cem: Coarsened Exact Matching

Version 1.0.29
August 16, 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. 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 monotonoic 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 and other 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, Linux, 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 seemlessly 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 verison 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 Evaluation and 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.[1]

We compute $\mathcal{L}_1$ statistic, as well as several unidimensional measures of imbalance via our `imbalance` function. In our running example:

```
> imbalance(group = LL$treated, data = LL[vars])

Multivariate Imbalance Measure L1=1.196530

Univariate Imbalance Measures

            statistic   type       L1 min 25%    50%   75%     max
age         1.792e-01 (diff) 0.184702   0   1   0.00  -1.0    -6.0
education   1.922e-01 (diff) 0.153789   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) 0.071792   0   0  69.73 584.9 -2139.0
re75        3.942e+01 (diff) 0.114874   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
```

Only the overall $\mathcal{L}_1$ statistic measure includes imbalance with respect to the full joint distribution, including all interactions, of the covariates; in the case of our example, $\mathcal{L}_1$ =1.197. The unidimensional measures in the table are all computed for each variable separately.

The first column in the table of unidimensional measures, labeled `statistic`, 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 continuous or dichotomous, and so `(diff)` appears in all rows. The second column, labeled `L1`, reports the $\mathcal{L}_1^j$ measure, which is $\mathcal{L}_1$ computed for each variable separately (which of course interactions). 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

[1]Of course, as with 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.

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, since even if the marginal distribution of every variable is perfectly balanced, the joint distribution can still be highly imbalanced.

For convenience, an alternative use of `imbalance` allows you to drop some variables within the function:

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

[1] "treated" "re78"

> imbalance(group = LL$treated, data = LL, drop = todrop)

Multivariate Imbalance Measure L1=1.196530

Univariate Imbalance Measures

            statistic   type       L1 min 25%    50%   75%     max
age         1.792e-01 (diff) 0.184702   0   1   0.00  -1.0    -6.0
education   1.922e-01 (diff) 0.153789   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) 0.071792   0   0  69.73 584.9 -2139.0
re75        3.942e+01 (diff) 0.114874   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
```

3.2 Coarsened Exact Matching

We now apply the coarsened exact matching algorithm by calling the function `cem`. The CEM algorithm performs exact matching on coarsened data to determine matches and then passes on the uncoarsened data from observations that were matched to estimate the causal effect. Exact matching works by first sorting all the observations into strata, each of which has identical values for all the coarsened pre-treatment covariates, and then discarding all observations within any stratum that does not have at least one observation for each unique value of the treatment variable.

To run this algorithm, we must choose a type of coarsening for each covariate. We show how this is done this via a fully automated procedures in Section 3.2.1. Then we show how to use explicit prior knowledge to choose the coarsening in Section 3.2.2, which is normally preferable when feasible.

In CEM, 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 to units within each strata to ensure that each has at least one observation assigned each value of the treated variable.

3.2.1 Automated Coarsening

In our running example we have a dichotomous treatment variable. In the following code, we match on all variables but `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 output object `mat` contains useful information about the match, including a (small) table about the number of observations in total, matched, and unmatched, as well as the results of a call to the `imbalance` function for information about the quality of the matched data (unless `eval.imbalance` is set to `FALSE`). Since `cem` bounds the imbalance ex ante, the most important information in `mat` is the number of observations matched. But the results also give the imbalance in the matched data using the same measures as that in the original data described in Section 3.1. Thus,

