Package ‘CBPS’

August 21, 2019

Version 0.21

Date 2019-08-20

Title Covariate Balancing Propensity Score

Depends R (>= 3.4), MASS, MatchIt, nnet, numDeriv, glmnet

Imports

Description Implements the covariate balancing propensity score (CBPS) proposed by Imai and Ratkovic (2014) <DOI:10.1111/rssb.12027>. The propensity score is estimated such that it maximizes the resulting covariate balance as well as the prediction of treatment assignment. The method, therefore, avoids an iteration between model fitting and balance checking. The package also implements optimal CBPS from Fan et al. (2016) <https://imai.fas.harvard.edu/research/CBPStheory.html>, several extensions of the CBPS beyond the cross-sectional, binary treatment setting. They include the CBPS for longitudinal settings so that it can be used in conjunction with marginal structural models from Imai and Ratkovic (2015) <DOI:10.1080/01621459.2014.956872>, treatments with three- and four-valued treatment variables, continuous-valued treatments from Fong, Hazlett, and Imai (2018) <DOI:10.1214/17-AOAS1101>, propensity score estimation with a large number of covariates from Ning, Peng, and Imai (2018) <arXiv:1812.08683>, and the situation with multiple distinct binary treatments administered simultaneously. In the future it will be extended to other settings including the generalization of experimental and instrumental variable estimates.

LazyLoad yes

LazyData yes

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

RoxygenNote 6.1.1

Suggests testthat

Author Christian Fong [aut, cre],
Marc Ratkovic [aut],
Kosuke Imai [aut],
Chad Hazlett [ctb],

Xiaolin Yang [ctb],
Sida Peng [ctb]

Maintainer Christian Fong <cjfong@umich.edu>

Date/Publication 2019-08-21 07:40:18 UTC

R topics documented:

`balance` *Optimal Covariate Balance*

Description

Returns the mean and standardized mean associated with each treatment group, before and after weighting. To access more comprehensive diagnostics for assessing covariate balance, consider using Noah Greifer's `cobalt` package.

Usage

```
balance(object, ...)
```

Arguments

`object`	A CBPS, npCBPS, or CBMSM object.
`...`	Additional arguments to be passed to balance.

Details

For binary and multi-valued treatments as well as marginal structural models, each of the matrices' rows are the covariates and whose columns are the weighted mean, and standardized mean associated with each treatment group. The standardized mean is the weighted mean divided by the standard deviation of the covariate for the whole population. For continuous treatments, returns the absolute Pearson correlation between the treatment and each covariate.

@aliases balance balance.npCBPS balance.CBPS balance.CBMSM

Value

Returns a list of two matrices, "original" (before weighting) and "balanced" (after weighting).

Author(s)

Christian Fong, Marc Ratkovic, and Kosuke Imai.

Examples

```
###
### Example: Assess Covariate Balance
###
data(LaLonde)
## Estimate CBPS
fit <- CBPS(treat ~ age + educ + re75 + re74 +
I(re75==0) + I(re74==0),
data = LaLonde, ATT = TRUE)
balance(fit)
```

`balance.CBPS` *Calculates the pre- and post-weighting difference in standardized means for covariate within each contrast*

Description

Calculates the pre- and post-weighting difference in standardized means for covariate within each contrast

Usage

```
## S3 method for class 'CBPS'
balance(object, ...)
```

Arguments

`object`	A CBPS, npCBPS, or CBMSM object.
`...`	Additional arguments to be passed to balance.

`balance.CBPSContinuous` *Calculates the pre- and post-weighting correlations between each covariate and the T*

Description

Calculates the pre- and post-weighting correlations between each covariate and the T

Usage

```
## S3 method for class 'CBPSContinuous'
balance(object, ...)
```

Arguments

`object`	A CBPS, npCBPS, or CBMSM object.
`...`	Additional arguments to be passed to balance.

`balance.npCBPS` *Calls the appropriate balance function based on the number of treatments*

Description

Calls the appropriate balance function based on the number of treatments

Usage

```
## S3 method for class 'npCBPS'
balance(object, ...)
```

Arguments

`object`	A CBPS, npCBPS, or CBMSM object.
`...`	Other parameters to be passed.

`Blackwell` *Blackwell Data for Covariate Balancing Propensity Score*

Description

This data set gives the outcomes a well as treatment assignments and covariates for the example from Blackwell (2013).

Format

A data frame consisting of 13 columns (including treatment assignment, time, and identifier vectors) and 570 observations.

Source

d.gone.neg is the treatment. d.gone.neg.l1, d.gone.neg.l2, and d.gone.neg.l3 are lagged treatment variables. camp.length, deminc, base.poll, base.und, and office covariates. year is the year of the particular race, and time goes from the first measurement (time = 1) to the election (time = 5). demName is the identifier, and demprcnt is the outcome.

References

Blackwell, Matthew. (2013). A framework for dynamic causal inference in political science. American Journal of Political Science 57, 2, 504-619.

`CBIV` *Covariate Balancing Propensity Score for Instrumental Variable Estimates (CBIV)*

Description

`CBIV` estimates propensity scores for compliance status in an instrumental variables setup such that both covariate balance and prediction of treatment assignment are maximized. The method, therefore, avoids an iterative process between model fitting and balance checking and implements both simultaneously.

