Package 'Matching'

April 14, 2022

Version 4.10-2

Date 2022-04-13

Title Multivariate and Propensity Score Matching with Balance Optimization

Maintainer Jasjeet Singh Sekhon <jas.sekhon@yale.edu>

Description Provides functions for multivariate and propensity score matching and for finding optimal balance based on a genetic search algorithm. A variety of univariate and multivariate metrics to determine if balance has been obtained are also provided. For details, see the paper by Jasjeet Sekhon (2007, <doi:10.18637/jss.v042.i07>).

Depends R (>= 2.6.0), MASS (>= 7.2-1), graphics, grDevices, stats

Suggests parallel, rgenoud (>= 2.12)

License GPL-3

URL https://github.com/JasjeetSekhon/Matching

NeedsCompilation yes

RoxygenNote 7.1.1

Author Jasjeet Singh Sekhon [aut, cre], Theo Saarinen [aut]

Repository CRAN

Date/Publication 2022-04-14 12:22:30 UTC

R topics documented:

`balanceUV` *Univariate Balance Tests*

Description

This function provides a number of univariate balance metrics. Generally, users should call `MatchBalance` and not this function directly.

Usage

```
balanceUV(Tr, Co, weights = rep(1, length(Co)), exact = FALSE, ks=FALSE,
          nboots = 1000, paired=TRUE, match=FALSE,
          weights.Tr=rep(1,length(Tr)), weights.Co=rep(1,length(Co)),
          estimand="ATT")
```

Arguments

`Tr`	A vector containing the treatment observations.
`Co`	A vector containing the control observations.
`weights`	A vector containing the observation specific weights. Only use this option when the treatment and control observations are paired (as they are after matching).
`exact`	A logical flag indicating if the exact Wilcoxon test should be used instead of the test with a correction. See `wilcox.test` for details.
`ks`	A logical flag for if the univariate bootstrap Kolmogorov-Smirnov (KS) test should be calculated. If the ks option is set to true, the univariate KS test is calculated for all non-dichotomous variables. The bootstrap KS test is consistent even for non-continuous variables. See `ks.boot` for more details.
`nboots`	The number of bootstrap samples to be run for the `ks` test. If zero, no bootstraps are done. Bootstrapping is highly recommended because the bootstrapped Kolmogorov-Smirnov test only provides correct coverage even for non-continuous covariates. At least 500 `nboots` (preferably 1000) are recommended for publication quality p-values.
`paired`	A flag for if the paired `t.test` should be used.
`match`	A flag for if the `Tr` and `Co` objects are the result of a call to `Match`.
`weights.Tr`	A vector of weights for the treated observations.
`weights.Co`	A vector of weights for the control observations.

`estimand` This determines if the standardized mean difference returned by the `sdiff` object is standardized by the variance of the treatment observations (which is done if the estimand is either "ATE" or "ATT") or by the variance of the control observations (which is done if the estimand is "ATC").

Value

`sdiff` This is the standardized difference between the treated and control units multiplied by 100. That is, 100 times the mean difference between treatment and control units divided by the standard deviation of the treatment observations alone if the estimand is either `ATT` or `ATE`. The variance of the control observations are used if the estimand is `ATC`.

`sdiff.pooled` This is the standardized difference between the treated and control units multiplied by 100 using the pooled variance. That is, 100 times the mean difference between treatment and control units divided by the pooled standard deviation as in Rosenbaum and Rubin (1985).

`mean.Tr` The mean of the treatment group.

`mean.Co` The mean of the control group.

`var.Tr` The variance of the treatment group.

`var.Co` The variance of the control group.

`p.value` The p-value from the two-sided weighted `t.test`.

`var.ratio` var.Tr/var.Co.

`ks` The object returned by `ks.boot`.

`tt` The object returned by two-sided weighted `t.test`.

`qqsummary` The return object from a call to `qqstats` with standardization—i.e., balance test based on the empirical CDF.

`qqsummary.raw` The return object from a call to `qqstats` without standardization–i.e., balance tests based on the empirical QQ-plot which retain the scale of the variable.

Author(s)

Jasjeet S. Sekhon, UC Berkeley, <sekhon@berkeley.edu>, http://sekhon.berkeley.edu/.

References

Sekhon, Jasjeet S. 2011. "Multivariate and Propensity Score Matching Software with Automated Balance Optimization." *Journal of Statistical Software* 42(7): 1-52. doi: 10.18637/jss.v042.i07

Diamond, Alexis and Jasjeet S. Sekhon. 2013. "Genetic Matching for Estimating Causal Effects: A General Multivariate Matching Method for Achieving Balance in Observational Studies." *Review of Economics and Statistics*. 95 (3): 932–945. http://sekhon.berkeley.edu/papers/GenMatch.pdf

Rosenbaum, Paul R. and Donald B. Rubin. 1985. "Constructing a Control Group Using Multivariate Matched Sampling Methods That Incorporate the Propensity Score." *The American Statistician* 39:1 33-38.

Hollander, Myles and Douglas A. Wolfe. 1973. *Nonparametric statistical inference*. New York: John Wiley & Sons.

See Also

Also see summary.balanceUV, qqstats ks.boot, Match, GenMatch, MatchBalance, GerberGreenImai, lalonde

Examples

```
data(lalonde)
attach(lalonde)

foo  <- balanceUV(re75[treat==1],re75[treat!=1])
summary(foo)
```

GenMatch *Genetic Matching*

Description

This function finds optimal balance using multivariate matching where a genetic search algorithm determines the weight each covariate is given. Balance is determined by examining cumulative probability distribution functions of a variety of standardized statistics. By default, these statistics include t-tests and Kolmogorov-Smirnov tests. A variety of descriptive statistics based on empirical-QQ (eQQ) plots can also be used or any user provided measure of balance. The statistics are not used to conduct formal hypothesis tests, because no measure of balance is a monotonic function of bias and because balance should be maximized without limit. The object returned by GenMatch can be supplied to the Match function (via the Weight.matrix option) to obtain causal estimates. GenMatch uses genoud to perform the genetic search. Using the cluster option, one may use multiple computers, CPUs or cores to perform parallel computations.

Usage

```
GenMatch(Tr, X, BalanceMatrix=X, estimand="ATT", M=1, weights=NULL,
         pop.size = 100, max.generations=100,
         wait.generations=4, hard.generation.limit=FALSE,
         starting.values=rep(1,ncol(X)),
         fit.func="pvals",
         MemoryMatrix=TRUE,
         exact=NULL, caliper=NULL, replace=TRUE, ties=TRUE,
         CommonSupport=FALSE, nboots=0, ks=TRUE, verbose=FALSE,
         distance.tolerance=1e-05,
         tolerance=sqrt(.Machine$double.eps),
         min.weight=0, max.weight=1000,
         Domains=NULL, print.level=2,
         project.path=NULL,
         paired=TRUE, loss=1,
         data.type.integer=FALSE,
         restrict=NULL,
         cluster=FALSE, balance=TRUE, ...)
```

Arguments

Tr	A vector indicating the observations which are in the treatment regime and those which are not. This can either be a logical vector or a real vector where 0 denotes control and 1 denotes treatment.
X	A matrix containing the variables we wish to match on. This matrix may contain the actual observed covariates or the propensity score or a combination of both.
BalanceMatrix	A matrix containing the variables we wish to achieve balance on. This is by default equal to X, but it can in principle be a matrix which contains more or less variables than X or variables which are transformed in various ways. See the examples.
estimand	A character string for the estimand. The default estimand is "ATT", the sample average treatment effect for the treated. "ATE" is the sample average treatment effect, and "ATC" is the sample average treatment effect for the controls.
M	A scalar for the number of matches which should be found. The default is one-to-one matching. Also see the ties option.
weights	A vector the same length as Y which provides observation specific weights.
pop.size	Population Size. This is the number of individuals genoud uses to solve the optimization problem. The theorems proving that genetic algorithms find good solutions are asymptotic in population size. Therefore, it is important that this value not be small. See genoud for more details.
max.generations	Maximum Generations. This is the maximum number of generations that genoud will run when optimizing. This is a *soft* limit. The maximum generation limit will be binding only if hard.generation.limit has been set equal to *TRUE*. Otherwise, wait.generations controls when optimization stops. See genoud for more details.
wait.generations	If there is no improvement in the objective function in this number of generations, optimization will stop. The other options controlling termination are max.generations and hard.generation.limit.
hard.generation.limit	This logical variable determines if the max.generations variable is a binding constraint. If hard.generation.limit is *FALSE*, then the algorithm may exceed the max.generations count if the objective function has improved within a given number of generations (determined by wait.generations).
starting.values	This vector's length is equal to the number of variables in X. This vector contains the starting weights each of the variables is given. The starting.values vector is a way for the user to insert *one* individual into the starting population. genoud will randomly create the other individuals. These values correspond to the diagonal of the Weight.matrix as described in detail in the Match function.
fit.func	The balance metric GenMatch should optimize. The user may choose from the following or provide a function: pvals: maximize the p.values from (paired) t-tests and Kolmogorov-Smirnov tests conducted for each column in BalanceMatrix. Lexical optimization is

