39570.7   Max.   :37431.7  
 NA's   :25.0000   NA's   :20.0000   NA's   :   20.0   NA's   :   16.0  
      re78          hispanic            u74               u75        
 Min.   :    0   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000  
 1st Qu.:    0   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000  
 Median : 4008   Median : 0.0000   Median : 0.0000   Median : 0.0000  
 Mean   : 5504   Mean   : 0.1053   Mean   : 0.4539   Mean   : 0.3991  
 3rd Qu.: 8782   3rd Qu.: 0.0000   3rd Qu.: 1.0000   3rd Qu.: 1.0000  
 Max.   :60308   Max.   : 1.0000   Max.   : 1.0000   Max.   : 1.0000  
 NA's   :   18   NA's   :19.0000   NA's   :17.0000   NA's   :23.0000  
```

Then we use Amelia package (Honaker, King and Blackwell, 2006) to do multiple imputation

```
> require(Amelia)
> imputed <- amelia(LL1, noms = c("black", "hispanic", "treated",
+     "married", "nodegree", "u74", "u75"))[1:5]

-- Imputation 1 --

 1  2  3  4  5

-- Imputation 2 --
```

```
 1  2  3  4  5

-- Imputation 3 --

 1  2  3  4  5  6  7  8

-- Imputation 4 --

 1  2  3  4

-- Imputation 5 --

 1  2  3  4
```

Now `imputed` contains 5 multiply imputed data of `LL1`. We pass this to the `multicem` function, we has an interface similar to `cem` but accepts a list of data frames and output a list of `cem` solutions. Each matching oslution is called `matchX` where `X` varies from 1 to the number of multiply imputed data sets.

```
> mat1 <- multicem("treated", datalist = imputed, drop = "re78")
> str(mat1, max.lev = 1)
```

```
List of 5
 $ match1:List of 20
  ..- attr(*, "class")= chr "cem.match"
 $ match2:List of 20
  ..- attr(*, "class")= chr "cem.match"
 $ match3:List of 20
  ..- attr(*, "class")= chr "cem.match"
 $ match4:List of 20
  ..- attr(*, "class")= chr "cem.match"
 $ match5:List of 20
  ..- attr(*, "class")= chr "cem.match"
 - attr(*, "class")= chr [1:2] "multicem" "list"
```

to see each matching solution one can use the usual approach

```
> mat1$match1$tab
```

```
           G0  G1
All       425 297
Matched   202 142
Unmatched 223 155
```

```
> mat1$match2$tab
```

```
           G0  G1
All       425 297
Matched   202 143
Unmatched 223 154

> mat1$match3$tab

           G0  G1
All       425 297
Matched   203 146
Unmatched 222 151

> mat1$match4$tab

           G0  G1
All       425 297
Matched   217 158
Unmatched 208 139

> mat1$match5$tab

           G0  G1
All       425 297
Matched   199 151
Unmatched 226 146
```

In the above example, `multicem` has no clue about which rows where originally missing, so each matching solution is different. Mixing together the output of each single match may be risky, thus `multicem` allows to specify also the original data set with missing data. In this case, `multicem` produces the same solutions assigning each multiply imputed observation in the strata where it fall most frequently. This is the correct way to use `multicem` and allow for correct combination of att estimate in each model. We give an example now

```
> mat2 <- multicem("treated", datalist = imputed, drop = "re78",
+     data = LL1)
> mat2$match1$tab

           G0  G1
All       425 297
Matched   203 147
Unmatched 222 150

> mat2$match2$tab
```

```
           G0  G1
All       425 297
Matched   203 147
Unmatched 222 150

> mat2$match3$tab

           G0  G1
All       425 297
Matched   203 147
Unmatched 222 150

> mat2$match4$tab

           G0  G1
All       425 297
Matched   203 147
Unmatched 222 150

> mat2$match5$tab

           G0  G1
All       425 297
Matched   203 147
Unmatched 222 150
```

Now we can estimate the att safely with the usual Rubin's formulas, i.e. the quantity of interest (qoi) $q_j, j = 1, \dots, m$ is estimated in each of the m multiply imputed data sets along with its variance $SE(q_j)^2$ (squared standard deviation). Then, the final estimate of the qoi $\bar{q}$ and its variance is given by

$$\bar{q} = \frac{1}{m}\sum_{j=1}^{m} q_j, \quad SE(\bar{q})^2 = \frac{1}{m}\sum_{j=1}^{m} SE(q_j)^2 + \left(1 + \frac{1}{m}\right) S_q^2$$

where $S_q^2 = \sum_{j=1}^{m}(q_j - \bar{q})^2/(m-1)$. We make use again of the function `att`.