Usage

```
CBIV(Tr, Z, X, iterations = 1000, method = "over", twostep = TRUE,
  twosided = TRUE, ...)
```

Arguments

`Tr`	A binary treatment variable.
`Z`	A binary encouragement variable.
`X`	A pre-treatment covariate matrix.
`iterations`	An optional parameter for the maximum number of iterations for the optimization. Default is 1000.
`method`	Choose "over" to fit an over-identified model that combines the propensity score and covariate balancing conditions; choose "exact" to fit a model that only contains the covariate balancing conditions. Our simulations suggest that "over" dramatically outperforms "exact."
`twostep`	Default is `TRUE` for a two-step GMM estimator, which will run substantially faster than continuous-updating. Set to `FALSE` to use the continuous-updating GMM estimator.
`twosided`	Default is `TRUE`, which allows for two-sided noncompliance with both always-takers and never-takers. Set to `FALSE` for one-sided noncompliance, which allows only for never-takers.
`...`	Other parameters to be passed through to `optim()`.

Details

Fits covariate balancing propensity scores for generalizing local average treatment effect estimates obtained from instrumental variables analysis.

Value

`coefficients`	A named matrix of coefficients, where the first column gives the complier coefficients and the second column gives the always-taker coefficients.
`fitted.values`	The fitted N x 3 compliance score matrix. The first column gives the estimated probability of being a complier, the second column gives the estimated probability of being an always-taker, and the third column gives the estimated probability of being a never-taker.
`weights`	The optimal weights: the reciprocal of the probability of being a complier.
`deviance`	Minus twice the log-likelihood of the CBIV fit.
`converged`	Convergence value. Returned from the call to `optim()`.
`J`	The J-statistic at convergence
`df`	The number of linearly independent covariates.
`bal`	The covariate balance associated with the optimal weights, calculated as the GMM loss of the covariate balance conditions.

Author(s)

Christian Fong

References

Imai, Kosuke and Marc Ratkovic. 2014. "Covariate Balancing Propensity Score." Journal of the Royal Statistical Society, Series B (Statistical Methodology). http://imai.princeton.edu/research/CBPS.html

Examples

```
###
### Example: propensity score matching
### (Need to fix when we have an actual example).

##Load the LaLonde data
data(LaLonde)
## Estimate CBPS
fit <- CBPS(treat ~ age + educ + re75 + re74 +
I(re75==0) + I(re74==0),
data = LaLonde, ATT = TRUE)
summary(fit)
```

CBMSM *Covariate Balancing Propensity Score (CBPS) for Marginal Structural Models*

Description

CBMSM estimates propensity scores such that both covariate balance and prediction of treatment assignment are maximized. With longitudinal data, the method returns marginal structural model weights that can be entered directly into a linear model. The method also handles multiple binary treatments administered concurrently.

Usage

```
CBMSM(formula, id, time, data, type = "MSM", twostep = TRUE,
  msm.variance = "approx", time.vary = FALSE, init = "opt", ...)
```

Arguments

formula	A formula of the form treat ~ X. The same covariates are used in each time period. At default values, a single set of coefficients is estimated across all time periods. To allow a different set of coefficients for each time period, set time.vary = TRUE.
id	A vector which identifies the unit associated with each row of treat and X.
time	A vector which identifies the time period associated with each row of treat and X.

`data`	An optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `CBMSM` is called.
`type`	"MSM" for a marginal structural model, with multiple time periods or "MultiBin" for multiple binary treatments at the same time period.
`twostep`	Set to `TRUE` to use a two-step estimator, which will run substantially faster than continuous-updating. Default is `FALSE`, which uses the continuous-updating estimator described by Imai and Ratkovic (2014).
`msm.variance`	Default is `FALSE`, which uses the low-rank approximation of the variance described in Imai and Ratkovic (2014). Set to `TRUE` to use the full variance matrix.
`time.vary`	Default is `FALSE`, which uses the same coefficients across time period. Set to `TRUE` to fit one set per time period.
`init`	Default is `"opt"`, which uses CBPS and logistic regression starting values, and chooses the one that achieves the best balance. Other options are "glm" and "CBPS"
`...`	Other parameters to be passed through to `optim()`

Details

Fits covariate balancing propensity scores for marginal structural models.

@aliases CBMSM CBMSM.fit

Value

`weights`	The optimal weights.
`fitted.values`	The fitted propensity score for each observation.
`y`	The treatment vector used.
`x`	The covariate matrix.
`id`	The vector id used in CBMSM.fit.
`time`	The vector time used in CBMSM.fit.
`model`	The model frame.
`call`	The matched call.
`formula`	The formula supplied.
`data`	The data argument.
`treat.hist`	A matrix of the treatment history, with each observation in rows and time in columns.
`treat.cum`	A vector of the cumulative treatment history, by individual.

Author(s)

Marc Ratkovic, Christian Fong, and Kosuke Imai; The CBMSM function is based on the code for version 2.15.0 of the glm function implemented in the stats package, originally written by Simon Davies. This documenation is likewise modeled on the documentation for glm and borrows its language where the arguments and values are the same.