conducted—see the loss option for details.
qqmean.mean: calculate the mean standardized difference in the eQQ plot for each variable. Minimize the mean of these differences across variables.
qqmean.max: calculate the mean standardized difference in the eQQ plot for each variable. Minimize the maximum of these differences across variables. Lexical optimization is conducted.
qqmedian.mean: calculate the median standardized difference in the eQQ plot for each variable. Minimize the median of these differences across variables.
qqmedian.max: calculate the median standardized difference in the eQQ plot for each variable. Minimize the maximum of these differences across variables. Lexical optimization is conducted.
qqmax.mean: calculate the maximum standardized difference in the eQQ plot for each variable. Minimize the mean of these differences across variables.
qqmax.max: calculate the maximum standardized difference in the eQQ plot for each variable. Minimize the maximum of these differences across variables. Lexical optimization is conducted.
Users may provide their own fit.func. The name of the user provided function should not be backquoted or quoted. This function needs to return a fit value that will be minimized, by lexical optimization if more than one fit value is returned. The function should expect two arguments. The first being the matches object returned by GenMatch—see below. And the second being a matrix which contains the variables to be balanced—i.e., the BalanceMatrix the user provided to GenMatch. For an example see http://sekhon.berkeley.edu/matching/R/my_fitfunc.R.

MemoryMatrix

This variable controls if genoud sets up a memory matrix. Such a matrix ensures that genoud will request the fitness evaluation of a given set of parameters only once. The variable may be *TRUE* or *FALSE*. If it is *FALSE*, genoud will be aggressive in conserving memory. The most significant negative implication of this variable being set to *FALSE* is that genoud will no longer maintain a memory matrix of all evaluated individuals. Therefore, genoud may request evaluations which it has previously requested. When the number variables in X is large, the memory matrix consumes a large amount of RAM.

genoud's memory matrix will require *significantly* less memory if the user sets hard.generation.limit equal to *TRUE*. Doing this is a good way of conserving memory while still making use of the memory matrix structure.

exact

A logical scalar or vector for whether exact matching should be done. If a logical scalar is provided, that logical value is applied to all covariates in X. If a logical vector is provided, a logical value should be provided for each covariate in X. Using a logical vector allows the user to specify exact matching for some but not other variables. When exact matches are not found, observations are dropped. distance.tolerance determines what is considered to be an exact match. The exact option takes precedence over the caliper option. Obviously, if exact matching is done using *all* of the covariates, one should not be using GenMatch unless the distance.tolerance has been set unusually high.

caliper

A scalar or vector denoting the caliper(s) which should be used when matching. A caliper is the distance which is acceptable for any match. Observations which are outside of the caliper are dropped. If a scalar caliper is provided, this caliper

is used for all covariates in X. If a vector of calipers is provided, a caliper value should be provided for each covariate in X. The caliper is interpreted to be in standardized units. For example, caliper=.25 means that all matches not equal to or within .25 standard deviations of each covariate in X are dropped. The ecaliper object which is returned by GenMatch shows the enforced caliper on the scale of the X variables. Note that dropping observations generally changes the quantity being estimated.

replace — A logical flag for whether matching should be done with replacement. Note that if FALSE, the order of matches generally matters. Matches will be found in the same order as the data are sorted. Thus, the match(es) for the first observation will be found first, the match(es) for the second observation will be found second, etc. Matching without replacement will generally increase bias. Ties are randomly broken when replace==FALSE—see the ties option for details.

ties — A logical flag for whether ties should be handled deterministically. By default ties==TRUE. If, for example, one treated observation matches more than one control observation, the matched dataset will include the multiple matched control observations and the matched data will be weighted to reflect the multiple matches. The sum of the weighted observations will still equal the original number of observations. If ties==FALSE, ties will be randomly broken. *If the dataset is large and there are many ties, setting* ties=FALSE *often results in a large speedup.* Whether two potential matches are close enough to be considered tied, is controlled by the distance.tolerance option.

CommonSupport — This logical flag implements the usual procedure by which observations outside of the common support of a variable (usually the propensity score) across treatment and control groups are discarded. The caliper option is to be preferred to this option because CommonSupport, consistent with the literature, only drops *outliers* and leaves *inliers* while the caliper option drops both. If CommonSupport==TRUE, common support will be enforced on the first variable in the X matrix. Note that dropping observations generally changes the quantity being estimated. Use of this option renders it impossible to use the returned object matches to reconstruct the matched dataset. Seriously, don't use this option; use the caliper option instead.

nboots — The number of bootstrap samples to be run for the ks test. By default this option is set to zero so no bootstraps are done. See ks.boot for additional details.

ks — A logical flag for if the univariate bootstrap Kolmogorov-Smirnov (KS) test should be calculated. If the ks option is set to true, the univariate KS test is calculated for all non-dichotomous variables. The bootstrap KS test is consistent even for non-continuous variables. By default, the bootstrap KS test is not used. To change this see the nboots option. If a given variable is dichotomous, a t-test is used even if the KS test is requested. See ks.boot for additional details.

verbose — A logical flag for whether details of each fitness evaluation should be printed. Verbose is set to FALSE if the cluster option is used.

distance.tolerance — This is a scalar which is used to determine if distances between two observations are different from zero. Values less than distance.tolerance are deemed to be equal to zero. This option can be used to perform a type of optimal matching.

`tolerance` This is a scalar which is used to determine numerical tolerances. This option is used by numerical routines such as those used to determine if a matrix is singular.

`min.weight` This is the minimum weight any variable may be given.

`max.weight` This is the maximum weight any variable may be given.

`Domains` This is a `ncol(X)` $\times 2$ matrix. The first column is the lower bound, and the second column is the upper bound for each variable over which `genoud` will search for weights. If the user does not provide this matrix, the bounds for each variable will be determined by the `min.weight` and `max.weight` options.

`print.level` This option controls the level of printing. There are four possible levels: 0 (minimal printing), 1 (normal), 2 (detailed), and 3 (debug). If level 2 is selected, `GenMatch` will print details about the population at each generation, including the best individual found so far. If debug level printing is requested, details of the `genoud` population are printed in the "genoud.pro" file which is located in the temporary `R` directory returned by the `tempdir` function. See the `project.path` option for more details. Because `GenMatch` runs may take a long time, it is important for the user to receive feedback. Hence, print level 2 has been set as the default.

`project.path` This is the path of the `genoud` project file. By default no file is produced unless `print.level=3`. In that case, `genoud` places its output in a file called "genoud.pro" located in the temporary directory provided by `tempdir`. If a file path is provided to the `project.path` option, a file will be created regardless of the `print.level`. The behavior of the project file, however, will depend on the `print.level` chosen. If the `print.level` variable is set to 1, then the project file is rewritten after each generation. Therefore, only the currently fully completed generation is included in the file. If the `print.level` variable is set to 2 or higher, then each new generation is simply appended to the project file. No project file is generated for `print.level=0`.

`paired` A flag for whether the paired `t.test` should be used when determining balance.

`loss` The loss function to be optimized. The default value, `1`, implies "lexical" optimization: all of the balance statistics will be sorted from the most discrepant to the least and weights will be picked which minimize the maximum discrepancy. If multiple sets of weights result in the same maximum discrepancy, then the second largest discrepancy is examined to choose the best weights. The processes continues iteratively until ties are broken.

If the value of `2` is used, then only the maximum discrepancy is examined. This was the default behavior prior to version 1.0. The user may also pass in any function she desires. Note that the option 1 corresponds to the `sort` function and option 2 to the `min` function. Any user specified function should expect a vector of balance statistics ("p-values") and it should return either a vector of values (in which case "lexical" optimization will be done) or a scalar value (which will be maximized). Some possible alternative functions are `mean` or `median`.

`data.type.integer`

By default, floating-point weights are considered. If this option is set to `TRUE`,

search will be done over integer weights. Note that before version 4.1, the default was to use integer weights.

