Package 'ctmle'

November 10, 2017

Type Package

Title Collaborative Targeted Maximum Likelihood Estimation

Version 0.1.1

Date 2017-11-01

Maintainer Cheng Ju <jucheng1992@gmail.com>

Description Implements the general template for collaborative targeted maximum likelihood estimation. It also provides several commonly used C-TMLE instantiation, like the vanilla/scalable variable-selection C-TMLE (Ju et al. (2017) <doi:10.1177/0962280217729845>) and the glmnet-C-TMLE algorithm (Ju et al. (2017) <arXiv:1706.10029>).

License GPL-2

Depends R (>= 2.14.0), SuperLearner, tmle, glmnet, stats

Imports

Suggests testthat, knitr, rmarkdown, dplyr

LazyLoad yes

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Cheng Ju [aut, cre],
Susan Gruber [aut],
Richard Wyss [ctb],
Jenny Haggstrom [ctb],
Mark van der Laan [aut, ths]

Repository CRAN

Date/Publication 2017-11-10 09:59:50 UTC

R topics documented:

---

bound *set outliers to min/max allowable values. It assumes x contains only numerical data*

Description

set outliers to min/max allowable values. It assumes x contains only numerical data

Usage

```
bound(x, bounds)
```

Arguments

x	input data
bounds	a vector with length 2, contains the min and max of the bound

Value

x truncated input x by min/max in bounds

Examples

```
x <- rnorm(1000)
x <- bound(x, c(-1, 1))
```

---

build_gn_seq *Help function to build the sequence of gn candidates in ctmleGeneral*

Description

This function helps building gn candidates for ctmleGeneral. It returns gn_candidates_cv, gn_candidates, and folds, which could be directly applied to ctmleGeneral.

Usage

```
build_gn_seq(A, W, SL.library, folds, verbose = TRUE)
```

Arguments

A	binary treatment indicator, 1 - treatment, 0 - control
W	vector, matrix, or dataframe containing baseline covariates for Q bar
SL.library	a vector of the names of the estimators for ctmle (need to be prepared in the format for SL, see more details in SuperLearner package), The theory of ctmle requires the estimators are ordered by the model complexity, with the last one be a consistent estimator.
folds	The list of indices for the ctmle cross-validation step
verbose	A boolean. If print out the training log for Super Learne

Value

gn_candidates_cv matrix or dataframe, each column stand for a estimate of propensity score. Estimate in the column with larger index should have smaller empirical loss

gn_candidates matrix or dataframe, each column stand for a the cross-validated estimate. For example, the (i,j)-th element is the predicted propensity score by j-th estimator, for i-th observation, when it is in the validation set