```
> out <- att(mat2, re78 ~ treated, data = imputed)

           (Intercept)  treated
Estimate     4527.4667 729.0929
Std. Error    426.2025 658.3930
```

In the case of multiple data sets, the output of `att` also contains a list of single att estimates for each mutiply imputed data, i.e.

```
> str(out)

List of 2
 $ mult:List of 5
  ..$ : num [1:2, 1:4] 4556.7  728.2  416.3  642.3   10.9 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "(Intercept)" "treated"
  .. .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
  ..$ : num [1:2, 1:4] 4492.4  754.7  425.5  656.6   10.6 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "(Intercept)" "treated"
  .. .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
  ..$ : num [1:2, 1:4] 4638.1  571.4  420.2  648.4   11.0 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "(Intercept)" "treated"
  .. .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
  ..$ : num [1:2, 1:4] 4485.2  704.5  415.3  640.8   10.8 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "(Intercept)" "treated"
  .. .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
  ..$ : num [1:2, 1:4] 4464.9  886.6  418.1  645.2   10.7 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "(Intercept)" "treated"
  .. .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
 $ est : num [1:2, 1:2] 4527  426  729  658
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "Estimate" "Std. Error"
  .. ..$ : chr [1:2] "(Intercept)" "treated"
```

The user can apply the `att` function also to the first example, but the estimate of the qoi is not well defined from the statistical point of view.

7 R Functions

7.1 `cem`: Coarsened Exact Matching

Description

Implementation of Coarsened Exact Matching

Usage

```
cem(treatment = NULL, data, cutpoints = NULL, drop = NULL,
    eval.imbalance = FALSE, k2k = FALSE,  method=NULL, mpower=2, verbose = 0)
```

Arguments

`treatment`	character, name of the treatment variable
`data`	a data.frame
`cutpoints`	named list each describing the cutpoints for the variables (the names are variable names). Each list element is either a vector of cutpoints, a number of cutpoints, or a method for automatic bin contruction. See Details.
`drop`	a vector of variable names in the data frame to ignore during matching
`eval.imbalance`	Boolean. See Details.
`k2k`	boolean, return k-to-k matching? Default = `FALSE`
`method`	distance method to use in `k2k` matching. See Details.
`mpower`	power of the Minkowski distance. See Details.
`verbose`	controls level of verbosity. Default=0.

Details

When specifying cutpoints, several automatic methods can be chosen among “`sturges`” (Surges’ rule, the default), “`fd`” (Freedman-Diaconis’ rule), “`scott`” (Scott’s rule) and “ss” (Shimazaki-Shinomoto’s rule). See references for a description of each rule.

`verbose`: a number greater or equal to 0. The higher, the more info are provided during the execution of the algorithm.

If `eval.imbalance` = `TRUE`, `cem$imbalance` contains the imbalance measure by absolute difference in means for numerical variables and chi-square distance for categorical variables. If `FALSE` (the default) then `cem$imbalance` is set to `NULL`.

If `k2k` is set to `TRUE`, the algorithm return strata with the same number of treated and control units per stratum, otherwise all the matched units are returned (default). When `k2k` = `TRUE`, the user can choose a `method` (between ‘`euclidean`’, ‘`maximum`’, ‘`manhattan`’, ‘`canberra`’, ‘`binary`’ and ‘`minkowski`’) for nearest neighbor matching inside each `cem` strata. By default `method` is set to ‘`NULL`’, which means random matching inside `cem` strata. For the Minkowski distance the power can be specified via the argument `mpower`'. For more information on `method != NULL`, refer to `dist` help page.

In case of missing data, `cem` gives a warning and treats missing values as distinct values and match observations with missing values in the same variable in the same stratum provided that all the remaining (corasened) covariates match.

Value

`call`	the call
`strata`	vector of stratum number in which each observation belongs, NA if the observation has not been matched
`n.strata`	number of strata generated
`vars`	report variables names used for the match
`drop`	variables removed from the match
`breaks`	named list of cutpoints, eventually NULL
`treatment`	name of the treatment variable
`groups`	factor, each observation belong to one group generated by the treatment variable
`n.groups`	number of groups identified by the treatment variable
`group.idx`	named list, index of observations belonging to each group
`group.len`	sizes of groups
`tab`	summary table of matched by group
`imbalance`	NULL or a vector of imbalances. See Details.

Author(s)

Stefano Iacus, Gary King, and Giusseppe Porro

References

Stefano Iacus, Gary King, Giusseppe Porro, "Matching for Casual Inference Without Balance Checking," http://gking.harvard.edu/files/abs/cem-abs.shtml

Examples

