

# cem: Coarsened Exact Matching

Version 1.0.15  
August 13, 2008

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Setup</b>	<b>2</b>
2.1	Software Requirements . . . . .	2
2.2	Installation . . . . .	3
2.3	Loading CEM . . . . .	3
2.4	Updating CEM . . . . .	3
<b>3</b>	<b>A User's Guide</b>	<b>3</b>
3.1	Basic Analysis of Unmatched Data . . . . .	4
3.2	Matching with CEM . . . . .	7
3.3	Choosing the coarsening . . . . .	9
3.4	Progressive coarsening . . . . .	11
3.5	Restricting the matching solution to a $k$ -to- $k$ match . . . . .	13
3.6	An example of estimation of SATT from <code>cem</code> output . . . . .	14
3.7	Working with missing data . . . . .	15
3.8	Working with multiply imputed data . . . . .	16
<b>4</b>	<b>Reference to CEM's Functions</b>	<b>20</b>
4.1	<code>cem</code> : Coarsened Exact Matching . . . . .	21
4.2	<code>att</code> : Example of ATT estimation from CEM output . . . . .	25
4.3	<code>DW</code> : Dehejia-Wahba dataset . . . . .	27
4.4	<code>eval.match</code> : Calculates several one dimensional imbalance measures . . . . .	28
4.5	<code>k2k</code> : Reduction to k2k Matching . . . . .	30
4.6	<code>L1.meas</code> : Evaluates L1 distance between multidimensional histograms . . . . .	32
4.7	<code>LL</code> : Lalonde dataset . . . . .	34
4.8	<code>relax.cem</code> : Diagnostic tool for CEM . . . . .	35
4.9	<code>shift.cem</code> : Diagnostic tool for CEM . . . . .	38

# 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. This algorithm, and its many attractive statistical properties, are described in

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

The paper shows that CEM is a monotonic imbalance bounding (MIB) 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 repeatedly 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, creating randomized blocks in experimental designs, and evaluating extreme counterfactuals.

## 2 Setup

### 2.1 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 works seamlessly with Zelig (<http://gking.harvard.edu/zelig>) for estimating causal effects after matching, but presently only offers a subset of features of the R version. A Stata version of CEM is also available at the CEM web site, <http://gking.harvard.edu/cem>.

## 2.2 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")
```

## 2.3 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)
```

## 2.4 Updating CEM

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

```
> update.packages()
```

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

```
> library(cem)
```

# 3 A User's Guide

We show here how to use CEM through a simple running example: the National Supported Work (NSW) Demonstration data, also known as the Lalonde data set (Lalonde, 1986). This program provided training to selected individuals for 12-18 months and help finding a job in the hopes of increasing their' earnings. The treatment variable, `treated`, is 1 for participants (the treatment group) and 0 for nonparticipants (the control group). The key outcome variable is earnings in 1978 (`re78`). The statistical goal is to estimate the sample average treatment effect on the treated (the "SATT").

Since participation in the program was not assigned strictly at random, we must control for a set of pretreatment variables by the CEM algorithm. These pre-treatment variables include 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.

Matching is not a method of estimation; it is a way to preprocess a data set so that estimation of SATT based on the matched data set will be less “model-dependent” (i.e., less a function of apparently small and indefensible modeling decisions) than when based on the original full data set. Matching involves pruning observations that have no close matches on pre-treatment covariates in both the treated and control groups. The result is typically less model-dependence, bias, and (by removing heterogeneity) inefficiency (King and Zeng, 2006; Ho et al., 2007; Iacus, King and Porro, 2008).

### 3.1 Basic Analysis of Unmatched Data

We begin with a naive estimate of SATT — the simple difference in means — which would be useful only if the in-sample distribution of pre-treatment covariates were the same in the treatment and control groups:

```
> 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)
```

The data include 297 treated units and 425 control units. The (unadjusted and therefore likely biased) difference in means is then:

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

```
[1] 886.3
```

Because the variable `treated` was not randomly assigned, the pre-treatment covariates differ between the treated and control groups. To see this, we focus on these pre-treatment covariates:

```
> vars <- c("age", "education", "black", "married", "nodegree",
+           "re74", "re75", "hispanic", "u74", "u75")
```

The overall imbalance is given by the  $\mathcal{L}_1$  statistic, introduced in Iacus, King and Porro (2008) as a comprehensive measure of global imbalance. It is based on the  $L1$  difference between the multidimensional histogram of all pretreatment covariates in the treated group and that in the control group. Perfect global balance is indicated by  $\mathcal{L}_1 = 0$ , and larger

values indicate larger imbalance between the groups. To use this measure, we require a list of bin sizes for the numerical variables. Our functions compute these automatically, or they can be set by the user, but either way we must keep them fixed as we match to ensure comparability.<sup>1</sup>