folds The list of indices for the ctmle cross-validation step

details The SuperLearner object used to generate gn_candidates_cv

Examples

```
N <- 1000
p = 100
V = 5
Wmat <- matrix(rnorm(N * p), ncol = p)
gcoef <- matrix(c(-1,-1,rep(-(3/((p)-2)),(p)-2)),ncol=1)

W <- as.data.frame(Wmat)
g <- 1/(1+exp(Wmat%*%gcoef / 3))
A <- rbinom(N, 1, prob = g)

folds <-by(sample(1:N,N), rep(1:V, length=N), list)

lasso_fit <- cv.glmnet(x = as.matrix(W), y = A, alpha = 1, nlambda = 100, nfolds = 10)
lasso_lambdas <- lasso_fit$lambda[lasso_fit$lambda <= lasso_fit$lambda.min][1:5]
# Build template for glmnet
SL.glmnet_new <- function (Y, X, newX, family, obsWeights, id, alpha = 1,
                           nlambda = 100, lambda = 0,...)
{
    # browser()
    if (!is.matrix(X)) {
          X <- model.matrix(~-1 + ., X)
         newX <- model.matrix(~-1 + ., newX)
   }
   fit <- glmnet::glmnet(x = X, y = Y,
                         lambda = lambda,
                         family = family$family, alpha = alpha)
```

```
   pred <- predict(fit, newx = newX, type = "response")
     fit <- list(object = fit)
   class(fit) <- "SL.glmnet"
   out <- list(pred = pred, fit = fit)
   return(out)
}

# Use a sequence of estimator to build gn sequence:
SL.cv1lasso <- function (... , alpha = 1, lambda = lasso_lambdas[1]){
   SL.glmnet_new(... , alpha = alpha, lambda = lambda)
}

SL.cv2lasso <- function (... , alpha = 1, lambda = lasso_lambdas[2]){
    SL.glmnet_new(... , alpha = alpha, lambda = lambda)
}

SL.cv3lasso <- function (... , alpha = 1, lambda = lasso_lambdas[3]){
    SL.glmnet_new(... , alpha = alpha, lambda = lambda)
}

SL.cv4lasso <- function (... , alpha = 1, lambda = lasso_lambdas[4]){
     SL.glmnet_new(... , alpha = alpha, lambda = lambda)
}

SL.library = c('SL.cv1lasso', 'SL.cv2lasso', 'SL.cv3lasso', 'SL.cv4lasso', 'SL.glm')

gn_seq <- build_gn_seq(A = A, W = W, SL.library = SL.library, folds = folds)

gn_seq$gn_candidates_cv
gn_seq$gn_candidates
```

`ctmleDiscrete` *Discrete Collaborative Targeted Minimum-loss based Estimation*

Description

This function computes the discrete Collaborative Targeted Minimum-loss based Estimator for variable selection. It includes the greedy C-TMLE algorithm (Gruber and van der Laan 2010), and scalable C-TMLE algorithm (Ju, Gruber, and Lendle et al. 2016) with a user-specified order.

Usage

```
ctmleDiscrete(Y, A, W, Wg = W, Q = NULL, preOrder = FALSE, order = NULL,
  patience = FALSE, Qbounds = NULL, cvQinit = FALSE, Qform = NULL,
  SL.library = NULL, alpha = 0.995, family = "gaussian", gbound = 0.025,
  like_type = "RSS", fluctuation = "logistic", verbose = FALSE,
  detailed = FALSE, PEN = FALSE, V = 5, folds = NULL,
  stopFactor = 10^6)
```

Arguments

Y	continuous or binary outcome variable
A	binary treatment indicator, 1 for treatment, 0 for control
W	vector, matrix, or dataframe containing baseline covariates for Q bar
Wg	vector, matrix, or dataframe containing baseline covariates for propensity score model (defaults to W if not supplied by user)
Q	n by 2 matrix of initial values for Q0W, Q1W in columns 1 and 2, respectively. Current version does not support SL for automatic initial estimation of Q bar
preOrder	boolean indicator for using scalable C-TMLE algorithm or not
order	the use-specified order of covariables. Only used when (preOrder = TRUE). If not supplied by user, it would automatically order covariates from W_1 to W_p
patience	a number to stop early when the score in the CV function does not improve after so many covariates. Used only when (preOrder = TRUE)
Qbounds	bound on initial Y and predicted values for Q.
cvQinit	if TRUE, cross-validate initial values for Q to avoid overfits
Qform	optional regression formula for estimating initial Q
SL.library	optional vector of prediction algorithms for data adaptive estimation of Q, defaults to glm, and glmnet
alpha	used to keep predicted initial values bounded away from (0,1) for logistic fluctuation, 0.995 (default)
family	family specification for working regression models, generally 'gaussian' for continuous outcomes (default), 'binomial' for binary outcomes
gbound	bound on P(A=1|W), defaults to 0.025
like_type	'RSS' or 'loglike'. The metric to use for forward selection and cross-validation
fluctuation	'logistic' (default) or 'linear', for targeting step
verbose	print status messages if TRUE
detailed	boolean number. If it is TRUE, return more detailed results
PEN	boolean. If true, penalized loss is used in cross-validation step
V	Number of folds. Only used if folds is not specified
folds	The list of indices for cross-validation step. We recommend the cv-splits in C-TMLE matchs that in gn_candidate_cv
stopFactor	Numerical value with default 1e6. If the current empirical likelihood is stopFactor times larger than the best previous one, the construction would stop

