

# Package ‘cem’

November 8, 2009

**Version** 1.0.128

**Date** 2009-11-07

**Title** CEM: Software for Coarsened Exact Matching

**Author** Stefano Iacus <stefano.iacus@unimi.it> Gary King <king@harvard.edu>  
Giuseppe Porro <giuseppe.porro@unimi.it>

**Maintainer** Gary King <king@harvard.edu>

**Depends** R (>= 2.6.0), nlme, lattice, randomForest

**Description** This program implements the coarsened exact matching algorithm (and many extensions) 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>

**License** GPL-2

**URL** <http://gking.harvard.edu/cem>

**Suggests** Amelia(>= 1.2-0), MatchIt

**Repository** CRAN

**Date/Publication** 2009-11-08 10:24:23

## R topics documented:

att	2
cem	5
DW	8
imbalance	9
imbspace	11
imbspace.plot	12
k2k	13
L1.meas	15
L1.profile	16

LeLonde . . . . .	18
LL . . . . .	19
pair . . . . .	20
relax.cem . . . . .	21
shift.cem . . . . .	24
<b>Index</b>	<b>26</b>

---

att

*Example of ATT estimation from CEM output*


---

## Description

An example of ATT estimation from CEM output

## Usage

```
att(obj, formula, data, model="linear", extrapolate=FALSE, ntree=2000)
## S3 method for class 'cem.att':
plot(x, obj, data, vars=NULL, plot=TRUE, ecolours, ...)
## S3 method for class 'cem.att':
summary(object, ...)
```

## Arguments

obj	a <code>cem.match</code> or <code>cem.match.list</code> object
formula	a model formula. See Details.
data	a single <code>data.frame</code> or a list of <code>data.frame</code> 's in case of <code>cem.match.list</code>
model	one model. See Details.
extrapolate	extrapolate the CEM restricted estimate to the whole data. Default = FALSE.
ntree	number of trees to generate in random forest model. Default = 2000.
x	the output from the <code>att</code> function
vars	a vector of variable names to be used in the parallel plots. By default all variables involved in data matching are used.
object	an object of class <code>cem.att</code> function
plot	if TRUE the plot is produced, otherwise only calculations are made.
ecolours	a vector of three colors respectively for positive, zero and negative treatment effect. Default <code>c("blue", "black", "red")</code> .
...	passed to the plot function or to <code>printCoefmat</code> for the method summary

## Details

Argument `model` can be `lm`, `linear` for linear regression model; `logit` for the the logistic model; `lme`, `linear-RE` for the linear model with random effects. Also `rf`, `forest` for the randomforest algorithm.

If the outcome is  $y$  and the treatment variable is  $T$ , then a formula like  $y \sim T$  will produce the simplest estimate the ATT: with `lm`, it is just the coefficient on  $T$ , which is the same as the difference in means, weighted by CEM stratum size. Users can add covariates to span any remaining imbalance after the match, such as  $y \sim T + \text{age} + \text{sex}$ , 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.

When `extrapolate = TRUE`, the estimate model is extrapolated to the whole set of data.

There is a `print` method for the output of `att`. Specifying the option `TRUE` in a `print` command gives complete output from the estimated model when available.

## Value

A matrix of estimates with their standard error, or a list in the case of `cem.match.list`. For `plot.att` a list of strata estimated treatment effect and group ("positive", "negative", "zero") and individual treatment and effect and group. The individual treatment effect and group is given by the treatment effect of the strata. Similarly for the group ("positive", "negative", "zero"). Also, colors associated to estimated treatment effects are returned for easy subsequent plotting.