`restrict` A matrix which restricts the possible matches. This matrix has one row for each restriction and three columns. The first two columns contain the two observation numbers which are to be restricted (for example 4 and 20), and the third column is the restriction imposed on the observation-pair. Negative numbers in the third column imply that the two observations cannot be matched under any circumstances, and positive numbers are passed on as the distance between the two observations for the matching algorithm. The most commonly used positive restriction is `0` which implies that the two observations will always be matched.

Exclusion restriction are even more common. For example, if we want to exclude the observation pair 4 and 20 and the pair 6 and 55 from being matched, the restrict matrix would be: `restrict=rbind(c(4,20,-1),c(6,55,-1))`

`cluster` This can either be an object of the 'cluster' class returned by one of the `makeCluster` commands in the `parallel` package or a vector of machine names so that `GenMatch` can setup the cluster automatically. If it is the latter, the vector should look like:
`c("localhost","musil","musil","deckard")`.
This vector would create a cluster with four nodes: one on the localhost another on "deckard" and two on the machine named "musil". Two nodes on a given machine make sense if the machine has two or more chips/cores. `GenMatch` will setup a SOCK cluster by a call to `makePSOCKcluster`. This will require the user to type in her password for each node as the cluster is by default created via `ssh`. One can add on usernames to the machine name if it differs from the current shell: "username@musil". Other cluster types, such as PVM and MPI, which do not require passwords, can be created by directly calling `makeCluster`, and then passing the returned cluster object to `GenMatch`. For an example of how to manually setup up a cluster with a direct call to `makeCluster` see `http://sekhon.berkeley.edu/matching/R/cluster_manual.R`. For an example of how to get around a firewall by ssh tunneling see: `http://sekhon.berkeley.edu/matching/R/cluster_manual_tunnel.R`.

`balance` This logical flag controls if load balancing is done across the cluster. Load balancing can result in better cluster utilization; however, increased communication can reduce performance. This option is best used if each individual call to `Match` takes at least several minutes to calculate or if the nodes in the cluster vary significantly in their performance. If cluster==FALSE, this option has no effect.

`...` Other options which are passed on to `genoud`.

Value

`value` The fit values at the solution. By default, this is a vector of p-values sorted from the smallest to the largest. There will generally be twice as many p-values as there are variables in `BalanceMatrix`, unless there are dichotomous variables in this matrix. There is one p-value for each covariate in `BalanceMatrix` which is the result of a paired t-test and another p-value for each non-dichotomous

variable in BalanceMatrix which is the result of a Kolmogorov-Smirnov test. Recall that these p-values cannot be interpreted as hypothesis tests. They are simply measures of balance.

par — A vector of the weights given to each variable in X.

Weight.matrix — A matrix whose diagonal corresponds to the weight given to each variable in X. This object corresponds to the Weight.matrix in the Match function.

matches — A matrix where the first column contains the row numbers of the treated observations in the matched dataset. The second column contains the row numbers of the control observations. And the third column contains the weight that each matched pair is given. These objects may not correspond respectively to the index.treated, index.control and weights objects which are returned by Match because they may be ordered in a different way. Therefore, end users should use the objects returned by Match because those are ordered in the way that users expect.

ecaliper — The size of the enforced caliper on the scale of the X variables. This object has the same length as the number of covariates in X.

Author(s)

Jasjeet S. Sekhon, UC Berkeley, <sekhon@berkeley.edu>, http://sekhon.berkeley.edu/.

References

Sekhon, Jasjeet S. 2011. "Multivariate and Propensity Score Matching Software with Automated Balance Optimization." *Journal of Statistical Software* 42(7): 1-52. doi: 10.18637/jss.v042.i07

Diamond, Alexis and Jasjeet S. Sekhon. 2013. "Genetic Matching for Estimating Causal Effects: A General Multivariate Matching Method for Achieving Balance in Observational Studies." *Review of Economics and Statistics*. 95 (3): 932–945. http://sekhon.berkeley.edu/papers/GenMatch.pdf

Sekhon, Jasjeet Singh and Walter R. Mebane, Jr. 1998. "Genetic Optimization Using Derivatives: Theory and Application to Nonlinear Models." *Political Analysis*, 7: 187-210. http://sekhon.berkeley.edu/genoud/genoud.pdf

See Also

Also see Match, summary.Match, MatchBalance, genoud, balanceUV, qqstats, ks.boot, GerberGreenImai, lalonde

Examples

```
data(lalonde)
attach(lalonde)

#The covariates we want to match on
X = cbind(age, educ, black, hisp, married, nodegr, u74, u75, re75, re74)

#The covariates we want to obtain balance on
BalanceMat <- cbind(age, educ, black, hisp, married, nodegr, u74, u75, re75, re74,
```

```
                    I(re74*re75))

#
#Let's call GenMatch() to find the optimal weight to give each
#covariate in 'X' so as we have achieved balance on the covariates in
#'BalanceMat'. This is only an example so we want GenMatch to be quick
#so the population size has been set to be only 16 via the 'pop.size'
#option. This is *WAY* too small for actual problems.
#For details see http://sekhon.berkeley.edu/papers/MatchingJSS.pdf.
#
genout <- GenMatch(Tr=treat, X=X, BalanceMatrix=BalanceMat, estimand="ATE", M=1,
                   pop.size=16, max.generations=10, wait.generations=1)

#The outcome variable
Y=re78/1000

#
# Now that GenMatch() has found the optimal weights, let's estimate
# our causal effect of interest using those weights
#
mout <- Match(Y=Y, Tr=treat, X=X, estimand="ATE", Weight.matrix=genout)
summary(mout)

#
#Let's determine if balance has actually been obtained on the variables of interest
#
mb <- MatchBalance(treat~age +educ+black+ hisp+ married+ nodegr+ u74+ u75+
                   re75+ re74+ I(re74*re75),
                   match.out=mout, nboots=500)

# For more examples see: http://sekhon.berkeley.edu/matching/R.
```

GerberGreenImai *Gerber and Green Dataset used by Imai*

Description

This is the dataset used by Imai (2005) to replicate and evaluate the field experiment done by Gerber and Green (2000). The accompanying demo replicates Imai's propensity score model which is then used to estimate the causal effect of get-out-the-vote telephone calls on turnout.

Usage

```
data(GerberGreenImai)
```

Format

A data frame with 10829 observations on the following 26 variables.

PERSONS Number persons in household

WARD Ward of residence

QUESTION Asked to commit to voting

MAILGRP Sent mail

PHONEGRP Phone batch \#1

PERSNGRP Personal contact attempted

APPEAL Content of message

CONTACT Personal contact occurred

MAILINGS Number of mailings sent

AGE Age of respondent

MAJORPTY Democratic or Republican

VOTE96.0 Abstained in 1996

VOTE96.1 Voted in 1996

MAILCALL Phone batch \#2

VOTED98 Voted in 1998

PHNSCRPT Script read to phone respondents

DIS.MC Contacted by phone in batch \#2

DIS.PHN Contacted by phone in batch \#1

PHN.C Contacted by phone

PHNTRT1 Phone contact attempted (no blood or blood/civic)

PHNTRT2 Phone contact attempted (no blood)

PHN.C1 Contact occurred in phntrt1

PHN.C2 Contact occurred in phntrt2

NEW New voter

phone Contacted by phone

AGE2 Age squared

Details

The demo provided, entitled `GerberGreenImai`, uses Imai's propensity score model to estimate the causal effect of get-out-the-vote telephone calls on turnout. The propensity score model fails to balance age.

References

Gerber, Alan S. and Donald P. Green. 2000. "The Effects of Canvassing, Telephone Calls, and Direct Mail on Voter Turnout: A Field Experiment." *American Political Science Review* 94: 653-663.

Gerber, Alan S. and Donald P. Green. 2005. "Correction to Gerber and Green (2000), replication of disputed findings, and reply to Imai (2005)." *American Political Science Review* 99: 301-313.

Imai, Kosuke. 2005. "Do Get-Out-The-Vote Calls Reduce Turnout? The Importance of Statistical Methods for Field Experiments." *American Political Science Review* 99: 283-300.

Hansen, Ben B. Hansen and Jake Bowers. forthcoming. "Attributing Effects to a Cluster Randomized Get-Out-The-Vote Campaign." *Journal of the American Statistical Association*.

See Also

Also see Match and MatchBalance, GenMatch, balanceUV, ks.boot lalonde

---

ks.boot *Bootstrap Kolmogorov-Smirnov*

Description

This function executes a bootstrap version of the univariate Kolmogorov-Smirnov test which provides correct coverage even when the distributions being compared are not entirely continuous. Ties are allowed with this test unlike the traditional Kolmogorov-Smirnov test.