References

Imai, Kosuke and Marc Ratkovic. 2014. "Covariate Balancing Propensity Score." Journal of the Royal Statistical Society, Series B (Statistical Methodology). http://imai.princeton.edu/research/CBPS.html

Imai, Kosuke and Marc Ratkovic. 2015. "Robust Estimation of Inverse Probability Weights for Marginal Structural Models." Journal of the American Statistical Association. http://imai.princeton.edu/research/MSM.html

See Also

plot.CBMSM

Examples

```
##Load Blackwell data

data(Blackwell)

## Quickly fit a short model to test
form0 <- "d.gone.neg ~ d.gone.neg.l1 + camp.length"
fit0<-CBMSM(formula = form0, time=Blackwell$time,id=Blackwell$demName,
data=Blackwell, type="MSM",  iterations = NULL, twostep = TRUE,
msm.variance = "approx", time.vary = FALSE)

## Not run:
##Fitting the models in Imai and Ratkovic  (2014)
##Warning: may take a few mintues; setting time.vary to FALSE
##Results in a quicker fit but with poorer balance
##Usually, it is best to use time.vary TRUE
form1<-"d.gone.neg ~ d.gone.neg.l1 + d.gone.neg.l2 + d.neg.frac.l3 +
camp.length + camp.length + deminc + base.poll + year.2002 +
year.2004 + year.2006 + base.und + office"

##Note that  init="glm" gives the published results but the default is now init="opt"
fit1<-CBMSM(formula = form1, time=Blackwell$time,id=Blackwell$demName,
data=Blackwell, type="MSM",  iterations = NULL, twostep = TRUE,
msm.variance = "full", time.vary = TRUE, init="glm")

fit2<-CBMSM(formula = form1, time=Blackwell$time,id=Blackwell$demName,
data=Blackwell, type="MSM",  iterations = NULL, twostep = TRUE,
msm.variance = "approx", time.vary = TRUE, init="glm")


##Assessing balance

bal1<-balance.CBMSM(fit1)
bal2<-balance.CBMSM(fit2)

##Effect estimation: Replicating Effect Estimates in
```

```
##Table 3 of Imai and Ratkovic (2014)

lm1<-lm(demprcnt[time==1]~fit1$treat.hist,data=Blackwell,
weights=fit1$glm.weights)
lm2<-lm(demprcnt[time==1]~fit1$treat.hist,data=Blackwell,
weights=fit1$weights)
lm3<-lm(demprcnt[time==1]~fit1$treat.hist,data=Blackwell,
weights=fit2$weights)

lm4<-lm(demprcnt[time==1]~fit1$treat.cum,data=Blackwell,
weights=fit1$glm.weights)
lm5<-lm(demprcnt[time==1]~fit1$treat.cum,data=Blackwell,
weights=fit1$weights)
lm6<-lm(demprcnt[time==1]~fit1$treat.cum,data=Blackwell,
weights=fit2$weights)



### Example: Multiple Binary Treatments Administered at the Same Time
n<-200
k<-4
set.seed(1040)
X1<-cbind(1,matrix(rnorm(n*k),ncol=k))

betas.1<-betas.2<-betas.3<-c(2,4,4,-4,3)/5
probs.1<-probs.2<-probs.3<-(1+exp(-X1 %*% betas.1))^-1

treat.1<-rbinom(n=length(probs.1),size=1,probs.1)
treat.2<-rbinom(n=length(probs.2),size=1,probs.2)
treat.3<-rbinom(n=length(probs.3),size=1,probs.3)
treat<-c(treat.1,treat.2,treat.3)
X<-rbind(X1,X1,X1)
time<-c(rep(1,nrow(X1)),rep(2,nrow(X1)),rep(3,nrow(X1)))
id<-c(rep(1:nrow(X1),3))
y<-cbind(treat.1,treat.2,treat.3) %*% c(2,2,2) +
X1 %*% c(-2,8,7,6,2) + rnorm(n,sd=5)

multibin1<-CBMSM(treat~X,id=id,time=time,type="MultiBin",twostep=TRUE)
summary(lm(y~-1+treat.1+treat.2+treat.3+X1, weights=multibin1$w))

## End(Not run)
```

`CBMSM.fit` *CBMSM.fit*

Description

CBMSM.fit

Usage

```
CBMSM.fit(treat, X, id, time, MultiBin.fit, twostep, msm.variance,
  time.vary, init, ...)
```

Arguments

treat	A vector of treatment assignments. For N observations over T time periods, the length of treat should be N*T.
X	A covariate matrix. For N observations over T time periods, X should have N*T rows.
id	A vector which identifies the unit associated with each row of treat and X.
time	A vector which identifies the time period associated with each row of treat and X.
MultiBin.fit	A parameter for whether the multiple binary treatments occur concurrently (FALSE) or over consecutive time periods (TRUE) as in a marginal structural model. Setting type = "MultiBin" when calling CBMSM will set MultiBin.fit to TRUE when CBMSM.fit is called.
twostep	Set to TRUE to use a two-step estimator, which will run substantially faster than continuous-updating. Default is FALSE, which uses the continuous-updating estimator described by Imai and Ratkovic (2014).
msm.variance	Default is FALSE, which uses the low-rank approximation of the variance described in Imai and Ratkovic (2014). Set to TRUE to use the full variance matrix.
time.vary	Default is FALSE, which uses the same coefficients across time period. Set to TRUE to fit one set per time period.
init	Default is "opt", which uses CBPS and logistic regression starting values, and chooses the one that achieves the best balance. Other options are "glm" and "CBPS"
...	Other parameters to be passed through to optim()

CBPS *Covariate Balancing Propensity Score (CBPS) Estimation*

Description

CBPS estimates propensity scores such that both covariate balance and prediction of treatment assignment are maximized. The method, therefore, avoids an iterative process between model fitting and balance checking and implements both simultaneously. For cross-sectional data, the method can take continuous treatments and treatments with a control (baseline) condition and either 1, 2, or 3 distinct treatment conditions.

Fits covariate balancing propensity scores.