## Author(s)

Stefano Iacus, Gary King, and Giuseppe Porro

## References

Stefano Iacus, Gary King, Giuseppe Porro, "Matching for Casual Inference Without Balance Checking: Coarsened Exact Matching," <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
homo1 <- att(mat, re78~treated, data=LL)
rand1 <- att(mat, re78~treated, data=LL, model="linear-RE")
rf1 <- att(mat, re78~treated, data=LL, model="rf")

homo2 <- att(mat, re78~treated, data=LL, extra=TRUE)
rand2 <- att(mat, re78~treated, data=LL, model="linear-RE", extra=TRUE)
```

```

rf2 <- att(mat, re78~treated, data=LL, model="rf", extra=TRUE)

homo1
summary(homo1)

rand1
rf1

homo2
rand2
rf2

plot( homo1, mat, LL, vars=c("age","education","re74","re75"))
plot( rand1, mat, LL, vars=c("age","education","re74","re75"))
plot( rf1, mat, LL, vars=c("age","education","re74","re75"))

plot( homo2, mat, LL, vars=c("age","education","re74","re75"))
plot( rand2, mat, LL, vars=c("age","education","re74","re75"))
plot( rf2, mat, LL, vars=c("age","education","re74","re75"))

# 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)

# example with missing data
# using multiply imputed data
# we use Amelia for multiple imputation
if(require(Amelia)){
  data(LL)
  n <- dim(LL)[1]
  k <- dim(LL)[2]

# we generate missing values in 30% of the rows of LL data
# 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))

imputed <- amelia(LL1)
imputed <- imputed$imputations[1:5]

mat <- cem("treated", datalist=imputed, data=LL1, drop="re78")

print(mat)

att(mat, re78 ~ treated, data=imputed)
}

```

---

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 optional multiply imputed data.frame's
cutpoints	named list each describing the cutpoints for numerical variables (the names are variable names). Each list element is either a vector of cutpoints, a number of cutpoints, or a method for automatic bin construction. See Details.
grouping	named list, each element of which is a list of groupings for a single categorical variable. See Details.
drop	a vector of variable names in the data frame to ignore during matching
eval.imbalance	Boolean. See Details.
k2k	boolean, restrict to 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 may be chosen, including “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 option is a list where each element is itself a list. For example, suppose for variable `quest1` you have the following possible levels "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.

By default, `cem` treats missing values as distinct categories and matches 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 (i.e., with missing cell values imputed). 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 `cem.match.list`, which is a list of objects of class `cem.match` plus a field called `unique` which is `true` only if `data` and `datalist` are not both `NULL`. A `cem.match` object is a list with the following slots:

<code>call</code>	the call
<code>strata</code>	vector of stratum number in which each observation belongs, <code>NA</code> if the observation has not been matched
<code>n.strata</code>	number of strata generated
<code>vars</code>	report variables names used for the match
<code>drop</code>	variables removed from the match
<code>breaks</code>	named list of cutpoints, eventually <code>NULL</code>
<code>treatment</code>	name of the treatment variable

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: Coarsened Exact Matching,” <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

# mahalanobis matching: we use MatchIt
if(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
# making use of Amelia for multiple imputation
if(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"))
  imputed <- imputed$imputations[1:5]
# 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)
}

```

---

 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.

---

imbalance	<i>Calculates several imbalance measures</i>
-----------	--

---

### 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 the L1 measure. See Details.
weights	weights

## Details

This function calculates 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)` to indicate the type of statistic being calculated.

If `breaks` is not specified, the Scott automated bin calculation is used (which coarsens less than Sturges, which used in `cem`). Please refer to `cem` help page. In this case, `breaks` are used to calculate the L1 measure.

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

## Value

An object of class `imbalance` which is a list with the following two elements

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

## Author(s)

Stefano Iacus, Gary King, and Giuseppe Porro

## References

Stefano Iacus, Gary King, Giuseppe Porro, "Matching for Casual Inference Without Balance Checking: Coarsened Exact Matching," <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")
```

---

imbspace

*Diagnostic tool for CEM*


---

**Description**

Diagnostic tools for CEM

**Usage**

