

CEM: Software for Coarsened Exact Matching*

Stefano M. Iacus[†] Gary King[‡] Giuseppe Porro[§]

Version
April 12, 2018

Contents

1	Introduction	2
	Overview	2
	Properties	2
	Goal	2
	Algorithm	3
	Measuring Balance	4
2	Setup	5
	2.1 Software Requirements	5
	2.2 Installation	5
	2.3 Loading CEM	5
	2.4 Updating CEM	5
3	A User’s Guide	6
	3.1 Basic Evaluation and Analysis of Unmatched Data	6
	3.2 Coarsened Exact Matching	8
	3.2.1 Automated Coarsening	9
	3.2.2 Coarsening by Explicit User Choice	9
	3.3 Progressive coarsening	11
	3.4 Restricting the Matching Solution to a k -to- k Match	14
	3.5 Estimating the Causal Effect from <code>cem</code> output	16

*Forthcoming, *Journal of Statistical Software*

[†]Department of Economics, Business and Statistics, University of Milan, Via Conservatorio 7, I-20124 Milan, Italy; stefano.iacus@unimi.it

[‡]Institute for Quantitative Social Science, 1737 Cambridge Street, Harvard University, Cambridge MA 02138; <http://GKing.harvard.edu>, king@harvard.edu, (617) 495-2027.

[§]Department of Economics and Statistics, University of Trieste, P.le Europa 1, I-34127 Trieste, Italy; giuseppe.porro@econ.units.it.

3.6	Matching and Missing Data	19
3.6.1	Matching on Missingness	20
3.6.2	Matching Multiply Imputed Data	21
3.7	Creating paired samples	23

1 Introduction

Overview This program is designed to improve causal inference via a method of matching that is widely applicable in observational data and easy to understand and use (if you understand how to draw a histogram, you will understand this method). The program implements the coarsened exact matching (CEM) algorithm, described below. CEM may be used alone or in combination with any existing matching method. This algorithm, and its statistical properties, are described in

Stefano M. Iacus, Gary King, and Giuseppe Porro, “Causal Inference Without Balance Checking: Coarsened Exact Matching”, <http://gking.harvard.edu/files/abs/cem-plus-abs.shtml>. and Stefano M. Iacus, Gary King, and Giuseppe Porro, “Multivariate Matching Methods That are Monotone Imbalance Bounding”, <http://gking.harvard.edu/files/abs/cem-math-abs.shtml>.

Properties CEM is a monotonic imbalance bounding (MIB) matching method — which means both that the maximum imbalance between the treated and control groups may be chosen by the user ex ante, rather than discovered through the usual laborious process of ex post checking and repeatedly reestimating, and that adjusting the maximum imbalance on one variable has no effect on the maximum imbalance of any other.

This paper also shows that CEM 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 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 matching method or statistical model they would have applied to the raw data. CEM also works for multicategory treatments, creating randomized blocks in experimental designs, and evaluating extreme counterfactuals.

Goal 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, 2011, 2012). If used for analyzing observational data,

applications of CEM (and all other methods of causal inference) require an assumption of ignorability (a.k.a. “no omitted variable bias” or “no confounding”).

The specific statistical goal is to estimate some version of a causal effect, such as the sample average treatment effect on the treated (the “SATT”). Thus, let Y_i be the dependent variable for unit i , T_i be a treatment variable, and X_i be a vector of pre-treatment control variables. Although CEM works as easily with multicategory treatment variables, we simplify this introductory description by assuming that T_i is dichotomous and takes on the value 1 for “treated” units and 0 for “control” units. Then define the treatment effect for treated units as the difference between two potential outcomes: $TE_i = Y_i(T_i = 1) - Y_i(T_i = 0)$, where $Y_i(T_i = 1) = Y_i$ is always observed and $Y_i(T_i = 0)$, the value that Y_i would have taken on if it were the case that $T_i = 0$, is always unobserved. Then $Y_i(T_i = 0)$ is estimated with Y_j from matched controls (i.e., among units for which $X_i \approx X_j$), either directly, $\hat{Y}_i(T_i = 0) = Y_j(T_j = 0)$, or via a model, $\hat{Y}_i(T_i = 0) = \hat{g}(X_j)$. Then SATT can be computed as a simple average: $SATT = \frac{1}{n_T} \sum_{i \in \{T_i=1\}} TE_i$.

Algorithm The CEM algorithm then involves three steps:

1. *Temporarily* coarsen each control variable in X as much as you are willing, for the purposes of matching. For example, years of education might be coarsened into grade school, middle school, high school, college, graduate school. Most researchers are intimately familiar with the concept and practice of coarsening, as it is widely used in applied data analyses in many fields, although unlike its present use coarsening for data analysis involves a permanent removal of information from the analysis and ultimate estimates.
2. Sort all units into strata, each of which has the same values of the coarsened X .
3. Prune from the data set the units in any stratum that do not include at least one treated and one control unit.