Then, in our data, we calculate the overall imbalance as:

```
> L1.meas(LL$treated, LL[vars])
```

```
Multivariate imbalance measure L1=1.469138
```

which means that  $\mathcal{L}_1 = 1.469$ . One can see (and save to ensure comparability) the breaks automatically generated by `L1.meas` with

```
> L1breaks <- L1.meas(LL$treated, LL[vars])$breaks
> L1breaks
```

```
$age
```

```
[1] 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56
```

```
$education
```

```
[1] 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0
[16] 10.5 11.0 11.5 12.0 12.5 13.0 13.5 14.0 14.5 15.0 15.5 16.0
```

```
$black
```

```
[1] 0.0 0.2 0.4 0.6 0.8 1.0
```

```
$married
```

```
[1] 0.0 0.2 0.4 0.6 0.8 1.0
```

```
$nodegree
```

```
[1] 0.0 0.2 0.4 0.6 0.8 1.0
```

```
$re74
```

```
[1] 0 2000 4000 6000 8000 10000 12000 14000 16000 18000 20000 22000
[13] 24000 26000 28000 30000 32000 34000 36000 38000 40000
```

```
$re75
```

```
[1] 0 2000 4000 6000 8000 10000 12000 14000 16000 18000 20000 22000
[13] 24000 26000 28000 30000 32000 34000 36000 38000
```

```
$hispanic
```

---

<sup>1</sup>Of course, like in drawing histograms, the choice of bins affects the final result. The important thing is to choose one and keep it the same throughout to allow for fair comparisons. The particular choice is less crucial.

```
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
$u74
```

```
[1] 0.0 0.2 0.4 0.6 0.8 1.0
```

```
$u75
```

```
[1] 0.0 0.2 0.4 0.6 0.8 1.0
```

To get a better feel for the components of this imbalance, the function `eval.match`, returns a table of imbalance statistics (in columns) for each variable (in rows). Only the overall  $\mathcal{L}_1$  statistic (which is also reported after the table by `eval.match`) measures imbalance in all interactions among the variables; the measures in the table are all computed for each variable separately. In our data:

```
> eval.match(group = LL$treated, data = LL[vars])
```

One dimensional measures of imbalance

	statistics	type	L1	min	25%	50%	75%	max
age	1.792e-01	(diff)	0.009412	0	1	0.00	-1.0	-6.0
education	1.922e-01	(diff)	0.196237	1	0	1.00	1.0	2.0
black	1.347e-03	(diff)	0.002694	0	0	0.00	0.0	0.0
married	1.070e-02	(diff)	0.021406	0	0	0.00	0.0	0.0
nodegree	-8.348e-02	(diff)	0.166956	0	-1	0.00	0.0	0.0
re74	-1.015e+02	(diff)	0.000000	0	0	69.73	584.9	-2139.0
re75	3.942e+01	(diff)	0.000000	0	0	294.18	660.7	490.4
hispanic	-1.867e-02	(diff)	0.037330	0	0	0.00	0.0	0.0
u74	-2.010e-02	(diff)	0.040198	0	0	0.00	0.0	0.0
u75	-4.509e-02	(diff)	0.090172	0	0	0.00	0.0	0.0

Multivariate imbalance measure L1=1.469138

The first column in this table, labeled `statistics`, reports the difference in means for numerical variables (indicated by the second column, `type`, reporting `(diff)`) or a chi-square difference for categorical variables (when the second column reports `(Chi2)`). In our running example, all variables are numerical or dichotomous, and so `(diff)` appears in all rows. The second column, labeled `L1`, reports the  $\mathcal{L}_1$  measure for each variable separately. The remaining columns in the table report the difference in the empirical quantile of the distributions of the two groups for the 0th (min), 25th, 50th, 75th, and 100th (max) percentiles for each variable.

This particular table shows that variables `re74` and `re75` are imbalanced in the raw data in many ways and variable `age` is balanced in means but not in the quantiles of the two distributions. This table also illustrates the point that balancing only the means between

the treated and control groups does not necessarily guarantee balance in the rest of the distribution. Most important, of course, is the overall  $\mathcal{L}_1$  measure reported after the table, since even if the marginal distribution of every variable is perfectly balanced, the joint distribution can be highly imbalanced.

An alternative use of `eval.match` allows you to drop some variables within the function:

```
> todrop <- c("treated", "re78")

[1] "treated" "re78"

> eval.match(group = LL$treated, data = LL, drop = todrop)
```

One dimensional measures of imbalance

	statistics	type	L1	min	25%	50%	75%	max
age	1.792e-01	(diff)	0.009412	0	1	0.00	-1.0	-6.0
education	1.922e-01	(diff)	0.196237	1	0	1.00	1.0	2.0
black	1.347e-03	(diff)	0.002694	0	0	0.00	0.0	0.0
married	1.070e-02	(diff)	0.021406	0	0	0.00	0.0	0.0
nodegree	-8.348e-02	(diff)	0.166956	0	-1	0.00	0.0	0.0
re74	-1.015e+02	(diff)	0.000000	0	0	69.73	584.9	-2139.0
re75	3.942e+01	(diff)	0.000000	0	0	294.18	660.7	490.4
hispanic	-1.867e-02	(diff)	0.037330	0	0	0.00	0.0	0.0
u74	-2.010e-02	(diff)	0.040198	0	0	0.00	0.0	0.0
u75	-4.509e-02	(diff)	0.090172	0	0	0.00	0.0	0.0

Multivariate imbalance measure L1=1.469138

## 3.2 Matching with CEM

We now apply `cem` which applies the exact matching algorithm to coarsened data to determine matches and then passes on the uncoarsened data that were matched to estimate the quantity of interest. The exact matching algorithm identifies strata with identical coarsened values of all the pre-treatment covariates and then keeps only those which include observations with at least one of every unique value of the treatment variable. Thus, the treatment variable may be dichotomous or mutichotomous. Alternatively, `cem` may be used for randomized block experiments without specifying a treatment variable; in this case no strata are deleted and the treatment variable is (randomly) assigned after matching.