Usage

```
ks.boot(Tr, Co, nboots=1000, alternative = c("two.sided","less","greater"),
        print.level=0)
```

Arguments

Tr	A vector containing the treatment observations.
Co	A vector containing the control observations.
nboots	The number of bootstraps to be performed. These are, in fact, really Monte Carlo simulations which are preformed in order to determine the proper p-value from the empiric.
alternative	indicates the alternative hypothesis and must be one of '"two.sided"' (default), '"less"', or '"greater"'. You can specify just the initial letter. See ks.test for details.
print.level	If this is greater than 1, then the simulation count is printed out while the simulations are being done.

Value

ks.boot.pvalue	The bootstrap p-value of the Kolmogorov-Smirnov test for the hypothesis that the probability densities for both the treated and control groups are the same.
ks	Return object from ks.test.
nboots	The number of bootstraps which were completed.

Author(s)

Jasjeet S. Sekhon, UC Berkeley, <sekhon@berkeley.edu>, http://sekhon.berkeley.edu/.

References

Sekhon, Jasjeet S. 2011. "Multivariate and Propensity Score Matching Software with Automated Balance Optimization." *Journal of Statistical Software* 42(7): 1-52. doi: 10.18637/jss.v042.i07

Diamond, Alexis and Jasjeet S. Sekhon. 2013. "Genetic Matching for Estimating Causal Effects: A General Multivariate Matching Method for Achieving Balance in Observational Studies." *Review of Economics and Statistics*. 95 (3): 932–945. http://sekhon.berkeley.edu/papers/GenMatch.pdf

Abadie, Alberto. 2002. "Bootstrap Tests for Distributional Treatment Effects in Instrumental Variable Models." *Journal of the American Statistical Association*, 97:457 (March) 284-292.

See Also

Also see summary.ks.boot, qqstats, balanceUV, Match, GenMatch, MatchBalance, GerberGreenImai, lalonde

Examples

```
#
# Replication of Dehejia and Wahba psid3 model
#
# Dehejia, Rajeev and Sadek Wahba. 1999.``Causal Effects in
# Non-Experimental Studies: Re-Evaluating the Evaluation of Training
# Programs.''Journal of the American Statistical Association 94 (448):
# 1053-1062.
#
data(lalonde)

#
# Estimate the propensity model
#
glm1  <- glm(treat~age + I(age^2) + educ + I(educ^2) + black +
             hisp + married + nodegr + re74  + I(re74^2) + re75 + I(re75^2) +
             u74 + u75, family=binomial, data=lalonde)


#
#save data objects
#
X  <- glm1$fitted
Y  <- lalonde$re78
Tr  <- lalonde$treat

#
# one-to-one matching with replacement (the "M=1" option).
# Estimating the treatment effect on the treated (the "estimand" option which defaults to 0).
#
rr  <- Match(Y=Y,Tr=Tr,X=X,M=1);
summary(rr)

#
# Do we have balance on 1975 income after matching?
```

```
#
ks <- ks.boot(lalonde$re75[rr$index.treated], lalonde$re75[rr$index.control], nboots=500)
summary(ks)
```

`lalonde` *Lalonde Dataset*

Description

Dataset used by Dehejia and Wahba (1999) to evaluate propensity score matching.

Usage

```
data(lalonde)
```

Format

A data frame with 445 observations on the following 12 variables.

age age in years.

educ years of schooling.

black indicator variable for blacks.

hisp indicator variable for Hispanics.

married indicator variable for martial status.

nodegr indicator variable for high school diploma.

re74 real earnings in 1974.

re75 real earnings in 1975.

re78 real earnings in 1978.

u74 indicator variable for earnings in 1974 being zero.

u75 indicator variable for earnings in 1975 being zero.

treat an indicator variable for treatment status.

Details

Two demos are provided which use this dataset. The first, `DehejiaWahba`, replicates one of the models from Dehejia and Wahba (1999). The second demo, `AbadieImbens`, replicates the models produced by Abadie and Imbens in their Matlab code. Many of these models are found to produce good balance for the Lalonde data.

References

Dehejia, Rajeev and Sadek Wahba. 1999."Causal Effects in Non-Experimental Studies: Re-Evaluating the Evaluation of Training Programs." *Journal of the American Statistical Association* 94 (448): 1053-1062.

LaLonde, Robert. 1986. "Evaluating the Econometric Evaluations of Training Programs." *American Economic Review* 76:604-620.

See Also

Also see Match, GenMatch, MatchBalance, balanceUV, ks.boot, GerberGreenImai

---

Match *Multivariate and Propensity Score Matching Estimator for Causal Inference*

Description

Match implements a variety of algorithms for multivariate matching including propensity score, Mahalanobis and inverse variance matching. The function is intended to be used in conjunction with the MatchBalance function which determines the extent to which Match has been able to achieve covariate balance. In order to do propensity score matching, one should estimate the propensity model before calling Match, and then send Match the propensity score to use. Match enables a wide variety of matching options including matching with or without replacement, bias adjustment, different methods for handling ties, exact and caliper matching, and a method for the user to fine tune the matches via a general restriction matrix. Variance estimators include the usual Neyman standard errors, Abadie-Imbens standard errors, and robust variances which do not assume a homogeneous causal effect. The GenMatch function can be used to *automatically find balance* via a genetic search algorithm which determines the optimal weight to give each covariate.

Usage

```
Match(Y=NULL, Tr, X, Z = X, V = rep(1, length(Y)), estimand = "ATT", M = 1,
      BiasAdjust = FALSE, exact = NULL, caliper = NULL, replace=TRUE, ties=TRUE,
      CommonSupport=FALSE,Weight = 1, Weight.matrix = NULL, weights = NULL,
      Var.calc = 0, sample = FALSE, restrict=NULL, match.out = NULL,
      distance.tolerance = 1e-05, tolerance=sqrt(.Machine$double.eps),
      version="standard")
```

Arguments

Y	A vector containing the outcome of interest. Missing values are not allowed. An outcome vector is not required because the matches generated will be the same regardless of the outcomes. Of course, without any outcomes no causal effect estimates will be produced, only a matched dataset.
Tr	A vector indicating the observations which are in the treatment regime and those which are not. This can either be a logical vector or a real vector where 0 denotes control and 1 denotes treatment.
X	A matrix containing the variables we wish to match on. This matrix may contain the actual observed covariates or the propensity score or a combination of both. All columns of this matrix must have positive variance or Match will return an error.
Z	A matrix containing the covariates for which we wish to make bias adjustments.
V	A matrix containing the covariates for which the variance of the causal effect may vary. Also see the Var.calc option, which takes precedence.

`estimand` A character string for the estimand. The default estimand is "ATT", the sample average treatment effect for the treated. "ATE" is the sample average treatment effect, and "ATC" is the sample average treatment effect for the controls.

`M` A scalar for the number of matches which should be found. The default is one-to-one matching. Also see the `ties` option.

`BiasAdjust` A logical scalar for whether regression adjustment should be used. See the `Z` matrix.

`exact` A logical scalar or vector for whether exact matching should be done. If a logical scalar is provided, that logical value is applied to all covariates in `X`. If a logical vector is provided, a logical value should be provided for each covariate in `X`. Using a logical vector allows the user to specify exact matching for some but not other variables. When exact matches are not found, observations are dropped. `distance.tolerance` determines what is considered to be an exact match. The `exact` option takes precedence over the `caliper` option.

`caliper` A scalar or vector denoting the caliper(s) which should be used when matching. A caliper is the distance which is acceptable for any match. Observations which are outside of the caliper are dropped. If a scalar caliper is provided, this caliper is used for all covariates in `X`. If a vector of calipers is provided, a caliper value should be provided for each covariate in `X`. The caliper is interpreted to be in standardized units. For example, `caliper=.25` means that all matches not equal to or within .25 standard deviations of each covariate in `X` are dropped. Note that dropping observations generally changes the quantity being estimated.

`replace` A logical flag for whether matching should be done with replacement. Note that if `FALSE`, the order of matches generally matters. Matches will be found in the same order as the data are sorted. Thus, the match(es) for the first observation will be found first, the match(es) for the second observation will be found second, etc. Matching without replacement will generally increase bias. Ties are randomly broken when `replace==FALSE` —see the `ties` option for details.

`ties` A logical flag for whether ties should be handled deterministically. By default `ties==TRUE`. If, for example, one treated observation matches more than one control observation, the matched dataset will include the multiple matched control observations and the matched data will be weighted to reflect the multiple matches. The sum of the weighted observations will still equal the original number of observations. If `ties==FALSE`, ties will be randomly broken. *If the dataset is large and there are many ties, setting* `ties=FALSE` *often results in a large speedup.* Whether two potential matches are close enough to be considered tied, is controlled by the `distance.tolerance` option.

