

Package: ebal (via r-universe)

August 22, 2024

Type Package

Title Entropy Reweighting to Create Balanced Samples

Version 0.1-8

Date 2022-06-08

Author Jens Hainmueller

Maintainer Jens Hainmueller <jhain@stanford.edu>

Description Package implements entropy balancing, a data preprocessing procedure described in Hainmueller (2008, <[doi:10.1093/pan/mpr025](https://doi.org/10.1093/pan/mpr025)>) that allows users to reweight a dataset such that the covariate distributions in the reweighted data satisfy a set of user specified moment conditions. This can be useful to create balanced samples in observational studies with a binary treatment where the control group data can be reweighted to match the covariate moments in the treatment group. Entropy balancing can also be used to reweight a survey sample to known characteristics from a target population.

License GPL (>= 2)

Depends methods

URL <https://web.stanford.edu/~jhain/>

NeedsCompilation no

Date/Publication 2022-06-09 08:10:24 UTC

Repository <https://jankee2022.r-universe.dev>

RemoteUrl <https://github.com/cran/ebal>

RemoteRef HEAD

RemoteSha 2965641abcf9ba2d2aa4115a04cb557613ac660

Contents

baltest.collect	2
---------------------------	---

eb	3
ebalance	4
ebalance.trim	6
getsquares	8
line.searcher	9
matrixmaker	10

Index	12
--------------	-----------

baltest.collect	<i>Collect Covariate Balance Statistics</i>
-----------------	---

Description

A function that summarizes the covariate balance statistics that are computed by MatchBalance(Matching) in a balance table.

Usage

```
baltest.collect(matchbal.out, var.names, after = TRUE)
```

Arguments

matchbal.out	An object from a call to MatchBalance(Matching)
var.names	A vector of covariate names.
after	A logical flag for whether the results from before or after Matching should be summarized. If TRUE baltest.collect summarizes the results from the covariate balance checks that MatchBalance computes in the matched data. If FALSE the results from the balance checks in the unmatched data are used.

Details

See MatchBalance(Matching) for details.

Value

A matrix that contains the covariate balance statistics in tabular format.

Author(s)

Jens Hainmueller

See Also

MatchBalance in the Matching package.

Examples

```
## load(Matching) to run this example
## create toy data: one treatment indicator and three covariates X1-3
#dat <- data.frame(treatment=rbinom(50,size=1,prob=.5),replicate(3,rnorm(50)))
#covarsname <- colnames(dat)[-1]

## run balance checks
#mout <- MatchBalance(treatment~X1+X2+X3,data=dat)

## summarize in balance table
#baltest.collect(matchbal.out=mout,var.names=covarsname,after=FALSE)
```

eb

Function for Entropy Balancing

Description

This function is called internally by `ebalance` and `ebalance.trim` to implement entropy balancing. This function would normally not be called manually by a user.

Usage

```
eb(tr.total = tr.total, co.x = co.x,
   coefs = coefs, base.weight = base.weight,
   max.iterations = max.iterations,
   constraint.tolerance = constraint.tolerance,
   print.level = print.level)
```

Arguments

<code>tr.total</code>	NA
<code>co.x</code>	NA
<code>coefs</code>	NA
<code>base.weight</code>	NA
<code>max.iterations</code>	NA
<code>constraint.tolerance</code>	NA
<code>print.level</code>	NA

Value

A list containing the results from the algorithm.

Author(s)

Jens Hainmueller

See Also

ebalance, ebalance.trim

Examples

```
##---- NA -----
```

ebalance

Entropy balancing

Description

This function implements entropy balancing, a data preprocessing procedure that allows users to reweight a dataset. The preprocessing is based on a maximum entropy reweighting scheme that assigns weights to each unit such that the covariate distributions in the reweighted data satisfy a set of moment conditions specified by the researcher. This can be useful to balance covariate distributions in observational studies with a binary treatment where the control group data can be reweighted to match the covariate moments in the treatment group. Entropy balancing can also be used to reweight a survey sample to known characteristics from a target population. The weights that result from entropy balancing can be passed to regression or other models to subsequently analyze the reweighted data.

By default, ebalance reweights the covariate distributions from a control group to match target moments that are computed from a treatment group such that the reweighted data can be used to analyze the average treatment effect on the treated.