In our running example we have a dichotomous treatment variable. We therefore match on all the pre-treatment variable. In this code, we drop `re78` which is the outcome variable and so should never be included. Hence we proceed specifying `"re78"` in argument `drop`:

```
> mat <- cem(treatment = "treated", data = LL, drop = "re78")
```

The object `mat` now contains useful information about the match. One is the summary of the matching solution

```
> mat
```

```

      G0  G1
All      425 297
Matched  222 163
Unmatched 203 134

```

from which we can see the number of (comparable) observations matched and thus retained, as well as those which were pruned because they were not comparable. The function `cem` also generates weights to be used in the evaluation of imbalance measures and statistical estimates of the treatment effect. These are contained in `mat$w`. Now we see at the imbalance measure on the original data

```
> eval.match(LL$treated, LL, drop = todrop, breaks = L1breaks)
```

One dimensional measures of imbalance

	statistics	type	L1	min	25%	50%	75%	max
age	1.792e-01	(diff)	0.009412	0	1	0.00	-1.0	-6.0
education	1.922e-01	(diff)	0.196237	1	0	1.00	1.0	2.0
black	1.347e-03	(diff)	0.002694	0	0	0.00	0.0	0.0
married	1.070e-02	(diff)	0.021406	0	0	0.00	0.0	0.0
nodegree	-8.348e-02	(diff)	0.166956	0	-1	0.00	0.0	0.0
re74	-1.015e+02	(diff)	0.000000	0	0	69.73	584.9	-2139.0
re75	3.942e+01	(diff)	0.000000	0	0	294.18	660.7	490.4
hispanic	-1.867e-02	(diff)	0.037330	0	0	0.00	0.0	0.0
u74	-2.010e-02	(diff)	0.040198	0	0	0.00	0.0	0.0
u75	-4.509e-02	(diff)	0.090172	0	0	0.00	0.0	0.0

Multivariate imbalance measure L1=1.469138

and on the matched data

```
> eval.match(LL$treated, LL, drop = todrop, weights = mat$w, breaks = L1breaks)
```

One dimensional measures of imbalance

	statistics	type	L1	min	25%	50%	75%	max
age	1.862e-01	(diff)	0.000e+00	0	0	0.0	1.00	1.0
education	1.022e-02	(diff)	2.045e-02	0	0	0.0	0.00	0.0
black	-1.110e-16	(diff)	1.249e-16	0	0	0.0	0.00	0.0
married	0.000e+00	(diff)	1.110e-16	0	0	0.0	0.00	0.0



nodegree	-1.110e-16	(diff)	1.110e-16	0	0	0.0	0.00	0.0
re74	7.198e+00	(diff)	0.000e+00	0	0	0.0	-70.86	416.4
re75	1.221e+01	(diff)	0.000e+00	0	0	234.5	140.79	-852.3
hispanic	0.000e+00	(diff)	1.110e-16	0	0	0.0	0.00	0.0
u74	0.000e+00	(diff)	5.551e-17	0	0	0.0	0.00	0.0
u75	0.000e+00	(diff)	0.000e+00	0	0	0.0	0.00	0.0

Multivariate imbalance measure L1=0.863087

From the above results it can be seen that a good match can produce substantial reduction in imbalance, not only in the means, but also in the marginal and joint distributions of the data.

Notice that `cem` can directly calculate the imbalance measures if we set the switch `eval.imbalance` to `TRUE`, indeed

```
> mat <- cem(treatment = "treated", data = LL, drop = "re78", eval.imbalance = TRUE,
+           L1.breaks = L1breaks)
> mat
```

	G0	G1
All	425	297
Matched	222	163
Unmatched	203	134

One dimensional measures of imbalance

	statistics	type	L1	min	25%	50%	75%	max
age	1.862e-01	(diff)	0.000e+00	0	0	0.0	1.00	1.0
education	1.022e-02	(diff)	2.045e-02	0	0	0.0	0.00	0.0
black	-1.110e-16	(diff)	1.249e-16	0	0	0.0	0.00	0.0
married	0.000e+00	(diff)	1.110e-16	0	0	0.0	0.00	0.0
nodegree	-1.110e-16	(diff)	1.110e-16	0	0	0.0	0.00	0.0
re74	7.198e+00	(diff)	0.000e+00	0	0	0.0	-70.86	416.4
re75	1.221e+01	(diff)	0.000e+00	0	0	234.5	140.79	-852.3
hispanic	0.000e+00	(diff)	1.110e-16	0	0	0.0	0.00	0.0
u74	0.000e+00	(diff)	5.551e-17	0	0	0.0	0.00	0.0
u75	0.000e+00	(diff)	0.000e+00	0	0	0.0	0.00	0.0