```
data(LL)

mybr = list(re74=hist(LL$re74,plot=FALSE)$breaks,
 re75 = hist(LL$re75,plot=FALSE)$breaks,
 age = hist(LL$age,plot=FALSE)$breaks,
 education = hist(LL$education,plot=FALSE)$breaks)

L1.meas(LL$treated, LL[,-c(1,9)], breaks=mybr)
eval.match(LL$treated, LL[,-c(1,9)], breaks=mybr)

# cem match: automatic bin choice
mat <- cem(treatment="treated",data=LL, drop="re78")
mat$tab
cem1.idx <- which(mat$matched)
# imbalance
L1.meas(LL$treated[cem1.idx], LL[cem1.idx,-c(1,9)], breaks=mybr)
eval.match(LL$treated[cem1.idx], LL[cem1.idx,-c(1,9)], breaks=mybr)
L1.meas(LL$treated[cem1.idx], LL[cem1.idx,-c(1,9)], breaks=mybr,weights=mat$w[cem1.idx])
eval.match(LL$treated[cem1.idx], LL[cem1.idx,-c(1,9)], breaks=mybr,weights=mat$w[cem1.idx])

# cem match: user choiced coarsening
re74cut <- hist(LL$re74, br=seq(0,max(LL$re74)+1000, by=1000),plot=FALSE)$breaks
re75cut <- hist(LL$re75, br=seq(0,max(LL$re75)+1000, by=1000),plot=FALSE)$breaks
agecut <- hist(LL$age, br=seq(15,55, length=14),plot=FALSE)$breaks
mycp <- list(re75=re75cut, re74=re74cut, age=agecut)
mat <- cem(treatment="treated",data=LL, drop="re78",cutpoints=mycp)
mat$tab
cem2.idx <- which(mat$matched)
#imbalance
L1.meas(LL$treated[cem2.idx], LL[cem2.idx,-c(1,9)], breaks=mybr)
eval.match(LL$treated[cem2.idx], LL[cem2.idx,-c(1,9)], breaks=mybr)
L1.meas(LL$treated[cem2.idx], LL[cem2.idx,-c(1,9)], breaks=mybr,weights=mat$w[cem2.idx])
eval.match(LL$treated[cem2.idx], LL[cem2.idx,-c(1,9)], breaks=mybr,weights=mat$w[cem2.idx])

# cem match: user choiced coarsening, k-to-k matching
mat <- cem(treatment="treated",data=LL, drop="re78",cutpoints=mycp,k2k=TRUE)
mat$tab
cem3.idx <- which(mat$matched)
#imbalance
L1.meas(LL$treated[cem3.idx], LL[cem3.idx,-c(1,9)], breaks=mybr)
eval.match(LL$treated[cem3.idx], LL[cem3.idx,-c(1,9)], breaks=mybr)
L1.meas(LL$treated[cem3.idx], LL[cem3.idx,-c(1,9)], breaks=mybr,weights=mat$w[cem3.idx])
eval.match(LL$treated[cem3.idx], LL[cem3.idx,-c(1,9)], breaks=mybr,weights=mat$w[cem3.idx])

# mahalnobis matching
require(MatchIt)
mah <- matchit(treated~age+education+re74+re75+black+hispanic+nodegree+married+u74+u75,
   distance="mahalanobis", data=LL)
mah
idx1 <- as.numeric(mah$match.matrix)
```