Usage

```
ebalance(Treatment, X, base.weight = NULL,
norm.constant = NULL, coefs = NULL,
max.iterations = 200,
constraint.tolerance = 1, print.level = 0)
```

Arguments

Treatment	A vector indicating the observations which are in the data group that should be reweighted (i.e. the control group) and those which are in the data group that should be used to compute the target moments (i.e. the treatment group). This can either be a logical vector or a real vector where 0 denotes control observations and 1 denotes treatment observations. By default the target moments are computed using the data from all treatment observations.
X	A matrix containing the variables that the researchers wants to include in the reweighting. To adjust the means of the covariates, the raw covariates can be included. To adjust the variances of the covariates, squared terms of the raw covariates can be included. To adjust co-moments, interaction terms can be included. All columns of this matrix must have positive variance and the matrix must be invertible. No missing data is allowed.

<code>base.weight</code>	An optional vector of base weights for the maximum entropy reweighting (one weight for each control unit). The default is uniform base weights.
<code>norm.constant</code>	An optional normalizing constant. By default the weights are normalized such that the sum of the weights for the reweighted control group match the number of observations in the treatment group.
<code>coefs</code>	An optional vector of model coefficients to start the reweighting.
<code>max.iterations</code>	Maximum number of iterations that will be run when attempting to reweight the data.
<code>constraint.tolerance</code>	This is the tolerance level used by <i>ebalance</i> to decide if the moments in the reweighted data are equal to the target moments.
<code>print.level</code>	Controls the level of printing: 0 (normal printing), 2 (detailed), and 3 (very detailed).

Details

If the user wants to pass different target moments these should be entered in a new row in the dataset which contains the target means for the covariate distributions.

Value

An list object of class *ebalance* with the following elements:

<code>target.margins</code>	A vector that contains the target moments coded from the covariate distributions of the treatment group.
<code>co.xdata</code>	A matrix that contains the covariate data from the control group.
<code>w</code>	A vector that contains the control group weights assigned by entropy balancing.
<code>coefs</code>	A vector that contains coefficients from the reweighting algorithm.
<code>maxdiff</code>	A scalar that contains the maximum deviation between the moments of the reweighted data and the target moments.
<code>constraint.tolerance</code>	The tolerance level used for the balance constraints.
<code>base.weight</code>	The base weight used.
<code>print.level</code>	The print level used.
<code>converged</code>	Logical flag if algorithm converged within tolerance.

Author(s)

Jens Hainmueller

References

Hainmueller, J. (2012) 'Entropy Balancing for Causal Effects: A Multivariate Reweighting Method to Produce Balanced Samples in Observational Studies', *Political Analysis* (Winter 2012) 20 (1): 25–46.

Zaslavsky, A. (1988), 'Representing local reweighting area adjustments by of households', Survey Methodology 14(2), 265–288.

Ireland, C. and Kullback, S. (1968), 'Contingency tables with given marginals', Biometrika 55, 179–188.

Kullback, S. (1959), Information Theory and Statistics, Wiley, NY.

See Also

Also see [ebalance.trim](#).

Examples

```
# create toy data: treatment indicator and three covariates X1-3
treatment <- c(rep(0,50),rep(1,30))
X          <- rbind(replicate(3,rnorm(50,0)),replicate(3,rnorm(30,.5)))
colnames(X) <- paste("x",1:3,sep="")

# entropy balancing
eb.out <- ebalance(Treatment=treatment,
                  X=X)

# means in treatment group data
apply(X[treatment==1,],2,mean)
# means in reweighted control group data
apply(X[treatment==0,],2,weighted.mean,w=eb.out$w)
# means in raw data control group data
apply(X[treatment==0,],2,mean)
```

ebalance.trim

Trimming of Weights for Entropy Balancing

Description

Function to trim weights obtained from entropy balancing. It takes the output from a call to ebalance and trims the data weights (subject to the moment conditions) such that the ratio of the maximum/minimum weight to the mean weight is reduced to satisfy a user specified target. If no user target is specified the maximum weight ratio is automatically trimmed as far as is feasible given the data.

Usage

```
ebalance.trim(ebalanceobj, max.weight = NULL,
             min.weight = 0, max.trim.iterations = 200,
             max.weight.increment = 0.92,
             min.weight.increment = 1.08,
             print.level = 0)
```

Arguments

- `ebalanceobj` An object from a call to `ebalance`.
- `max.weight` Optional target for the ratio of the maximum to mean weight.
- `min.weight` Optional target for the ratio of the minimum to mean weight.
- `max.trim.iterations`
 Maximum number of trimming iterations.
- `max.weight.increment`
 Increment for iterative trimming of the ratio of the maximum to mean weight (a scalar between 0-1, .92 indicates that the attempted reduction in the max ratio is 8 percent).
- `min.weight.increment`
 Increment for iterative trimming of the ratio of the minimum to mean weight (a scalar > 1, 1.08 indicates that the attempted reduction in the max ratio is 8 percent).
- `print.level` Controls the level of printing: 0 (normal printing), 2 (detailed), and 3 (very detailed).