Following these three steps, the researcher can apply any method to the matched data that they might have to the raw data to estimate the causal effect, with the addition of a weight that equalizes the number of treated and control units within each stratum. Thus, any existing method of matching may be used within CEM strata to further prune the data in other ways, in which case the combined approach still inherits all of CEM’s properties. Whether or not another method of matching is applied, one must compute the causal effect, either by a (weighted) difference in means in Y among the treated and control units, or with the application of a statistical model.

If the coarsened bins are set to zero width, then CEM returns the exact matching solution, in which case model dependence will be eliminated (other than the ignorability assumption), but too few observations may be left. If instead the coarsened bins are set too wide, then few observations will be discarded, but differences within the large strata must be spanned with a statistical model, in which case model dependence may be an issue.

What if the level of coarsening is set as large as the researcher finds reasonable, but the number of observations is still too small? This of course may happen with a single continuous covariate with little or no coarsening or due to higher order interactions among a large set of discrete covariates. We offer below a “progressive coarsening” procedure that may help you rethink some of your coarsening choices by indicating how many more observations you would recover by loosening up the coarsening level for each variable. But if the remaining sample is still too small, the only possibilities involve collecting more data; setting the coarsening level artificially large and relying on theory to rule out some of the model dependence; or living with the model dependence and increasing the uncertainty with which you draw your inferences and represent your conclusions. To be clear, in this situation, you are in a bind and no method of matching or analysis is likely to save you. Statistics of course is not magic and can only get you so far given limited information. When in this situation, it is best to recognize the limits of your data relative to the question you are asking, and decide whether to devote whatever resources at your disposal to collecting more data, developing better theory, dealing with the uncertainty, or choosing a different research project.

Measuring Balance Although CEM is MIB, the actual degree of imbalance achieved in the matched sample may be lower than the chosen maximum, and so we also introduce a simple and comprehensive *multivariate* imbalance measure (Iacus, King and Porro, 2011). The measure is based on the \mathcal{L}_1 difference between the multidimensional histogram of all pretreatment covariates in the treated group and that in the control group. To do this, we first choose the number of bins for each continuous variable via standard automated univariate histogram methods and with categorical variables left as is. These bin sizes must be defined separately from and prior to the coarsening levels chosen for CEM. Although this initial choice poses all the usual issues and potential problems when choosing bins in drawing histograms, we use it only as a fixed reference to evaluate pre and post matching imbalance. (We also offer a function, which are logically similar to ROC curve for classification problems, that compute this measure for a very large number of coarsenings to see whether one matching solution dominates others; see `L1.profile`.)

Our functions compute these bin sizes automatically using automated histogram methods (and with smaller bins than would typically be chosen in running CEM), or they can optionally be set by the user, so long as this level is fixed through all subsequent matching. Then, we cross-tabulate the discretized variables as $X_1 \times \dots \times X_k$ for the treated and control groups separately, and record the k -dimensional relative frequencies for the treated $f_{\ell_1 \dots \ell_k}$ and control $g_{\ell_1 \dots \ell_k}$ units. Finally, our measure of imbalance is the absolute difference over all the cell values:

$$\mathcal{L}_1(f, g) = \frac{1}{2} \sum_{\ell_1 \dots \ell_k} |f_{\ell_1 \dots \ell_k} - g_{\ell_1 \dots \ell_k}| \quad (1)$$

and where the summation is over all cells of the multivariate histogram, but is feasible to compute because it contains at most n nonzero terms. The \mathcal{L}_1 measure¹ varies in $[0, 1]$. Perfect (up to discretization) global balance results in $\mathcal{L}_1 = 0$, and $\mathcal{L}_1 = 1$ indicates complete

¹Prior to version 1.0.90 of the `cem` package the \mathcal{L}_1 measure did not include the factor $\frac{1}{2}$.

separation of the multidimensional histograms. Any value in the interval $(0, 1)$ indicates the amount of difference between k -dimensional frequencies of the two groups.

CEM also offers several other measures of imbalance such as the global difference in means and the difference in means within the strata that are defined by every matching method.

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 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://r.iq.harvard.edu", type="source")
```

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`).

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. We modify these data by adding a (fictitious) variable to illustrate discrete responses called `q1`, the answer to a survey question asking before assignment for an opinion about this job training program, with possible responses *strongly agree*, *agree*, *neutral*, *strongly disagree*, *disagree*, and *no opinion*; note that the last category is not on the same ordered scale as the other responses. Ten percent of the observations have missing data (added randomly by us to illustrate how CEM deals with missingness). We call this new data set the LeLonde (intentionally misspelling Lalonde); the original, unmodified Lalonde (1986) data are contained in data set LL.

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)
> data(LeLonde)
```

We remove missing data from the the data set before starting the analysis (we show better procedures for dealing with missing data in Section 3.6).

```
> Le <- data.frame(na.omit(LeLonde))
```

and then compute the size of the treated and control groups:

```

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