```
idx2 <- as.numeric(rownames(mah$match.matrix))
mah.idx <- match( c(idx1,idx2), rownames(LL))
#imbalance
L1.meas(LL$treated[mah.idx], LL[mah.idx,-c(1,9)], breaks=mybr)
eval.match(LL$treated[mah.idx], LL[mah.idx,-c(1,9)], breaks=mybr)
```

7.2 `att`: Example of ATT estimation from CEM output

Description

An example of ATT estimation from CEM output

Usage

```
att(obj, formula, data, model="lm", family="binomial")
```

Arguments

`obj`	a `cem` or `multicem` object
`data`	a single data.frame or a list of data.frame's in case of `multicem`
`formula`	formula type specification of model. See Details.
`model`	either `lm` or `glm`. See Details.
`family`	used if model is `glm`, otherwise ignored.

Details

Argument `data` must be a single data frame or a list of (mulitply imputed) data frames.

Argument `model` can be `lm` or `glm` if the outcome variable in the ATT estimation is, e.g., a binary outcome. If the outcome is `y` and the treatment variable is `T`, then a `formula` like `y ~ T` is enough to estimate the ATT: it is just the coefficient of `T`. User can add covariates to span any remaining imbalance after the match, such as `y ~ T + age + sex`, to adjust for variables `age` and `sex`.

In the case of multiply imputed datasets, the model is applied to each single matched data and the ATT and is the standard error estimated using the standard formulas for combining results of multiply imputed data.

Value

A matrix of estimates with their standard error, or a list in case of `multicem`.

Examples

```
 data(LL)

# cem match: automatic bin choice
mat <- cem(treatment="treated",data=LL, drop="re78")
mat$tab
mat$k2k

# ATT estimate
```

```
att(mat, re78~treated,  data=LL)

# reduce the match into k2k using euclidean distance within cem strata
mat2 <- k2k(mat, LL, "euclidean", 1)
mat2$tab
mat2$k2k

# ATT estimate after k2k
att(mat2, re78~treated, data=LL)

# using multiply imputated data
require(Amelia)

data(LL)
n <- dim(LL)[1]
k <- dim(LL)[2]

# we generate missing values in 30
# randomly in one colum per row
LL1 <- LL
idx <- sample(1:n, .3*n)
invisible(sapply(idx, function(x) LL1[x,sample(2:k,1)] <<- NA))

# we use Amelia for multiple imputation

imputed <- amelia(LL1)

mat <- multicem("treated", datalist=imputed[1:5], drop="re78")

out <- att(mat, re78 ~ treated, data=imputed[1:5])

str(out)
```

7.3 `DW`: Dehejia-Wahba dataset

Description

A subset of the Lalonde dataset (see cited reference).

Usage

```
data(DW)
```

Format

A data frame with 445 observations on the following 10 variables.

`treated` treated variable indicator

`age` age

`education` years of education

`black` race indicator variable

`married` marital status indicator variable

`nodegree` indicator variable of not possessing a degree

`re74` real earnings in 1974

`re75` real earnings in 1975

`re78` real earnings in 1978 (post treatment outcome)

`hispanic` ethnic indicator variable

`u74` unemployment in 1974 indicator variable

`u75` unemployment in 1975 indicator variable

Source

see references

References

Dehejia, R., Wahba, S. (1999) "Causal Effects in Nonexperimental Studies: Reevaluating the Evaluation of Training Programs," *Journal of the American Statistical Association*, 94, 1053-1062.

7.4 `eval.match`: Calculates several one dimensional imbalance measures

Description

Calculates several one dimensional imbalance measures for the original and matched data sets

Usage

```
eval.match(group, data, breaks = NULL, weights)
```

Arguments

`group`	the group variable
`data`	the data
`breaks`	a list of vectors of cutpoints used to calculate L1 measure. See Details.
`weights`	weights

Details

This function calculate several imbalance measures. For numeric variables the difference in means (under the column `statistics`, the difference in quantiles and the L1 measure is calculated. For categorical variables the L1 measure and the Chi-squared distance (under column `statistics`) is calculated.

If the `breaks` are not specified, the same approach as in `cem` is used. Please refer to `cem` help page. In this case, breaks are used to calculate the L1 measure.

Value

`value`	Table of imbalance measures

Author(s)

Stefano Iacus, Gary King, and Giusseppe Porro

References

Stefano Iacus, Gary King, Giusseppe Porro, “Matching for Casual Inference Without Balance Checking,” http://gking.harvard.edu/files/abs/cem-abs.shtml

Examples

