Package ‘cem’

May 8, 2020

Type Package
Title Coarsened Exact Matching
Version 1.1.20
Author Iacus, Stefano M., King, Gary, Porro, Giuseppe
Depends tcltk, lattice
Imports MatchIt, combinat, randomForest, nlme
Suggests Amelia(>= 1.2-0)
Maintainer Stefano M. Iacus <stefano.iacus@unimi.it>
Description Implementation of the Coarsened Exact Matching algorithm.
License GPL-2
URL http://gking.harvard.edu/cem/
NeedsCompilation no
Repository CRAN
Date/Publication 2020-05-08 10:50:07 UTC

R topics documented:

att 2
cem 5
cemspace 9
combine.spacegraphs 10
DW 11
imbalance 12
imbspace 14
imbspace.plot 15
k2k 16
L1.meas 18
L1.profile 19
LeLonde 21
LL 22
LLvsPSID 23
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, ecolors, ...)
## S3 method for class 'cem.att'
summary(object, ...)

Arguments

- obj: a cem.match or cem.match.list object
- formula: a model formula. See Details.
- data: a single data.frame or a list of data.frame's in case of cem.match.list
- 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 att 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 cem.att function
- plot: if TRUE the plot is produced, otherwise only calculations are made.
- ecolors: a vector of three colors respectively for positive, zero and negative treatment effect. Default c("blue", "black", "red").
- ...: passed to the plot function or to printCoefmat for the method summary
Details

Argument model can be lm, linear for linear regression model; logit for 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


Examples

data(LL)

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

# ATT estimate
homo1 <- att(mat, re78-treated, data=LL)
rand1 <- att(mat, re78-treated, data=LL, model="linear-RE")
rfl <- 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
## Not run:
if(require(Amelia)){
data(LL)
n <- dim(LL)[1]
k <- dim(LL)[2]

# we generate missing values in 30
# randomly in one column 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)
cem

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

## End(Not run)

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 = FALSE, k2k=FALSE, method=NULL, mpower=2, L1.breaks = NULL, L1.grouping = NULL, verbose = 0, baseline.group="1", keep.all=FALSE)

#### Arguments

- **treatment** character, name of the treatment variable
- **baseline.group** character, name of the baseline level treatment. See Details.
- **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.
- **L1.grouping** as grouping but only needed in the calculation of the L1 measure not in matching.
- **verbose** controls level of verbosity. Default=0.
- **keep.all** if FALSE the coarsened dataset is not returned. Default=FALSE
Details

For multilevel (and a binary) treatment variables, the cem weights are calculated with respect to the baseline. Therefore, matched units with treatment variable equal to the baseline level receive weight 1, the others the usual cem weights. Unless specified, by default baseline is set to "1". If this level is not one of the possible values taken by the treatment variable, then the baseline is set to the first level of the treatment variable.

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, cem$imbalance contains the imbalance measure by absolute difference in means for numerical variables and chi-square distance for categorical variables. If FALSE (the default) then cem$imbalance is set to NULL. If 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. If k2k is set to TRUE also keep.all is set to TRUE.

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 \texttt{cem.match} if only \texttt{data} is not \texttt{NULL} or an object of class \texttt{cem.match.list}, which is a list of objects of class \texttt{cem.match} plus a field called \texttt{unique} which is true only if \texttt{data} and \texttt{datalist} are not both \texttt{NULL}. A \texttt{cem.match} object is a list with the following slots:

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

Author(s)

Stefano Iacus, Gary King, and Giuseppe Porro

References


Examples

```r
## Not run:
data(LL)

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

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

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

# cem match: user choiced coarsening
re74cut <- hist(LL$re74, br=seq(0,max(LL$re74)+1000, by=1000), plot=FALSE)$breaks
```
re75cut <- hist(LL$re75, br=seq(0,max(LL$re75)+1000, by=1000), plot=FALSE)$breaks
agecut <- hist(LL$age, br=seq(15,55, length=14), plot=FALSE)$breaks
mycp <- list(re75=re75cut, re74=re74cut, age=agecut)
mat <- cem(treatment="treated", data=LL, drop="re78", cutpoints=mycp)
mat

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

# mahalnobis matching: 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)
}

## End(Not run)
**Description**

Exploration tool for CEM

**Usage**

cemspace(treatment=NULL, data = NULL, R=100, grouping = NULL, drop=NULL, L1.breaks = NULL, L1.grouping=NULL, plot = TRUE, fixed = NULL, minimal = 1, maximal = 5, M=250, raw.profile=NULL, keep.weights=FALSE)

