Package 'drtmle'

April 23, 2022

Title Doubly-Robust Nonparametric Estimation and Inference

Version 1.1.1

Description Targeted minimum loss-based estimators of counterfactual means and causal effects that are doubly-robust with respect both to consistency and asymptotic normality (Benkeser et al (2017), <doi:10.1093/biomet/asx053>; MJ van der Laan (2014), <doi:10.1515/ijb-2012-0038>).

Depends R (>= 3.5.0)

Imports SuperLearner, np, future.apply

Suggests testthat, knitr, rmarkdown, gam, quadprog, nloptr, parallel, foreach, stringi

License MIT + file LICENSE

URL https://github.com/benkeser/drtmle

BugReports https://github.com/benkeser/drtmle/issues

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation no

Author David Benkeser [aut, cre, cph] (<https://orcid.org/0000-0002-1019-8343>), Nima Hejazi [ctb] (<https://orcid.org/0000-0002-7127-2789>)

Maintainer David Benkeser <benkeser@emory.edu>

Repository CRAN

Date/Publication 2022-04-23 08:00:06 UTC

R topics documented:

adaptive_iptw *Compute asymptotically linear IPTW estimators with super learning for the propensity score*

Description

Compute asymptotically linear IPTW estimators with super learning for the propensity score

Usage

```
adaptive_iptw(W, A, Y, DeltaY = as.numeric(!is.na(Y)),
  DeltaA = as.numeric(!is.na(A)), stratify = FALSE, family = if (all(Y
  %in% c(0, 1))) {     stats::binomial() } else {     stats::gaussian() },
  a_0 = unique(A[!is.na(A)]), SL_g = NULL, glm_g = NULL, SL_Qr = NULL,
  glm_Qr = NULL, returnModels = TRUE, verbose = FALSE, maxIter = 2,
  tolIC = 1/length(Y), tolg = 0.01, cvFolds = 1, gn = NULL, ...)
```

Arguments

W	A data.frame of named covariates
A	A numeric vector of binary treatment assignment (assumed to be equal to 0 or 1)
Y	A numeric numeric of continuous or binary outcomes.
DeltaY	A numeric indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
DeltaA	A numeric indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
stratify	A logical indicating whether to estimate the missing outcome regression separately for observations with different levels of A (if TRUE) or to pool across A (if FALSE).
family	A family object equal to either binomial() or gaussian(), to be passed to the SuperLearner or glm function.
a_0	A vector of numeric treatment values at which to return marginal mean estimates.
SL_g	A vector of characters describing the super learner library to be used for each of the propensity score regressions (DeltaA, A, and DeltaY). To use the same library for each of the regressions (or if there is no missing data in A nor Y), a single library may be input. See link{SuperLearner::SuperLearner} for details on how super learner libraries can be specified.
glm_g	A list of characters describing the formulas to be used for each of the propensity score regressions (DeltaA, A, and DeltaY). To use the same formula for each of the regressions (or if there is no missing data in A nor Y), a single character formula may be input.

SL_Qr	A vector of characters or a list describing the Super Learner library to be used for the reduced-dimension outcome regression.
glm_Qr	A character describing a formula to be used in the call to glm for reduced-dimension outcome regression. Ignored if SL_Qr!=NULL. The formula should use the variable name 'gn'.
returnModels	A logical indicating whether to return model fits for the propensity score and reduced-dimension regressions.
verbose	A logical indicating whether to print status updates.
maxIter	A numeric that sets the maximum number of iterations the TMLE can perform in its fluctuation step.
tolIC	A numeric that defines the stopping criteria based on the empirical mean of the influence function.
tolg	A numeric indicating the minimum value for estimates of the propensity score.
cvFolds	A numeric equal to the number of folds to be used in cross-validated fitting of nuisance parameters. If cvFolds = 1, no cross-validation is used.
gn	An optional list of propensity score estimates. If specified, the function will ignore the nuisance parameter estimation specified by SL_g and glm_g. The entries in the list should correspond to the propensity for the observed values of W, with order determined by the input to a_0 (e.g., if a_0 = c(0,1) then gn[[1]] should be propensity of A = 0 and gn[[2]] should be propensity of A = 1).
...	Other options (not currently used).