`CommonSupport` This logical flag implements the usual procedure by which observations outside of the common support of a variable (usually the propensity score) across treatment and control groups are discarded. The `caliper` option is to be preferred to this option because `CommonSupport`, consistent with the literature, only drops *outliers* and leaves *inliers* while the caliper option drops both. If `CommonSupport==TRUE`, common support will be enforced on the first variable in the `X` matrix. Note that dropping observations generally changes the quantity being estimated. Use of this option renders it impossible to use the returned objects `index.treated` and `index.control` to reconstruct the matched dataset.

The returned object mdata will, however, still contain the matched dataset. Seriously, don't use this option; use the caliper option instead.

Weight — A scalar for the type of weighting scheme the matching algorithm should use when weighting each of the covariates in X. The default value of 1 denotes that weights are equal to the inverse of the variances. 2 denotes the Mahalanobis distance metric, and 3 denotes that the user will supply a weight matrix (Weight.matrix). Note that if the user supplies a Weight.matrix, Weight will be automatically set to be equal to 3.

Weight.matrix — This matrix denotes the weights the matching algorithm uses when weighting each of the covariates in X—see the Weight option. This square matrix should have as many columns as the number of columns of the X matrix. This matrix is usually provided by a call to the GenMatch function which finds the optimal weight each variable should be given so as to achieve balance on the covariates.

For most uses, this matrix has zeros in the off-diagonal cells. This matrix can be used to weight some variables more than others. For example, if X contains three variables and we want to match as best as we can on the first, the following would work well:

```
> Weight.matrix <-diag(3)
> Weight.matrix[1,1] <-1000/var(X[,1])
> Weight.matrix[2,2] <-1/var(X[,2])
> Weight.matrix[3,3] <-1/var(X[,3])
```

This code changes the weights implied by the inverse of the variances by multiplying the first variable by a 1000 so that it is highly weighted. In order to enforce exact matching see the exact and caliper options.

weights — A vector the same length as Y which provides observation specific weights.

Var.calc — A scalar for the variance estimate that should be used. By default Var.calc=0 which means that homoscedasticity is assumed. For values of Var.calc > 0, robust variances are calculated using Var.calc matches.

sample — A logical flag for whether the population or sample variance is returned.

distance.tolerance — This is a scalar which is used to determine if distances between two observations are different from zero. Values less than distance.tolerance are deemed to be equal to zero. This option can be used to perform a type of optimal matching

tolerance — This is a scalar which is used to determine numerical tolerances. This option is used by numerical routines such as those used to determine if a matrix is singular.

restrict — A matrix which restricts the possible matches. This matrix has one row for each restriction and three columns. The first two columns contain the two observation numbers which are to be restricted (for example 4 and 20), and the third column is the restriction imposed on the observation-pair. Negative numbers in the third column imply that the two observations cannot be matched under any circumstances, and positive numbers are passed on as the distance between the two observations for the matching algorithm. The most commonly used positive restriction is 0 which implies that the two observations will always be matched.

Exclusion restrictions are even more common. For example, if we want to exclude the observation pair 4 and 20 and the pair 6 and 55 from being matched, the restrict matrix would be: `restrict=rbind(c(4,20,-1),c(6,55,-1))`

`match.out` The return object from a previous call to `Match`. If this object is provided, then `Match` will use the matches found by the previous invocation of the function. Hence, `Match` will run faster. This is useful when the treatment does not vary across calls to `Match` and one wants to use the same set of matches as found before. This often occurs when one is trying to estimate the causal effect of the same treatment (`Tr`) on different outcomes (`Y`). When using this option, be careful to use the same arguments as used for the previous invocation of `Match` unless you know exactly what you are doing.

`version` The version of the code to be used. The "fast" C/C++ version of the code does not calculate Abadie-Imbens standard errors. Additional speed can be obtained by setting `ties=FALSE` or `replace=FALSE` if the dataset is large and/or has many ties. The "legacy" version of the code does not make a call to an optimized C/C++ library and is included only for historical compatibility. The "fast" version of the code is significantly faster than the "standard" version for large datasets, and the "legacy" version is much slower than either of the other two.

Details

This function is intended to be used in conjunction with the `MatchBalance` function which checks if the results of this function have actually achieved balance. The results of this function can be summarized by a call to the `summary.Match` function. If one wants to do propensity score matching, one should estimate the propensity model before calling `Match`, and then place the fitted values in the `X` matrix—see the provided example.

The `GenMatch` function can be used to *automatically find balance* by the use of a genetic search algorithm which determines the optimal weight to give each covariate. The object returned by `GenMatch` can be supplied to the `Weight.matrix` option of `Match` to obtain estimates.

`Match` is often much faster with large datasets if `ties=FALSE` or `replace=FALSE`—i.e., if matching is done by randomly breaking ties or without replacement. Also see the `Matchby` function. It provides a wrapper for `Match` which is much faster for large datasets when it can be used.

Three demos are included: `GerberGreenImai`, `DehejiaWahba`, and `AbadieImbens`. These can be run by calling the `demo` function such as by `demo(DehejiaWahba)`.

Value

`est` The estimated average causal effect.

`se` The Abadie-Imbens standard error. This standard error has correct coverage if `X` consists of either covariates or a known propensity score because it takes into account the uncertainty of the matching procedure. If an estimated propensity score is used, the uncertainty involved in its estimation is not accounted for although the uncertainty of the matching procedure itself still is.

`est.noadj` The estimated average causal effect without any `BiasAdjust`. If `BiasAdjust` is not requested, this is the same as `est`.

`se.standard` The usual standard error. This is the standard error calculated on the matched data using the usual method of calculating the difference of means (between treated and control) weighted by the observation weights provided by `weights`. Note that the standard error provided by `se` takes into account the uncertainty of the matching procedure while `se.standard` does not. Neither `se` nor `se.standard` take into account the uncertainty of estimating a propensity score. `se.standard` does not take into account any `BiasAdjust`. Summary of both types of standard error results can be requested by setting the `full=TRUE` flag when using the `summary.Match` function on the object returned by `Match`.

`se.cond` The conditional standard error. The practitioner should not generally use this.

`mdata` A list which contains the matched datasets produced by `Match`. Three datasets are included in this list: `Y`, `Tr` and `X`.

`index.treated` A vector containing the observation numbers from the original dataset for the treated observations in the matched dataset. This index in conjunction with `index.control` can be used to recover the matched dataset produced by `Match`. For example, the `X` matrix used by `Match` can be recovered by `rbind(X[index.treated,],X[index.con` The user should generally just examine the output of `mdata`.

`index.control` A vector containing the observation numbers from the original data for the control observations in the matched data. This index in conjunction with `index.treated` can be used to recover the matched dataset produced by `Match`. For example, the `X` matrix used by `Match` can be recovered by `rbind(X[index.treated,],X[index.control,])`. The user should generally just examine the output of `mdata`.

`index.dropped` A vector containing the observation numbers from the original data which were dropped (if any) in the matched dataset because of various options such as `caliper` and `exact`. If no observations were dropped, this index will be `NULL`.

`weights` A vector of weights. There is one weight for each matched-pair in the matched dataset. If all of the observations had a weight of 1 on input, then each matched-pair will have a weight of 1 on output if there are no ties.

`orig.nobs` The original number of observations in the dataset.

`orig.wnobs` The original number of weighted observations in the dataset.

`orig.treated.nobs` The original number of treated observations (unweighted).

`nobs` The number of observations in the matched dataset.

`wnobs` The number of weighted observations in the matched dataset.

`caliper` The `caliper` which was used.

`ecaliper` The size of the enforced caliper on the scale of the `X` variables. This object has the same length as the number of covariates in `X`.

`exact` The value of the `exact` function argument.

`ndrops` The number of weighted observations which were dropped either because of caliper or exact matching. This number, unlike `ndrops.matches`, takes into account observation specific weights which the user may have provided via the `weights` argument.

`ndrops.matches` The number of matches which were dropped either because of caliper or exact matching.

Author(s)

Jasjeet S. Sekhon, UC Berkeley, <sekhon@berkeley.edu>, http://sekhon.berkeley.edu/.