```
> mat

           G0  G1
All       425 297
Matched   222 163
Unmatched 203 134


Multivariate Imbalance Measure L1=0.804709

Univariate Imbalance Measures

           statistic   type        L1 min 25%   50%    75%    max
age        1.862e-01 (diff) 1.820e-01   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) 8.006e-02   0   0   0.0 -70.86  416.4
re75       1.221e+01 (diff) 1.427e-01   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
```

We can see from these results the number of observations matched and thus retained, as well as those which were pruned because they were not comparable. By comparing the imbalance results to the original imbalance table given in the previous section, we can see that a good match can produce a substantial reduction in imbalance, not only in the means, but also in the marginal and joint distributions of the data.

The function `cem` also generates weights for use in the evaluation of imbalance measures and estimates of the causal effect (stored in `mat$w`).

3.2.2 Coarsening by Explicit User Choice

The power and simplicity of CEM comes from choosing the coarsening yourself rather than using the automated algorithm as in the previous section. Choosing the coarsening enables you to set the maximum level of imbalance ex ante, which is a direct function of the coarsening you choose. As importantly, the coarsening is a fundamentally substantive act, almost synonymous with the measurement of the original variables. In other words, if you know something about the data you are analyzing, you have enough information to choose the the coarsening. (And if you don't know something about the data, why might ask why you are analyzing it in the first place!)

In general, we want to set the coarsening for each variable so that substantively indistinguishable values are grouped and assigned the same numerical value. Groups may be of different sizes if appropriate. 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
Graduate school	>16

Using these natural breaks in the data to create the coarsening is generally a good approach and certainly better than using fixed bin sizes that disregard these meaningful breaks. This information is represented in the `cutpoints` option of `cem`. Because in our data, no respondents fall in 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 define the cutpoints as:

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

and run `cem` adding only the `cutpoints` option, leaving the rest unchanged:

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

           G0  G1
All       425 297
Matched   254 186
Unmatched 171 111


Multivariate Imbalance Measure L1=0.889174

Univariate Imbalance Measures

           statistic   type        L1 min 25%   50%    75%    max
age        1.443e-01 (diff) 1.685e-01   0   0   0.0   0.00    1.0
education -1.437e-02 (diff) 4.950e-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) 6.567e-02   0   0 369.3 233.61  416.4
re75       4.501e+01 (diff) 9.837e-02   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
```

As we can see, this matching solution differs from that resulting from our automated approach in the previous section. For comparison, the automatic cutpoints produced by `cem` are stored in the output object in slot `breaks`. So, for example, our automated coarsening produced:

```
> 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
```

whereas we can recover our personal choice of cutpoints as

```
> mat1$breaks$education

[1]  0.0  6.5  8.5 12.5 17.0
```

3.3 Progressive coarsening

Although the maximum imbalance is fixed ex ante by the user's coarsening choices, the number of observations matched is determined as a consequence of the matching procedure. If

you are dissatisfied with the number of observations available after matching, and you feel that it is substantively appropriate to coarsen further, then just increase the coarsening (by using fewer cutpoints). The result will be additional matches and of course a concommitant increase in the maximum possible imbalance between the treated and control groups. This is easy with CEM because, unlike most other methods, CEM is a monotonic imbalance bounding (MIB) method, which means that increasing the imbalance on one variable (through a change in coarsening) will not change the maximum imbalance on any other variable. MIB thus enables you to tinker with the solution one variable at a time to quickly produce a satisfactory result, if one is feasible.

If, however, you feel that additional coarsening is not appropriate, than too few obserations may indicate that your data contains insufficient information to estimate the causal effects of interest without model dependence; in that situation, you either give up or will have to attempt adjusting for the pre-treatment covariates via modeling assumptions.

Suppose, instead, that you are unsure whether to coarsen further or how much to coarsen, and are willing to entertain alternative matching solutions. We offer here an automated way to compute these solutions. The idea is to relax the initial `cem` solution selectively and automatically, to prune equivalent solutions, and to present them in a convenient manner so that users can ascertain where the difficulties in matching in these data can be found and what choices would produce which outcomes in terms of the numbers of observations matched.

We start by illustrating what happens when we relax a CEM solution "by hand". The following three runs show the effect on the matching solution (in terms of the number of observations and imbalance) when the coarsening for one variable (`age`) is relaxed from 10 to 6 to 3 bins. As can be seen, fewer cutpoints (which means larger bins) produces more matched units and high maximum (and in this case actual) imbalance:

```
> cem("treated", LL, cutpoints = list(age = 10), drop = "re78")

           G0  G1
All       425 297
Matched   228 161
Unmatched 197 136


Multivariate Imbalance Measure L1=0.870948

Univariate Imbalance Measures

           statistic   type        L1 min 25%   50%    75%    max
age        1.494e-01 (diff) 1.161e-01   0   1   0.0    0.0    1.0
education  1.097e-02 (diff) 2.195e-02   0   0   0.0    0.0    0.0
black      0.000e+00 (diff) 1.388e-17   0   0   0.0    0.0    0.0
married    6.939e-18 (diff) 1.180e-16   0   0   0.0    0.0    0.0
```

```
nodegree   0.000e+00 (diff) 0.000e+00   0   0   0.0    0.0    0.0
re74      -5.508e+01 (diff) 1.192e-01   0   0   0.0 -350.4  416.4
re75      -3.020e+01 (diff) 9.398e-02   0   0 234.5   51.9 -852.3
hispanic   6.939e-18 (diff) 6.939e-18   0   0   0.0    0.0    0.0
u74        1.110e-16 (diff) 1.110e-16   0   0   0.0    0.0    0.0
u75        0.000e+00 (diff) 0.000e+00   0   0   0.0    0.0    0.0
```