@aliases CBPS CBPS.fit print.CBPS

Usage

```
CBPS(formula, data, na.action, ATT = 1, iterations = 1000,
  standardize = TRUE, method = "over", twostep = TRUE,
  sample.weights = NULL, baseline.formula = NULL,
  diff.formula = NULL, ...)
```

Arguments

- `formula` — An object of class `formula` (or one that can be coerced to that class): a symbolic description of the model to be fitted.
- `data` — An optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `CBPS` is called.
- `na.action` — A function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset.
- `ATT` — Default is 1, which finds the average treatment effect on the treated interpreting the second level of the treatment factor as the treatment. Set to 2 to find the ATT interpreting the first level of the treatment factor as the treatment. Set to 0 to find the average treatment effect. For non-binary treatments, only the ATE is available.
- `iterations` — An optional parameter for the maximum number of iterations for the optimization. Default is 1000.
- `standardize` — Default is `TRUE`, which normalizes weights to sum to 1 within each treatment group. For continuous treatments, normalizes weights to sum up to 1 for the entire sample. Set to `FALSE` to return Horvitz-Thompson weights.
- `method` — Choose "over" to fit an over-identified model that combines the propensity score and covariate balancing conditions; choose "exact" to fit a model that only contains the covariate balancing conditions.
- `twostep` — Default is `TRUE` for a two-step estimator, which will run substantially faster than continuous-updating. Set to `FALSE` to use the continuous-updating estimator described by Imai and Ratkovic (2014).
- `sample.weights` — Survey sampling weights for the observations, if applicable. When left NULL, defaults to a sampling weight of 1 for each observation.
- `baseline.formula` — Used only to fit iCBPS (see Fan et al). Currently only works with binary treatments. A formula specifying the balancing covariates in the baseline outcome model, i.e., E(Y(0)|X).
- `diff.formula` — Used only to fit iCBPS (see Fan et al). Currently only works with binary treatments. A formula specifying the balancing covariates in the difference between the treatment and baseline outcome model, i.e., E(Y(1)-Y(0)|X).
- `...` — Other parameters to be passed through to `optim()`.

Value

`fitted.values` The fitted propensity score

`linear.predictor`
X * beta

`deviance` Minus twice the log-likelihood of the CBPS fit

`weights` The optimal weights. Let $\pi_i = f(T_i|X_i)$. For binary ATE, these are given by $\frac{T_i}{\pi_i} + \frac{(1-T_i)}{(1-\pi_i)}$. For binary ATT, these are given by $\frac{n}{n_t} * \frac{T_i - \pi_i}{1-\pi_i}$. For multi_valued treatments, these are given by $\sum_{j=0}^{J-1} T_{i,j}/\pi_{i,j}$. For continuous treatments, these are given by $\frac{f(T_i)}{f(T_i|X_i)}$. These expressions for weights are all before standardization (i.e. with standardize=`FALSE`). Standardization will make weights sum to 1 within each treatment group. For continuous treatment, standardization will make all weights sum to 1. If sampling weights are used, the weight for each observation is multiplied by the survey sampling weight.

`y` The treatment vector used

`x` The covariate matrix

`model` The model frame

`converged` Convergence value. Returned from the call to `optim()`.

`call` The matched call

`formula` The formula supplied

`data` The data argument

`coefficients` A named vector of coefficients

`sigmasq` The sigma-squared value, for continuous treatments only

`J` The J-statistic at convergence

`mle.J` The J-statistic for the parameters from maximum likelihood estimation

`var` The covariance matrix for the coefficients.

`Ttilde` For internal use only.

`Xtilde` For internal use only.

`beta.tilde` For internal use only.

`simgasq.tilde` For internal use only.

Author(s)

Christian Fong, Marc Ratkovic, Kosuke Imai, and Xiaolin Yang; The CBPS function is based on the code for version 2.15.0 of the glm function implemented in the stats package, originally written by Simon Davies. This documentation is likewise modeled on the documentation for glm and borrows its language where the arguments and values are the same.

References

Imai, Kosuke and Marc Ratkovic. 2014. "Covariate Balancing Propensity Score." Journal of the Royal Statistical Society, Series B (Statistical Methodology). http://imai.princeton.edu/research/CBPS.html

Fong, Christian, Chad Hazlett, and Kosuke Imai. 2018. "Covariate Balancing Propensity Score for a Continuous Treatment." The Annals of Applied Statistics. http://imai.princeton.edu/research/files/CBGPS.pdf

Fan, Jianqing and Imai, Kosuke and Liu, Han and Ning, Yang and Yang, Xiaolin. "Improving Covariate Balancing Propensity Score: A Doubly Robust and Efficient Approach." Unpublished Manuscript. http://imai.princeton.edu/research/CBPStheory.html