```

Thus, the data include 258 treated units and 392 control units. The (unadjusted and therefore likely biased) difference in means is then:

```

> mean(Le$re78[tr]) - mean(Le$re78[ct])

[1] 759

```

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", "q1")

```

The overall imbalance is given by the \mathcal{L}_1 statistic:

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

```

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

```

```

Multivariate Imbalance Measure: L1=0.902
Percentage of local common support: LCS=5.8%

```

Univariate Imbalance Measures:

	statistic	type	L1	min	25%	50%	75%	max
age	-0.252373	(diff)	5.102e-03	0	0	0.0	-1.0	-6.0
education	0.153635	(diff)	8.464e-02	1	0	1.0	1.0	1.0
black	-0.010323	(diff)	1.032e-02	0	0	0.0	0.0	0.0
married	-0.009551	(diff)	9.551e-03	0	0	0.0	0.0	0.0
nodegree	-0.081217	(diff)	8.122e-02	0	-1	0.0	0.0	0.0
re74	-18.160447	(diff)	5.551e-17	0	0	284.1	806.3	-2139.0
re75	101.501762	(diff)	5.551e-17	0	0	485.6	1238.4	490.4
hispanic	-0.010145	(diff)	1.014e-02	0	0	0.0	0.0	0.0
u74	-0.045582	(diff)	4.558e-02	0	0	0.0	0.0	0.0
u75	-0.065555	(diff)	6.556e-02	0	0	0.0	0.0	0.0
q1	7.494021	(Chi2)	1.067e-01	NA	NA	NA	NA	NA

Only the overall \mathcal{L}_1 statistic measure includes imbalance with respect to the joint distribution, including all interactions, of the covariates; in the case of our example, $\mathcal{L}_1 = 0.902$. 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`)). The second column, labeled `L1`, reports the \mathcal{L}_1^j measure, which is \mathcal{L}_1 computed for the j -th variable separately (which of course does not include 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. When the variable type is `Chi2`, the only variable-by-variable measure that is defined in this table is \mathcal{L}_1^j ; others are reported missing.

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

As an aside, we note that for convenience that the function `imbalance` allows you to drop some variables before computation:

```
todrop <- c("treated", "re78")
imbalance(group=Le$treated, data=Le, drop=todrop)
```

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 = Le, drop = "re78",keep.all=TRUE)
```

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. Argument `keep.all=TRUE` returns the coarsened data set in the output of `cem`. Thus,

```
> mat
      G0  G1
All      392 258
Matched   95  84
Unmatched 297 174
```

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. By controlling the coarsening, you also put an explicit bound on the degree of model dependence and the SATT estimation error.

Fortunately, 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 almost surely have enough information to choose the coarsening. (And if you don't know something about the data, you 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. Recall that any coarsening during CEM is used only for matching; the original values of the variables are passed on to the analysis stage for all matched observations.

The function `cem` treats categorical and numerical variables differently.

For *categorical* variables, we use the `grouping` option. For example, variable `q1` has the following levels

```
> levels(Le$q1)

[1] "agree"           "disagree"        "neutral"
[4] "no opinion"       "strongly agree"  "strongly disagree"
```

Notice that the levels are not ordered in the original data set. One can possibly transform the variable `q1` from `factor` to ordered factor using the command `ordered` in R or may want to group them in three groups as follows:

```
> q1.grp <- list(c("strongly agree", "agree"), c("neutral", "no opinion"), c("strongly disagree", "disagree"))
```

For *numerical* variables, we use the `cutpoints` option. Thus, 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 usually better than caliper matching in this context, as it would disregard these meaningful breaks. (The venerable technique of caliper matching of course may be useful for certain other types of data.) Because in our data, no respondents fall in the last category,

```
> table(Le$education)

 3  4  5  6  7  8  9 10 11 12 13 14 15
 1  5  4  6 12 55 106 146 173 113 19  9  1
```

we define the cutpoints as:

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

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

```
> mat1 <- cem(treatment = "treated", data = Le, drop = "re78",
+ cutpoints = list(education=educut), grouping=list(q1=q1.grp))
> mat1
```

```
      G0  G1
All      392 258
Matched  158 115
Unmatched 234 143
```

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 4.2 5.4 6.6 7.8 9.0 10.2 11.4 12.6 13.8 15.0
```

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 concomitant increase in the maximum possible imbalance between the treated and control groups. This is easy with CEM because CEM is a monotonic imbalance bounding (MIB) method, which means that increasing the imbalance on one variable (by widening the coarsened bin sizes) 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 observations 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.

For categorical variables, the algorithm considers the numerical values associated to each level of the variable. In R the numerical values associated to the levels go from 1 to the number of levels, say k . The coarsening occurs by partitioning the interval $[1, k]$ into intervals of growing size starting from, say, $\epsilon = 1$. So in all cases, coarsening occurs by grouping adjacent levels from “left” to “right”. This is of course not completely appropriate for pure unordered categorical variables, but to treat them in a proper way additional combinatorics would be necessary. The progressive coarsening is instead intended as an instrument which gives a feeling on which variable is more likely to prevent matching for a given data set.

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", Le, cutpoints = list(age=10), drop="re78", grouping=list(q1=q1.grp))
```

```

      G0  G1
All      392 258
Matched  142 100
Unmatched 250 158
```

```
> cem("treated", Le, cutpoints = list(age=6), drop="re78", grouping=list(q1=q1.grp))
```

```

      G0  G1
All      392 258
Matched  169 125
Unmatched 223 133
```

```
> cem("treated", Le, cutpoints = list(age=3), drop="re78", grouping=list(q1=q1.grp))
```