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

```
           G0  G1
All       425 297
Matched   261 186
Unmatched 164 111


Multivariate Imbalance Measure L1=0.905692

Univariate Imbalance Measures

           statistic   type        L1 min 25%   50%    75%    max
age        1.743e-01 (diff) 1.481e-01   0   0  0.00    0.0    7.0
education -1.776e-15 (diff) 1.075e-02  -1   0  0.00    0.0    0.0
black      0.000e+00 (diff) 0.000e+00   0   0  0.00    0.0    0.0
married    0.000e+00 (diff) 0.000e+00   0   0  0.00    0.0    0.0
nodegree   1.110e-16 (diff) 1.110e-16   0   0  0.00    0.0    0.0
re74      -8.674e+00 (diff) 8.672e-02   0   0  0.00 -106.9  416.4
re75      -6.146e+01 (diff) 7.595e-02   0   0 33.77 -138.5 -852.3
hispanic   0.000e+00 (diff) 0.000e+00   0   0  0.00    0.0    0.0
u74        0.000e+00 (diff) 0.000e+00   0   0  0.00    0.0    0.0
u75        0.000e+00 (diff) 1.110e-16   0   0  0.00    0.0    0.0
```

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

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


Multivariate Imbalance Measure L1=1.124454

Univariate Imbalance Measures
```

```
           statistic   type        L1 min 25%   50%     75%    max
age        5.836e-01 (diff) 1.559e-01   0   0   0.0    1.00    7.0
education -5.229e-03 (diff) 8.886e-04  -1   0   0.0    0.00    0.0
black      0.000e+00 (diff) 0.000e+00   0   0   0.0    0.00    0.0
married    0.000e+00 (diff) 0.000e+00   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      -1.661e+01 (diff) 1.193e-01   0   0 143.6  -14.51  636.8
re75       1.882e+00 (diff) 1.255e-01   0   0 129.0 -130.62 -852.3
hispanic   0.000e+00 (diff) 0.000e+00   0   0   0.0    0.00    0.0
u74       -5.551e-17 (diff) 1.665e-16   0   0   0.0    0.00    0.0
u75        0.000e+00 (diff) 0.000e+00   0   0   0.0    0.00    0.0
```

We automate this *progressive coarsening* procedure here in the `relax.cem` function. This function starts with the output of `cem` and relaxes variables one (`depth=1`), two (`depth=2`), or three (`depth=3`) at a time, while optionally keeping unchanged a chosen subset of the variables which we know well or have important effects on the outcome (`fixed`). The function also allows one to specify the minimal number of breaks of each variable (the default limit being 1). We begin with this example:

```
> tab <- relax.cem(mat, LL, depth = 1, 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, one per group (i.e. treated and control). Each row of the tables contain 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, and so we provide a method `plot` to view these tables more conveniently. The output of `plot(tab)` is plotted 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 (color-coded by variable). On the y-axis on the right is the absolute number of treated units matched, while the left side y-axis reports the same number in percentages. The numbers below the dots in the graphs represent the $\mathcal{L}_1$ measure corresponding to that matching solution. This graph also gives a feeling of the MIB 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 treated units, have been matched, or a combination of these. For more information refer to the reference manual for the function `relax.plot` which can be called directly instead of plot.