Multivariate imbalance measure L1=0.863087

### 3.3 Choosing the coarsening

When information about the real nature of the data is available, it is possible to specify directly the coarsening of each variable. For example, in the US educational system, the following discretization of years of education corresponds to different levels of school

grade school (0-6)  
middle school (7-8)  
high school (9-12)  
college (13-16)  
grad school (>16)

none of the observations belong to the last category

```
> table(LL$education)
```

```
 3  4  5  6  7  8  9 10 11 12 13 14 15 16
1  6  5  7 15 62 110 162 195 122 23 11  2  1
```

so we can define the cutpoints as follows

```
> educut <- c(0, 6.5, 8.5, 12.5, 17)
```

and run cem letting all the rest unchanged

```
> mat1 <- cem(treatment = "treated", data = LL, drop = "re78",
+             cutpoints = list(education = educut), eval.imbalance = TRUE,
+             L1.breaks = L1breaks)
> mat1
```

```
      G0  G1
All      425 297
Matched  254 186
Unmatched 171 111
```

One dimensional measures of imbalance

	statistics	type	L1	min	25%	50%	75%	max
age	1.443e-01	(diff)	0.000e+00	0	0	0.0	0.00	1.0
education	-1.437e-02	(diff)	7.398e-02	0	0	-1.0	0.00	-2.0
black	-1.110e-16	(diff)	1.249e-16	0	0	0.0	0.00	0.0
married	0.000e+00	(diff)	1.110e-16	0	0	0.0	0.00	0.0
nodegree	0.000e+00	(diff)	0.000e+00	0	0	0.0	0.00	0.0
re74	4.805e+01	(diff)	0.000e+00	0	0	369.3	233.61	416.4
re75	4.501e+01	(diff)	0.000e+00	0	0	226.6	-29.57	-852.3
hispanic	0.000e+00	(diff)	0.000e+00	0	0	0.0	0.00	0.0
u74	-5.551e-17	(diff)	5.551e-17	0	0	0.0	0.00	0.0
u75	-5.551e-17	(diff)	5.551e-17	0	0	0.0	0.00	0.0

Multivariate imbalance measure L1=1.097357

As seen, the matching solution is different and looking at the substance of the problem may help in the discovery of a good matching solution. Just for curiosity, the automatic cutpoints produced by `cem` are stored in the output object in the slot `breaks`. So, for example, in the first matching we have

```
> mat$breaks$education
[1] 3.0 4.3 5.6 6.9 8.2 9.5 10.8 12.1 13.4 14.7 16.0
```

and in our second example we recover our personal choice of cutpoints

```
> mat1$breaks$education
[1] 0.0 6.5 8.5 12.5 17.0
```

### 3.4 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, cutpoints = list(age = 10), drop = "re78")
```

	G0	G1
All	425	297
Matched	228	161
Unmatched	197	136

```
> cem("treated", LL, cutpoints = list(age = 6), drop = "re78")
```

	G0	G1
All	425	297
Matched	261	186
Unmatched	164	111

```
> cem("treated", LL, cutpoints = list(age = 3), drop = "re78")
```

	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

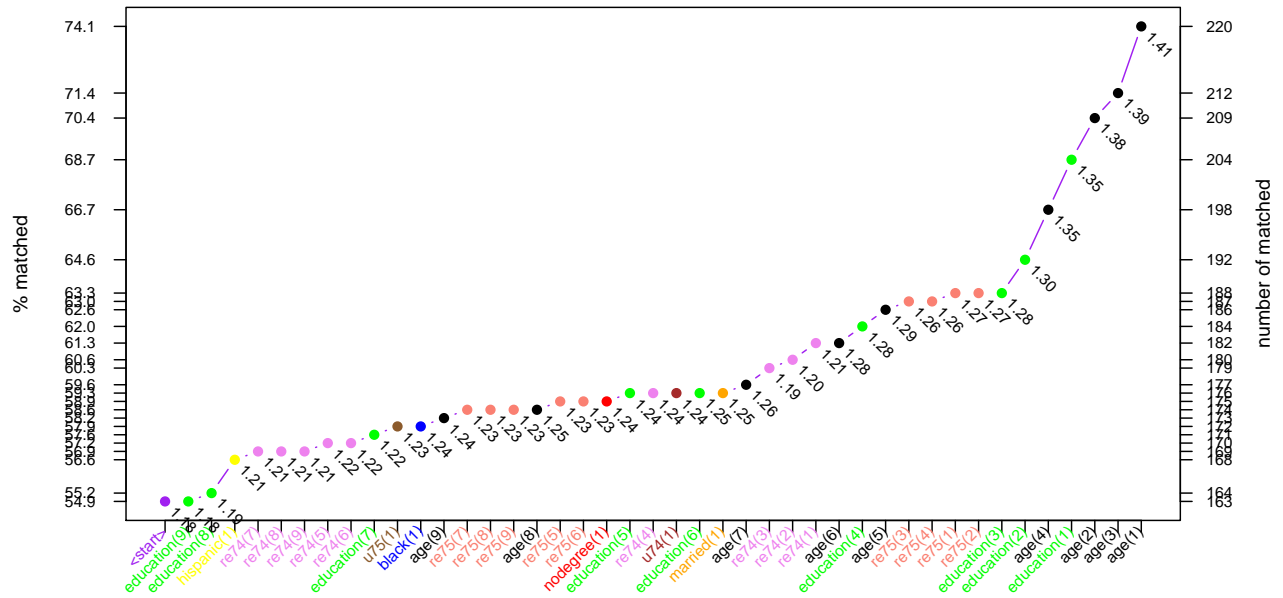


Figure 1: Example of graphical output of `relax.cem`.

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

Executing 42 different relaxations