References

Sekhon, Jasjeet S. 2011. "Multivariate and Propensity Score Matching Software with Automated Balance Optimization." *Journal of Statistical Software* 42(7): 1-52. doi: 10.18637/jss.v042.i07

Diamond, Alexis and Jasjeet S. Sekhon. 2013. "Genetic Matching for Estimating Causal Effects: A General Multivariate Matching Method for Achieving Balance in Observational Studies." *Review of Economics and Statistics*. 95 (3): 932–945. http://sekhon.berkeley.edu/papers/GenMatch.pdf

Abadie, Alberto and Guido Imbens. 2006. "Large Sample Properties of Matching Estimators for Average Treatment Effects." *Econometrica* 74(1): 235-267.

Imbens, Guido. 2004. Matching Software for Matlab and Stata.

See Also

Also see summary.Match, GenMatch, MatchBalance, Matchby, balanceUV, qqstats, ks.boot, GerberGreenImai, lalonde

Examples

```
# Replication of Dehejia and Wahba psid3 model
#
# Dehejia, Rajeev and Sadek Wahba. 1999.``Causal Effects in
# Non-Experimental Studies: Re-Evaluating the Evaluation of Training
# Programs.''Journal of the American Statistical Association 94 (448):
# 1053-1062.

data(lalonde)

#
# Estimate the propensity model
#
glm1  <- glm(treat~age + I(age^2) + educ + I(educ^2) + black +
             hisp + married + nodegr + re74  + I(re74^2) + re75 + I(re75^2) +
             u74 + u75, family=binomial, data=lalonde)


#
#save data objects
#
X  <- glm1$fitted
Y  <- lalonde$re78
Tr  <- lalonde$treat

#
# one-to-one matching with replacement (the "M=1" option).
# Estimating the treatment effect on the treated (the "estimand" option defaults to ATT).
```

```
#
rr  <- Match(Y=Y, Tr=Tr, X=X, M=1);
summary(rr)

# Let's check the covariate balance
# 'nboots' is set to small values in the interest of speed.
# Please increase to at least 500 each for publication quality p-values.
mb  <- MatchBalance(treat~age + I(age^2) + educ + I(educ^2) + black +
                    hisp + married + nodegr + re74  + I(re74^2) + re75 + I(re75^2) +
                    u74 + u75, data=lalonde, match.out=rr, nboots=10)
```

MatchBalance *Tests for Univariate and Multivariate Balance*

Description

This function provides a variety of balance statistics useful for determining if balance exists in any unmatched dataset and in matched datasets produced by the Match function. Matching is performed by the Match function, and MatchBalance is used to determine if Match was successful in achieving balance on the observed covariates.

Usage

```
MatchBalance(formul, data = NULL, match.out = NULL, ks = TRUE,
             nboots=500, weights=NULL, digits=5, paired=TRUE, print.level=1)
```

Arguments

formul	This formula does *not* estimate any model. The formula is simply an efficient way to use the R modeling language to list the variables we wish to obtain univariate balance statistics for. The dependent variable in the formula is usually the treatment indicator. One should include many functions of the observed covariates. Generally, one should request balance statistics on more higher-order terms and interactions than were used to conduct the matching itself.
data	A data frame which contains all of the variables in the formula. If a data frame is not provided, the variables are obtained via lexical scoping.
match.out	The output object from the Match function. If this output is included, MatchBalance will provide balance statistics for both before and after matching. Otherwise balance statistics will only be reported for the raw unmatched data.
ks	A logical flag for whether the univariate bootstrap Kolmogorov-Smirnov (KS) test should be calculated. If the ks option is set to true, the univariate KS test is calculated for all non-dichotomous variables. The bootstrap KS test is consistent even for non-continuous variables. See ks.boot for more details.
weights	An optional vector of observation specific weights.

`nboots` The number of bootstrap samples to be run. If zero, no bootstraps are done. Bootstrapping is highly recommended because the bootstrapped Kolmogorov-Smirnov test provides correct coverage even when the distributions being compared are not continuous. At least 500 `nboots` (preferably 1000) are recommended for publication quality p-values.

`digits` The number of significant digits that should be displayed.

`paired` A flag for whether the paired `t.test` should be used after matching. Regardless of the value of this option, an unpaired `t.test` is done for the unmatched data because it is assumed that the unmatched data were not generated by a paired experiment.

`print.level` The amount of printing to be done. If zero, there is no printing. If one, the results are summarized. If two, details of the computations are printed.

Details

This function can be used to determine if there is balance in the pre- and/or post-matching datasets. Difference of means between treatment and control groups are provided as well as a variety of summary statistics for the empirical CDF (eCDF) and empirical-QQ (eQQ) plot between the two groups. The eCDF results are the standardized mean, median and maximum differences in the empirical CDF. The eQQ results are summaries of the raw differences in the empirical-QQ plot.

Two univariate tests are also provided: the t-test and the bootstrap Kolmogorov-Smirnov (KS) test. These tests should not be treated as hypothesis tests in the usual fashion because we wish to maximize balance without limit. The bootstrap KS test is highly recommended (see the `ks` and `nboots` options) because the bootstrap KS is consistent even for non-continuous distributions. Before matching, the two sample t-test is used; after matching, the paired t-test is used.

Two multivariate tests are provided. The KS and Chi-Square null deviance tests. The KS test is to be preferred over the Chi-Square test because the Chi-Square test is not testing the relevant hypothesis. The null hypothesis for the KS test is equal balance in the estimated probabilities between treated and control. The null hypothesis for the Chi-Square test, however, is all of the parameters being insignificant; a comparison of residual versus null deviance. If the covariates being considered are discrete, this KS test is asymptotically nonparametric as long as the logit model does not produce zero parameter estimates.

`NA`'s are handled by the `na.action` option. But it is highly recommended that `NA`'s not simply be deleted, but one should check to make sure that missingness is balanced.

Value

`BeforeMatching` A list containing the before matching univariate balance statistics. That is, a list containing the results of the `balanceUV` function applied to all of the covariates described in `formul`. Note that the univariate test results for all of the variables in `formul` are printed if `verbose` > `0`.

`AfterMatching` A list containing the after matching univariate balance statistics. That is, a list containing the results of the `balanceUV` function applied to all of the covariates described in `formul`. Note that the univariate test results for all of the variables

in `formul` are printed if `verbose > 0`. This object is `NULL`, if no matched dataset was provided.

`BMsmallest.p.value`
: The smallest p.value found across all of the *before* matching balance tests (including t-tests and KS-tests.

`BMsmallestVarName`
: The name of the variable with the `BMsmallest.p.value` (a vector in case of ties).

`BMsmallestVarNumber`
: The number of the variable with the `BMsmallest.p.value` (a vector in case of ties).

`AMsmallest.p.value`
: The smallest p.value found across all of the *after* matching balance tests (including t-tests and KS-tests.

`AMsmallestVarName`
: The name of the variable with the `AMsmallest.p.value` (a vector in case of ties).

`AMsmallestVarNumber`
: The number of the variable with the `AMsmallest.p.value` (a vector in case of ties).

Author(s)

Jasjeet S. Sekhon, UC Berkeley, <sekhon@berkeley.edu>, http://sekhon.berkeley.edu/.

References

Sekhon, Jasjeet S. 2011. "Multivariate and Propensity Score Matching Software with Automated Balance Optimization." *Journal of Statistical Software* 42(7): 1-52. doi: 10.18637/jss.v042.i07

Diamond, Alexis and Jasjeet S. Sekhon. 2013. "Genetic Matching for Estimating Causal Effects: A General Multivariate Matching Method for Achieving Balance in Observational Studies." *Review of Economics and Statistics*. 95 (3): 932–945. http://sekhon.berkeley.edu/papers/GenMatch.pdf

Abadie, Alberto. 2002. "Bootstrap Tests for Distributional Treatment Effects in Instrumental Variable Models." *Journal of the American Statistical Association*, 97:457 (March) 284-292.

Hall, Peter. 1992. *The Bootstrap and Edgeworth Expansion*. New York: Springer-Verlag.

Wilcox, Rand R. 1997. *Introduction to Robust Estimation*. San Diego, CA: Academic Press.

William J. Conover (1971), *Practical nonparametric statistics*. New York: John Wiley & Sons. Pages 295-301 (one-sample "Kolmogorov" test), 309-314 (two-sample "Smirnov" test).

Shao, Jun and Dongsheng Tu. 1995. *The Jackknife and Bootstrap*. New York: Springer-Verlag.