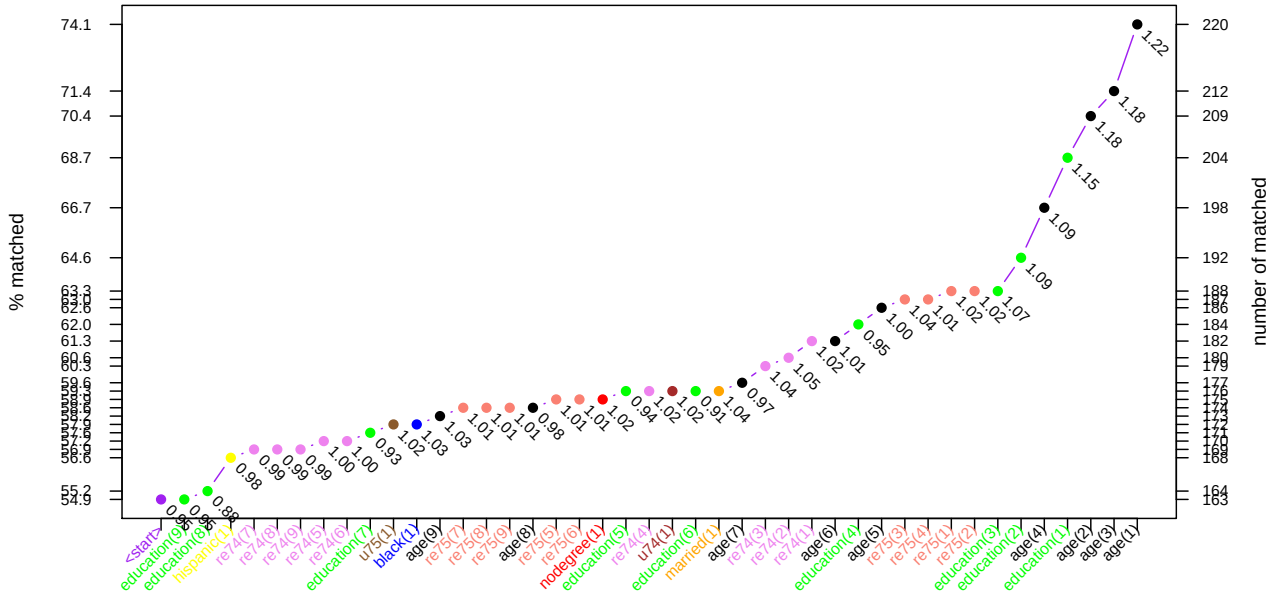

Figure 1: Example of the graphical output of `relax.cem`.

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)
```

3.4 Restricting the matching solution to a k-to-k match

By default, CEM uses maximal information, resulting in strata that may include different numbers of treated and control units. To compensate for the differential strata sizes, `cem` also returns weights to be used in subsequent analyses. Although this is generally the best option, a user with enough data may opt for a k-to-k solution to avoid the slight inconvenience of needing to use weights.

The function `k2k` accomplishes this by pruning observations from a `cem` solution within each stratum until the solution contains the same number of treated and control units in all strata. Pruning occurs within a stratum (for which observations are indistuinguishable to cem proper) by using nearest neighbor selection using a distance function specified by the user (including `euclidean`, `maximum`, `manhattan`, `canberra`, `binary`, or `minkowski`). By default `method` is set to `NULL`, which means random matching inside `cem` strata, an option that may reduce the chance for bias. (For the Minkowski distance the power can be specified

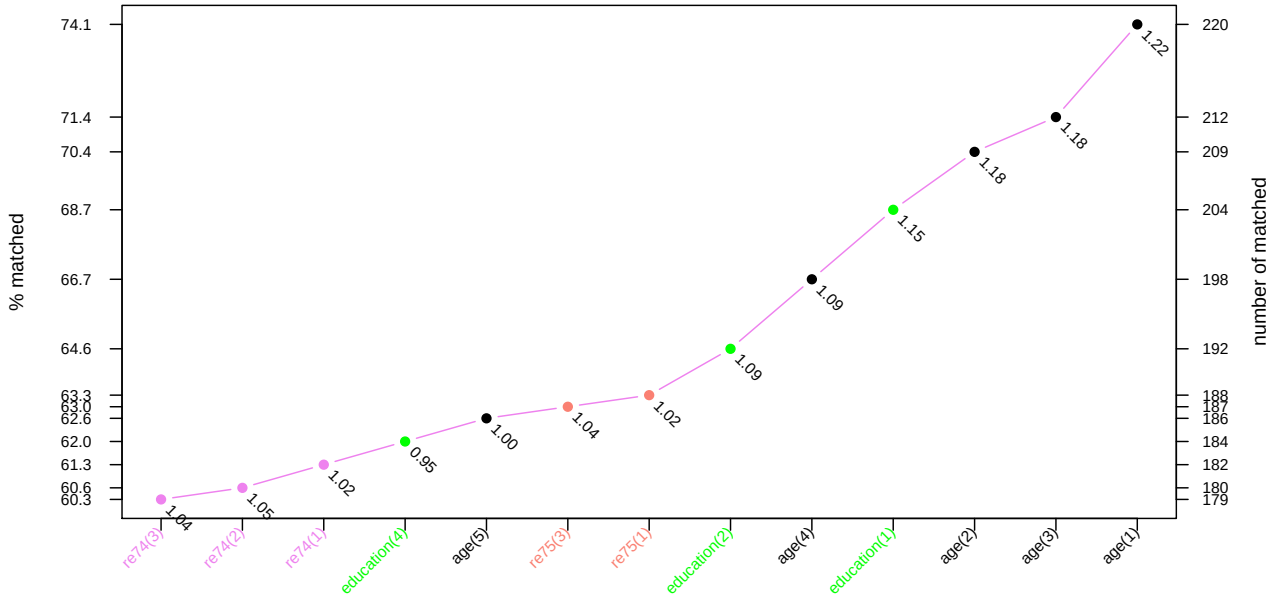

Figure 2: Example of reduced graphical output of `relax.cem`.

via the argument `mpower`. For more information on `method != NULL`, refer to `dist` help page.)

Here is an example of this approach. First, by running `cem`:

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

           G0  G1
All       425 297
Matched   222 163
Unmatched 203 134


Multivariate Imbalance Measure L1=0.804709

Univariate Imbalance Measures

           statistic   type        L1 min 25%   50%    75%    max
age        1.862e-01 (diff) 1.820e-01   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) 8.006e-02   0   0   0.0 -70.86  416.4
re75       1.221e+01 (diff) 1.427e-01   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
```

```
> mat$k2k
```

```
[1] FALSE
```