See Also

summary.CBPS

Examples

```
###
### Example: propensity score matching
###

##Load the LaLonde data
data(LaLonde)
## Estimate CBPS
fit <- CBPS(treat ~ age + educ + re75 + re74 +
I(re75==0) + I(re74==0),
data = LaLonde, ATT = TRUE)
summary(fit)
## Not run:
## matching via MatchIt: one to one nearest neighbor with replacement
library(MatchIt)
m.out <- matchit(treat ~ fitted(fit), method = "nearest",
 data = LaLonde, replace = TRUE)

### Example: propensity score weighting
###
## Simulation from Kang and Shafer (2007).
set.seed(123456)
n <- 500
X <- mvrnorm(n, mu = rep(0, 4), Sigma = diag(4))
prop <- 1 / (1 + exp(X[,1] - 0.5 * X[,2] +
 0.25*X[,3] + 0.1 * X[,4]))
treat <- rbinom(n, 1, prop)
y <- 210 + 27.4*X[,1] + 13.7*X[,2] + 13.7*X[,3] + 13.7*X[,4] + rnorm(n)

##Estimate CBPS with a misspecified model
X.mis <- cbind(exp(X[,1]/2), X[,2]*(1+exp(X[,1]))^(-1)+10,
  (X[,1]*X[,3]/25+.6)^3, (X[,2]+X[,4]+20)^2)
fit1 <- CBPS(treat ~ X.mis, ATT = 0)
summary(fit1)
```

```
## Horwitz-Thompson estimate
mean(treat*y/fit1$fitted.values)
## Inverse propensity score weighting
sum(treat*y/fit1$fitted.values)/sum(treat/fit1$fitted.values)

rm(list=c("y","X","prop","treat","n","X.mis","fit1"))

### Example: Continuous Treatment as in Fong, Hazlett,
### and Imai (2018).  See
### https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/AIF4PI
### for a real data example.
set.seed(123456)
n <- 1000
X <- mvrnorm(n, mu = rep(0,2), Sigma = diag(2))
beta <- rnorm(ncol(X)+1, sd = 1)
treat <- cbind(1,X)%*%beta + rnorm(n, sd = 5)

treat.effect <- 1
effect.beta <- rnorm(ncol(X))
y <- rbinom(n, 1, (1 + exp(-treat.effect*treat -
   X%*%effect.beta))^-1)

fit2 <- CBPS(treat ~ X)
summary(fit2)
summary(glm(y ~ treat + X, weights = fit2$weights,
family = "quasibinomial"))

rm(list=c("n", "X", "beta", "treat", "treat.effect",
  "effect.beta", "y", "fit2"))

### Simulation example: Improved CBPS (or iCBPS) from Fan et al
set.seed(123456)
n <- 500
X <- mvrnorm(n, mu = rep(0, 4), Sigma = diag(4))
prop <- 1 / (1 + exp(X[,1] - 0.5 * X[,2] + 0.25*X[,3] + 0.1 * X[,4]))
treat <- rbinom(n, 1, prop)
y1 <- 210 + 27.4*X[,1] + 13.7*X[,2] + 13.7*X[,3] + 13.7*X[,4] + rnorm(n)
y0 <- 210 + 13.7*X[,2] + 13.7*X[,3] + 13.7*X[,4] + rnorm(n)
##Estimate iCBPS with a misspecificied model
X.mis <- cbind(exp(X[,1]/2), X[,2]*(1+exp(X[,1]))^(-1)+10,
   (X[,1]*X[,3]/25+.6)^3, (X[,2]+X[,4]+20)^2)
fit1 <- CBPS(treat ~ X.mis, baseline.formula=~X.mis[,2:4],
 diff.formula=~X.mis[,1], ATT = FALSE)
summary(fit1)

## End(Not run)
```

CBPS.fit *CBPS.fit determines the proper routine (what kind of treatment) and calls the approporiate function. It also pre- and post-processes the data*

Description

CBPS.fit determines the proper routine (what kind of treatment) and calls the approporiate function. It also pre- and post-processes the data

Usage

```
CBPS.fit(treat, X, baselineX, diffX, ATT, method, iterations, standardize,
  twostep, sample.weights = sample.weights, ...)
```

Arguments

`treat`	A vector of treatment assignments. Binary or multi-valued treatments should be factors. Continuous treatments should be numeric.
`X`	A covariate matrix.
`baselineX`	Similar to `baseline.formula`, but in matrix form.
`diffX`	Similar to `diff.formula`, but in matrix form.
`ATT`	Default is 1, which finds the average treatment effect on the treated interpreting the second level of the treatment factor as the treatment. Set to 2 to find the ATT interpreting the first level of the treatment factor as the treatment. Set to 0 to find the average treatment effect. For non-binary treatments, only the ATE is available.
`method`	Choose "over" to fit an over-identified model that combines the propensity score and covariate balancing conditions; choose "exact" to fit a model that only contains the covariate balancing conditions.
`iterations`	An optional parameter for the maximum number of iterations for the optimization. Default is 1000.
`standardize`	Default is `TRUE`, which normalizes weights to sum to 1 within each treatment group. For continuous treatments, normalizes weights to sum up to 1 for the entire sample. Set to `FALSE` to return Horvitz-Thompson weights.
`twostep`	Default is `TRUE` for a two-step estimator, which will run substantially faster than continuous-updating. Set to `FALSE` to use the continuous-updating estimator described by Imai and Ratkovic (2014).
`sample.weights`	Survey sampling weights for the observations, if applicable. When left NULL, defaults to a sampling weight of 1 for each observation.
`...`	Other parameters to be passed through to `optim()`.

Value

CBPS.fit object

hdCBPS — *hdCBPS high dimensional CBPS method to parses the formula object and passes the result to hdCBPS.fit, which calculates ATE using CBPS method in a high dimensional setting.*

Description

hdCBPS high dimensional CBPS method to parses the formula object and passes the result to hdCBPS.fit, which calculates ATE using CBPS method in a high dimensional setting.