See Also

Also see `Match`, `GenMatch`, `balanceUV`, `qqstats`, `ks.boot`, `GerberGreenImai`, `lalonde`

Examples

```
#
# Replication of Dehejia and Wahba psid3 model
#
# Dehejia, Rajeev and Sadek Wahba. 1999.``Causal Effects in
# Non-Experimental Studies: Re-Evaluating the Evaluation of Training
# Programs.''Journal of the American Statistical Association 94 (448):
# 1053-1062.

data(lalonde)

#
# Estimate the propensity model
#
glm1  <- glm(treat~age + I(age^2) + educ + I(educ^2) + black +
             hisp + married + nodegr + re74  + I(re74^2) + re75 + I(re75^2) +
             u74 + u75, family=binomial, data=lalonde)


#
#save data objects
#
X  <- glm1$fitted
Y  <- lalonde$re78
Tr  <- lalonde$treat

#
# one-to-one matching with replacement (the "M=1" option).
# Estimating the treatment effect on the treated (the "estimand" option which defaults to 0).
#
rr  <- Match(Y=Y,Tr=Tr,X=X,M=1);

#Let's summarize the output
summary(rr)

# Let's check the covariate balance
# 'nboots' is set to small values in the interest of speed.
# Please increase to at least 500 each for publication quality p-values.
mb  <- MatchBalance(treat~age + I(age^2) + educ + I(educ^2) + black +
                    hisp + married + nodegr + re74  + I(re74^2) + re75 + I(re75^2) +
                    u74 + u75, data=lalonde, match.out=rr, nboots=10)
```

Matchby *Grouped Multivariate and Propensity Score Matching*

Description

This function is a wrapper for the Match function which separates the matching problem into subgroups defined by a factor. This is equivalent to conducting exact matching on each level of a factor. Matches within each level are found as determined by the usual matching options. This function is

much faster for large datasets than the Match function itself. For additional speed, consider doing matching without replacement—see the replace option. This function is more limited than the Match function. For example, Matchby cannot be used if the user wishes to provide observation specific weights.

Usage

```
Matchby(Y, Tr, X, by, estimand = "ATT", M = 1, ties=FALSE, replace=TRUE,
        exact = NULL, caliper = NULL, AI=FALSE, Var.calc=0,
        Weight = 1, Weight.matrix = NULL, distance.tolerance = 1e-05,
     tolerance = sqrt(.Machine$double.eps), print.level=1, version="Matchby", ...)
```

Arguments

Y	A vector containing the outcome of interest. Missing values are not allowed.
Tr	A vector indicating the observations which are in the treatment regime and those which are not. This can either be a logical vector or a real vector where 0 denotes control and 1 denotes treatment.
X	A matrix containing the variables we wish to match on. This matrix may contain the actual observed covariates or the propensity score or a combination of both.
by	A "factor" in the sense that as.factor(by) defines the grouping, or a list of such factors in which case their interaction is used for the grouping.
estimand	A character string for the estimand. The default estimand is "ATT", the sample average treatment effect for the treated. "ATE" is the sample average treatment effect (for all), and "ATC" is the sample average treatment effect for the controls.
M	A scalar for the number of matches which should be found. The default is one-to-one matching. Also see the ties option.
ties	A logical flag for whether ties should be handled deterministically. By default ties==TRUE. If, for example, one treated observation matches more than one control observation, the matched dataset will include the multiple matched control observations and the matched data will be weighted to reflect the multiple matches. The sum of the weighted observations will still equal the original number of observations. If ties==FALSE, ties will be randomly broken. *If the dataset is large and there are many ties, setting* ties=FALSE *often results in a large speedup.* Whether two potential matches are close enough to be considered tied, is controlled by the distance.tolerance option.
replace	Whether matching should be done with replacement. Note that if FALSE, the order of matches generally matters. Matches will be found in the same order as the data is sorted. Thus, the match(es) for the first observation will be found first and then for the second etc. Matching without replacement will generally increase bias so it is not recommended. *But if the dataset is large and there are many potential matches, setting* replace=false *often results in a large speedup and negligible or no bias.* Ties are randomly broken when replace==FALSE—see the ties option for details.
exact	A logical scalar or vector for whether exact matching should be done. If a logical scalar is provided, that logical value is applied to all covariates of X. If a logical

vector is provided, a logical value should be provided for each covariate in X. Using a logical vector allows the user to specify exact matching for some but not other variables. When exact matches are not found, observations are dropped. distance.tolerance determines what is considered to be an exact match. The exact option takes precedence over the caliper option.

caliper — A scalar or vector denoting the caliper(s) which should be used when matching. A caliper is the distance which is acceptable for any match. Observations which are outside of the caliper are dropped. If a scalar caliper is provided, this caliper is used for all covariates in X. If a vector of calipers is provided, a caliper value should be provide for each covariate in X. The caliper is interpreted to be in standardized units. For example, caliper=.25 means that all matches not equal to or within .25 standard deviations of each covariate in X are dropped.

AI — A logical flag for if the Abadie-Imbens standard error should be calculated. It is computationally expensive to calculate with large datasets. Matchby can only calculate AI SEs for ATT. To calculate AI errors with other estimands, please use the Match function. See the Var.calc option if one does not want to assume homoscedasticity.

Var.calc — A scalar for the variance estimate that should be used. By default Var.calc=0 which means that homoscedasticity is assumed. For values of Var.calc > 0, robust variances are calculated using Var.calc matches.

Weight — A scalar for the type of weighting scheme the matching algorithm should use when weighting each of the covariates in X. The default value of 1 denotes that weights are equal to the inverse of the variances. 2 denotes the Mahalanobis distance metric, and 3 denotes that the user will supply a weight matrix (Weight.matrix). Note that if the user supplies a Weight.matrix, Weight will be automatically set to be equal to 3.

Weight.matrix — This matrix denotes the weights the matching algorithm uses when weighting each of the covariates in X—see the Weight option. This square matrix should have as many columns as the number of columns of the X matrix. This matrix is usually provided by a call to the GenMatch function which finds the optimal weight each variable should be given so as to achieve balance on the covariates.

For most uses, this matrix has zeros in the off-diagonal cells. This matrix can be used to weight some variables more than others. For example, if X contains three variables and we want to match as best as we can on the first, the following would work well:

```
> Weight.matrix <-diag(3)
> Weight.matrix[1,1] <-1000/var(X[,1])
> Weight.matrix[2,2] <-1/var(X[,2])
> Weight.matrix[3,3] <-1/var(X[,3])
```

This code changes the weights implied by the inverse of the variances by multiplying the first variable by a 1000 so that it is highly weighted. In order to enforce exact matching see the exact and caliper options.

distance.tolerance — This is a scalar which is used to determine if distances between two observations are different from zero. Values less than distance.tolerance are deemed to be equal to zero. This option can be used to perform a type of optimal matching

tolerance	This is a scalar which is used to determine numerical tolerances. This option is used by numerical routines such as those used to determine if a matrix is singular.
print.level	The level of printing. Set to '0' to turn off printing.
version	The version of the code to be used. The "Matchby" C/C++ version of the code is the fastest, and the end-user should not change this option.
...	Additional arguments passed on to Match.

Details

Matchby is much faster for large datasets than Match. But Matchby only implements a subset of the functionality of Match. For example, the restrict option cannot be used, Abadie-Imbens standard errors are not provided and bias adjustment cannot be requested. Matchby is a wrapper for the Match function which separates the matching problem into subgroups defined by a factor. This is the equivalent to doing exact matching on each factor, and the way in which matches are found within each factor is determined by the usual matching options.

Note that by default ties=FALSE *although the default for the* Match *in* GenMatch *functions is* TRUE. *This is done because randomly breaking ties in large datasets often results in a great speedup.* For additional speed, consider doing matching without replacement which is often much faster when the dataset is large—see the replace option.

There will be slight differences in the matches produced by Matchby and Match because of how the covariates are weighted. When the data is broken up into separate groups (via the by option), Mahalanobis distance and inverse variance will imply different weights than when the data is taken as whole.

Value

est	The estimated average causal effect.
se.standard	The usual standard error. This is the standard error calculated on the matched data using the usual method of calculating the difference of means (between treated and control) weighted so that ties are taken into account.
se	The Abadie-Imbens standard error. This is only calculated if the AI option is TRUE. This standard error has correct coverage if X consists of either covariates or a known propensity score because it takes into account the uncertainty of the matching procedure. If an estimated propensity score is used, the uncertainty involved in its estimation is not accounted for although the uncertainty of the matching procedure itself still is.
index.treated	A vector containing the observation numbers from the original dataset for the treated observations in the matched dataset. This index in conjunction with index.control can be used to recover the matched dataset produced by Matchby. For example, the X matrix used by Matchby can be recovered by rbind(X[index.treated,],X[index.c
index.control	A vector containing the observation numbers from the original data for the control observations in the matched data. This index in conjunction with index.treated can be used to recover the matched dataset produced by Matchby. For example, the Y matrix for the matched dataset can be recovered by c(Y[index.treated],Y[index.control]).

weights	The weights for each observation in the matched dataset.
orig.nobs	The original number of observations in the dataset.
nobs	The number of observations in the matched dataset.
wnobs	The number of weighted observations in the matched dataset.
orig.treated.nobs	The original number of treated observations.
ndrops	The number of matches which were dropped because there were not enough observations in a given group and because of caliper and exact matching.
estimand	The estimand which was estimated.
version	The version of Match which was used.