Value

best_k the index of estimate that selected by cross-validation

est estimate of psi_0

CI IC-based 95

pvalue pvalue for the null hypothesis that Psi = 0

likelihood sum of squared residuals, based on selected estimator evaluated on all obs or, logistic loglikelihood if like_type != 'RSS'

varIC empirical variance of the influence curve adjusted for estimation of g

varDstar empirical variance of the influence curve

var.psi variance of the estimate

varIC.cv cross-validated variance of the influence curve

penlikelihood.cv penalized cross-validated likelihood

cv.res all cross-validation results for each fold

Examples

```
## Not run:
N <- 1000
p = 10
Wmat <- matrix(rnorm(N * p), ncol = p)
beta1 <- 4+2*Wmat[,1]+2*Wmat[,2]+2*Wmat[,5]+2*Wmat[,6]+2*Wmat[,8]
beta0 <- 2+2*Wmat[,1]+2*Wmat[,2]+2*Wmat[,5]+2*Wmat[,6]+2*Wmat[,8]
tauW <- 2
tau <- 2
gcoef <- matrix(c(-1,-1,rep(-(3/((p)-2)),(p)-2)),ncol=1)
Wm <- as.matrix(Wmat)
g <- 1/(1+exp(Wm%*%gcoef))
A <- rbinom(N, 1, prob = g)
sigma <- 1
epsilon <-rnorm(N,0,sigma)
Y  <- beta0 + tauW*A + epsilon

# Initial estimate of Q
Q <- cbind(rep(mean(Y[A == 0]), N), rep(mean(Y[A == 1]), N))

# User-suplied initial estimate
time_greedy <- system.time(
ctmle_discrete_fit1 <- ctmleDiscrete(Y = Y, A = A, W = data.frame(Wmat), Q = Q,
                                     preOrder = FALSE)
)

# If there is no input Q, then intial Q would be estimated by SL with Sl.library
ctmle_discrete_fit2 <- ctmleDiscrete(Y = Y, A = A, W = data.frame(Wmat),
                                     preOrder = FALSE, detailed = TRUE)

# scalable C-TMLE with pre-order option; order is user-specified,
# If 'order' is  not specified takes order from W1 to Wp.
time_preorder <- system.time(
ctmle_discrete_fit3 <- ctmleDiscrete(Y = Y, A = A, W = data.frame(Wmat), Q = Q,
                                     preOrder = TRUE,
                                     order = rev(1:p), detailed = TRUE)
)

# Compare the running time
time_greedy
```

```
time_preorder

## End(Not run)
```

---

ctmleGeneral *General Template for Collaborative Targeted Maximum Likelihood Estimation*

Description

This function computes the Collaborative Targeted Maximum Likelihood Estimator.

Usage

```
ctmleGeneral(Y, A, W, Wg = W, Q, ctmletype, gn_candidates,
  gn_candidates_cv = NULL, alpha = 0.995, family = "gaussian",
  gbound = 0.025, like_type = "RSS", fluctuation = "logistic",
  verbose = FALSE, detailed = FALSE, PEN = FALSE, g1W = NULL,
  g1WPrev = NULL, V = 5, folds = NULL, stopFactor = 10^6)
```