Value

An object of class "adaptive_iptw".

iptw_tmle A list of point estimates and covariance matrix for the IPTW estimator based on a targeted propensity score.

iptw_tmle_nuisance A list of the final TMLE estimates of the propensity score ($gnStar) and reduced-dimension regression ($QrnStar) evaluated at the observed data values.

iptw_os A list of point estimates and covariance matrix for the one-step correct IPTW estimator.

iptw_os_nuisance A list of the initial estimates of the propensity score and reduced-dimension regression evaluated at the observed data values.

iptw A list of point estimates for the standard IPTW estimator. No estimate of the covariance matrix is provided because theory does not support asymptotic Normality of the IPTW estimator if super learning is used to estimate the propensity score.

gnMod The fitted object for the propensity score. Returns NULL if returnModels = FALSE.

QrnMod The fitted object for the reduced-dimension regression that guards against misspecification of the outcome regression. Returns NULL if returnModels = FALSE.

a_0 The treatment levels that were requested for computation of covariate-adjusted means.

call The call to adaptive_iptw.

Examples

```
# load super learner
library(SuperLearner)
# simulate data
set.seed(123456)
n <- 100
W <- data.frame(W1 = runif(n), W2 = rnorm(n))
A <- rbinom(n, 1, plogis(W$W1 - W$W2))
Y <- rbinom(n, 1, plogis(W$W1 * W$W2 * A))
# fit iptw with maxIter = 1 to run fast

fit1 <- adaptive_iptw(
  W = W, A = A, Y = Y, a_0 = c(1, 0),
  SL_g = c("SL.glm", "SL.mean", "SL.step"),
  SL_Qr = "SL.npreg", maxIter = 1
)
```

---

average_est_cov_list *Helper function for averaging lists of estimates generated in the main* `for` *loop of* `drtmle`

Description

Helper function for averaging lists of estimates generated in the main `for` loop of `drtmle`

Usage

```
average_est_cov_list(est_cov_list)
```

Arguments

`est_cov_list`	A list with named entries `est` and `cov`

---

average_ic_list *Helper function to average convergence results and drtmle influence function estimates over multiple fits*

Description

Helper function to average convergence results and drtmle influence function estimates over multiple fits

Usage

```
average_ic_list(ic_list)
```

Arguments

ic_list	List of influence function estimates

---

ci *Compute confidence intervals for drtmle and adaptive_iptw@*

Description

Compute confidence intervals for drtmle and adaptive_iptw@

Usage

```
ci(...)
```

Arguments

...	Arguments to be passed to method

---

ci.adaptive_iptw *Confidence intervals for adaptive_iptw objects*

Description

Estimate confidence intervals for objects of class "adaptive_iptw"

Usage

```
## S3 method for class 'adaptive_iptw'
ci(object, est = c("iptw_tmle"), level = 0.95, contrast = NULL, ...)
```

Arguments

object	An object of class "adaptive_iptw"
est	A vector indicating for which estimators to return a confidence interval. Possible estimators include the TMLE IPTW ("iptw_tmle", recommended), the one-step IPTW ("iptw_os", not recommended), the standard IPTW ("iptw", recommended only for comparison to the other two estimators).
level	The nominal coverage probability of the desired confidence interval (should be between 0 and 1). Default computes 95\ intervals.

`contrast` Specifies the parameter for which to return confidence intervals. If `contrast=NULL`, then confidence intervals for the marginal means are computed. If instead, `contrast` is a numeric vector of ones, negative ones, and zeros to define linear combinations of the various means (e.g., to estimate an average treatment effect, see example). Finally, `contrast` can be a list with named functions `f`, `f_inv`, `h`, and `fh_grad`. The first two functions should take as input argument `eff`. Respectively, these specify which transformation of the effect measure to compute the confidence interval for and the inverse transformation to put the confidence interval back on the original scale. The function `h` defines the contrast to be estimated and should take as input `est`, a vector of the same length as `object$a_0`, and output the desired contrast. The function `fh_grad` is the gradient of the function `h`. See examples and vignette for more information.

`...` Other options (not currently used).

Value

An object of class `"ci.adaptive_iptw"` with point estimates and confidence intervals of the specified level.

