Section 9: Matching without replacement and Genetic Matching

Yotam Shem-Tov

Fall 2015

Matching without replacement

- The standard approach is to minimize the Mahalanobis distance matrix (In GenMatch we use a weighted Mahalanobis distance)
- We can also use other distance functions such as Euclidean distance, however Mahalanobis distance has attractive properties
- Matching with replacement will yield the same results using Euclidean, absolute, or Mahalanobis distance
- Next we consider Matching using Mahalnobis distance without replacement

Matching without replacement using Mahalanobis distance

- Suppose we have $X_1, \ldots, X_n$ with each variable X_i having k components (covariates): $X_i = (X_{i1}, X_{i2}, \ldots, X_{ik})$
- Suppose we have the $k \times k$ covariance matrix of these covariates. (either by estimating this matrix, or given to us). Call this covariance matrix S
- The Mahalanobis distance between X_i and X_j is,

$$md(X_i, X_j) = \sqrt{[(X_i - X_j)^T S^{-1} (X_i - X_j)]}$$

- Next we review a numerical example to understand Mahalanobis distances.

- The data:

```
set.seed(12345); n=10
x1 = c(rnorm(n/2),rnorm(n/2,mean=1))
x2 = c(rexp(n/2),rexp(n/2,rate=1.2))
tr= c(rep(1,n/2),rep(0,n/2))
X1 <- cbind(x1[tr==1],x2[tr==1])
X0 <- cbind(x1[tr==0],x2[tr==0])
```

- Write code that generates a distance matrix. The rows are the indexes of the treated observations and the columns are the indexes of the control observations.

```
Sigma = var(cbind(x1,x2))
for (j in c(1:(n/2))){
  for (i in c(1:(n/2))){
    dis.mat[i,j] = sqrt(matrix(X1[i,]-X0[j,],ncol=2)
    %*%solve(Sigma)%*%matrix(X1[i,]-X0[j,],ncol=1))
  }
}
rownames(dis.mat)=which(tr==1)
colnames(dis.mat)=which(tr==0)
```

The numerical example results

```
> print(dis.mat)
          6        7        8         9         10
1 1.9977643 1.494551 2.888388 0.5034378 0.7517883
2 2.2497128 1.347747 3.371233 0.9576349 0.9326353
3 1.0323755 2.473257 3.233327 1.3153698 0.2869124
4 0.9285015 3.138516 2.617814 1.7100774 1.3305647
5 2.0960122 1.482263 3.331142 0.9266216 0.7788730
```

Matching without replacement using Mahalanobis distance

- How to implement Mahalanobis distance in *R*:
  1. The *Match* function in the package *Matching*.
  2. Write your own function that calculates Mahalanobis distance, and creates matches to minimizes the distance matrix between the treated and the control units
- The code for both options is in the next slides using data from Dehejia and Wahba (1999). The results are very similar, the differences result in the minimization of the Mahalanobis distance matrix

Mahalanobis distance: Matching w/t replacement

Dehejia and Wahba (1999)

```
### Using Jas's "Match" function:
match = Match(Tr=treat1,X=x1,replace=FALSE,Weight=2)
# or using the option "Weight=1", there is not much difference

dm2 = d1[c(match$index.treated,match$index.control),]
```

Mahalanobis distance: Matching w/t replacement

```
smahal = function(z,X) {
  X = as.matrix(X)
  n = dim(X)[1]
  rownames (X)<-1:n
  k<-dim(X)[2]
  m<-sum(z)
  for (j in 1:k) {
    X[,j] = rank(X[,j])
  }
  cv <- cov(X)
  vuntied <- var(1:n)
  rat <- sqrt(vuntied/diag(cv))
  cv <- diag(rat)%*%cv%*%diag(rat)
  # distance matrix containing: rows are the treatment group and columns are the co
  out <- matrix(NA,m,n-m)
  Xc = X[z==0,]; Xt = X[z==1,];
  rownames(out)=rownames(X)[z==1]
  colnames(out)=rownames(X)[z==0]
  library(MASS)
  icov <- ginv(cv)
  for (i in 1:m) {
    out[i,] = mahalanobis(Xc,Xt[i,],icov,inverted=T)
  }
  out }
```