```
.....[20%]....[40%].....[60%]....[80%]....[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 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 method `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 `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 `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

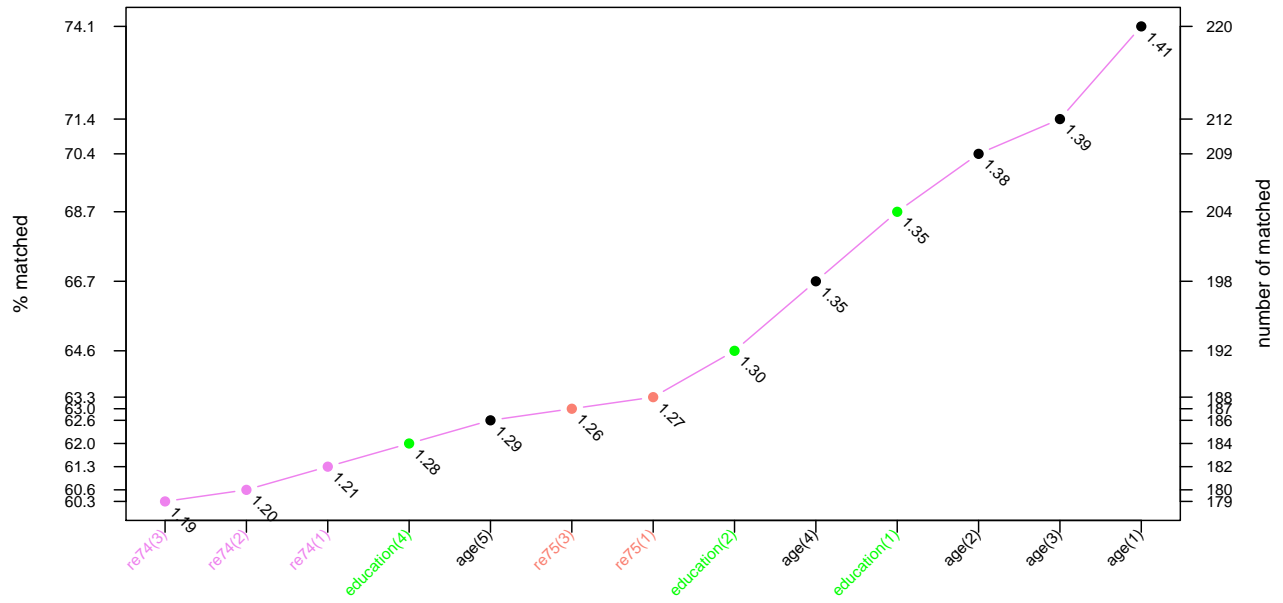


Figure 2: Example of reduced graphical output of `relax.cem`.

treated units has been matched, or a combination of these. For more information refer to the man page of the function `relax.plot` which can be called directly instead of `plot`. Here is one example of use of `plot` in which we specify that only solutions with at least 60% of the treated units are matched and duplicated solutions are removed. The output can be seen in Figure 2

```
> plot(tab, group = "1", perc = 0.6, unique = TRUE)
```

### 3.5 Restricting the matching solution to a $k$ -to- $k$ match

CEM usually returns strata containing a different number of treated and control units along with weights to be used in the subsequent analysis. Although this is the best option, it might happen that the user need a  $k$ -to- $k$  solution. This is obtained by pruning observations from each strata in order to have the same number of treated and control units. Because, up to coarsening, the observation in a stratum are not distinguishable to CEM itself, the user can specify a proper distance to prune locally (in each strata) the CEM solution. The function `k2k` allow for this. The user may choose between several distances specifying the `method` argument. In particular, the current choices are: ‘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. More methods or distances may be added in future releases of the package. Here follows an example of use

```
> mat <- cem(treatment = "treated", data = LL, drop = "re78")
> mat
```

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

```
> mat$k2k
```

```
[1] FALSE
```

and now pruning using the euclidean distance within CEM strata

```
> mat2 <- k2k(mat, LL, "euclidean", 1)
> mat2
```

```
      G0  G1
All      425 297
Matched   139 139
Unmatched 286 158
```

```
> mat2$k2k
```

```
[1] TRUE
```