```
imbspace(obj, data, depth = 1, L1.breaks = NULL,
plot = TRUE, fixed = NULL, minimal = 1, maximal = 6,
M=250, raw.profile=NULL)
```

**Arguments**

<code>obj</code>	an object of class <code>cem.match</code>
<code>data</code>	the original data.
<code>depth</code>	if 1, relaxes up to dropping one var, if 2 relaxes (up to dropping) two vars, etc.
<code>L1.breaks</code>	list of cutpoints for the calculation of the L1 measure.
<code>plot</code>	plot the space of solutions?
<code>fixed</code>	vector of variable names which will not be relaxed.
<code>minimal</code>	the minimal number of intervals acceptable after relaxation. Should be a named list of positive integers or if a number, this is applied to all variables.
<code>maximal</code>	the maximal number of intervals acceptable after relaxation. Should be a named list of positive integers or if a number, this is applied to all variables.
<code>M</code>	number of possible random coarsening for the L1 measure
<code>raw.profile</code>	and object of class <code>L1profile</code> . If passed, the <code>L1.breaks</code> are ignored.

**Details**

This is a diagnostic tool to help the user in the search of different choices of coarsenings. The algorithm tries all possible combination of coarsenings into intervals between `minimal` and `maximal` one variable at time, for pairs, triplets, etc depending on the value of `depth`.

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

**Value**

`val` an invisible object of class `imbalance.space`.

**Author(s)**

Stefano Iacus, Gary King, and Giuseppe Porro

## References

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

## See Also

[imbspace.plot](#)

---

imbspace.plot      *Plot of imbalance space diagnostic tool for CEM*

---

## Description

Plot of imbalance space diagnostic tool for CEM

## Usage

```
imbspace.plot(obj, group="1")
```

## Arguments

obj	an object of class <code>imbalance.space</code>
group	character string denoting group id. Defaults to "1".

## Details

For an interactive device a two panels plot is given. On the left panel the user can select a CEM solution and the number of cutpoints used in that matching solution is plotted as a parallel plot on the right plot. On exit (right-click on the left panel), the function returns all the cem solutions highlighted in the last selection of the user.

For non-interactive devices, only the space of the solutions are plotted.

This plot shows the tradeoff in matching as a function of imbalance and sample size.

The imbalance of the raw data is represented as a red plot and the initial CEM solution as a green plot. All solutions below the green dot and left to it are better than the user choice in terms of imbalance and number of units matched.

## Value

tab	an invisible object containing the selection of cem solutions and their coarsenings.
-----	--

## Author(s)

Stefano Iacus, Gary King, and Giuseppe Porro

## References

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

## See Also

[imbspace](#)

## Examples

```
require(cem)

data(LL)

mat <- cem("treated", LL, drop=c("re78","treated"), cut=list(age=4, edu=4, re74=3, re75=3))
mat

imb.raw <- L1.profile(LL$treated, LL[, mat$vars], M=250, plot=FALSE)

imbsp <- imbspace(mat, LL,depth=2, raw.profile=imb.raw, maximal=6, minimal=2, fixed=c("hispa

tmp <- plot(imbsp)
tmp
```

---

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 <code>cem</code>
data	the original data.frame used by <code>cem</code>
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: Coarsened Exact Matching," <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)
```

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 in order to measure the level of imbalance in a matching solution.

If `breaks` is not specified, the Scott automated bin calculation is used (which coarsens less than Sturges, which used in `cem`). Please refer to `cem` help page. In this case, breaks are used to calculate the L1 measure.

When choosing `breaks` for L1, a very fine coarsening (many cut points) produces values of L1 close to 1. A very mild coarsening (very few cutpoints), is not able to discriminate, i.e. L1 close to 0 (particularly true when the number of observations is small with respect to the number of continuous variables). The `L1.profile` function shows how to compare matching solutions for any level of (i.e., without regard to) coarsening.

This code also calculate the Local Common Support (LCS) measure, which is the proportion of non empty k-dimensional cells of the histogram which contain at least one observation per group.