Mahalanobis distance: Matching w/t replacement

Dehejia and Wahba (1999)

```
source("smahal.R")
### data without the experimental controls with CPS 3
d1 = read.dta("cps1re74.dta")

### CPS-1 vs NSW treatment group ###
x1 = d1[,c("age","ed","black","hisp","married","nodeg","re74","re75"
treat1 = d1$treat

# Calculating the Mahalanobis distance matrix using the function "sm
distance.matrix = smahal(treat1,x1)

library("optmatch")
pair.match <-pairmatch(distance.matrix, controls=1,data=d1)

# The matched data set:
dm = d1[is.na(pair.match)=="FALSE",]
```

Mahalanobis distance: Matching w/t replacement

	Jas's *Match* function		Code using package *optmatch*	
	Ave. control	T-test	Ave. control	T-test
age	25.189	0.445	25.330	0.562
ed	10.314	0.878	10.276	0.755
black	0.843	1.000	0.832	0.779
hisp	0.059	1.000	0.059	1.000
married	0.189	1.000	0.189	1.000
nodeg	0.708	1.000	0.708	1.000
re74	2631.729	0.270	2559.173	0.352
re75	2215.897	0.049	2211.832	0.060

- The treated units are not presented as they are the same in both matched data sets.
- The difference in the results follows from the numerical procedure used to minimize the Mahalanobis distance matrix. The *Match* package and the *optmatch* package use different algorithms.
- I recommend using the *Match* function, but understand what it does

What is the Mahalanobis distance doing: A special case

- Suppose covariates are ellipsoidal (in a crude sense, the data looks like an oval).
- Special case of ellipsoidal: Multivariate normal
- Transform the covariates into a circle, where the marginal SD is 1 for any covariate
- Equivalent to "Eliminating the correlation between covariates and scaling all covariates so that they are on the same scale"
- Calculate the distance within this circle
- Mahalanobis distance can be computed for any other original shape of data... but nice properties can be derived if the data is ellipsoidal

Equal Percent Bias Reduction (EPBR)

- Suppose we are matching treated units to control units
- matching method is Equal Percent Bias Reducing (EPBR) if, for ALL covariates X:

$$\begin{aligned}&\mathbb{E}(X|T=1) - \mathbb{E}(X|\textit{Matched Controls})\\ =&\gamma\left(\mathbb{E}(X|T=1) - \mathbb{E}(X|T=0)\right)\end{aligned}$$

  where $0 \leq \gamma \leq 1$.
- In words, the average covariate imbalance shrinks by γ for each covariate
- If data is ellipsoidal, then the matching with the Mahalanobis distance is EPBR
- EPBR may not be that attractive of a property; especially if you think that balance over some covariates (height, weight) is more important than others (number of hair follicles)
- EPBR shrinks the bias only on average, the bias in other percentiles is not guaranteed to reduce

General notes on matching

- EPBR does not mean that your estimated treatment effect will be less biased. It could be more biased if you do not have the right X s. Moreover, balance on covariates that you do not include could get worse
- Measurement error in some of the covariates can bias the matching treatment effect estimations. Is it attenuation bias?
- Is the following suggestion a good placebo test?

  "As a placebo test, redo the matching without the previous outcome and test the previous outcome. The two groups should be balanced on the previous outcome"

-

How can you tell if matching worked?

- Matching has necessary testable implication, which can be verified in the data. This are only necessary conditions, **not sufficient** conditions for inference. We cannot test for balance in unobservable...
- The objective of matching is to find people in the control group that look like people that are treated
- If matching works, the distribution of the covariates for the treated group should be the same as the distribution for the matched control group
- If data is quantitative, compare means of covariates between the treatment group and the matched control group (t-test)
- If data is categorical, compare proportions across the covariates. (Fisher exact test)
- These methods compare single points (a mean or proportion) between these two groups
- Kolmogorov-Smirnov test can test if theres discrepancies across the entire distribution

Example: Kane and Rouse (1995), Labor-Market Returns to Two- and Four-Year College

- We will use the paper, "Labor-Market Returns to Two-Year and Four-Year College" by Kane and Rouse (1995) to demonstrate matching without replacement and Genetic matching
- The data:

```
Outcome:  educ86, Years of Education Completed (as of 1986)
Treatment: twoyr (1 if started in a two year college,
0 if started in a four year college.
```

- We are interested in the effect of the treatment (two-years collage) on the outcome (years of education completed)

Example: Kane and Rouse (1995)