Value

An list object of class `ebalance.trim` with the following elements:

- `target.margins` A vector that contains the target moments coded from the covariate distributions of the treatment group.
- `co.xdata` A matrix that contains the covariate data from the control group.
- `w` A vector that contains the control group weights assigned by trimming entropy balancing algorithm.
- `coefs` A vector that contains coefficients from the reweighting algorithm.
- `maxdiff` A scalar that contains the maximum deviation between the moments of the reweighted data and the target moments.
- `norm.constant` Normalizing constant used.
- `constraint.tolerance`
 The tolerance level used for the balance constraints.
- `max.iterations` Maximum number of trimming iterations used.
- `base.weight` The base weight used.
- `converged` Logical flag if algorithm converged within tolerance.

Author(s)

Jens Hainmueller

References

- Hainmueller, J. (2012) 'Entropy Balancing for Causal Effects: A Multivariate Reweighting Method to Produce Balanced Samples in Observational Studies', *Political Analysis* (Winter 2012) 20 (1): 25–46.
- Zaslavsky, A. (1988), 'Representing local reweighting area adjustments by of households', *Survey Methodology* 14(2), 265–288.
- Ireland, C. and Kullback, S. (1968), 'Contingency tables with given marginals', *Biometrika* 55, 179–188.
- Kullback, S. (1959), *Information Theory and Statistics*, Wiley, NY.

See Also

Also see [ebalance.trim](#).

Examples

```
# create toy data: treatment indicator and three covariates X1-3
treatment <- c(rep(0,50),rep(1,30))
X         <- rbind(replicate(3,rnorm(50,0)),replicate(3,rnorm(30,.5)))
colnames(X) <- paste("x",1:3,sep="")

# entropy balancing
eb.out <- ebalance(Treatment=treatment,
                  X=X)

# means in treatment group data
apply(X[treatment==1,],2,mean)
# means in reweighted control group data
apply(X[treatment==0,],2,weighted.mean,w=eb.out$w)
# means in raw data control group data
apply(X[treatment==0,],2,mean)

# trim weights
eb.out.tr <- ebalance.trim(eb.out)
# means in reweighted control group data
apply(X[treatment==0,],2,weighted.mean,w=eb.out.tr$w)

# untrimmed and trimmed weights
round(summary(eb.out$w),2)
round(summary(eb.out.tr$w),2)
```

getsquares

Generate Matrix of Squared Terms

Description

Takes a matrix of covariates and generates a new matrix that contains the original covariates and all squared terms. Squared terms for binary covariates are omitted.

Usage

```
getsquares(mat)
```

Arguments

mat n by k numeric matrix of covariates.

Value

n by k*2 numeric matrix that contains the original covariates plus all squared terms.

Author(s)

Jens Hainmueller

See Also

See [matrixmaker](#)

Examples

```
# create toy matrix
mold <- replicate(3,rnorm(50))
colnames(mold) <- paste("x",1:3,sep="")
head(mold)
# create new matrix
mnew <- getsquares(mold)
head(mnew)
```

line.searcher

Optimal step length search for entropy balancing algorithm

Description

Function called internally by `ebalance` and `ebalance.trim` to compute optimal step length for entropy balancing algorithm. This function would normally not be called manually by a user.

Usage

```
line.searcher(Base.weight, Co.x,
Tr.total, coefs, Newton, ss)
```

Arguments

Base.weight	NA
Co.x	NA
Tr.total	NA
coefs	NA
Newton	NA
ss	NA

Value

A list with the results from the search.

Author(s)

Jens Hainmueller

See Also

ebalance, ebalance.trim

Examples

```
##---- NA -----
```

matrixmaker

Generate Matrix of One-way Interactions and Squared Terms

Description

Takes a matrix of covariates and generates a new matrix that contains the original covariates, all one-way interaction terms, and all squared terms.

Usage

```
matrixmaker(mat)
```

Arguments

mat n by k numeric matrix of covariates.

Value

n by $(k*(k+1))/2 + 1$ matrix of covariates with original covariates, all one-way interaction terms, and all squared terms.

Author(s)

Jens Hainmueller

See Also

See [getsquares](#)

Examples

```
# create toy matrix
mold <- replicate(3,rnorm(50))
colnames(mold) <- paste("x",1:3,sep="")
head(mold)
# create new matrix
mnew <- matrixmaker(mold)
head(mnew)
```

Index

baltest.collect, [2](#)

eb, [3](#)

ebalance, [3](#), [4](#)

ebalance.trim, [3](#), [6](#), [6](#), [8](#)

getsquares, [8](#), [11](#)

line.searcher, [9](#)

matrixmaker, [9](#), [10](#)