Arguments

Y	continuous or binary outcome variable
A	binary treatment indicator, 1 for treatment, 0 for control
W	vector, matrix, or dataframe containing baseline covariates for Q bar
Wg	vector, matrix, or dataframe containing baseline covariates for propensity score model (defaults to W if not supplied by user)
Q	n by 2 matrix of initial values for Q0W, Q1W in columns 1 and 2, respectively. Current version does not support SL for automatic initial estimation of Q bar
ctmletype	1 or 3. Type of general C-TMLE. Type 1 uses cross-validation to select best gn, while Type 3 directly solves extra clever covariates.
gn_candidates	matrix or dataframe, each column stand for a estimate of propensity score. Estimate in the column with larger index should have smaller empirical loss
gn_candidates_cv	matrix or dataframe, each column stand for a the cross-validated estimate. For example, the (i,j)-th element is the predicted propensity score by j-th estimator, for i-th observation, when it is in the validation set
alpha	used to keep predicted initial values bounded away from (0,1) for logistic fluctuation, 0.995 (default)
family	family specification for working regression models, generally 'gaussian' for continuous outcomes (default), 'binomial' for binary outcomes
gbound	bound on P(A=1|W), defaults to 0.025
like_type	'RSS' or 'loglike'. The metric to use for forward selection and cross-validation
fluctuation	'logistic' (default) or 'linear', for targeting step

verbose	print status messages if TRUE
detailed	boolean number. If it is TRUE, return more detailed results
PEN	boolean. If true, penalized loss is used in cross-validation step
g1W	Only used when type is 3. a user-supplied propensity score estimate.
g1WPrev	Only used when type is 3. a user-supplied propensity score estimate, with small fluctuation compared to g1W.
V	Number of folds. Only used if folds is not specified
folds	The list of indices for cross-validation step. We recommend the cv-splits in C-TMLE matchs that in gn_candidate_cv
stopFactor	Numerical value with default 1e6. If the current empirical likelihood is stopFactor times larger than the best previous one, the construction would stop

Value

best_k the index of estimate that selected by cross-validation

est estimate of psi_0

CI IC-based 95

pvalue pvalue for the null hypothesis that Psi = 0

likelihood sum of squared residuals, based on selected estimator evaluated on all obs or, logistic loglikelihood if like_type != "RSS"

varIC empirical variance of the influence curve adjusted for estimation of g

varDstar empirical variance of the influence curve

var.psi variance of the estimate

varIC.cv cross-validated variance of the influence curve

penlikelihood.cv penalized cross-validatedlikelihood

cv.res all cross-validation results for each fold

Examples

```
N <- 1000
p = 100
V = 5
Wmat <- matrix(rnorm(N * p), ncol = p)
gcoef <- matrix(c(-1,-1,rep(-(3/((p)-2)),(p)-2)),ncol=1)

W <- as.data.frame(Wmat)
g <- 1/(1+exp(Wmat%*%gcoef / 3))
A <- rbinom(N, 1, prob = g)

# Build potential outcome pairs, and the observed outcome Y
beta1 <- 4+2*Wmat[,1]+2*Wmat[,2]+2*Wmat[,5]+2*Wmat[,6]+2*Wmat[,8]
beta0 <- 2+2*Wmat[,1]+2*Wmat[,2]+2*Wmat[,5]+2*Wmat[,6]+2*Wmat[,8]

tau = 2
sigma <- 1
```

```
epsilon <-rnorm(N,0,sigma)
Y  <- beta0 + tau * A + epsilon
# Initial estimate of Q
Q <- cbind(rep(mean(Y[A == 1]), N), rep(mean(Y[A == 0]), N))

folds <-by(sample(1:N,N), rep(1:V, length=N), list)

lasso_fit <- cv.glmnet(x = as.matrix(W), y = A, alpha = 1, nlambda = 100, nfolds = 10)
lasso_lambdas <- lasso_fit$lambda[lasso_fit$lambda <= lasso_fit$lambda.min][1:5]
# Build template for glmnet
SL.glmnet_new <- function (Y, X, newX, family, obsWeights, id, alpha = 1,
                           nlambda = 100, lambda = 0,...)
{
    # browser()
    if (!is.matrix(X)) {
          X <- model.matrix(~-1 + ., X)
         newX <- model.matrix(~-1 + ., newX)
   }
   fit <- glmnet::glmnet(x = X, y = Y,
                         lambda = lambda,
                         family = family$family, alpha = alpha)
   pred <- predict(fit, newx = newX, type = "response")
     fit <- list(object = fit)
   class(fit) <- "SL.glmnet"
   out <- list(pred = pred, fit = fit)
   return(out)
}

# Use a sequence of LASSO estimators to build gn sequence:
SL.cv1lasso <- function (... , alpha = 1, lambda = lasso_lambdas[1]){
   SL.glmnet_new(... , alpha = alpha, lambda = lambda)
}

SL.cv2lasso <- function (... , alpha = 1, lambda = lasso_lambdas[2]){
    SL.glmnet_new(... , alpha = alpha, lambda = lambda)
}

SL.cv3lasso <- function (... , alpha = 1, lambda = lasso_lambdas[3]){
    SL.glmnet_new(... , alpha = alpha, lambda = lambda)
}

SL.cv4lasso <- function (... , alpha = 1, lambda = lasso_lambdas[4]){
     SL.glmnet_new(... , alpha = alpha, lambda = lambda)
}

SL.library = c('SL.cv1lasso', 'SL.cv2lasso', 'SL.cv3lasso', 'SL.cv4lasso', 'SL.glm')

# Build the sequence. See more details in the function build_gn_seq, and package SuperLearner
gn_seq <- build_gn_seq(A = A, W = W, SL.library = SL.library, folds = folds)


# Use the output of build_gn_seq for ctmle general templates
ctmle_fit <- ctmleGeneral(Y = Y, A = A, W = W, Q = Q, ctmletype = 1,
```

```
gn_candidates = gn_seq$gn_candidates,
gn_candidates_cv = gn_seq$gn_candidates_cv,
folds = gn_seq$folds, V = length(folds))
```