	Ave. Treat	Ave. control	T-test	Wilcoxon	KS
female	0.501	0.510	0.757	0.757	1.000
black	0.056	0.098	0.002	0.007	0.611
hispanic	0.135	0.096	0.035	0.023	0.709
bytest	59.255	60.919	0.000	0.000	0.000
dadmiss	0.131	0.119	0.525	0.515	1.000
dadvoc	0.093	0.069	0.123	0.097	0.991
dadsome	0.147	0.114	0.091	0.073	0.882
dadcoll	0.210	0.346	0.000	0.000	0.000
mommiss	0.068	0.040	0.035	0.016	0.959
momvoc	0.098	0.088	0.536	0.525	1.000
momsome	0.159	0.159	0.976	0.976	1.000
momcoll	0.145	0.246	0.000	0.000	0.002
fincome	23057.110	26534.917	0.000	0.007	0.026
fincmiss	0.051	0.061	0.426	0.446	1.000
ownhome	0.823	0.844	0.300	0.286	0.998
perwhite	75.956	79.176	0.027	0.002	0.008
urban	0.166	0.221	0.009	0.013	0.264

Balance table: Pre-matching

Example: Kane and Rouse (1995)

```
female (1 = Female/0 = Male)
black (1 = Black/0 = Not-Black)
hispanic (1 = Hispanic/0 = Not-Hispanic)
bytest (Base Year Composite Test Score)
Father's Highest Education Level:
  dadmiss1 (Education Missing)
  dadvoc (1 = Father attended Vocational/Technical School, 0 if not)
  dadsome (1 = Father attended some college but did not graduate, 0 if not)
  dadcoll (1 = Father college graduate, 0 if not)
Mother's Highest Education Level:
  mommiss (Education Missing)
  momvoc (1= Mother attended Vocational/Technical School, 0 if not)
  momsome (1= Mother attended some college but did not graduate, 0 if not)
 momcoll (1= Mother College Graduate, 0 if not)
fincome (Family Income)
fincmiss (Family Income Missing)
ownhome (1= Family Owns Home, 0 if not)
perwhite (Percentage of Student Body White in students high school)
urban (1 if students high school in Urban Area)
cue80 (County unemployment rate in 1980
stypc80 (State Personal Income per capita)
stwmfg80 (State Hourly Wage (mfg), 1980)
```

Example: Kane and Rouse (1995)

	Model 1	Model 2	Model 3
(Intercept)	15.30^{***}	11.04^{***}	11.20^{***}
	(0.04)	(0.61)	(0.63)
twoyr	-1.36^{***}	-1.22^{***}	-1.21^{***}
	(0.08)	(0.08)	(0.08)
female		0.03	0.03
		(0.07)	(0.07)
black		−0.23	-0.27^{*}
		(0.13)	(0.14)
fincome		0.00^{***}	0.00^{*}
		(0.00)	(0.00)
bytest		0.07^{***}	0.06^{***}
		(0.01)	(0.01)
urban		0.03	−0.03
		(0.09)	(0.09)
hispanic			0.14
			(0.13)
⋮			⋮
R^2	0.13	0.16	0.19
Adj. R^2	0.13	0.16	0.18
Num. obs.	1818	1818	1818

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

Example: Kane and Rouse (1995), matching w/t replacement

	Ave. Treat	Ave. control	T-test	Wilcoxon	KS
female	0.501	0.517	0.633	0.633	1.000
black	0.056	0.054	0.881	0.881	1.000
hispanic	0.135	0.121	0.541	0.540	1.000
bytest	59.255	59.689	0.040	0.019	0.040
dadmiss	0.131	0.119	0.606	0.606	1.000
dadvoc	0.093	0.093	1.000	1.000	1.000
dadsome	0.147	0.138	0.696	0.696	1.000
dadcoll	0.210	0.212	0.933	0.934	1.000
mommiss	0.068	0.068	1.000	1.000	1.000
momvoc	0.098	0.093	0.817	0.817	1.000
momsome	0.159	0.156	0.925	0.926	1.000
momcoll	0.145	0.142	0.923	0.923	1.000
fincome	23057.110	22638.695	0.701	0.894	1.000
fincmiss	0.051	0.051	1.000	1.000	1.000
ownhome	0.823	0.848	0.311	0.311	0.999
perwhite	75.956	79.883	0.028	0.002	0.012
urban	0.166	0.166	1.000	1.000	1.000