**Value**

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

L1	The numerical value of the L1 measure
breaks	A list of cutpoints used to calculate the L1 measure
LCS	The numerical value of the Local Common Support proportion

**Author(s)**

Stefano Iacus, Gary King, and Giuseppe Porro

## References

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

## Examples

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

---

<code>L1.profile</code>	<i>Calculates L1 distance for different coarsenings</i>
-------------------------	---

---

## Description

Calculates L1 distance for different coarsenings

## Usage

```
L1.profile(group, data, drop = NULL, min.cut = 2, max.cut = 12,
weights, plot = TRUE, add = FALSE, col = "red",
lty = 1, M=100, useCP=NULL)
```

## Arguments

<code>group</code>	the group variable
<code>data</code>	the data
<code>drop</code>	a vector of variable names in the data frame to ignore
<code>min.cut</code>	minimum number of cut points per variable
<code>max.cut</code>	maximum number of cut points per variable
<code>weights</code>	weights
<code>useCP</code>	a list which elements is a list of cutpoints, usually passed from a previous instance of <code>L1.profile</code> . If not <code>NULL</code> these coarsenings are used instead of generating them randomly.
<code>M</code>	number of random coarsenings
<code>plot</code>	plot a graph?
<code>add</code>	add graph to an existing plot? Makes sense only if <code>plot</code> is <code>TRUE</code>
<code>col</code>	draw in specified color
<code>lty</code>	draw using specified lty

**Details**

The L1 measure depends on the coarsening chosen to calculate it, and as such the comparison of different matching solutions may differ depending on this somewhat arbitrary choice. This function computes L1 for a random range of possible coarsenings. The point of this function is that if one matching solution has a lower L1 than another, then it dominates without regard to the choice of coarsening. A graphic display conveys the results succinctly. (The logic is similar to that for ROC curves used for classification algorithms.) (This degree of coarsening should remain fixed for different CEM runs.)

For each variables the function generates a random number of cutpoints between `min.cut` and `max.cut` in which to cut the support of each variable. This procedure is repeated `M` times. The out is sorted in increasing values of L1 just for graphical representation.

Non numeric variables are not grouped, i.e. no coarsening occurs.

A `plot` method exists for the returned object.

**Value**

An invisible object of class `L1profile` which contains a named list of coarsenings and values of the L1 measure for each coarsening.

**Author(s)**

Stefano Iacus, Gary King, and Giuseppe Porro

**References**

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

**Examples**

```
data(LL)
for(i in c(4:6,10:12))
  LL[[i]] <- factor(LL[[i]])

imb0 <- L1.profile(LL$treated,LL, drop=c("treated","re78"))

if(require(MatchIt)){
  m2 <- matchit(treated ~ black + hispanic + married + nodegree + u74 + u75 + education +
    age + re74 + re75, data=LL, distance="logit")

  m3 <- matchit(treated ~ black + hispanic + married + nodegree + u74 + u75 + education +
    age + re74 + re75, data=LL, distance="mahalanobis")

  L1.profile(LL$treated,LL, drop=c("treated","re78"),
    weights=m2$w, add=TRUE, col="green", lty=2, useCP=imb0$CP)

  L1.profile(LL$treated,LL, drop=c("treated","re78"),
    weights=m3$w, add=TRUE, col="orange", lty=3, useCP=imb0$CP)
}
```

```

m1 <- cem("treated", LL, drop="re78")

L1.profile(LL$treated,LL, drop=c("treated","re78"),
  weights=m1$w>0, add=TRUE, col="blue", lty=4, useCP=imb0$CP)

legend(5, 0.9, legend=c("raw data", "pscore", "mahalanobis", "cem"), lty=1:4, col=c("red", "

```

---

 LeLonde

---

*Modified Lalonde dataset*


---

### Description

This is a modified version of the Lalonde experimental dataset used for explanatory reasons only.