```
 data(LL)

mybr = list(re74=hist(LL$re74,plot=FALSE)$breaks,
 re75 = hist(LL$re75,plot=FALSE)$breaks,
 age = hist(LL$age,plot=FALSE)$breaks,
 education = hist(LL$education,plot=FALSE)$breaks)

L1.meas(LL$treated, LL[,-c(1,9)],breaks=mybr)
eval.match(LL$treated, LL[,-c(1,9)], breaks=mybr)

# cem match: automatic bin choice
mat <- cem(treatment="treated",data=LL, drop="re78")
mat$tab
cem1.idx <- which(mat$matched)
# imbalance
L1.meas(LL$treated[cem1.idx], LL[cem1.idx,-c(1,9)],breaks=mybr)
eval.match(LL$treated[cem1.idx], LL[cem1.idx,-c(1,9)], breaks=mybr)
eval.match(LL$treated[cem1.idx], LL[cem1.idx,-c(1,9)], breaks=mybr,weights=mat$w[cem1.idx])
```

7.5 `k2k`: Reduction to k2k Matching

Description

Reduces a CEM output to a k2k matching

Usage

```
k2k(obj, data, method=NULL, mpower=2, verbose=0)
```

Arguments

`obj`	an object as output from `cem`
`data`	the original data.frame used by `cem`
`method`	distance method to use in `k2k` matching. See Details.
`mpower`	power of the Minkowski distance. See Details.
`verbose`	controls level of verbosity. Default=0.

Details

This function transforms a typical `cem` matching solution to a `k`-to-`k` match, with `k` variable along strata: i.e., in each stratum generated by `cem`, the match is reduce to have the same number of treated and control units. (This option will delete some data that matched well, and thus likely increase the variance, but it means that subsequent analyses do not require weights.)

The user can choose a `method` (between '`euclidean`', '`maximum`', '`manhattan`', '`canberra`', '`binary`' and '`minkowski`') for nearest neighbor matching inside each `cem` strata. By default `method` is set to '`NULL`', which means random matching inside `cem` strata. For the Minkowski distance the power can be specified via the argument `mpower`'. For more information on `method != NULL`, refer to `dist` help page.

After `k2k` the weights of each matched observation are set to unity.

Value

`obj`	a cem object

Author(s)

Stefano Iacus, Gary King, and Giusseppe Porro

References

Stefano Iacus, Gary King, Giusseppe Porro, "Matching for Casual Inference Without Balance Checking," http://gking.harvard.edu/files/abs/cem-abs.shtml

Examples

```
data(LL)

# cem match: automatic bin choice
mat <- cem(treatment="treated",data=LL, drop="re78")
mat$tab
mat$k2k

# ATT estimate
summary(lm(re78 ~ treated, data=LL, weights=mat$w))

# transform the match into k2k
mat2 <- k2k(mat, LL, "euclidean", 1)
mat2$tab
mat2$k2k

# ATT estimate after k2k
summary(lm(re78 ~ treated, data=LL, weights=mat2$w))
```

7.6 `L1.meas`: Evaluates L1 distance between multidimensional histograms

Description

Evaluates L1 distance between multidimensional histograms

Usage

```
L1.meas(group, data, breaks = NULL, weights)
```

Arguments

`group`	the group variable
`data`	the data
`breaks`	a list of vectors of cutpoints; if not specified, automatic choice will be made
`weights`	weights

Details

This function calculates the L1 distance on the k-dimensional histogram.

If the `breaks` are not specified, the same approach as in `cem` is used. Please refer to `cem` help page. In this case, breaks are used to calculate the L1 measure.

Value

`value`	the L1 measure

Author(s)

Stefano Iacus, Gary King, and Giusseppe Porro

References

Stefano Iacus, Gary King, Giusseppe Porro, "Matching for Casual Inference Without Balance Checking," http://gking.harvard.edu/files/abs/cem-abs.shtml

Examples