Balance table: W/T-matching

Example: Kane and Rouse (1995), Regressions after matching

	Model 1	Model 2	Model 3
(Intercept)	15.18***	10.25***	10.33***
	(0.08)	(1.07)	(1.10)
twoyr	−1.24***	−1.21***	−1.22***
	(0.11)	(0.11)	(0.11)
female		−0.10	−0.09
		(0.11)	(0.11)
black		−0.35	−0.45
		(0.24)	(0.25)
fincome		0.00***	0.00
		(0.00)	(0.00)
bytest		0.08***	0.08***
		(0.02)	(0.02)
urban		−0.07	−0.17
		(0.15)	(0.15)
hispanic			0.28
			(0.21)
⋮			⋮
R^2	0.13	0.16	0.19
Adj. R^2	0.13	0.16	0.17
Num. obs.	858	858	858

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

Genetic matching

- The objective of matching is to create comparable treatment and control units
- Matching on the Propensity score or using Mahalanobis distance, or a mix using both, do we maximize balance? *No, they optimize on P-score or on Mahalanobis distance*
- Genetic Matching incorporates balancing directly in the algorithm
  - Generalizing Mahalanobis matching
  - Prioritizes differences on certain covariates
  - Optimizes matching for some target: balance on X_p

Weighted Mahalanobis Matching

- Why weight? Some covariates may be more important than others
- Intuitively, we already weight covariates either at 0 or 1 in measuring $md(X_i, X_j)$
- Define X_p as a full matrix of covariates and X_k as the k-dimensional matrix we match on, where $k < p$
- Let $\tilde{w}$ be a vector of weights so that,

$$\tilde{w} = \{w_1 = 0, w_2 = 0, \ldots, w_k = 0, w_{k+1} = 1, \ldots, w_p = 1\}$$

- Generalize this weighting in a flexible way, Let $\tilde{w} \in \{\underline{w}, \overline{w}\}$

Weighted Mahalanobis Matching

- Incorporate these weights in $W_{p\times p}$ to rescale unit distance between unit i and unit j,

$$wmd(X_i, X_j) = \sqrt{(X_i - X_j)^T \left(S^{-\frac{1}{2}}\right)^T W \left(S^{-\frac{1}{2}}\right)(X_i - X_j)}$$

- Where W is a $p \times p$ positive definite weight matrix, with $\tilde{w}$ the diagonal elements of W, the off diagonal element of W are 0
- "The choice of setting the non-diagonal elements of W to zero is made for reasons of computational power alone", Mebane and Sekhon (2011)
- $\left(S^{\frac{1}{2}}\right)$ is the Cholesky decomposition of the sample covariance matrix S,

$$S = \left(S^{-\frac{1}{2}}\right)^T \left(S^{-\frac{1}{2}}\right)$$

- When $W = I_{p\times p}$, then $wmd(X_i, X_j) = md(X_i, X_j)$

Weighted Mahalanobis Matching

Suppose we have,

$$(x_1, x_2) \sim N\left((0,0), \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{S}\right)$$

This is equivalent to $x_1 \sim N(0,1)$, $x_2 \sim N(0,1)$, and $Cov(x_1, x_2) = 0$. Then,

$$\begin{aligned} wmd(X_i, X_j) &= \sqrt{(X_i - X_j)^T \left(S^{-\frac{1}{2}}\right)^T W \left(S^{-\frac{1}{2}}\right)(X_i - X_j)} \\ &= \sqrt{(X_i - X_j)^T W (X_i - X_j)} \\ &= \sqrt{(x_{1i} - x_{1j},\ x_{2i} - x_{2j}) \begin{pmatrix} w_1 & 0 \\ 0 & w_2 \end{pmatrix} \begin{pmatrix} x_{1i} - x_{1j} \\ x_{2i} - x_{2j} \end{pmatrix}} \\ &= \sqrt{(x_{1i} - x_{1j})^2 w_1 + (x_{2i} - x_{2j})^2 w_2} \end{aligned}$$