Author(s)

Jasjeet S. Sekhon, UC Berkeley, <sekhon@berkeley.edu>, http://sekhon.berkeley.edu/.

References

Sekhon, Jasjeet S. 2011. "Multivariate and Propensity Score Matching Software with Automated Balance Optimization." *Journal of Statistical Software* 42(7): 1-52. doi: 10.18637/jss.v042.i07

Diamond, Alexis and Jasjeet S. Sekhon. 2013. "Genetic Matching for Estimating Causal Effects: A General Multivariate Matching Method for Achieving Balance in Observational Studies." *Review of Economics and Statistics*. 95 (3): 932–945. http://sekhon.berkeley.edu/papers/GenMatch.pdf

Abadie, Alberto and Guido Imbens. 2006. "Large Sample Properties of Matching Estimators for Average Treatment Effects." *Econometrica* 74(1): 235-267.

Imbens, Guido. 2004. Matching Software for Matlab and Stata.

See Also

Also see Match, summary.Matchby, GenMatch, MatchBalance, balanceUV, qqstats, ks.boot, GerberGreenImai, lalonde

Examples

```
#
# Match exactly by racial groups and then match using the propensity score within racial groups
#

data(lalonde)

#
# Estimate the Propensity Score
#
glm1  <- glm(treat~age + I(age^2) + educ + I(educ^2) +
             hisp + married + nodegr + re74  + I(re74^2) + re75 + I(re75^2) +
             u74 + u75, family=binomial, data=lalonde)
```

```
#save data objects
#
X  <- glm1$fitted
Y  <- lalonde$re78
Tr <- lalonde$treat

# one-to-one matching with replacement (the "M=1" option) after exactly
# matching on race using the 'by' option.  Estimating the treatment
# effect on the treated (the "estimand" option defaults to ATT).
rr  <- Matchby(Y=Y, Tr=Tr, X=X, by=lalonde$black, M=1);
summary(rr)

# Let's check the covariate balance
# 'nboots' is set to small values in the interest of speed.
# Please increase to at least 500 each for publication quality p-values.
mb  <- MatchBalance(treat~age + I(age^2) + educ + I(educ^2) + black +
                    hisp + married + nodegr + re74  + I(re74^2) + re75 + I(re75^2) +
                    u74 + u75, data=lalonde, match.out=rr, nboots=10)
```

`qqstats` *QQ Summary Statistics*

Description

This function calculates a set of summary statistics for the QQ plot of two samples of data. The summaries are useful for determining if the two samples are from the same distribution. If `standardize==TRUE`, the empirical CDF is used instead of the empirical-QQ plot. The later retains the scale of the variable.

Usage

```
qqstats(x, y, standardize=TRUE, summary.func)
```

Arguments

`x`	The first sample.
`y`	The second sample.
`standardize`	A logical flag for whether the statistics should be standardized by the empirical cumulative distribution functions of the two samples.
`summary.func`	A user provided function to summarize the difference between the two distributions. The function should expect a vector of the differences as an argument and return summary statistic. For example, the `quantile` function is a legal function to pass in.

Value

meandiff	The mean difference between the QQ plots of the two samples.
mediandiff	The median difference between the QQ plots of the two samples.
maxdiff	The maximum difference between the QQ plots of the two samples.
summarydiff	If the user provides a summary.func, the user requested summary difference is returned.
summary.func	If the user provides a summary.func, the function is returned.

Author(s)

Jasjeet S. Sekhon, UC Berkeley, <sekhon@berkeley.edu>, http://sekhon.berkeley.edu/.

References

Sekhon, Jasjeet S. 2011. "Multivariate and Propensity Score Matching Software with Automated Balance Optimization." *Journal of Statistical Software* 42(7): 1-52. doi: 10.18637/jss.v042.i07

Diamond, Alexis and Jasjeet S. Sekhon. Forthcoming. "Genetic Matching for Estimating Causal Effects: A General Multivariate Matching Method for Achieving Balance in Observational Studies." *Review of Economics and Statistics*. http://sekhon.berkeley.edu/papers/GenMatch.pdf

See Also

Also see ks.boot, balanceUV, Match, GenMatch, MatchBalance, GerberGreenImai, lalonde

Examples

```
#
# Replication of Dehejia and Wahba psid3 model
#
# Dehejia, Rajeev and Sadek Wahba. 1999.``Causal Effects in
# Non-Experimental Studies: Re-Evaluating the Evaluation of Training
# Programs.''Journal of the American Statistical Association 94 (448):
# 1053-1062.
#

data(lalonde)

#
# Estimate the propensity model
#
glm1  <- glm(treat~age + I(age^2) + educ + I(educ^2) + black +
             hisp + married + nodegr + re74  + I(re74^2) + re75 + I(re75^2) +
             u74 + u75, family=binomial, data=lalonde)


#
#save data objects
#
X  <- glm1$fitted
```

```
Y  <- lalonde$re78
Tr  <- lalonde$treat

#
# one-to-one matching with replacement (the "M=1" option).
# Estimating the treatment effect on the treated (the "estimand" option which defaults to 0).
#
rr  <- Match(Y=Y,Tr=Tr,X=X,M=1);
summary(rr)

#
# Do we have balance on 1975 income after matching?
#
qqout  <- qqstats(lalonde$re75[rr$index.treated], lalonde$re75[rr$index.control])
print(qqout)
```

---

summary.balanceUV *Summarizing output from balanceUV*

Description

summary method for class balanceUV

Usage

```
  ## S3 method for class 'balanceUV'
summary(object, ..., digits=5)
```

Arguments

object	An object of class "balanceUV", usually, a result of a call to balanceUV.
digits	The number of significant digits that should be displayed.
...	Other options for the generic summary function.

Author(s)

Jasjeet S. Sekhon, UC Berkeley, <sekhon@berkeley.edu>, http://sekhon.berkeley.edu/.

See Also

Also see balanceUV, Match, GenMatch, MatchBalance, qqstats, ks.boot, GerberGreenImai, lalonde

summary.ks.boot *Summarizing output from ks.boot*

Description

summary method for class ks.boot

Usage

```
  ## S3 method for class 'ks.boot'
summary(object, ..., digits=5)
```

Arguments

object	An object of class "ks.boot", usually, a result of a call to ks.boot.
digits	The number of significant digits that should be displayed.
...	Other options for the generic summary function.

Author(s)

Jasjeet S. Sekhon, UC Berkeley, <sekhon@berkeley.edu>, http://sekhon.berkeley.edu/.

See Also

Also see ks.boot, balanceUV, qqstats, Match, GenMatch, MatchBalance, GerberGreenImai, lalonde

summary.Match *Summarizing output from Match*

Description

summary method for class Match

Usage

```
  ## S3 method for class 'Match'
summary(object, ... , full=FALSE, digits=5)
```

Arguments

object	An object of class "Match", usually, a result of a call to Match.
full	A flag for whether the unadjusted estimates and naive standard errors should also be summarized.
digits	The number of significant digits that should be displayed.
...	Other options for the generic summary function.

Author(s)

Jasjeet S. Sekhon, UC Berkeley, <sekhon@berkeley.edu>, http://sekhon.berkeley.edu/.

See Also

Also see Match, GenMatch, MatchBalance, balanceUV, qqstats, ks.boot, GerberGreenImai, lalonde

---

summary.Matchby *Summarizing output from Matchby*

Description

summary method for class Matchby

Usage

```
  ## S3 method for class 'Matchby'
summary(object, ... , digits=5)
```

Arguments

object	An object of class "Matchby", usually, a result of a call to Matchby.
digits	The number of significant digits that should be displayed.
...	Other options for the generic summary function.

Author(s)

Jasjeet S. Sekhon, UC Berkeley, <sekhon@berkeley.edu>, http://sekhon.berkeley.edu/.

See Also

Also see Matchby, Match, GenMatch, MatchBalance, balanceUV, qqstats, ks.boot, GerberGreenImai, lalonde

Index