Usage

```
hdCBPS(formula, data, na.action, y, ATT = 0, iterations = 1000,
  method = "linear")
```

Arguments

`formula`	An object of class formula (or one that can be coerced to that class): a symbolic description of the model to be fitted.
`data`	An optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which CBPS is called.
`na.action`	A function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset.
`y`	An outcome variable.
`ATT`	Option to calculate ATT
`iterations`	An optional parameter for the maximum number of iterations for the optimization. Default is 1000.
`method`	Choose among "linear", "binomial", and "possion".

Value

`ATT`	Average treatment effect on the treated.
`ATE`	Average treatment effect.
`s`	Standard Error.
`fitted.values`	The fitted propensity score
`coefficients1`	Coefficients for the treated propensity score
`coefficients0`	Coefficients for the untreated propensity score
`model`	The model frame

Author(s)

Sida Peng

`LaLonde` *LaLonde Data for Covariate Balancing Propensity Score*

Description

This data set gives the outcomes a well as treatment assignments and covariates for the econometric evaluation of training programs in LaLonde (1986).

Format

A data frame consisting of 12 columns (including a treatment assignment vector) and 3212 observations.

Source

Data from the National Supported Work Study. A benchmark matching dataset. Columns consist of an indicator for whether the observed unit was in the experimental subset; an indicator for whether the individual received the treatment; age in years; schooling in years; indicators for black and Hispanic; an indicator for marriage status, one of married; an indicator for no high school degree; reported earnings in 1974, 1975, and 1978; and whether the 1974 earnings variable is missing. Data not missing 1974 earnings are the Dehejia-Wahba subsample of the LaLonde data. Missing values for 1974 earnings set to zero. 1974 and 1975 earnings are pre-treatment. 1978 earnings is taken as the outcome variable.

References

LaLonde, R.J. (1986). Evaluating the econometric evaluations of training programs with experimental data. American Economic Review 76, 4, 604-620.

`npCBPS` *Non-Parametric Covariate Balancing Propensity Score (npCBPS) Estimation*

Description

`npCBPS` is a method to estimate weights interpretable as (stabilized) inverse generlized propensity score weights, w_i = f(T_i)/f(T_i|X), without actually estimating a model for the treatment to arrive at f(T|X) estimates. In brief, this works by maximizing the empirical likelihood of observing the values of treatment and covariates that were observed, while constraining the weights to be those that (a) ensure balance on the covariates, and (b) maintain the original means of the treatment and covariates.

In the continuous treatment context, this balance on covariates means zero correlation of each covariate with the treatment. In binary or categorical treatment contexts, balance on covariates implies equal means on the covariates for observations at each level of the treatment. When given a numeric treatment the software handles it continuously. To handle the treatment as binary or categorical is must be given as a factor.

Furthermore, we apply a Bayesian variant that allows the correlation of each covariate with the treatment to be slightly non-zero, as might be expected in a a given finite sample.

Estimates non-parametric covariate balancing propensity score weights.

@aliases npCBPS npCBPS.fit

Usage

```
npCBPS(formula, data, na.action, corprior = 0.01, print.level = 0, ...)
```

Arguments

`formula`	An object of class `formula` (or one that can be coerced to that class): a symbolic description of the model to be fitted.
`data`	An optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `CBPS` is called.
`na.action`	A function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset.
`corprior`	Prior hyperparameter controlling the expected amount of correlation between each covariate and the treatment. Specifically, the amount of correlation between the k-dimensional covariates, X, and the treatment T after weighting is assumed to have prior distribution MVN(0,sigma^2 I_k). We conceptualize sigma^2 as a tuning parameter to be used pragmatically. It's default of 0.1 ensures that the balance constraints are not too harsh, and that a solution is likely to exist. Once the algorithm works at such a high value of sigma^2, the user may wish to attempt values closer to 0 to get finer balance.
`print.level`	Controls verbosity of output to the screen while npCBPS runs. At the default of print.level=0, little output is produced. It print.level>0, it outputs diagnostics including the log posterior (log_post), the log empirical likelihood associated with the weights (log_el), and the log prior probability of the (weighted) correlation of treatment with the covariates.
`...`	Other parameters to be passed.

Value

`weights`	The optimal weights
`y`	The treatment vector used
`x`	The covariate matrix
`model`	The model frame
`call`	The matched call
`formula`	The formula supplied
`data`	The data argument

log.p.eta	The log density for the (weighted) correlation of the covariates with the treatment, given the choice of prior (corprior)
log.el	The log empirical likelihood of the observed data at the chosen set of IPW weights.
eta	A vector describing the correlation between the treatment and each covariate on the weighted data at the solution.
sumw0	The sum of weights, provided as a check on convergence. This is always 1 when convergence occurs unproblematically. If it differs from 1 substantially, no solution perfectly satisfying the conditions was found, and the user may consider a larger value of corprior.

Author(s)

Christian Fong, Chad Hazlett, and Kosuke Imai

References

Fong, Christian, Chad Hazlett, and Kosuke Imai. "Parametric and Nonparametric Covariate Balancing Propensity Score for General Treatment Regimes." Unpublished Manuscript. http://imai.princeton.edu/research/files/CBGPS.pdf

Examples

```
##Generate data
data(LaLonde)

## Restricted two only two covariates so that it will run quickly.
## Performance will remain good if the full LaLonde specification is used
fit <- npCBPS(treat ~ age + educ, data = LaLonde, corprior=.1/nrow(LaLonde))
plot(fit)
```

---

npCBPS.fit *npCBPS.fit*

Description

npCBPS.fit

Usage

```
npCBPS.fit(treat, X, corprior, print.level, ...)
```

Arguments

treat
: A vector of treatment assignments. Binary or multi-valued treatments should be factors. Continuous treatments should be numeric.