and now pruning to a k-to-k solution, 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


Multivariate Imbalance Measure L1=0.804709

Univariate Imbalance Measures

           statistic   type        L1 min 25%   50%    75%    max
age        1.862e-01 (diff) 1.820e-01   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) 8.006e-02   0   0   0.0 -70.86  416.4
re75       1.221e+01 (diff) 1.427e-01   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
```

```
> mat2$k2k
```

```
[1] TRUE
```

Alternatively, we can produce the same result in one step by adding the `k2k=TRUE` option to the original `cem` call.

3.5 Estimating the Causal Effect from `cem` output

Using the output from `cem`, we can estimate SATT via the `att` function. The simplest approach requires a weighted difference in means (unless `k2k` was used, in which case no weights are required). For convenience, we compute this as a regression of the outcome variable on a constant and the treatment variable,

```
> 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
```

where the SATT estimate is the coefficient on the `treated` variable, in our case 550.96. The function `att` allows for R's standard `formula` interface and, by default, uses `lm` to estimate the model using the weights produced by `cem`.

If exact matching (i.e., without coarsening) was chosen this procedure is appropriate as is. In other situations, with some coarsening, some imbalance remains in the matched data. The remaining imbalance is strictly bounded by the level of coarsening, which can be seen by any remaining variation within the coarsened bins. Thus, a reasonable approach in this common situation is to attempt to adjust for the remaining imbalance via a statistical model. (Modeling assumptions for models applied to the matched data are much less consequential than they would otherwise be because CEM is known to strictly bound the level of model dependence.) To apply a statistical model to control for the remaining imbalance, we use the `formula` interface in `att`. For 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
```

The user can also specify `glm` modeling in the case of binary, count, or other noncontinuous outcome variables. For more information, see the reference manual entry for `att`.

3.6 Matching and Missing Data

Almost all previous methods of matching assume the absence of any missing values. In contrast, CEM offers two valid approaches to dealing with missing values (item nonresponse) — either as values to match directly, as we describe in Section 3.6.1, or by a special procedure to deal with multiply imputed data, as in Section 3.7.

3.6.1 Matching on Missingness

[*** Need to change this example to show how we can coarsen NA with other observed values. At present, the following example is accurate but pretty artificial. ***]

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")

Missing values exist in the data!