### Usage

```
data(LL)
```

### Format

A data frame with 722 observations on the following 11 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
q1 answer to survey question n1

```

### Details

This data is a copy of the original Lalonde (1986) data set (see [LL](#)) with 10% of missing data and an additional variable `q1` which is the fictitious answer to the questionnaire on “Agreement on this job training program”.

**Source**

see references

**References**

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

---

LL	<i>Lalonde dataset</i>
----	------------------------

---

**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.

---

`pair`*Produces a paired sample out of a CEM match solution*

---

**Description**

Produces a paired sample out of a CEM match solution

**Usage**

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

**Arguments**

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

**Details**

This function returns a vector of paired matched units index.

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.

**Value**

<code>obj</code>	a list with the fields <code>paired</code> , <code>full.paired</code> , <code>reservoir</code> and <code>reservoir2</code> . The latter contain the indexes of the unmatched units.
------------------	---

**Author(s)**

Stefano Iacus, Gary King, and Giuseppe Porro

**References**

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

**Examples**

```

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)
table(psample$paired)
psample$paired[1:100]

table(psample$full.paired)
psample$full.paired[1:10]

# 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,])
table(psample$full.paired)

```

---

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, eval.imbalance=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.

<code>shifts</code>	a vector of proportions of shifts.
<code>minimal</code>	the minimal number of intervals acceptable after relaxation. Should be a named list of positive integers.
<code>group</code>	character string denoting group id. Defaults to "1".
<code>perc</code>	only plot if percentage of matched units is greater than <code>perc</code> .
<code>unique</code>	only plot different solutions (in terms of matched units).
<code>depth</code>	if 1, relaxes up to dropping one var, if 2 relaxes (up to dropping) two vars, etc.
<code>use.coarsened</code>	used coarsened values for continuous variables.
<code>colors</code>	If TRUE each variable is plotted in a different colour.
<code>eval.imbalance</code>	If TRUE L1 measure is evaluated at each iteration.
<code>...</code>	passed to the <code>relax.plot</code> function.

## Details

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

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

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

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

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

If `L1.breaks` are NULL they are taken from the `cem` object if available or calculated automatically 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: Coarsened Exact Matching,” <http://gking.harvard.edu/files/abs/cem-abs.shtml>

**See Also**

[cem](#)

**Examples**

```
## Not run:
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=0.01)
tab3 <- relax.cem(mat, LL, depth=1, minimal=list(age=3, education=3),
fixed=c("re74", "re75"))
tab4 <- relax.cem(mat, LL, depth=2, minimal=list(age=4,
education=3, re75=6), plot=FALSE, fixed="re74")

relax.plot(tab4)
relax.plot(tab4, unique=TRUE)
relax.plot(tab4, perc=0.7)

## End(Not run)
```

---

`shift.cem`*Diagnostic tool for CEM*

---

## Description

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

## Usage

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

## Arguments

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

## Details

For each variable, shift all the cutpoints left and right by `shifts` times the smallest epsilon of the coarsening. Shifting to the right produces a new cell on the left; shift to the left, adds a new cell to the coarsening on the right. Only positive proportions should be used; the algorithm will produce shifting on the left or on the right. The best shifting of the original `cem` match is produced as output, where best is defined in terms of the maximal total number of matched units  $m_T+m_C$  (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  $m_T$ , the number of control units matched  $m_C$ , and the sum  $m_T+m_C$ , 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: Coarsened Exact Matching,” <http://gking.harvard.edu/files/abs/cem-abs.shtml>

## See Also

[cem](#)

**Examples**

```
## Not run:
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)

## End(Not run)
```

# Index

## \*Topic **datagen**

cem, 4  
imbalance, 9  
imbspace, 10  
imbspace.plot, 11  
k2k, 13  
L1.meas, 14  
L1.profile, 15  
pair, 19  
relax.cem, 20  
shift.cem, 23

## \*Topic **datasets**

DW, 8  
LeLonde, 17  
LL, 18

## \*Topic **multivariate**

att, 2  
cem, 4  
k2k, 13  
pair, 19

att, 2

cem, 4, 9, 14, 21, 22, 24

dist, 6, 13, 19

DW, 8

imbalance, 9

imbspace, 10, 12

imbspace.plot, 11, 11

k2k, 13

L1.meas, 9, 14

L1.profile, 15

LeLonde, 17

LL, 18, 18

pair, 19

plot.cem.att (att), 2

relax.cem, 20

relax.plot (relax.cem), 20

shift.cem, 22, 23

summary.cem.att (att), 2