```
data(LL)
L1.meas(LL$treated,LL[,-c(1,9)])
```

7.7 `LL`: Lalonde dataset

Description

Lalonde experimental dataset (see cited reference).

Usage

```
data(LL)
```

Format

A data frame with 722 observations on the following 10 variables.

`treated` treatment variable indicator

`age` age

`education` years of education

`black` race indicator variable

`married` marital status indicator variable

`nodegree` indicator variable for not possessing a degree

`re74` real earnings in 1974

`re75` real earnings in 1975

`re78` real earnings in 1978 (post-treatment outcome)

`hispanic` ethnic indicator variable

`u74` unemployment in 1974 indicator variable

`u75` unemployment in 1975 indicator variable

Source

see references

References

Lalonde, R. (1986) “Evaluating the Econometric Evaluations of Training Programs,” *American Economic Review*, 76, 604-620.

7.8 `multicem`: Coarsened Exact Matching for Multiply Imputed Data

Description

Implementation of Coarsened Exact Matching for Multiply Imputed Data

Usage

```
multicem(treatment = NULL, datalist, data = NULL, cutpoints = NULL, drop = NULL,
    eval.imbalance = FALSE, k2k = FALSE,  method=NULL, mpower=2, verbose = 0)
```

Arguments

`treatment`	character, name of the treatment variable
`datalist`	a list of imputed data.frame's
`data`	original data.frame with missing values
`cutpoints`	named list each describing the cutpoints for the variables (the names are variable names). Each list element is either a vector of cutpoints, a number of cutpoints, or a method for automatic bin contruction. See Details.
`drop`	a vector of variable names in the data frame to ignore during matching
`eval.imbalance`	boolean. See Details.
`k2k`	boolean, return k-to-k matching? Default = `FALSE`
`method`	distance method to use in `k2k` matching. See Details.
`mpower`	power of the Minkowski distance. See Details.
`verbose`	controls level of verbosity. Default=0.

Details

Argument `datalist` is a list of (multiply imputed) data frames. If `data` is not specified, the function `cem` is applied independently to each element of the list, resulting in separately matched data sets with different numbers of treated and control units.

When `data` is specified, each multiply imputed observation is assigned to the stratum in which it has been matched most frequently. In this case, the algorithm outputs the same matching solution for each multiply imputed data set (i.e., an observation, and the number of treated and control units matched, in one data set has the same meaning in all, and is the same for all)

All the remaining arguments are passed to `cem` as specified.

Value

An object of class `multicem`, i.e. a list of objects of class `cem`

Author(s)

Stefano Iacus, Gary King, and Giusseppe Porro

References

Stefano Iacus, Gary King, Giusseppe Porro, "Matching for Casual Inference Without Balance Checking," http://gking.harvard.edu/files/abs/cem-abs.shtml

Examples

```
require(Amelia)

data(LL)
n <- dim(LL)[1]
k <- dim(LL)[2]

set.seed(123)

LL1 <- LL
idx <- sample(1:n, .3*n)
invisible(sapply(idx, function(x) LL1[x,sample(2:k,1)] <<- NA))

imputed <- amelia(LL1,noms=c("black","hispanic","treated","married","nodegree","u74","u75"))

# without information on which observation has missing values
mat1 <- multicem("treated", datalist=imputed, drop="re78")
#str(mat1, max.lev=1)
mat1$match1$tab
mat1$match2$tab

# ATT estimation
out <- att(mat1, re78 ~ treated, data=imputed)

# with information about missingness
mat2 <- multicem("treated", datalist=imputed, drop="re78", data=LL1)
#str(mat2, max.lev=1)
mat2$match1$tab
mat2$match2$tab

# ATT estimation
out <- att(mat2, re78 ~ treated, data=imputed)
```

7.9 `relax.cem`: Diagnostic tool for CEM

Description

Diagnostic tools for CEM

Usage