> mat3

           G0  G1
All       425 297
Matched   206 152
Unmatched 219 145


Multivariate Imbalance Measure L1=0.901912

Univariate Imbalance Measures

           statistic   type        L1 min 25%    50%    75%   max
age       -1.643e-01 (diff) 1.649e-01   0  -1   -1.0    0.0    -2
education -1.096e-02 (diff) 2.193e-02   0   0    0.0    0.0     0
black     -1.110e-16 (diff) 1.110e-16   0   0    0.0    0.0     0
married   -1.388e-17 (diff) 1.388e-17   0   0    0.0    0.0     0
nodegree   0.000e+00 (diff) 0.000e+00   0   0    0.0    0.0     0
re74       2.540e+01 (diff) 1.242e-01   0   0    0.0  140.2 -1326
```

```
re75      -4.153e+01 (diff) 1.488e-01   0   0 -325.3 -169.4 -1200
hispanic   0.000e+00 (diff) 0.000e+00   0   0    0.0    0.0     0
u74        0.000e+00 (diff) 0.000e+00   0   0    0.0    0.0     0
u75       -5.551e-17 (diff) 5.551e-17   0   0    0.0    0.0     0
```

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


Multivariate Imbalance Measure L1=0.912774

Univariate Imbalance Measures

           statistic   type        L1 min 25%    50%    75%   max
age       -1.823e-01 (diff) 1.693e-01   0   0   -1.0    0.0    -2
education -1.126e-02 (diff) 2.252e-02   0   0    0.0    0.0     0
black     -1.110e-16 (diff) 1.110e-16   0   0    0.0    0.0     0
married    0.000e+00 (diff) 1.110e-16   0   0    0.0    0.0     0
nodegree  -1.110e-16 (diff) 1.110e-16   0   0    0.0    0.0     0
re74       2.540e+01 (diff) 1.276e-01   0   0    0.0  311.5 -1326
re75      -4.280e+01 (diff) 1.394e-01   0   0 -236.9 -336.1 -1200
hispanic   6.939e-18 (diff) 6.939e-18   0   0    0.0    0.0     0
u74        0.000e+00 (diff) 0.000e+00   0   0    0.0    0.0     0
u75        0.000e+00 (diff) 0.000e+00   0   0    0.0    0.0     0
```

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.

3.7 Matching Multiply Imputed Data

Consider a data set to be matched, some of which is missing. One approach to analyzing data with missing values is *multiple imputation*, which involves creating m (usually about

$m = 5$) data sets, each of which is the same as the original except that the missing values have been imputed in each. Uncertainty in the values of the missing cells is represented by variation in the imputations across the different imputed data sets (King et al., 2001).

As an example we create a version of the Lalonde data with missing values which we create. (Normally of course one would not add missing values!). Thus,

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

Now `LL1` contains 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
```

Now we use `Amelia` package (Honaker, King and Blackwell, 2006) to create multiply imputed data sets:

```
> 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 a list of 5 multiply imputed versions of `LL1`. We pass this list to the `cem` function in the argument `datalist` and `cem` produces a set of multiply imputed solutions, as usual with the original uncoarsened values of the variables, but now assigning each multiply imputed observation to the strata where it falls most frequently. The output of `cem` is a list of `cem.match` solutions (named `match1`, `match2`,..., `match5`). (Be sure to also name the original data frame in option `data` or `cem` will merely run the basic algorithm five separate times on each of the input data sets, a procedure that can be useful for batch processing of data to be matched, but is not recommended for multiply imputed data sets since the strata will not be the same across the data sets.) For example:

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

           G0  G1
All       425 297
Matched   203 147
Unmatched 222 150


Multivariate Imbalance Measure L1=1.047998
```

```
Univariate Imbalance Measures

           statistic   type      L1     min 25%   50%    75%    max
age         0.003073 (diff) 0.11111    0.00   0   1.0    0.0    1.0
education   0.001755 (diff) 0.03175    0.00   0   0.0    0.0    0.0
black      -0.005442 (diff) 0.05442    0.00   0   0.0    0.0    0.0
married    -0.008095 (diff) 0.05374    0.00   0   0.0    0.0    0.0
nodegree    0.006317 (diff) 0.04179    0.00   0   0.0    0.0    0.0
re74      -37.169126 (diff) 0.18021   98.68   0   0.0 -116.8  889.5
re75       32.514138 (diff) 0.17732 -648.39   0 202.6  147.5 -640.9
hispanic    0.001212 (diff) 0.01509    0.00   0   0.0    0.0    0.0
u74         0.008800 (diff) 0.05556    0.00   0   0.0    0.0    0.0
u75         0.005442 (diff) 0.04082    0.00   0   0.0    0.0    0.0
```

Now we estimate SATT via the usual multiple imputation combining formulas (averaging the point estimates and within and between variances, as usual; see King et al. 2001). The function `att` implements these procedures:

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

           (Intercept) treated
Estimate        4527.5   729.1
Std. Error       426.2   658.4
```