`ctmleGlmnet` *Collaborative Targeted Maximum Likelihood Estimation for hyper-parameter tuning of LASSO*

Description

This function computes the Collaborative Maximum Likelihood Estimation for hyper-parameter tuning of LASSO.

Usage

```
ctmleGlmnet(Y, A, W, Wg = W, Q, lambdas = NULL, ctmletype, V = 5,
  folds = NULL, alpha = 0.995, family = "gaussian", gbound = 0.025,
  like_type = "RSS", fluctuation = "logistic", verbose = FALSE,
  detailed = FALSE, PEN = FALSE, g1W = NULL, g1WPrev = NULL,
  stopFactor = 10^6)
```

Arguments

`Y`	continuous or binary outcome variable
`A`	binary treatment indicator, 1 for treatment, 0 for control
`W`	vector, matrix, or dataframe containing baseline covariates for Q bar
`Wg`	vector, matrix, or dataframe containing baseline covariates for propensity score model (defaults to W if not supplied by user)
`Q`	n by 2 matrix of initial values for Q0W, Q1W in columns 1 and 2, respectively. Current version does not support SL for automatic initial estimation of Q bar
`lambdas`	numeric vector of lambdas (regularization parameter) for glmnet estimation of propensity score, with decreasing order. We recommend the first lambda is selected by external cross-validation.
`ctmletype`	1, 2 or 3. Type of general C-TMLE. Type 1 uses cross-validation to select best gn, Type 3 directly solves extra clever covariates, and Type 2 uses both cross-validation and extra covariate. See more details in !!!
`V`	Number of folds. Only used if folds is not specified
`folds`	The list of indices for cross-validation step. We recommend the cv-splits in C-TMLE matchs that in gn_candidate_cv
`alpha`	used to keep predicted initial values bounded away from (0,1) for logistic fluctuation, 0.995 (default)
`family`	family specification for working regression models, generally 'gaussian' for continuous outcomes (default), 'binomial' for binary outcomes
`gbound`	bound on P(A=1|W), defaults to 0.025

like_type	'RSS' or 'loglike'. The metric to use for forward selection and cross-validation
fluctuation	'logistic' (default) or 'linear', for targeting step
verbose	print status messages if TRUE
detailed	boolean number. If it is TRUE, return more detailed results
PEN	boolean. If true, penalized loss is used in cross-validation step
g1W	Only used when type is 3. a user-supplied propensity score estimate.
g1WPrev	Only used when type is 3. a user-supplied propensity score estimate, with small fluctuation compared to g1W.
stopFactor	Numerical value with default 1e6. If the current empirical likelihood is stopFactor times larger than the best previous one, the construction would stop