Weighted Mahalanobis Matching

- If $w_1 = 2w_2$ then,

$$wmd(X_i, X_j) = \sqrt{(x_{1i} - x_{1j})^2 \underbrace{2w_2}_{w_1} + (x_{1i} - x_{1j})^2 w_2}$$

$$\approx \sqrt{2(x_{1i} - x_{1j})^2 + (x_{2i} - x_{2j})^2}$$

- x_1 receives twice as weight as x_2 in the distance calculation between X_i and X_j, e.g we penalize more for differences in x_1
- Weighting rescales (upweights) relative differences on some covariates over other differences
- The weighted Mahalanobis distance matching is affine invariant. For example, dividing all weights by w_1, affine transformation of *wmd* will not change the matched stratas
- How do we choose the weights $\tilde{w}$? *Genetic optimization*

Evolutionary Algorithm to Select Weights

- The relevant task is to identify the k weights
- Pick weights to minimizes differences on X_p across treated and control units after matching - *This is our objective function*

- Genetic Matching
  - Evolutionary algorithm to optimize weights over multiple generations
  - Optimize means reduces imbalance on included X covariates

Genetic Matching Workflow

Basic argument for the functions *GenMatch* and *Match* in the Matching *R* package

- Tr - Treatment indicator
- X_k - is covariates for matching (Match Matrix)
- X_p - is covariates for balancing("Balance Matrix)
- Note: X_p can include **any**, **all** or **none** of X_k

The optimization is performed using the package *rgenoud*. This optimization package can be used for other optimization problems, for example *maximum likelihood*

Genetic Matching Workflow

- At starting generation g_0:
  1. Select starting $\tilde{w}_0$
  2. Devise m weight vectors using evolutionary string operators (called populations)
  3. For each weight vector:
     a. Compute $wmd(X_k, \tilde{w}_0)$
     b. Match i treated to j control units using $wmd(X_{ki}, X_{kj}, \tilde{w}_0)$ distances
     c. Compute a balance metric $\mathbb{B}(s)$, a function of all s matched pairs
  4. Select the m^{th} weights $(\tilde{w}_0^m)$ that maximizes $\mathbb{B}(s)$
- At next generation g_1 to g_n:
  5. Retain $(\tilde{w}_{t-1}^m)$ as the best starting value from g_{t1}
  6. Repeat steps 2 to 4 from above, maximizing $\mathbb{B}_t(s)$ at each step
  7. Stop by user control or once no more fitness gains over last few generations

Genetic Matching

Load the package 'Matching'

Matching includes the functions GenMatch(), Matching(), and MatchBalance().

Genetic Matching

Use GenMatch() to find the matches

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, ...)

Genetic Matching

Use GenMatch() to find the matches 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, ...)

Tr is the vector of treatment assignments.

Genetic Matching

Use GenMatch() to find the matches

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, ...)

X is the vector of variables that we want to match on.

Genetic Matching

Use GenMatch() to find the matches

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, ...)

BalanceMatrix is the vector of variables that we to achieve balance on (defaults to **X**).

Genetic Matching

Use GenMatch() to find the matches

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, ...)

estimand is the parameter of interest. It can be the "ATT", "ATE", or "ATC".

Genetic Matching

Use GenMatch() to find the matches

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, ...)

M is the number of control units that you match to each treated unit.
M=1 means one-to-one matching.

Genetic Matching

Use GenMatch() to find the matches

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, ...)

weights is a vector with the same length as Y that specifies the weights to be placed on the observations. This is **not** the weights denoting the importance of the covariates.

Genetic Matching

Use GenMatch() to find the matches

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.fun="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, ...)

caliper is a vector denoting the maximum difference apart each match can be for every covariate. Units with no suitable matches will be dropped.

Genetic Matching

Use GenMatch() to find the matches

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.fun="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, ...)

replace allows you to choose whether to match with or without replacement.

Genetic Matching

Use GenMatch() to find the matches

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.fun="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, ...)

ties allows you to deal with cases where there are several equally suitable control units for one treated unit. If **ties** is set at TRUE, GenMatch will average over the control units. If **ties** is FALSE, ties will be broken with a coin flip.

Genetic Matching

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.fun="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, ...)

paired allows you to choose whether GenMatch uses pairedt-tests.

Genetic Matching

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.fun="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, ...)

loss is the loss function. It should take the vector of p-values for the covariates and return some value that GenMatch will try to maximize. The default is to sort the p-values and minimize the maximum discrepancy.

Genetic Matching

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")

Y is the vector of outcomes.

Genetic Matching

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")

Tr is the treatment assignment vector.

Genetic Matching

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")

X is the matrix of control variables.

Genetic Matching

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")

Weight can be set at 2 to do Mahalanobis Distance matching. Leave this argument blank if you want to do Genetic matching.