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,
    grouping = NULL, drop=NULL, eval.imbalance = TRUE, k2k=FALSE,
        method=NULL, mpower=2, L1.breaks = NULL, verbose = 0)
```

Arguments

`treatment`	character, name of the treatment variable
`data`	a data.frame
`datalist`	a list of imputed data.frame's
`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.
`grouping`	named list. Each element is a list of groupings for a single variable. 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.
`L1.breaks`	list of cutpoints for the calculation of the L1 measure.
`verbose`	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.

The `grouping` is a list where each element is itself a list. For example, suppose for variable `quest1` you have the following possible leves ■`no answer`■, `NA`, ■`negative`■, ■`neutral`■, ■`positive`■ and you want to collect (■`no answer`■, `NA`, ■`neutral`■) into a single group, then the `grouping` argument should contain `list(quest1=list(c(`■`no answer`■`, NA, `■`neutral`■`)))`. Or if you have a discrete variable `elements` with values

1:10 and you want to collect it into groups "1:3,NA", "4", "5:9", "10" you specify in grouping the following list list(elements=list(c(1:3,NA), 5:9)). Values not defined in the grouping are left as they are. If cutpoints and groupings are defined for the same variable, the groupings take precedence and the corresponding cutpoints are set to NULL.

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 data contains missing data, the imbalance measures are not calculated.

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 plus a field called unique which is true only if data and datalist are not both nNULL. A cem.match object is a list with the following slots:

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 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)


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

imbalance(LL$treated, LL, drop=todrop)

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

# 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

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

# mahalnobis matching
require(MatchIt)
mah <- matchit(treated~age+education+re74+re75+black+hispanic+nodegree+married+u74+u75,
   distance="mahalanobis", data=LL)
mah
#imbalance
imbalance(LL$treated, LL, drop=todrop, weights=mah$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")
mat1

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

# with information about missingness
mat2 <- cem("treated", datalist=imputed, drop="re78", data=LL1)
mat2

# 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

obj	a cem.atch 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.

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
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
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 <- 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 imbalance: Calculates several imbalance measures

Description

Calculates several imbalance measures for the original and matched data sets

Usage

```
imbalance(group, data, drop=NULL, breaks = NULL, weights)
```

Arguments

group	the group variable
data	the data
drop	a vector of variable names in the data frame to ignore
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 statistic, the difference in quantiles and the L1 measure is calculated. For categorical variables the L1 measure and the Chi-squared distance (under column statistic) 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 og class imbalance which is a list with the following two elements

tab	Table of imbalance measures
L1	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)

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

imbalance(LL$treated, LL, drop=todrop)

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

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

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	an object of class cem.match

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
mat$k2k

# ATT estimate
att(mat, re78 ~ treated, data=LL)

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

`group`	the group variable
`data`	the data
`drop`	a vector of variable names in the data frame to ignore
`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.

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

`breaks`	A list of cutpoints used to calculate the L1 measure
`value`	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

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).

If `L1.breaks` are `NULL` they are taken from the `cem` object if available or calculated atumatically as in `cem`.

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 and the L1breaks used

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)

mat <- cem(treatment="treated",data=LL, drop="re78")
mat
tab <- relax.cem(mat, LL, 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, depth=1, minimal=list(re74=6, age=3, education=3, re75=5))
tab2 <- relax.cem(mat, LL, depth=1, minimal=list(re74=6, age=3, education=3, re75=5), shifts
tab3 <- relax.cem(mat, LL, depth=1, minimal=list(age=3, education=3), fixed=c("re74","re75")

# uncomment to run. Might be slow
# tab4 <- relax.cem(mat, LL, depth=2, minimal=list(age=4, education=3,re75=6), plot=FALSE, r
# 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

`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

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, James Honaker, Anne Joseph and Kenneth Scheve. 2001. "Analyzing Incomplete Political Science Data: An Alternative Algorithm for Multiple Imputation." *American Political Science Review* 95(1, March):49–69. http://gking.harvard.edu/files/abs/evil-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.