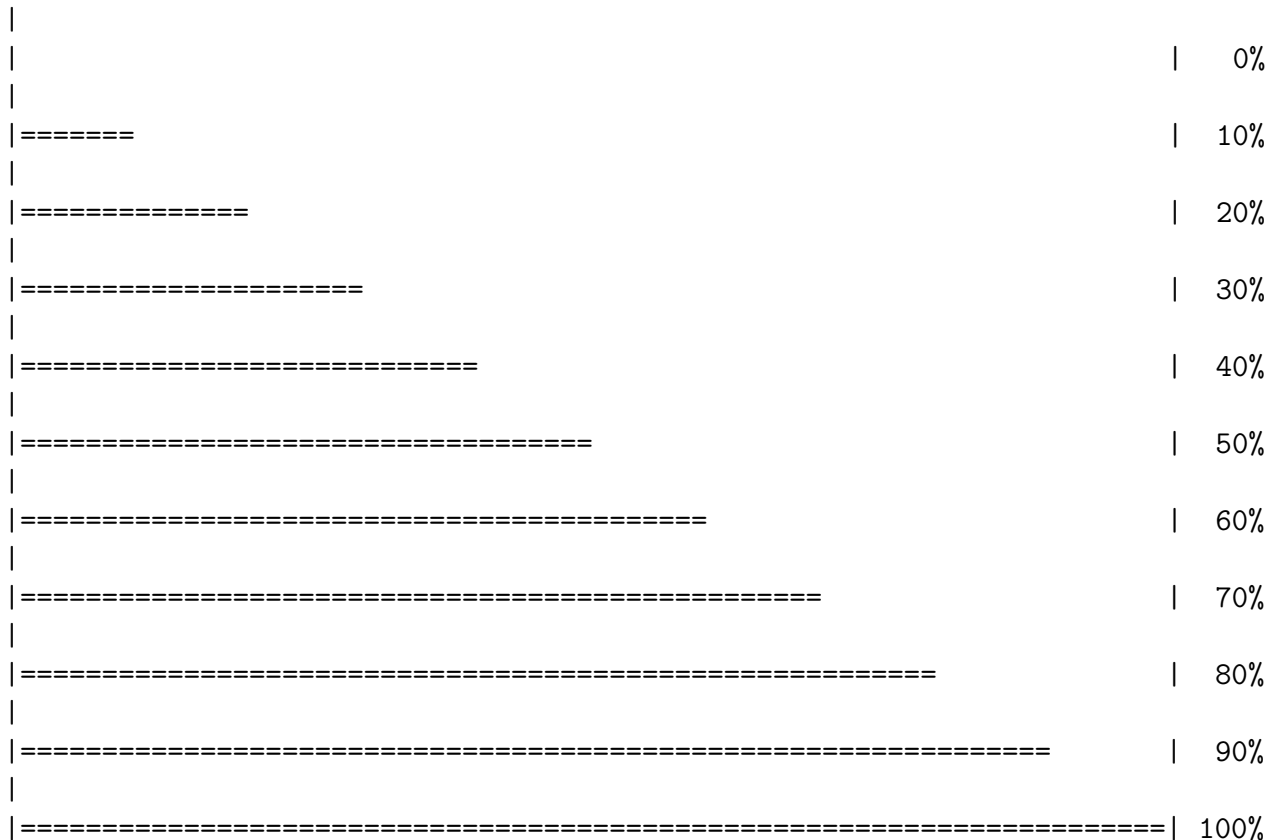
```

      G0  G1
All      392 258
Matched  217 147
Unmatched 175 111
```

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 (the argument `perc=0.3` is passed to the plot function and implies that only the solutions with at least 30% of the units are matched)

```
> tab <- relax.cem(mat, Le, depth=1, perc=0.3)
```