**Arguments**

treatment character, name of the treatment variable.
data a data.frame.
R number of possible random coarsening for the CEM.
grouping named list, each element of which is a list of groupings for a single categorical variable. For more details see cem.
drop a vector of variable names in the data frame to ignore during matching
L1.breaks list of cutpoints for the calculation of the L1 measure.
L1.grouping as grouping but only needed in the calculation of the L1 measure not in matching.
plot plot the space of solutions?
fixed vector of variable names which will not be relaxed.
minimal 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.
maximal 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.
M number of possible random coarsening for the L1 measure
raw.profile an object of class L1profile. If passed, the L1.breaks are ignored and set to median cutpoints of L1 profile.
keep.weights if TRUE, for each matching solutions the CEM-weights are stored.

**Details**

This is a tool to help the user to explore different cem solutions by choosing random coarsenings. The algorithm tries R random choices of coarsenings into intervals between minimal and maximal for numerical, integer or ordered factors. It drops or include dichotomous or boolean variables.

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

If you want to relax a given cem solution, use the function imbspace instead.
combine.spacegraphs  

Value
val an invisible object of class imbalance.space.

Author(s)
Stefano Iacus, Gary King, and Giuseppe Porro

References

See Also
imbspace.plot, cemspace

Examples
## Not run:
data(LL)
tmp <- cemspace("treated", LL, drop="re78", M=50)
## End(Not run)

Description
Combine two spacegraph objects so that their results can be plotted together. Both spacegraphs must be from the same dataset using the same distance metric.

Usage
combine.spacegraphs(x,y)

Arguments
x a spacegraph object to be combined
y a spacegraph object to be combined

Details
This allows users to combine two spacegraph objects rather than having to re-run the spacegraph command from the start.
Inputs must be created using spacegraph.
Value

val an object of class spacegraph.

Author(s)

Richard Nielsen

See Also

spacegraph

Examples

## Not run:
data(LL)

sp1 <- spacegraph("treated", LL, drop="re78", M=5,
R=list(cem=5, psm=5, mdm=0))

## Note that we must use the same L1 measure from the first spacegraph!
sp2 <- spacegraph("treated", LL, drop="re78", raw.profile=sp1$raw.profile,
R=list(cem=0, psm=0, mdm=5))

sp3 <- combine.spacegraphs(sp1, sp2)
plot(sp3)

## End(Not run)

### 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

imbalance

Calculates several imbalance measures

Description
Calculates several imbalance measures for the original and matched data sets

Usage
imbalance(group, data, drop=NULL, breaks = NULL, weights, grouping = NULL)

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
grouping named list, each element of which is a list of groupings for a single categorical variable. See Details.
**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.

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.

See L1.meas help page for details.

**Value**

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

- tab: Table of imbalance measures
- L1: The global L1 measure of imbalance

**Author(s)**

Stefano Iacus, Gary King, and Giuseppe Porro

**References**


**Examples**

```r
## Not run:
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")

## End(Not run)

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

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

**See Also**
`imbspace.plot`

**Description**
Plot of imbalance space diagnostic tool for CEM

**Usage**
imbspace.plot(obj, group="1", data, explore=TRUE)

**Arguments**
- **obj**: an object of class `imbalance.space`
- **group**: character string denoting group id. Defaults to "1".
- **data**: data for running additional matching solutions.
- **explore**: if TRUE the user can interact and find new solutions.

**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**


**See Also**

imbspace

**Examples**

```r
## Not run:
require(cem)
data(LL)

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

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("hispanic", "black", "married",
"nodegree","u74","u75"), plot=FALSE)

tmp <- plot(imbsp,data=LL,explore=interactive())
tmp

## End(Not run)
```

---

**k2k**

*Reduction to k2k Matching*

**Description**

Reduces a CEM output to a k2k matching

**Usage**

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

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

Details

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

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

After k2k the weights of each matched observation are set to unity. Please notice that option keep.all=TRUE must be used in `cem` calls otherwise k2k will not work.

Value

- **obj**: an object of class `cem.match`

Author(s)

Stefano Iacus, Gary King, and Giuseppe Porro

References


Examples

data(LL)

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

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

# transform the match into k2k
```r
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**  
```r
L1.meas(group, data, drop=NULL, breaks = NULL, weights, grouping = NULL)
```

**Arguments**  
- `group`  
- `data`  
- `drop`  
- `breaks`  
- `weights`  
- `grouping` named list, each element of which is a list of groupings for a single categorical variable. See Details.