Genetic Matching

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")

Weight.matrix takes the output from the GenMatch() function.

Genetic Matching

```
$est
      [,1]
[1,] 20933

$se
[1] 28768.51

$est.noadj
[1] 20933

$se.standard
[1] 23754.81
```

Genetic Matching

$est is the estimated treatment effect (adjusted for bias if you BiasAdjust=TRUE)

$se is the estimated standard error (adjusted for the uncertainty in the matching procedure)

$est.noadj is the estimated treatment effect (not adjusted for bias)

$se.standard is the standard error estimated the normal way

Genetic Matching

```
$index.treated
 [1]  6  8 11 28 30 34 35 42 43 45 50 51 52 56 60

$index.control
 [1] 48  3 48 48 17 36 40 46 25 10 48 64 15 64 19
```

Genetic Matching

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

Genetic Matching

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

formul is not the regular formula. It should be written as Treat $\sim$ Control 1 + Control 2 +...

Genetic Matching

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

data is the data frame.

Genetic Matching

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

match.out is the output of Match(). If you do not include this, MatchBalance will only return the balance before matching.

Genetic Matching

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

paired determines if the t.tests on the matched data is paired.

Genetic Matching

```
> mat=Match(Y=data$b2004, Tr=data$etouch, X=with(data,cbind(income, b2000, hispanic
size)), Weight.matrix=gen)
>
> MatchBalance(etouch ~ income + b2000 + hispanic + size, data=data, match.out=mat)
```

Genetic Matching

```
***** (V1) income *****
                        Before Matching     After Matching
mean treatment........      39282              39282
mean control..........      34261              39358
std mean diff.........     105.39             -1.5994

mean raw eQQ diff.....     5574.9              1128.3
med  raw eQQ diff.....       6222                1045
max  raw eQQ diff.....       7100                4158

mean eCDF diff........    0.26883            0.066667
med  eCDF diff........    0.25513            0.066667
max  eCDF diff........    0.52949                 0.2

var ratio (Tr/Co).....    0.56619              1.0204
T-test p-value........  0.0023728             0.88572
KS Bootstrap p-value.. < 2.22e-16                0.85
KS Naive p-value......  0.0016245             0.92509
KS Statistic..........    0.52949                 0.2
```

GenMatch: Custom loss function

1. Your custom loss function should take a vector containing all t-tests p-values, followed by all the ks-test p-values, for the covariates in the BalanceMatrix.
2. Therefore, the length of the vector it will receive is twice the length of the number of covariates in the BalanceMatrix
3. Your loss function should return either a number or a vector
4. If it returns a number, then GenMatch will try to maximize that number
5. If it returns a vector, then GenMatch will try to maximize the first number in that vector. In cases where multiple matching schemes result in the same first number, then GenMatch will look at the second number, and then the third number, and so on

GenMatch: Custom loss function

What does this loss function do?

```
loss.function=function(x) {
  p.vals = x
  return(sort(p.vals))
}
```

Answer: Maximize the minimum p-value, with ties are broken by looking at the second lowest p-value. This is the default loss function.

GenMatch: Custom loss function

What does this loss function do?

```
loss.function=function(x) {
  p.vals = x
  return(min(p.vals))
}
```

Answer: Maximize the minimum p-value. This is the loss function if you set loss=2

GenMatch: Custom loss function

What does this loss function do?

```
loss.function=function(x) {
  p.vals = x
  return(mean(p.vals))
}
```

Answer: Maximize the mean of the p-values

GenMatch: Custom loss function

What does this loss function do?

```
loss.function=function(x) {
  p.vals = x
  p.vals[1]=p.vals[1]*0.5
  p.vals[length(p.vals)/
2+1]=p.vals[length(p.vals)/2+1]*0.5
  return(sort(p.vals))
}
```

Answer: Maximize the minimum p-value, giving more weight to the first covariate (we will treat the p-values for the first covariate as half their actual value

GenMatch: Custom loss function

What does this loss function do?

```
loss.function = function(x) {
  p.vals = x
  if(sum(x < initial) > 0) {
    p.vals = 0.1*p.vals
  }
  return(sort(p.vals))
}
```

Answer: Restricts the search to only the weighting schemes where balance on every covariate is better than before matching. (Note: If GenMatch cannot find any such balancing scheme, the loss function defaults to maximize the minimum balance