Executing 47 different relaxations



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

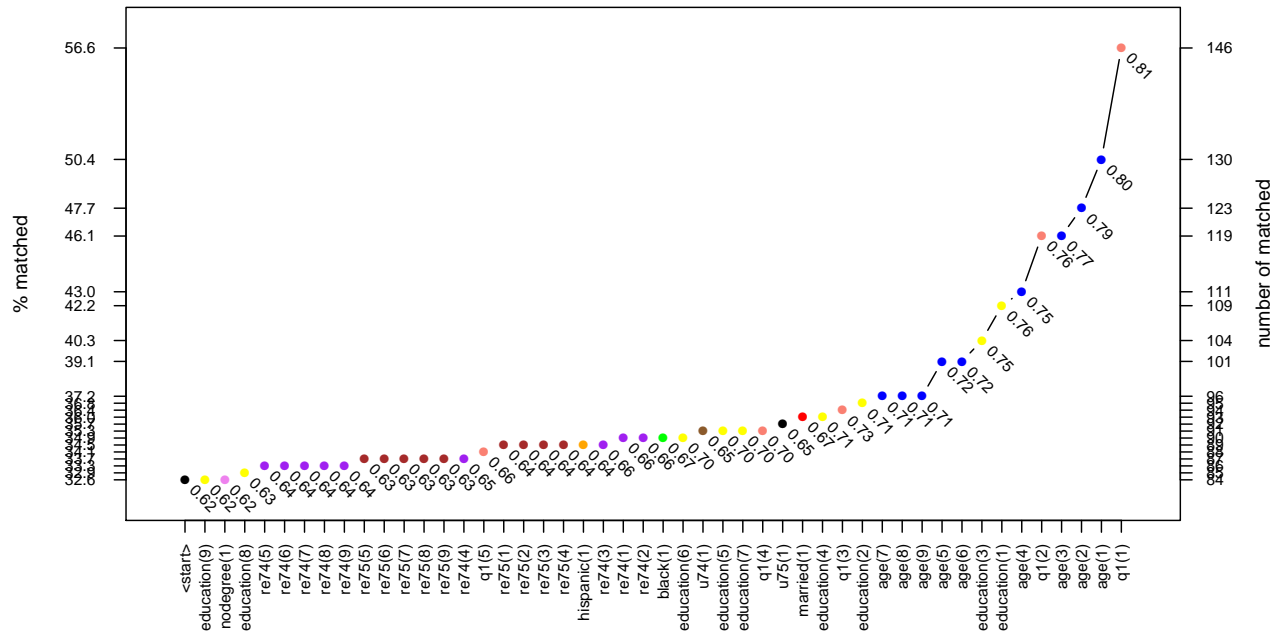


Figure 1: Example of the graphical output of `relax.cem`.

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`.

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=0.35, 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 indistinguishable to `cem` proper) by using nearest neighbor selection using a distance function specified by

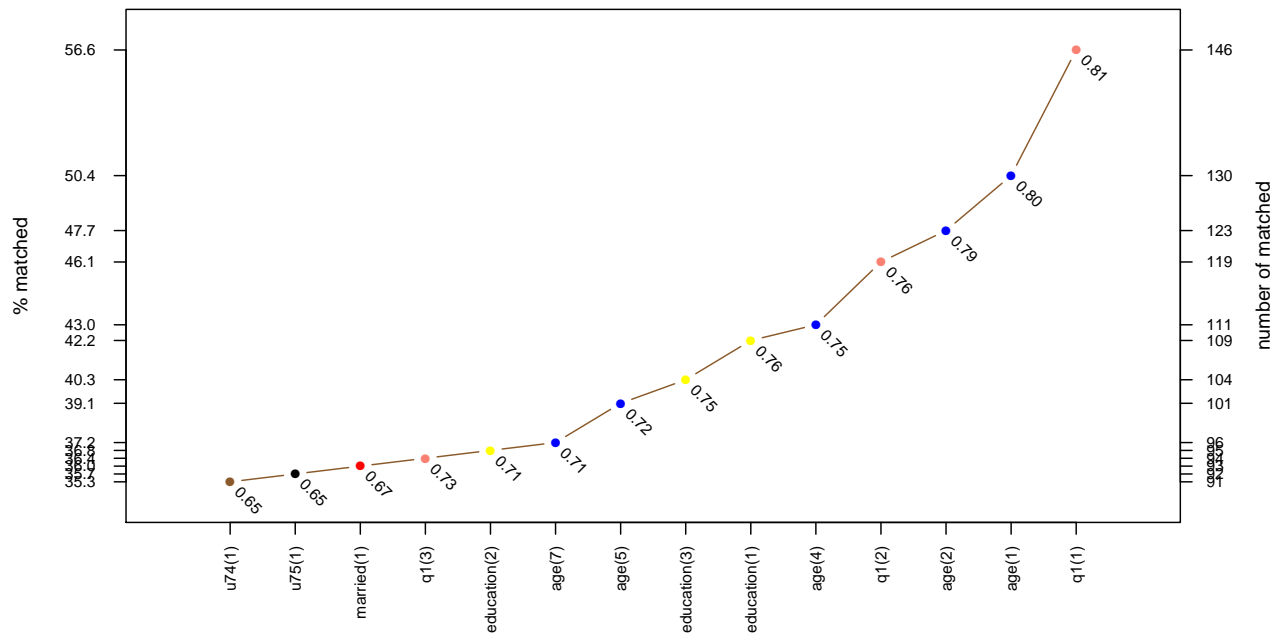


Figure 2: Example of reduced graphical output of `relax.cem`.