**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 cut points), 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 `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.

The \texttt{L1.profile} function shows how to compare matching solutions for any level of (i.e., without regard to) coarsening.

This code also calculates 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.

\textbf{Value}

An object of class \texttt{L1.meas} which is a list with the following fields:

- \texttt{L1} The numerical value of the L1 measure
- \texttt{breaks} A list of cutpoints used to calculate the L1 measure
- \texttt{LCS} The numerical value of the Local Common Support proportion

\textbf{Author(s)}

Stefano Iacus, Gary King, and Giuseppe Porro

\textbf{References}


\textbf{Examples}

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

\texttt{L1.profile} \hspace{1cm} \textit{Calculates L1 distance for different coarsenings}

\textbf{Description}

Calculates L1 distance for different coarsenings

\textbf{Usage}

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

- **group**: the group variable
- **data**: the data
- **drop**: a vector of variable names in the data frame to ignore
- **min.cut**: minimum number of cut points per variable
- **max.cut**: maximum number of cut points per variable
- **weights**: weights
- **useCP**: a list which elements is a list of cutpoints, usually passed from a previous instance of `L1.profile`. If not NULL these coarsenings are used instead of generating them randomly.
- **M**: number of random coarsenings
- **plot**: plot a graph?
- **add**: add graph to an existing plot? Makes sense only if `plot` is TRUE
- **col**: draw in specified color
- **lty**: draw using specified lty
- **grouping**: named list, each element of which is a list of groupings for a single categorical variable. See Details.
- **progress**: if TRUE, feedback on progress is given. See Details.

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 output is sorted in increasing values of L1 just for graphical representation.

Non numeric variables are grouped randomly unless they appear specified in the `grouping` argument.

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
LeLonde

References


Examples

```r
## Not run:
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", "green", "orange", "blue"))
## End(Not run)
```

LeLonde

*Modified Lalonde dataset*

Description

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

Usage

data(LL)
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

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 questionaire on “Agreement on this job training program”.

Source
see references

References

Description
Lalonde experimental dataset (see cited reference).

Usage
data(LL)
LLvsPSID

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

LLvsPSID Lalonde treated units versus PSID control individuals

Description
The Lalonde set of treated units versus PSID (Panel Study of Income Dynamics) control individuals

Usage
data(LLvsPSID)

Format
A data frame with 2787 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

Details

These two sets of treated and control units can be hardly matched.

Source

see references

References


| 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

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

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

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

Author(s)

Stefano Iacus, Gary King, and Giuseppe Porro

References


Examples

```r
## Not run:
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)

## End(Not run)
```
pscoreSelect

Heuristic search of the best propensity score model specification

Description

Heuristic search of the best propensity score model specification

Usage

pscoreSelect(formula, data, C.L=2*(pnorm(-1,0,1)), C.Q=0.1)

Arguments

data the original data.
formula formula type specification.
C.L if at least one likelihood ratio test statistic is greater than C.L, then the covariate with higher likelihood ratio test statistic is added linearly to the model.
C.Q if the highest likelihood ratio statistic is greater than C.Q, then interaction terms are included in the pscore model specification.

Details

This is a tool to help the user to the search for the best propensity score model specification along the lines suggested by Imbens and Rubin (forthcoming). The output of the function is a model formula to be passed to glm or such, in order to estimate the propensity score model and then perform propensity score matching.

This tool is useful in combination with imbspace.plot.

Value

val an invisible object of class list.

Author(s)

Richard Nielsen

References


See Also

cemspace
Examples

```r
## Not run:
data(LL)
mod <- pscoreSelect( treated ~ age + education + black + married + nodegree +
re74 + re75 + hispanic + u74 + u75, data=LL)
print(mod)
```

## End(Not run)

relax.cem

Diagnostic tool for CEM

Description

Diagnostic tools for CEM

Usage

```r
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, use.weights=FALSE, ...)
relax.plot(tab, group="1", max.terms=50, perc=.5, unique=FALSE, colors=TRUE)
```

Arguments

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

tab an invisible object containing the tabs and the L1breaks used

Details

relax.cem starts from a cem solution (as given by cem) which has to be run with argument keep.all=TRUE.
relax.cem 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