Examples

```
# load super learner
library(SuperLearner)
# fit adaptive_iptw
set.seed(123456)
n <- 200
W <- data.frame(W1 = runif(n), W2 = rnorm(n))
A <- rbinom(n, 1, plogis(W$W1 - W$W2))
Y <- rbinom(n, 1, plogis(W$W1 * W$W2 * A))

fit1 <- adaptive_iptw(
  W = W, A = A, Y = Y, a_0 = c(1, 0),
  SL_g = c("SL.glm", "SL.mean", "SL.step"),
  SL_Qr = "SL.glm"
)

# get confidence intervals for each mean
ci_mean <- ci(fit1)

# get confidence intervals for ATE
ci_ATE <- ci(fit1, contrast = c(1, -1))

# get confidence intervals for risk ratio
# by inputting own contrast function
# this computes CI on log scale and back transforms
myContrast <- list(
  f = function(eff) {
    log(eff)
  },
  f_inv = function(eff) {
    exp(eff)
```

```
  },
  h = function(est) {
    est[1] / est[2]
  },
  fh_grad = function(est) {
    c(1 / est[1], -1 / est[2])
  }
)
ci_RR <- ci(fit1, contrast = myContrast)
```

`ci.drtmle` *Confidence intervals for drtmle objects*

Description

Confidence intervals for drtmle objects

Usage

```
## S3 method for class 'drtmle'
ci(object, est = c("drtmle"), level = 0.95, contrast = NULL, ...)
```

Arguments

`object`	An object of class `"drtmle"`
`est`	A vector indicating for which estimators to return a confidence interval. Possible estimators include the TMLE with doubly robust inference (`"drtmle"`, recommended), the AIPTW with additional correction for misspecification (`"aiptw_c"`, not recommended), the standard TMLE (`"tmle"`, recommended only for comparison to "drtmle"), the standard AIPTW (`"aiptw"`, recommended only for comparison to "drtmle"), and G-computation (`"gcomp"`, not recommended).
`level`	The nominal coverage probability of the desired confidence interval (should be between 0 and 1). Default computes 95\ intervals.
`contrast`	Specifies the parameter for which to return confidence intervals. If `contrast=NULL`, then confidence intervals for the marginal means are computed. If instead, `contrast` is a numeric vector of ones, negative ones, and zeros to define linear combinations of the various means (e.g., to estimate an average treatment effect, see example). Finally, `contrast` can be a list with named functions `f`, `f_inv`, `h`, and `fh_grad`. The first two functions should take as input argument `eff`. Respectively, these specify which transformation of the effect measure to compute the confidence interval for and the inverse transformation to put the confidence interval back on the original scale. The function `h` defines the contrast to be estimated and should take as input `est`, a vector of the same length as `object$a_0`, and output the desired contrast. The function `fh_grad` is the gradient of the function `h`. See examples and vignette for more information.
`...`	Other options (not currently used).

Value

An object of class "ci.drtmle" with point estimates and confidence intervals of the specified level.

Examples

```
# load super learner
library(SuperLearner)
# simulate data
set.seed(123456)
n <- 100
W <- data.frame(W1 = runif(n), W2 = rnorm(n))
A <- rbinom(n, 1, plogis(W$W1 - W$W2))
Y <- rbinom(n, 1, plogis(W$W1 * W$W2 * A))

# fit drtmle with maxIter = 1 to run fast
fit1 <- drtmle(
  W = W, A = A, Y = Y, a_0 = c(1, 0),
  family = binomial(),
  stratify = FALSE,
  SL_Q = c("SL.glm", "SL.mean"),
  SL_g = c("SL.glm", "SL.mean"),
  SL_Qr = "SL.npreg",
  SL_gr = "SL.npreg", maxIter = 1
)

# get confidence intervals for each mean
ci_mean <- ci(fit1)

# get confidence intervals for ATE
ci_ATE <- ci(fit1, contrast = c(1, -1))

# get confidence intervals for risk ratio by
# computing CI on log scale and back-transforming
myContrast <- list(
  f = function(eff) {
    log(eff)
  },
  f_inv = function(eff) {
    exp(eff)
  },
  h = function(est) {
    est[1] / est[2]
  },
  fh_grad = function(est) {
    c(1 / est[1], -1 / est[2])
  }
)
ci_RR <- ci(fit1, contrast = myContrast)
```

drtmle *TMLE estimate of the average treatment effect with doubly-robust inference*