### 3.6 An example of estimation of SATT 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    4.686e+03 550.9626
Std. Error    3.980e+02 611.6134
t value       1.178e+01  0.9008
Pr(>|t|)      1.578e-27  0.3682
```

The SATT estimate is the coefficient of the `treated` variable, in our case 550.96. 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`. Via the `formula` interface, it is also possible to specify more flexible models to span the remaining imbalance due to not completely balanced covariates. Here follows an example

```
> est2 <- att(mat, re78 ~ treated + re74 + re75, data = LL)
> est2
```

	(Intercept)	treated	re74	re75
Estimate	4.257e+03	551.9584	0.4436	-0.1800
Std. Error	4.338e+02	607.6120	0.3110	0.3596
t value	9.813e+00	0.9084	1.4265	-0.5005
Pr(> t )	2.045e-20	0.3642	0.1545	0.6170

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`.

### 3.7 Working with missing data

The function `cem` allows to work with missing data directly but it is not a method intended for data imputation. Missing data are treated by `cem` as distinct values of variables in the following way: (variable by variable) after the coarsening on the non-missing values took place, NA are collected in a separate stratum. In the next example, we copy of the LL data onto LL2 and generate randomly missing data in the earnings variables `re74`

```
> set.seed(123)
> LL2 <- LL
> n <- dim(LL)[1]
> LL2$re74[sample(1:n, 50)] <- NA
> summary(LL2$re74)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
0	0	734	3640	5160	39600	50

Now we run `cem` on the LL2 data with missing values and on a copy of the same data LL4 where the rows of LL2 containing missing values are omitted. For comparability, we use the same cutpoints we used in Section 3.2 on the complete LL data. The cutpoints are contained in `mat$breaks`

```
> mat3 <- cem("treated", LL2, cutpoints = mat$breaks, drop = "re78")
> mat3
```

	G0	G1
All	425	297
Matched	206	152
Unmatched	219	145

and we compare the above with the solution obtained by dropping the observations with missing data

```
> LL4 <- na.omit(LL2)
> mat4 <- cem("treated", LL4, cutpoints = mat$breaks, drop = "re78")
> mat4
```

```
      G0  G1
All      396 276
Matched   203 148
Unmatched 193 128
```

and, as expected, the two solutions differ but not that much. The gain (in terms of number of matched units) decreases as the number of covariates increases. As remarked, `cem` does not do any missing data imputation, so at the analysis step the user might want to use other imputation techniques conditionally on the CEM solution. But the suggested approach is to do multiple imputation first and applying `cem` after on the multiply imputed data sets as the next Section explains.

### 3.8 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. :0.000	Min. :17.0	Min. : 3.0	Min. : 0.000
1st Qu.:0.000	1st Qu.:19.0	1st Qu.: 9.0	1st Qu.: 1.000
Median :0.000	Median :23.0	Median :10.0	Median : 1.000
Mean :0.411	Mean :24.5	Mean :10.3	Mean : 0.804
3rd Qu.:1.000	3rd Qu.:27.0	3rd Qu.:11.0	3rd Qu.: 1.000
Max. :1.000	Max. :54.0	Max. :16.0	Max. : 1.000
	NA's :18.0	NA's :21.0	NA's :19.000

married	nodegree	re74	re75
Min. : 0.000	Min. : 0.000	Min. : 0	Min. : 0
1st Qu.: 0.000	1st Qu.: 1.000	1st Qu.: 0	1st Qu.: 0
Median : 0.000	Median : 1.000	Median : 824	Median : 935



Mean	: 0.159	Mean	: 0.778	Mean	: 3687	Mean	: 3057
3rd Qu.:	0.000	3rd Qu.:	1.000	3rd Qu.:	5272	3rd Qu.:	4064
Max.	: 1.000	Max.	: 1.000	Max.	: 39571	Max.	: 37432
NA's	: 25.000	NA's	: 20.000	NA's	: 20	NA's	: 16
	re78		hispanic		u74		u75
Min.	: 0	Min.	: 0.000	Min.	: 0.000	Min.	: 0.000
1st Qu.:	0	1st Qu.:	0.000	1st Qu.:	0.000	1st Qu.:	0.000
Median	: 4008	Median	: 0.000	Median	: 0.000	Median	: 0.000
Mean	: 5504	Mean	: 0.105	Mean	: 0.454	Mean	: 0.399
3rd Qu.:	8782	3rd Qu.:	0.000	3rd Qu.:	1.000	3rd Qu.:	1.000
Max.	: 60308	Max.	: 1.000	Max.	: 1.000	Max.	: 1.000
NA's	: 18	NA's	: 19.000	NA's	: 17.000	NA's	: 23.000

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 `cem` function in the argument `datalist` as a list of data frame and `cem` returns a list of `cem.match` solutions. Each matching solution is called `matchX` where `X` varies from 1 to the number of multiply imputed data sets.

```
> mat1 <- cem("treated", datalist = imputed, drop = "re78")
> mat1
```

```
$match1
      G0  G1
All      425 297
Matched   202 142
Unmatched 223 155
```

```
$match2
      G0  G1
All      425 297
Matched   202 143
Unmatched 223 154
```

```
$match3
      G0  G1
All      425 297
Matched   203 146
Unmatched 222 151
```

```
$match4
      G0  G1
All      425 297
Matched   217 158
Unmatched 208 139
```

```
$match5
      G0  G1
All      425 297
Matched   199 151
Unmatched 226 146
```

```
attr("class")
[1] "multicem" "list"
```