X
: A covariate matrix.

corprior
: Prior hyperparameter controlling the expected amount of correlation between each covariate and the treatment. Specifically, the amount of correlation between the k-dimensional covariates, X, and the treatment T after weighting is assumed to have prior distribution MVN(0,sigma^2 I_k). We conceptualize sigma^2 as a tuning parameter to be used pragmatically. It's default of 0.1 ensures that the balance constraints are not too harsh, and that a solution is likely to exist. Once the algorithm works at such a high value of sigma^2, the user may wish to attempt values closer to 0 to get finer balance.

print.level
: Controls verbosity of output to the screen while npCBPS runs. At the default of print.level=0, little output is produced. It print.level>0, it outputs diagnostics including the log posterior (log_post), the log empirical likelihood associated with the weights (log_el), and the log prior probability of the (weighted) correlation of treatment with the covariates.

...
: Other parameters to be passed.

`plot.CBMSM` *Plotting CBPS Estimation for Marginal Structural Models*

Description

Plots the absolute difference in standardized means before and after weighting.

Usage

```
## S3 method for class 'CBMSM'
plot(x, covars = NULL, silent = TRUE,
  boxplot = FALSE, ...)
```

Arguments

x
: an object of class "CBMSM".

covars
: Indices of the covariates to be plotted (excluding the intercept). For example, if only the first two covariates from `balance` are desired, set `covars` to 1:2. The default is `NULL`, which plots all covariates.

silent
: If set to `FALSE`, returns the absolute imbalance for each treatment history pair before and after weighting. This helps the user to create his or her own customized plot. Default is `TRUE`, which returns nothing.

boxplot
: If set to `TRUE`, returns a boxplot summarizing the imbalance on the covariates instead of a point for each covariate. Useful if there are many covariates.

...
: Additional arguments to be passed to plot.

Details

Covariate balance is improved if the plot's points are below the plotted line of y=x.

Value

The x-axis gives the imbalance for each covariate-treatment history pair without any weighting, and the y-axis gives the imbalance for each covariate-treatment history pair after CBMSM weighting. Imbalance is measured as the absolute difference in standardized means for the two treatment histories. Means are standardized by the standard deviation of the covariate in the full sample.

Author(s)

Marc Ratkovic and Christian Fong

See Also

CBMSM, plot

---

plot.CBPS *Plotting Covariate Balancing Propensity Score Estimation*

Description

This function plots the absolute difference in standardized means before and after weighting. To access more sophisticated graphics for assessing covariate balance, consider using Noah Greifer's `cobalt` package.

Usage

```
## S3 method for class 'CBPS'
plot(x, covars = NULL, silent = TRUE, boxplot = FALSE,
  ...)
```

Arguments

`x`	an object of class "CBPS" or "npCBPS", usually, a result of a call to `CBPS` or `npCBPS`.
`covars`	Indices of the covariates to be plotted (excluding the intercept). For example, if only the first two covariates from `balance` are desired, set `covars` to 1:2. The default is `NULL`, which plots all covariates.
`silent`	If set to `FALSE`, returns the imbalances used to construct the plot. Default is `TRUE`, which returns nothing.
`boxplot`	If set to `TRUE`, returns a boxplot summarizing the imbalance on the covariates instead of a point for each covariate. Useful if there are many covariates.
`...`	Additional arguments to be passed to plot.

Details

The "Before Weighting" plot gives the balance before weighting, and the "After Weighting" plot gives the balance after weighting.

@aliases plot.CBPS plot.npCBPS

Value

For binary and multi-valued treatments, plots the absolute difference in standardized means by contrast for all covariates before and after weighting. This quantity for a single covariate and a given pair of treatment conditions is given by $\frac{\sum_{i=1}^{n} w_i*(T_i==1)*X_i}{\sum_{i=1}^{n}(T_i==1)*w_i} - \frac{\sum_{i=1}^{n} w_i*(T_i==0)*X_i}{\sum_{i=1}^{n}(T_i==0)*w_i}$. For continuous treatments, plots the weighted absolute Pearson correlation between the treatment and each covariate. See https://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient#Weighted_correlation_coefficient.

Author(s)

Christian Fong, Marc Ratkovic, and Kosuke Imai.

See Also

CBPS, plot

`plot.CBPSContinuous` *Plot the pre-and-post weighting correlations between X and T*

Description

Plot the pre-and-post weighting correlations between X and T

Usage

```
## S3 method for class 'CBPSContinuous'
plot(x, covars = NULL, silent = TRUE,
  boxplot = FALSE, ...)
```

Arguments

`x`	an object of class “CBPS” or “npCBPS”, usually, a result of a call to `CBPS` or `npCBPS`.
`covars`	Indices of the covariates to be plotted (excluding the intercept). For example, if only the first two covariates from `balance` are desired, set `covars` to 1:2. The default is `NULL`, which plots all covariates.
`silent`	If set to `FALSE`, returns the imbalances used to construct the plot. Default is `TRUE`, which returns nothing.
`boxplot`	If set to `TRUE`, returns a boxplot summarizing the imbalance on the covariates instead of a point for each covariate. Useful if there are many covariates.
`...`	Additional arguments to be passed to balance.

plot.npCBPS — *Calls the appropriate plot function, based on the number of treatments*

Description

Calls the appropriate plot function, based on the number of treatments

Usage

```
## S3 method for class 'npCBPS'
plot(x, covars = NULL, silent = TRUE, ...)
```