See Also

cem

Examples

```r
## Not run:
data(LL)

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

## End(Not run)
```

search.match

Heuristic search of match solutions

Description

Heuristic search of match solutions

Usage

```r
search.match(data, treatment, vars, depth=3, min.vars =1, group=1, useCP, ...)
```
Arguments

- **data**: the original data.
- **treatment**: name of the treatment variable.
- **depth**: level of interaction and squares. See Details.
- **vars**: vector of variables’ names to match on.
- **min.vars**: minimum number of variables to consider in the model.
- **group**: the identifier of the treated group, usually 1 or the level of the fact variable treatment.
- **useCP**: the cutpoints for the calculation of the L1 measure
- ... passed to `matchit`

Details

This is a tool to help the user in the search of different choices models for matching. For example, for the search of different propensity score models. The tool tries all submodels of \( k \) variables starting from one covariate up to the full model. Then adds interactions to the full model trying all pairs, triplets etc according to the parameter depth. Then, for continuous variables only, adds squared terms to the full model.

This tool is useful in combination with `imbspace.plot`.

Value

- **val**: an invisible object of class `list`.

Author(s)

Stefano Iacus, Gary King, and Giuseppe Porro

References


See Also

`imbspace.plot`
shift.cem

Diagnostic tool for CEM

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

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

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

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

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

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

Value
- tab: an invisible object containing a new cem object

Author(s)
Stefano Iacus, Gary King, and Giuseppe Porro

References

See Also
cem
Examples

```r
## 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)

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

## End(Not run)
```

Description

Randomly compute many different matching solutions

Usage

```r
spacegraph(treatment=NULL, data = NULL,
R=list("cem"=50,"psm"=0,"mdm"=0,"matchit"=0),
grouping = NULL, drop=NULL,
L1.breaks = NULL, L1.grouping=NULL, fixed = NULL,
minimal = 1, maximal = 15, M=100,
raw.profile=NULL, keep.weights=FALSE, progress=TRUE,
rgrouping=FALSE, groups=NULL, psmpoly=1, mdmpoly=1,
other.matches=NULL, heuristic=FALSE, linear.pscore=FALSE)
```

Arguments

treatment character, name of the treatment variable
data

- a data.frame

drop

- a vector of variable names in the data frame to ignore during matching.

R

- a named list that gives the number of possible random solutions for each matching method. Allowed methods are `cem`, `psm`, `mdm`, and `matchit`.

grouping

- named list, each element of which is a list of groupings for a single categorical variable. For more details see `cem`.

L1.breaks

- list of cutpoints for the calculation of the L1 measure.

L1.grouping

- as `grouping` but only needed in the calculation of the L1 measure not in matching.

fixed

- vector of variable names which will not be relaxed.

minimal

- 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.

maximal

- 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.

M

- number of possible random coarsening for the L1 measure.

raw.profile

- an object of class `L1profile`. If passed, the L1.breaks are ignored and set to median cutpoints of L1 profile.

keep.weights

- if TRUE, for each matching solutions the CEM-weights are stored.

rgrouping

- Boolean, specifies whether levels of categorical variables should be randomly grouped together by CEM.

groups

- named list, each element of which is a list of allowable groupings for a single categorical variable.

psmpoly

- numeric, specifying the order of polynomials to include in the propensity score models. At the moment, only `psmpoly=1` is available and other values will throw warnings.

mdmpoly

- numeric, specifying the order of polynomials to include in Mahalanobis matching. At the moment, only `mdmpoly=1` is available and other values will throw warnings.

other.matches

- This argument allows plotting of user-specified matching solutions. The solutions must be supplied in a specific format: as a list of data frames, where each data frame contains the observation IDs, observations weights, and the method. These must be provided in three columns of the data frame, with names (exactly) "id", "weight", "method". See the example.

heuristic

- Boolean, if TRUE spacegraph uses a heuristic method to randomly select covariates for MDM and PSM rather than sampling from all possible combinations of covariates. The heuristic tends to select most of the main effects and a small number of interactions. This method is automatically applied with large numbers of covariates.

progress

- show progress bars.

linear.pscore

- does `linear.pscore`
Details

Spacegraph is a tool to help the user to the search for optimal matching solutions by generating many matching solutions from a variety of matching algorithms (currently CEM, Mahalanobis distance matching, and propensity score matching are supported). The resulting object can be plotted with `plot()` to show where each solution falls along the bias-variance tradeoff.