Description

TMLE estimate of the average treatment effect with doubly-robust inference

Usage

```
drtmle(Y, A, W, DeltaA = as.numeric(!is.na(A)),
  DeltaY = as.numeric(!is.na(Y)), a_0 = unique(A[!is.na(A)]), family = if
  (all(Y %in% c(0, 1))) {     stats::binomial() } else {
  stats::gaussian() }, stratify = FALSE, SL_Q = NULL, SL_g = NULL,
  SL_Qr = NULL, SL_gr = NULL, n_SL = 1, avg_over = "drtmle",
  se_cv = "none", se_cvFolds = ifelse(se_cv == "partial", 10, 1),
  targeted_se = se_cv != "partial", glm_Q = NULL, glm_g = NULL,
  glm_Qr = NULL, glm_gr = NULL, adapt_g = FALSE, guard = c("Q", "g"),
  reduction = "univariate", returnModels = FALSE, returnNuisance = TRUE,
  cvFolds = 1, maxIter = 3, tolIC = 1/length(Y), tolg = 0.01,
  verbose = FALSE, Qsteps = 2, Qn = NULL, gn = NULL,
  use_future = FALSE, ...)
```

Arguments

Y	A numeric continuous or binary outcomes.
A	A numeric vector of discrete-valued treatment assignment.
W	A data.frame of named covariates.
DeltaA	A numeric vector of missing treatment indicator (assumed to be equal to 0 if missing 1 if observed).
DeltaY	A numeric vector of missing outcome indicator (assumed to be equal to 0 if missing 1 if observed).
a_0	A numeric vector of fixed treatment values at which to return marginal mean estimates.
family	A family object equal to either binomial() or gaussian(), to be passed to the SuperLearner or glm function.
stratify	A boolean indicating whether to estimate the outcome regression separately for different values of A (if TRUE) or to pool across A (if FALSE).
SL_Q	A vector of characters or a list describing the Super Learner library to be used for the outcome regression. See SuperLearner for details.
SL_g	A vector of characters describing the super learner library to be used for each of the propensity score regressions (DeltaA, A, and DeltaY). To use the same library for each of the regressions (or if there is no missing data in A nor Y), a single library may be input. See SuperLearner for details on how super learner libraries can be specified.

SL_Qr — A vector of characters or a list describing the Super Learner library to be used for the reduced-dimension outcome regression.

SL_gr — A vector of characters or a list describing the Super Learner library to be used for the reduced-dimension propensity score.

n_SL — Number of repeated Super Learners to run (default 1) for the each nuisance parameter. Repeat Super Learners more times to obtain more stable inference.

avg_over — If multiple Super Learners are run, on which scale should the results be aggregated. Options include: "SL" = repeated nuisance parameter estimates are averaged before subsequently generating a single vector of point estimates based on the averaged models; "drtmle" = repeated vectors of point estimates are generated and averaged. Both can be specified, recognizing that this adds considerable computational expense. In this case, the final estimates are the average of n_SL point estimates where each is built by averaging n_SL fits. If NULL, no averaging is performed (in which case n_SL should be set equal to 1).

se_cv — Should cross-validated nuisance parameter estimates be used for computing standard errors? Options are "none" = no cross-validation is performed; "partial" = only applicable if Super Learner is used for nuisance parameter estimates; "full" = full cross-validation is performed. See vignette for further details. Ignored if cvFolds > 1, since then cross-validated nuisance parameter estimates are used by default and it is assumed that you want full cross-validated standard errors.

se_cvFolds — If cross-validated nuisance parameter estimates are used to compute standard errors, how many folds should be used in this computation. If se_cv = "partial", then this option sets the number of folds used by the SuperLearner fitting procedure.

targeted_se — A boolean indicating whether the targeted nuisance parameters should be used in standard error computation or the initial estimators. If se_cv is not set to "none", this option is ignored and standard errors are computed based on non-targeted, cross-validated nuisance parameter fits.

glm_Q — A character describing a formula to be used in the call to glm for the outcome regression. Ignored if SL_Q!=NULL.

glm_g — A list of characters describing the formulas to be used for each of the propensity score regressions (DeltaA, A, and DeltaY). To use the same formula for each of the regressions (or if there are no missing data in A nor Y), a single character formula may be input. In general the formulas can reference any variable in colnames(W), unless adapt_g = TRUE in which case the formulas should reference variables QaW where a takes values in a_0.

glm_Qr — A character describing a formula to be used in the call to glm for reduced-dimension outcome regression. Ignored if SL_Qr!=NULL. The formula should use the variable name 'gn'.

glm_gr — A character describing a formula to be used in the call to glm for the reduced-dimension propensity score. Ignored if SL_gr!=NULL. The formula should use the variable name 'Qn' and 'gn' if reduction='bivariate' and 'Qn' otherwise.