Arguments

`x`	an object of class "CBPS" or "npCBPS", usually, a result of a call to `CBPS` or `npCBPS`.
`covars`	Indices of the covariates to be plotted (excluding the intercept). For example, if only the first two covariates from `balance` are desired, set `covars` to 1:2. The default is `NULL`, which plots all covariates.
`silent`	If set to `FALSE`, returns the imbalances used to construct the plot. Default is `TRUE`, which returns nothing.
`...`	Additional arguments to be passed to balance.

print.CBPS — *Print coefficients and model fit statistics*

Description

Print coefficients and model fit statistics

Usage

```
## S3 method for class 'CBPS'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`x`	an object of class "CBPS" or "npCBPS", usually, a result of a call to `CBPS` or `npCBPS`.
`digits`	the number of digits to keep for the numerical quantities.
`...`	Additional arguments to be passed to summary.

summary.CBPS — *Summarizing Covariate Balancing Propensity Score Estimation*

Description

Prints a summary of a fitted CBPS object.

Usage

```
## S3 method for class 'CBPS'
summary(object, ...)
```

Arguments

`object`	an object of class "CBPS", usually, a result of a call to CBPS.
`...`	Additional arguments to be passed to summary.

Details

Prints a summmary of a CBPS object, in a format similar to glm. The variance matrix is calculated from the numerical Hessian at convergence of CBPS.

Value

`call`	The matched call.
`deviance.residuals`	The five number summary and the mean of the deviance residuals.
`coefficients`	A table including the estimate for the each coefficient and the standard error, z-value, and two-sided p-value for these estimates.
`J`	Hansen's J-Statistic for the fitted model.
`Log-Likelihood`	The log-likelihood of the fitted model.

Author(s)

Christian Fong, Marc Ratkovic, and Kosuke Imai.

See Also

CBPS, summary

vcov.CBPS *Calculate Variance-Covariance Matrix for a Fitted CBPS Object*

Description

vcov.CBPS Returns the variance-covariance matrix of the main parameters of a fitted CBPS object.

Usage

```
## S3 method for class 'CBPS'
vcov(object, ...)
```

Arguments

object	An object of class formula (or one that can be coerced to that class): a symbolic description of the model to be fitted.
...	Additional arguments to be passed to vcov.CBPS

Details

This is the CBPS implementation of the generic function vcov().

Value

A matrix of the estimated covariances between the parameter estimates in the linear or non-linear predictor of the model.

Author(s)

Christian Fong, Marc Ratkovic, and Kosuke Imai.

References

This documentation is modeled on the documentation of the generic vcov.

See Also

vcov

Examples

```
###
### Example: Variance-Covariance Matrix
###

##Load the LaLonde data
data(LaLonde)
## Estimate CBPS via logistic regression
```

```
fit <- CBPS(treat ~ age + educ + re75 + re74 + I(re75==0) + I(re74==0),
    data = LaLonde, ATT = TRUE)
## Get the variance-covariance matrix.
vcov(fit)
```

`vcov_outcome` *Calculate Variance-Covariance Matrix for Outcome Model*

Description

`vcov_outcome` Returns the variance-covariance matrix of the main parameters of a fitted CBPS object.

This adjusts the standard errors of the weighted regression of Y on Z for uncertainty in the weights.

@aliases vcov_outcome vcov_outcome.CBPSContinuous

Usage

```
vcov_outcome(object, Y, Z, delta, tol = 10^(-5), lambda = 0.01)
```

Arguments

`object`	A fitted CBPS object.
`Y`	The outcome.
`Z`	The covariates (including the treatment and an intercept term) that predict the outcome.
`delta`	The coefficients from regressing Y on Z, weighting by the cbpsfit$weights.
`tol`	Tolerance for choosing whether to improve conditioning of the "M" matrix prior to conversion. Equal to 1/(condition number), i.e. the smallest eigenvalue divided by the largest.
`lambda`	The amount to be added to the diagonal of M if the condition of the matrix is worse than tol.

Value

A matrix of the estimated covariances between the parameter estimates in the weighted outcome regression, adjusted for uncertainty in the weights.

Author(s)

Christian Fong, Chad Hazlett, and Kosuke Imai.

References

Lunceford and Davididian 2004.

Examples

```
###
### Example: Variance-Covariance Matrix
###

##Load the LaLonde data
data(LaLonde)
## Estimate CBPS via logistic regression
fit <- CBPS(treat ~ age + educ + re75 + re74 + I(re75==0) + I(re74==0),
    data = LaLonde, ATT = TRUE)
## Get the variance-covariance matrix.
vcov(fit)
```

---

vcov_outcome.CBPSContinuous

vcov_outcome

Description

vcov_outcome

Usage

```
## S3 method for class 'CBPSContinuous'
vcov_outcome(object, Y, Z, delta, tol = 10^(-5),
  lambda = 0.01)
```

Arguments

object	A fitted CBPS object.
Y	The outcome.
Z	The covariates (including the treatment and an intercept term) that predict the outcome.
delta	The coefficients from regressing Y on Z, weighting by the cbpsfit$weights.
tol	Tolerance for choosing whether to improve conditioning of the "M" matrix prior to conversion. Equal to 1/(condition number), i.e. the smallest eigenvalue divided by the largest.
lambda	The amount to be added to the diagonal of M if the condition of the matrix is worse than tol.

Value

Variance-Covariance Matrix for Outcome Model

Index