Value

best_k the index of estimate that selected by cross-validation

est estimate of psi_0

CI IC-based 95

pvalue pvalue for the null hypothesis that Psi = 0

likelihood sum of squared residuals, based on selected estimator evaluated on all obs or, logistic loglikelihood if like_type != 'RSS'

varIC empirical variance of the influence curve adjusted for estimation of g

varDstar empirical variance of the influence curve

var.psi variance of the estimate

varIC.cv cross-validated variance of the influence curve

penlikelihood.cv penalized cross-validatedlikelihood

cv.res all cross-validation results for each fold

Examples

```
## Not run:
set.seed(123)
N <- 1000
p = 10
Wmat <- matrix(rnorm(N * p), ncol = p)
beta1 <- 4+2*Wmat[,1]+2*Wmat[,2]+2*Wmat[,5]+2*Wmat[,6]+2*Wmat[,8]
beta0 <- 2+2*Wmat[,1]+2*Wmat[,2]+2*Wmat[,5]+2*Wmat[,6]+2*Wmat[,8]
tau <- 2
gcoef <- matrix(c(-1,-1,rep(0,(p)-2)),ncol=1)
Wm <- as.matrix(Wmat)
g <- 1/(1+exp(Wm%*%gcoef / 3))
A <- rbinom(N, 1, prob = g)
sigma <- 1
epsilon <-rnorm(N,0,sigma)
Y  <- beta0 + tau * A + epsilon
# ctmleGlmnet must provide user-specified Q
W_tmp <- data.frame(Wm[,1:3])
```

```
treated<- W_tmp[which(A==1),]
untreated<-W_tmp[which(A==0),]
Y1<-Y[which(A==1)]
Y0<-Y[which(A==0)]
# Initial Q-estimate
beta1hat <- predict(lm(Y1~.,data=treated),newdata=W_tmp)
beta0hat <- predict(lm(Y0~., data=untreated),newdata=W_tmp)
Q <- matrix(c(beta0hat,beta1hat),ncol=2)
W = Wm
glmnet_fit <- cv.glmnet(y = A, x = Wm,
                        family = 'binomial', nlambda = 40)
start = which(glmnet_fit$lambda==glmnet_fit$lambda.min))
end = length(glmnet_fit$lambda)
lambdas <-glmnet_fit$lambda[start:end]
ctmle_fit1 <- ctmleGlmnet(Y=Y, A=A,
                          W=data.frame(W=W),
                          Q = Q, lambdas = lambdas,
                          ctmletype=1, alpha=.995,
                          family="gaussian",
                          gbound=0.025,like_type="loglik" ,
                          fluctuation="logistic",
                          verbose=FALSE,
                          detailed=FALSE, PEN=FALSE,
                          V=5, stopFactor=10^6)

## End(Not run)
```