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 via the argument `mpower`. For more information on `method != NULL`, refer to `dist` help page.) The option `keep.all=TRUE` must be used in `cem` calls otherwise `k2k` will not work.

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

```
> mat <- cem(treatment="treated",data=Le, drop="re78",keep.all=TRUE)
> mat
```

```

      G0  G1
All    392 258
Matched  95  84
Unmatched 297 174
```

```
> mat$k2k
```

```
[1] FALSE
```

and now pruning to a *k-to-k* solution, using the euclidean distance within CEM strata:

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

```

      G0  G1
All    392 258
Matched 70 70
Unmatched 322 188

```

```
> 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,

```

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

```

```

      G0  G1
All    425 297
Matched 222 163
Unmatched 203 134

```

Linear regression model on CEM matched data:

```

SATT point estimate: 550.962564 (p.value=0.368242)
95% conf. interval: [-647.777701, 1749.702830]

```

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 a linear model to estimate the `att` 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, data = LL)
> est2
```

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

Linear regression model on CEM matched data:

```
SATT point estimate: 553.113736 (p.value=0.362760)
95% conf. interval: [-636.606542, 1742.834014]
```

The user can also specify the option `model` which accepts one of the following string arguments

- `linear` or `lm` (the default) for linear model, when the treatment effect is supposed to be homogeneous

```
> att(mat, re78 ~ treated + re74 , data = LL, model="linear")
```

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

Linear regression model on CEM matched data:

```
SATT point estimate: 553.113736 (p.value=0.362760)
95% conf. interval: [-636.606542, 1742.834014]
```

- `linear-RE` or `lme` for linear model with random effects in cem strata, for non-homogeneous treatment effect

```
> att(mat, re78 ~ treated + re74 , data = LL, model="linear-RE")
```

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

Linear random effect model on CEM matched data:

```
SATT point estimate: 552.448961 (p.value=0.000000)
95% conf. interval: [364.067068, 740.830853]
```

- `logistic` or `logit` for dichotomous response variable², for homogeneous treatment effect.
- `forest` or `rf` for random forest model, also for non-homogeneous treatment effect. It accepts continuous, dichotomous or counting outcomes.

```
> att(mat, re78 ~ treated + re74 , data = LL, model="forest")
```

```

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

Random forest model on CEM matched data:

```
SATT point estimate: 498.558006 (p.value=0.579560)
95% conf. interval: [-1265.175241, 2262.291253]
```

All the above models run on the CEM matched subsamples, so the quantity of interest may change in case of non-homogeneous treatment effect. The option `extrapolate`, if set `TRUE`, extrapolates each of the above models also to the set of treated units not matched. In this case the quantity of interest is kept fixed but the estimation is more model dependent.

```
> att(mat, re78 ~ treated + re74 , data = LL, model="linear", extra=TRUE)
```

```

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

Linear regression model with extrapolation:

```
SATT point estimate: 674.337762 (p.value=0.347286)
95% conf. interval: [-236.091964, 1584.767489]
```

```
> att(mat, re78 ~ treated + re74 , data = LL, model="linear-RE", extra=TRUE)
```

```

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

Linear random effect model with extrapolation:

²We do not provide an example here, model the syntax is the same for the other models.

```
SATT point estimate: 902.087484 (p.value=0.000000)
95% conf. interval: [816.567245, 987.607724]
```

```
> att(mat, re78 ~ treated + re74 , data = LL, model="rf", extra=TRUE)
```

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

Random forest model with extrapolation:

```
SATT point estimate: 95.338234 (p.value=0.884590)
95% conf. interval: [-1191.987223, 1382.663692]
```

As Figure 3 shows, it is also possible to plot the results of the SATT estimation as follows

```
> est3 <- att(mat, re78 ~ treated + re74 , data = LL)
> est3
```

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

Linear regression model on CEM matched data:

```
SATT point estimate: 553.113736 (p.value=0.362760)
95% conf. interval: [-636.606542, 1742.834014]
```