The `spacegraph` function currently calculates two measures of balance for each solution: the L1 metric (see `L1.meas`) and the difference in means of the covariates. Typically, analysts look at the difference in means separately for each variable, but this can’t be plotted on a two-dimensional graph. We summarize the difference in means by calculating the average difference in means for all of the covariates. Specifically, we calculate the difference in means for each variable as \( \frac{\text{mean(treated)} - \text{mean(control)}}{\text{sd(treated)}} \) and then average across all covariates.

Value

val an object of class `spacegraph` that can be used directly with `plot()` to produce a `spacegraph`.

Author(s)

Richard Nielsen, Stephano Iacus, Gary King, and Guiseppe Porro

See Also

`combine.spacegraphs`

Examples

```r
## Not run:
data(LL)
sp <- spacegraph("treated", LL, drop="re78", M=5,
R=list(cem=5,psm=5, mdm=5))
plot(sp)
```

## ABOUT THE PLOTTING TOOL:
## The circled solution is the current selection.
## Solutions that are strictly better are also circled.
## The gui provides the exact call to re-run the selected matching solution.
## The call can also be edited, re-run by clicking the button, and
## automatically added to the existing spacegraph.
## CEM solutions can also be adjusted variable by variable and re-run.

```r
## Some plotting parameters can be changed
plot(sp, main="Comparison of Matching Methods",
ylab="L1", xlim=c(300,50), ylim=c(0,.7))
```

## You can specify whether the x-axis shows treated units,
## control units, or all units using the argument N, which
## can take the values "treated", "control", or "all".
## Default is "treated".

plot(sp, N="all")

## You can specify how the x-axis is scaled. Setting scale.var=T
## gives you the scaling as 1/sqrt(n). Setting scale.var=F gives
## scales it linearly. Default is scale.var=T.

plot(sp, scale.var=F)

## You can also specify which measure of balance to use
## by specifying the argument "balance.metric" as
## "L1", "mdiff" (Avg. standardized difference in means),
## or "mdisc" (Average Mahalanobis Discrepancy). Default is "L1".

plot(sp, balance.metric="mdiff")
plot(sp, balance.metric="mdisc")

## Matching solutions from other methods can be included in
## a spacegraph by using the argument "other.matches".
## First, Run a matching method. Here, propensity scores from MatchIt.
library(MatchIt)
m.out <- matchit(formula=treated ~ education+age, data=LL, method = "nearest")

## Put the required information into a list of data frames.
## Note, there are many ways to do this.
mymatches <- list(data.frame(names(m.out$w)))
names(mymatches[[1]])[1] <- "id"
mymatches[[1]]$weight <- m.out$w
mymatches[[1]]$method <- "matchit psm"

sp <- spacegraph("treated", LL, drop="re78", M=5,
R=list(cem=5, psm=5, mdm=5), other.matches=mymatches)
plot(sp)

## End(Not run)
Index

*Topic datagen
  cem, 5
  cemspace, 9
  combine.spacegraphs, 10
  imbalance, 12
  imbspace, 14
  imbspace.plot, 15
  k2k, 16
  L1.meas, 18
  L1.profile, 19
  pair, 24
  pscoreSelect, 26
  relax.cem, 27
  search.match, 29
  shift.cem, 31
  spacegraph, 32

*Topic datasets
  DW, 11
  LeLonde, 21
  LL, 22
  LLvsPSID, 23

*Topic multivariate
  att, 2
  cem, 5
  k2k, 16
  pair, 24
  L1.meas, 13, 18, 34
  L1.profile, 19, 19
  LeLonde, 21
  LL, 22, 22
  LLvsPSID, 23
  pair, 24
  plot.cem.att(att), 2
  pscoreSelect, 26
  relax.cem, 27
  relax.plot(relax.cem), 27
  search.match, 29
  shift.cem, 28, 31
  spacegraph, 10, 11, 32
  summary.cem.att(att), 2
  att, 2
  cem, 5, 9, 13, 18, 28, 29, 31, 33
  cemspace, 9, 10, 26
  combine.spacegraphs, 10, 34
  dist, 6, 17, 25
  DW, 11
  imbalance, 12
  imbspace, 9, 14, 16
  imbspace.plot, 9, 10, 14, 15, 15, 26, 30
  k2k, 16