`print.ctmle` *print a ctmle object*

Description

print a ctmle object

Usage

```
## S3 method for class 'ctmle'
print(x, ...)
```

Arguments

`x`	a ctmle object
`...`	other parameter

Examples

```
## Not run:
N <- 1000
p = 10
```

```
Wmat <- matrix(rnorm(N * p), ncol = p)
beta1 <- 4+2*Wmat[,1]+2*Wmat[,2]+2*Wmat[,5]+2*Wmat[,6]+2*Wmat[,8]
beta0 <- 2+2*Wmat[,1]+2*Wmat[,2]+2*Wmat[,5]+2*Wmat[,6]+2*Wmat[,8]
tauW <- 2
tau <- 2
gcoef <- matrix(c(-1,-1,rep(-(3/((p)-2)),(p)-2)),ncol=1)
Wm <- as.matrix(Wmat)
g <- 1/(1+exp(Wm%*%gcoef))
A <- rbinom(N, 1, prob = g)
sigma <- 1
epsilon <-rnorm(N,0,sigma)
Y  <- beta0 + tauW*A + epsilon

# Initial estimate of Q
Q <- cbind(rep(mean(Y[A == 0]), N), rep(mean(Y[A == 1]), N))

# User-suplied initial estimate
time_greedy <- system.time(
ctmle_discrete_fit1 <- ctmleDiscrete(Y = Y, A = A, W = data.frame(Wmat), Q = Q,
                                     preOrder = FALSE)
)
ctmle_summary = summary(ctmle_discrete_fit1)
ctmle_summary
ctmle_discrete_fit1

## End(Not run)
```

`print.summary.ctmle` *print the summary of a ctmle object*

Description

print the summary of a ctmle object

Usage

```
## S3 method for class 'summary.ctmle'
print(x, ...)
```

Arguments

`x`	a summary.ctmle object
`...`	other parameter

Examples

```
## Not run:
N <- 1000
p = 10
```

```
Wmat <- matrix(rnorm(N * p), ncol = p)
beta1 <- 4+2*Wmat[,1]+2*Wmat[,2]+2*Wmat[,5]+2*Wmat[,6]+2*Wmat[,8]
beta0 <- 2+2*Wmat[,1]+2*Wmat[,2]+2*Wmat[,5]+2*Wmat[,6]+2*Wmat[,8]
tauW <- 2
tau <- 2
gcoef <- matrix(c(-1,-1,rep(-(3/((p)-2)),(p)-2)),ncol=1)
Wm <- as.matrix(Wmat)
g <- 1/(1+exp(Wm%*%gcoef))
A <- rbinom(N, 1, prob = g)
sigma <- 1
epsilon <-rnorm(N,0,sigma)
Y  <- beta0 + tauW*A + epsilon

# Initial estimate of Q
Q <- cbind(rep(mean(Y[A == 0]), N), rep(mean(Y[A == 1]), N))

# User-suplied initial estimate
time_greedy <- system.time(
ctmle_discrete_fit1 <- ctmleDiscrete(Y = Y, A = A, W = data.frame(Wmat), Q = Q,
                                     preOrder = FALSE)
)
ctmle_summary = summary(ctmle_discrete_fit1)
ctmle_summary
ctmle_discrete_fit1

## End(Not run)
```

`summary.ctmle` *Summarise a ctmle object*

Description

Summarise a ctmle object

Usage

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

Arguments

`object`	a ctmle object
`...`	other parameter

Examples

```
## Not run:
N <- 1000
p = 10
```

```
Wmat <- matrix(rnorm(N * p), ncol = p)
beta1 <- 4+2*Wmat[,1]+2*Wmat[,2]+2*Wmat[,5]+2*Wmat[,6]+2*Wmat[,8]
beta0 <- 2+2*Wmat[,1]+2*Wmat[,2]+2*Wmat[,5]+2*Wmat[,6]+2*Wmat[,8]
tauW <- 2
tau <- 2
gcoef <- matrix(c(-1,-1,rep(-(3/((p)-2)),(p)-2)),ncol=1)
Wm <- as.matrix(Wmat)
g <- 1/(1+exp(Wm%*%gcoef))
A <- rbinom(N, 1, prob = g)
sigma <- 1
epsilon <-rnorm(N,0,sigma)
Y  <- beta0 + tauW*A + epsilon

# Initial estimate of Q
Q <- cbind(rep(mean(Y[A == 0]), N), rep(mean(Y[A == 1]), N))

# User-suplied initial estimate
time_greedy <- system.time(
ctmle_discrete_fit1 <- ctmleDiscrete(Y = Y, A = A, W = data.frame(Wmat), Q = Q,
                                     preOrder = FALSE)
)
ctmle_summary = summary(ctmle_discrete_fit1)
ctmle_summary
ctmle_discrete_fit1

## End(Not run)
```

Index