```
> plot(est3, mat, LL, vars=c("education", "age", "re74", "re75"))
```

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). In the first, where we treat missing values as one of the values of the variables, is appropriate when “NA” is a valid value that is not really missing (such as when “no opinion” really means no opinion); see Section 3.6.1. The other is a special procedure to allow for multiply imputed data in CEM, as described in Section 3.6.2.

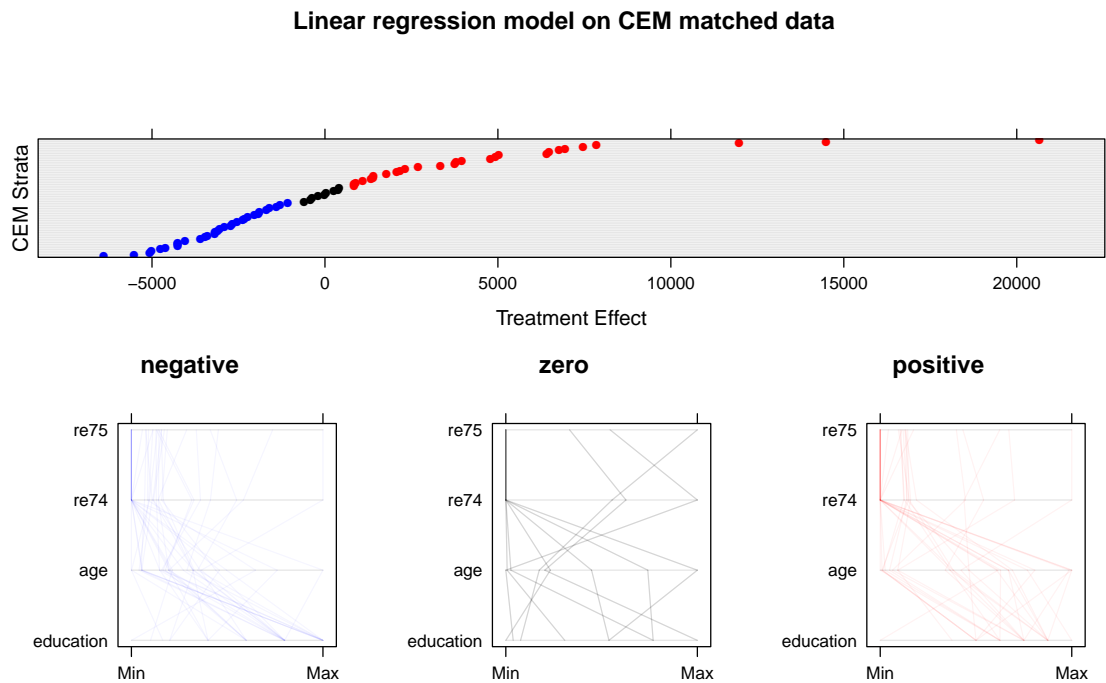


Figure 3: Example of a plot of the output of `att`. The top panel gives observation-level causal effect estimates sorted in numerical order and colored in ranges – negative (in blue), not significantly different from zero (black), or positive (red). For each range of effects, the bottom panel gives parallel plots; each line in a parallel plot represents the (covariate) characteristics of a single observation.

3.6.1 Matching on Missingness

In the next example, we use our original `LeLonde` data with missing values and we compare the result with `Le` from which we dropped the `NA` values. For comparability, we use the same cutpoints we used in Section 3.2 on the `Le` data. The cutpoints are contained in `mat$breaks`

```
> mat3 <- cem("treated", LeLonde, drop="re78", cutpoints = mat$breaks, grouping=list(q
```

The data contain missing values. CEM will match on them; see the manual for other options

```
> mat3
```

```

      G0  G1
All    425 297
Matched 134 101
Unmatched 291 196
```

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

```
> mat4 <- cem("treated", Le, drop="re78", cutpoints = mat$breaks, grouping=list(q1=q1.
> mat4
```

```

          G0  G1
All      392 258
Matched  132 100
Unmatched 260 158
```

and, as expected, the two solutions differ somewhat. The gain (in terms of number of matched units) decreases as the number of covariates increases.

3.6.2 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 take the original LeLonde data with missing values

```
> summary(LeLonde)
```

```

      treated      age      education      black      married
Min.   :0.000  Min.   :17.0  Min.    : 3.0  Min.   :0.0  Min.   :0.000
1st Qu.:0.000  1st Qu.:19.0  1st Qu.: 9.0  1st Qu.:1.0  1st Qu.:0.000
Median :0.000  Median :23.0  Median :10.0  Median :1.0  Median :0.000
Mean   :0.411  Mean   :24.5  Mean   :10.3  Mean   :0.8  Mean   :0.163
3rd Qu.:1.000  3rd Qu.:27.0  3rd Qu.:11.0  3rd Qu.:1.0  3rd Qu.:0.000
Max.   :1.000  Max.   :55.0  Max.   :16.0  Max.   :1.0  Max.   :1.000
      NA's   :8      NA's   :8      NA's   :1      NA's   :9

      nodegree      re74      re75      re78
Min.   :0.000  Min.   :  0  Min.   :  0  Min.   :  0
1st Qu.:1.000  1st Qu.:  0  1st Qu.:  0  1st Qu.:  0
Median :1.000  Median : 824  Median : 941  Median : 4033
Mean   :0.778  Mean   : 3662  Mean   : 3051  Mean   : 5486
3rd Qu.:1.000  3rd Qu.: 5237  3rd Qu.: 3993  3rd Qu.: 8813
Max.   :1.000  Max.   :39571  Max.   :37432  Max.   :60308
NA's   :5      NA's   :8      NA's   :4      NA's   :7