```
relax.cem(obj, data, depth=1, verbose = 1, L1.breaks=NULL, plot=TRUE, fixed=NULL,
 shifts=NULL, minimal=NULL, use.coarsened=TRUE)
relax.plot(tab, group="1", max.terms=50, perc=.5, unique=FALSE, colors=TRUE)
```

Arguments

`obj`	an object of class cem.
`data`	the original data.
`verbose`	controls the level of verbosity.
`L1.breaks`	list of cutpoints for the calculation of the L1 measure.
`plot`	plot the solutions?
`tab`	the output table from `relax.cem`.
`fixed`	vector of variable names which will not be relaxed.
`max.terms`	plot only the last best results of `relax.cem`.
`shifts`	a vector of proportions of shifts.
`minimal`	the minimal number of intervals acceptable after relaxation. Should be a nameed list of positive integers.
`group`	character string denoting group id. Defaults to `"1"`.
`perc`	only plot if percentage of matched units is greater than `perc`.
`unique`	only plot different solutions (in terms of matched units).
`depth`	if 1, relaxes up to dropping one var, if 2 relaxes (up to dropping) two vars, etc.
`use.coarsened`	used coarsened values for continuous variables.
`colors`	If `TRUE` each variable is plotted in a different colour.

Details

`relax.cem` starts from a cem solution (as given by `cem`) and tries several relaxed coarsenings on the variables. Coarsenings corresponds to dividing the support of each variable into a decreasing number of intervals of the same length (even if in the starting solution intervals are of different lengths). Because CEM is MIB, the number of matched units increases as the number of intervals decrease. All variables are coarsened into `k` intervals along a sequence which starts from the original number of intervals and decreases to 10 intervals by 2, then continues from 10 down to 1 intervals by 1. If `minimal` is specified, variables are coarsened down to that minimal value.

To observe MIB property of CEM `use.coarsened` (default) should be set to `TRUE`; otherwise the coarsening of the continuous variable will be recalculated at each iteration and there is no guarantee of monotonicity.

`relax.cem` outputs a list of tables. Each table is named `Ggroup` where `group` is the id of the group. Each `Ggroup` table is ordered in increasing order of matched units of group `group`. Columns `PercGgroup` and `Ggroup` report percentage and absolute number of matched units for each `group`. Column `Relaxed` indicates which relaxation has been done, with something like `"V1(4), V3(5)"`, which means ”variable `V1` has been split in 4 intervals of the same length and variable `V3` into five intervals”. Thus, the number of intervals is reported in parenthases and if equal to 1 means that the corresponding variable is excluded from affecting the match (i.e. all observations are assigned to the same interval).

If `shifts` is not null, each coarsening is shifted accordingly (see `shift.cem` for additional details). In case of shifting “`S:`” appears in the labels.

The `relax.plot`, plot all the different relaxation in increasing order of number of treated units matched. For each coarsening it also reports the value of the L1 measure. The table generated by `relax.cem` may contain many entries. By default, only a portion of best coarsenings are plotted (option `max.terms`). In addition, the user can specify to plot the corasening for which at least a certain percentage of treated units have been matched (option `perc`, by default 50 In addition, of several different coarsenings which lead to the same number of treated units matched, the user can specify to plot only one of them using the option `unique` = `TRUE` (default).

Value

`tab`	an invisible object containing the tabs

Author(s)

Stefano Iacus, Gary King, and Giusseppe Porro

References

Stefano Iacus, Gary King, Giusseppe Porro, “Matching for Casual Inference Without Balance Checking,” http://gking.harvard.edu/files/abs/cem-abs.shtml

See Also

`cem`

Examples

```
data(LL)

mybr = list(re74=hist(LL$re74,plot=FALSE)$breaks,
  re75 = hist(LL$re75,plot=FALSE)$breaks,
  age = hist(LL$age,plot=FALSE)$breaks,
  education = hist(LL$education,plot=FALSE)$breaks)

mat <- cem(treatment="treated",data=LL, drop="re78")
mat$tab

tab <- relax.cem(mat, LL, L1.breaks=mybr, depth=1, plot=FALSE)

relax.plot(tab, group="1")
relax.plot(tab, group="1", unique=TRUE)
relax.plot(tab, group="1", perc=0.6)
relax.plot(tab, group="1", perc=0.6,unique=TRUE)

tab1 <- relax.cem(mat, LL, L1.breaks=mybr, depth=1, minimal=list(re74=6, age=3, education=3
tab2 <- relax.cem(mat, LL, L1.breaks=mybr, depth=1, minimal=list(re74=6, age=3, education=3
tab3 <- relax.cem(mat, LL, L1.breaks=mybr, depth=1, minimal=list(age=3, education=3), fixed=