In the above example, `cem` has no clue about which rows were originally missing, so each matching solution is different. Mixing together the output of each single match may be

risky, thus `cem` allows to specify also the original data set with missing data. In this case, `cem` produces the same solutions assigning each multiply imputed observation in the strata where it fall most frequently. This is the correct way to use `cem` in the presence of multiple imputation and allow for correct combination of SATT estimate in each model. We give an example now

```
> mat2 <- cem("treated", datalist = imputed, drop = "re78", data = LL1)
> mat2
```

```
$match1
```

	G0	G1
All	425	297
Matched	203	147
Unmatched	222	150

```
$match2
```

	G0	G1
All	425	297
Matched	203	147
Unmatched	222	150

```
$match3
```

	G0	G1
All	425	297
Matched	203	147
Unmatched	222	150

```
$match4
```

	G0	G1
All	425	297
Matched	203	147
Unmatched	222	150

```
$match5
```

	G0	G1
All	425	297
Matched	203	147
Unmatched	222	150

```
attr("class")
[1] "multicem" "list"
```

Now we can estimate the SATT 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 are 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      4527.5      729.1
Std. Error    426.2      658.4
```

In the case of multiple data sets, the output of `att` also contains a list of single SATT estimates for each multiply imputed data. Typing `str(out)` reveals the structure of the output. 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.

## 4 Reference to CEM's Functions

## 4.1 cem: Coarsened Exact Matching

### Description

Implementation of Coarsened Exact Matching

### Usage

```
cem(treatment=NULL, data = NULL, datalist=NULL, cutpoints = NULL, drop=NULL,  
    eval.imbalance = FALSE, k2k=FALSE, method=NULL, mpower=2,  
    L1.breaks = NULL, verbose = 0)
```

### Arguments

<code>treatment</code>	character, name of the treatment variable
<code>data</code>	a data.frame
<code>datalist</code>	a list of imputed data.frame's
<code>cutpoints</code>	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 construction. See Details.
<code>drop</code>	a vector of variable names in the data frame to ignore during matching
<code>eval.imbalance</code>	Boolean. See Details.
<code>k2k</code>	boolean, return k-to-k matching? Default = <b>FALSE</b>
<code>method</code>	distance method to use in <b>k2k</b> matching. See Details.
<code>mpower</code>	power of the Minkowski distance. See Details.
<code>L1.breaks</code>	list of cutpoints for the calculation of the L1 measure.
<code>verbose</code>	controls level of verbosity. Default=0.

### Details

When specifying cutpoints, several automatic methods can be chosen among “**sturges**” (Sturges’ 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` (the default), `cem$imbalance` contains the imbalance measure by absolute difference in means for numerical variables and chi-square distance for categorical variables. If **FALSE** then `cem$imbalance` is set to **NULL**.

If `L1.breaks` is missing, the default rule to calculate cutpoints is the Scott’s rule.

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.

If argument `data` is non `NULL` and `datalist` is `NULL` CEM is applied to the single data set in `data`.

Argument `datalist` is a list of (multiply imputed) data frames. If `data` is `NULL`, 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` and `datalist` are both non `NULL`, 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)

## Value

Returns an object of class `cem.match` if only `data` is not `NULL` or an object of class `multicem`, which is a list of object of class `cem.match`. A `cem.match` object is a list with the following slots:

<code>call</code>	the call
<code>strata</code>	vector of stratum number in which each observation belongs, NA if the observation has not been matched
<code>n.strata</code>	number of strata generated
<code>vars</code>	report variables names used for the match
<code>drop</code>	variables removed from the match
<code>breaks</code>	named list of cutpoints, eventually <code>NULL</code>
<code>treatment</code>	name of the treatment variable
<code>groups</code>	factor, each observation belong to one group generated by the treatment variable
<code>n.groups</code>	number of groups identified by the treatment variable
<code>group.idx</code>	named list, index of observations belonging to each group

<code>group.len</code>	sizes of groups
<code>tab</code>	summary table of matched by group
<code>imbalance</code>	NULL or a vector of imbalances. See Details.

## 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>

## Examples

```
data(LL)

mybr = list(re74=seq(0, 40000, by=5000),
  re75 = seq(0, 40000, by=5000),
  age = seq(15, 55, by=5),
  education = 3:16)

todrop <- c("treated","re78")

eval.match(LL$treated, LL, drop=todrop, breaks=mybr)

# cem match: automatic bin choice
mat <- cem(treatment="treated", data=LL, drop="re78")
mat$tab
# imbalance
eval.match(LL$treated, LL, drop=todrop, breaks=mybr,weights=mat$w)

# 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$tab
#imbalance
eval.match(LL$treated, LL, drop=todrop, breaks=mybr,weights=mat$w)