      hispanic      u74      u75
Min.   :0.000  Min.   :0.000  Min.   :0.000  agree      :111
1st Qu.:0.000  1st Qu.:0.000  1st Qu.:0.000  disagree   :121
```

Median	:0.000	Median	:0.000	Median	:0.000	neutral	:129
Mean	:0.104	Mean	:0.453	Mean	:0.402	no opinion	:117
3rd Qu.	:0.000	3rd Qu.	:1.000	3rd Qu.	:1.000	strongly agree	:121
Max.	:1.000	Max.	:1.000	Max.	:1.000	strongly disagree	:118
NA's	:11	NA's	:3	NA's	:3	NA's	: 5

Now we use `Amelia` package (Honaker, King and Blackwell, 2010) to create multiply imputed data sets:

```

> require(Amelia)
> set.seed(123)
> imputed <- amelia(LeLonde,noms=c("black","hispanic","treated","married","nodegree",
+ "u74","u75","q1"))

-- Imputation 1 --

  1  2  3  4

-- Imputation 2 --

  1  2  3  4

-- Imputation 3 --

  1  2  3

-- Imputation 4 --

  1  2  3  4

-- Imputation 5 --

  1  2  3

> imputed <- imputed$imputations[1:5]

```

Now `imputed` contains a list of 5 multiply imputed versions of `LeLonde`. 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=LeLonde, grouping=list(q1)
> mat2
```

```

          G0  G1
All      425 297
Matched  115  93
Unmatched 310 204
```

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)
> out
```

Linear regression model on CEM matched data:

```
SATT point estimate: -133.133916 (p.value=1.127755)
95% conf. interval: [-1755.801146, 1489.533314]
```

3.7 Creating paired samples

In some cases, it is useful to apply CEM so some data to create paired matched samples. Given an output of `cem`, the function `pair` produces two sets of indexes corresponding to pair matched units.

```
> data(LL)
> # cem match: automatic bin choice
> mat <- cem(data=LL, drop="re78")
> # we want a set of paired units
> psample <- pair(mat, data=LL)
```

```
Total number of units paired in CEM strata: 352
Total number of units matched: 722
```

each pair of observation has a different strata number

```
> table(psample$paired)
```

```

 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
```

```

61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2

```

not all observations can be matched in cem strata

```
> psample$paired[1:100]
```

```

15993 15994 15995 15996 15997 15998 15999 16000 16001 16002 16003 16004 16005
   NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
16006 16007 16008 16009 16010 16011 16012 16013 16014 16015 16016 16017 16018
   NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
16019 16020 16021 16022 16023 16024 16025 16026 16027 16028 16029 16030 16031
   NA    NA    NA    NA    NA     1    NA    NA    NA     1    NA    NA    NA
16032 16033 16034 16035 16036 16037 16038 16039 16040 16041 16042 16043 16044
   NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
16045 16046 16047 16048 16049 16050 16051 16052 16053 16054 16055 16056 16057
    3    NA    NA     4    NA    NA    NA    NA    NA    NA    NA    NA    NA
16058 16059 16060 16061 16062 16063 16064 16065 16066 16067 16068 16069 16070
   NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA     3     4
16071 16072 16073 16074 16075 16076 16077 16078 16079 16080 16081 16082 16083
   NA     6    NA    NA     7    NA    NA    NA    NA    NA    NA    NA    NA
16084 16085 16086 16087 16088 16089 16090 16091 16092
    8    NA    NA     9    NA     9    NA    NA    NA

```

the remaining observations are then matched and the final list of all paired units is contained in the file full.paired

```
> table(psample$full.paired)
```

```

 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60

```



```

  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
361
  2

```

```
> psample$full.paired[1:10]
```

```
15993 15994 15995 15996 15997 15998 15999 16000 16001 16002
  289   313   259   356   273   257   275   351   204   264
```

paired; if they cannot be paired, the function indicates which units are left without a mate.

```
> # cem match: automatic bin choice, we drop one row from the data set
> mat1 <- cem(data=LL[-1,], drop="re78")
> # we want a set of paired units but we have an odd number of units in the data
> psample <- pair(mat1, data=LL[-1,])
```

Total number of units paired in CEM strata: 352
 Total number of units matched: 720
 Unit corresponding to row `15994', not paired

```
> table(psample$full.paired)
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

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. 2010. “Amelia II: A Program for Missing Data.”. <http://gking.harvard.edu/amelia>.
- Iacus, Stefano M., Gary King and Giuseppe Porro. 2011. “Multivariate Matching Methods that are Monotonic Imbalance Bounding.” *Journal of the American Statistical Association* 106:345–361. <http://gking.harvard.edu/files/abs/cem-math-abs.shtml>.
- Iacus, Stefano M., Gary King and Giuseppe Porro. 2012. “Causal Inference Without Balance Checking: Coarsened Exact Matching.” *Political Analysis* . <http://gking.harvard.edu/files/abs/cem-plus-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):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.