# uncomment to run. Might be slow
# tab4 <- relax.cem(mat, LL, L1.breaks=mybr, depth=2, minimal=list(age=4, education=3,re75=6
# relax.plot(tab4)
# relax.plot(tab4, unique=TRUE)
# relax.plot(tab4, perc=0.7)
```

7.10 `shift.cem`: Diagnostic tool for CEM

Description

Diagnostic tools for CEM. Applies leftward and rightward shifts of the cutpoints.

Usage

```
shift.cem(obj, data, shifts=NULL, verbose=0, plot=TRUE)
```

Arguments

`obj`	and object of class cem
`data`	the original data
`shifts`	a vector of proportions of shifts
`verbose`	controls the level of verbosity
`plot`	whether to plot a graphic representation of the search

Details

For each variable, shift all the cutpoints left and right by `shifts` times the smallest epsilon of the coarsening. Shifting to the right produces a new cell on the left; shift to the left, adds a new cell to the coarsening on the right. Only positive proportions should be used; the algorithm will produce shifting on the left or on the right. The best shifting of the original cem match is produced as output, where best is defined in terms of the maximal total number of matched units `mT+mC` (see below).

By default, the function returns minimal information about the execution of the algorithm. By setting a value greater than 0 in option `verbose` more feedback on the process is returned.

Option `plot` = `TRUE` plots the number of treated units matched `mT`, the number of control units matched `mC`, and the sum `mT+mC`, as a function of the shifts.

Value

`tab`	an invisible object containing a new cem object

Author(s)

Stefano Iacus, Gary King, and Giuseppe Porro

References

Stefano Iacus, Gary King, Giuseppe Porro, "Matching for Casual Inference Without Balance Checking," http://gking.harvard.edu/files/abs/cem-abs.shtml

See Also

`cem`

Examples

```
data(LL)

m74 <- max(LL$re74, na.rm=TRUE)
s74 <- seq(0,m74,by=sd(LL$re74))
l74 <- length(s74)
if(max(s74) < m74) s74 <- c(s74, m74)

m75 <- max(LL$re75, na.rm=TRUE)
s75 <- seq(0,m75,by=sd(LL$re75))
l75 <- length(s75)
if(max(s75) < m75) s75 <- c(s75, m75)

mybr = list(re74=s74,
 re75 = s75,
 age = hist(LL$age,plot=FALSE)$breaks,
 education = hist(LL$education,plot=FALSE)$breaks)

mat <- cem(treatment="treated",data=LL, drop="re78",cut=mybr)
mat$tab

shift.cem(mat, data=LL, shifts=seq(0.01, 0.5, length=10), verb=1)
```

References

Freedman, D. and P. Diaconis. 1981. "On the histogram as a density estimator: L_2 theory." *Probability Theory and Related Fields* 57:453–476.

Ho, Daniel E., Kosuke Imai, Gary King and Elizabeth A. Stuart. Forthcoming. "MatchIt: Nonparametric Preprocessing for Parametric Causal Inference." *Journal of Statistical Software* . http://gking.harvard.edu/matchit.

Honaker, James, Gary King and Matthew Blackwell. 2006. "Amelia II: A Program for Missing Data.". http://gking.harvard.edu/amelia.

Iacus, Stefano M., Gary King and Giuseppe Porro. 2008. "Matching for Causal Inference Without Balance Checking.". http://gking.harvard.edu/files/abs/cem-abs.shtml.

Lalonde, Robert. 1986. "Evaluating the Econometric Evaluations of Training Programs." *American Economic Review* 76:604–620.

Scott, D.W. 1992. *Multivariate density estimation. Theory, practice and visualization.* New York: John Wiley & Sons, Inc.

Shimazaki, H. and S. Shinomoto. 2007. "A Method for Selecting the Bin Size of a Time Histogram." *Neural Computation* 19(6):1503–1527.