# cem match: user choiced coarsening, k-to-k matching
mat <- cem(treatment="treated",data=LL, drop="re78",cutpoints=mycp,k2k=TRUE)
mat$tab
```

```

#imbalance
eval.match(LL$treated, LL, drop=todrop, breaks=mybr,weights=mat$w)

# mahalanobis matching
require(MatchIt)
mah <- matchit(treated~age+education+re74+re75+black+hispanic+nodegree+married+u74+u75,
  distance="mahalanobis", data=LL)
mah
#imbalance
eval.match(LL$treated, LL, drop=todrop, breaks=mybr,weights=mah$weights)

# Multiply Imputed data
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 <- cem("treated", datalist=imputed, drop="re78")
#str(mat1, max.lev=1)
mat1$match1$stab
mat1$match2$stab

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

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

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

```



## 4.2 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

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

### Details

Argument `data` must be a single data frame or a list of (multiply 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`.

### 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>

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

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

str(out)
```

## 4.3 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.

## 4.4 `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, drop=NULL, breaks = NULL, weights)
```

### Arguments

<code>group</code>	the group variable
<code>data</code>	the data
<code>drop</code>	a vector of variable names in the data frame to ignore
<code>breaks</code>	a list of vectors of cutpoints used to calculate L1 measure. See Details.
<code>weights</code>	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. Column `type` reports either (`diff`) or (`Chi2`) according to the type of statistic being 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.

This function also calculate the global L1 imbalance measure. If `breaks` is missing, the default rule to calculate cutpoints is the Scott's rule. See `L1.meas` help page for details.

### Value

An object of class `eval.match` which is a list with the following two elements

<code>tab</code>	Table of imbalance measures
<code>L1</code>	The global L1 measure of imbalance

### 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>

## Examples

```
data(LL)

mybr = list(re74=seq(0, 40000, by=5000),
  re75 = seq(0, 40000, by=5000),
  age = seq(15, 55, by=5),
  education = 3:16)

todrop <- c("treated","re78")

L1.meas(LL$treated, LL, drop=todrop, breaks=mybr)
eval.match(LL$treated, LL, drop=todrop, breaks=mybr)

# cem match: automatic bin choice
mat <- cem(treatment="treated", data=LL, drop="re78")
mat$tab
# imbalance
eval.match(LL$treated, LL, drop=todrop, breaks=mybr, weights=mat$w)
```

## 4.5 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

<code>obj</code>	an object as output from <code>cem</code>
<code>data</code>	the original <code>data.frame</code> used by <code>cem</code>
<code>method</code>	distance method to use in k2k matching. See Details.
<code>mpower</code>	power of the Minkowski distance. See Details.
<code>verbose</code>	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

<code>obj</code>	an object of class <code>cem.match</code>
------------------	---

### 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>

## 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)

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

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

## 4.6 L1.meas: Evaluates L1 distance between multidimensional histograms

### Description

Evaluates L1 distance between multidimensional histograms

### Usage

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

### Arguments

<code>group</code>	the group variable
<code>data</code>	the data
<code>drop</code>	a vector of variable names in the data frame to ignore
<code>breaks</code>	a list of vectors of cutpoints; if not specified, automatic choice will be made
<code>weights</code>	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.

If **breaks** is missing, the default rule to calculate cutpoints is the Scott's rule.

### Value

An object of class **L1.meas** which is a list with the following fields

<code>breaks</code>	A list of cutpoints used to calculate the L1 measure
<code>value</code>	The numerical value of the L1 measure

### 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>



## Examples

```
data(LL)
L1.meas(LL$treated,LL, drop=c("treated","re78"))
```

## 4.7 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.

## 4.8 `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

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

Calling directly `plot` on the output of `cem.relax` has the same effect of calling directly `relax.plot`.

## Value

`tab`                    an invisible object containing the tabs

## 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)
```

```
mybr = list(re74=seq(0, 40000, by=5000),
  re75 = seq(0, 40000, by=5000),
  age = seq(15, 55, by=5),
  education = 3:16)
```

```
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")
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=0)
# relax.plot(tab4)
# relax.plot(tab4, unique=TRUE)
# relax.plot(tab4, perc=0.7)
```

## 4.9 `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

<code>obj</code>	and object of class <code>cem</code>
<code>data</code>	the original data
<code>shifts</code>	a vector of proportions of shifts
<code>verbose</code>	controls the level of verbosity
<code>plot</code>	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

<code>tab</code>	an invisible object containing a new <code>cem</code> 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

- 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>.
- Ho, Daniel, Kosuke Imai, Gary King and Elizabeth Stuart. 2007. “Matching as Nonparametric Preprocessing for Reducing Model Dependence in Parametric Causal Inference.” *Political Analysis* 15:199–236. <http://gking.harvard.edu/files/abs/matchp-abs.shtml>.
- 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>.
- King, Gary and Langche Zeng. 2006. “The Dangers of Extreme Counterfactuals.” *Political Analysis* 14(2):131–159. <http://gking.harvard.edu/files/abs/counterft-abs.shtml>.
- Lalonde, Robert. 1986. “Evaluating the Econometric Evaluations of Training Programs.” *American Economic Review* 76:604–620.