adapt_g	A boolean indicating whether the propensity score should be outcome adaptive. If TRUE then the propensity score is estimated as the regression of A onto covariates QaW for a in each value contained in a_0. See vignette for more details.
guard	A character vector indicating what pattern of misspecifications to guard against. If guard contains "Q", then the TMLE guards against misspecification of the outcome regression by estimating the reduced-dimension outcome regression specified by glm_Qr or SL_Qr. If guard contains "g" then the TMLE (additionally) guards against misspecification of the propensity score by estimating the reduced-dimension propensity score specified by glm_gr or SL_gr. If guard is set to NULL, then only standard TMLE and one-step estimators are computed.
reduction	A character equal to "univariate" for a univariate misspecification correction (default) or "bivariate" for the bivariate version.
returnModels	A boolean indicating whether to return model fits for the outcome regression, propensity score, and reduced-dimension regressions.
returnNuisance	A boolean indicating whether to return the estimated nuisance regressions evaluated on the observed data. Defaults to TRUE. If n_SL is large and "drtmle" is in avg_over, then consider setting to FALSE in order to reduce size of resultant object.
cvFolds	A numeric equal to the number of folds to be used in cross-validated fitting of nuisance parameters. If cvFolds = 1, no cross-validation is used. Alternatively, cvFolds may be entered as a vector of fold assignments for observations, in which case its length should be the same length as Y.
maxIter	A numeric that sets the maximum number of iterations the TMLE can perform in its fluctuation step.
tolIC	A numeric that defines the stopping criteria based on the empirical mean of the influence function.
tolg	A numeric indicating the minimum value for estimates of the propensity score.
verbose	A boolean indicating whether to print status updates.
Qsteps	A numeric equal to 1 or 2 indicating whether the fluctuation submodel for the outcome regression should be fit using a single minimization (Qsteps = 1) or a backfitting-type minimization (Qsteps=2). The latter was found to be more stable in simulations and is the default.
Qn	An optional list of outcome regression estimates. If specified, the function will ignore the nuisance parameter estimation specified by SL_Q and glm_Q. The entries in the list should correspond to the outcome regression evaluated at A and the observed values of W, with order determined by the input to a_0 (e.g., if a_0 = c(0,1) then Qn[[1]] should be outcome regression at A = 0 and Qn[[2]] should be outcome regression at A = 1).
gn	An optional list of propensity score estimates. If specified, the function will ignore the nuisance parameter estimation specified by SL_g and glm_g. The entries in the list should correspond to the propensity for the observed values of W, with order determined by the input to a_0 (e.g., if a_0 = c(0,1) then gn[[1]] should be propensity of A = 0 and gn[[2]] should be propensity of A = 1).
use_future	Boolean indicating whether to use future_lapply or instead to just use lapply. The latter can be easier to run down errors.
...	Other options (not currently used).

Value

An object of class "drtmle".

drtmle A list of doubly-robust point estimates and a doubly-robust covariance matrix

nuisance_drtmle A list of the final TMLE estimates of the outcome regression ($QnStar), propensity score ($gnStar), and reduced-dimension regressions ($QrnStar, $grnStar) evaluated at the observed data values.

ic_drtmle A list of the empirical mean of the efficient influence function ($eif) and the extra pieces of the influence function resulting from misspecification. All should be smaller than tolIC (unless maxIter was reached first). Also includes a matrix of the influence function values at the estimated nuisance parameters evaluated at the observed data.

aiptw_c A list of doubly-robust point estimates and a non-doubly-robust covariance matrix. Theory does not guarantee performance of inference for these estimators, but simulation studies showed they often perform adequately.

nuisance_aiptw A list of the initial estimates of the outcome regression, propensity score, and reduced-dimension regressions evaluated at the observed data values.

tmle A list of doubly-robust point estimates and non-doubly-robust covariance for the standard TMLE estimator.

aiptw A list of doubly-robust point estimates and non-doubly-robust covariance matrix for the standard AIPTW estimator.

gcomp A list of non-doubly-robust point estimates and non-doubly-robust covariance matrix for the standard G-computation estimator. If super learner is used there is no guarantee of correct inference for this estimator.

QnMod The fitted object for the outcome regression. Returns NULL if returnModels = FALSE.

gnMod The fitted object for the propensity score. Returns NULL if returnModels = FALSE.

QrnMod The fitted object for the reduced-dimension regression that guards against misspecification of the outcome regression. Returns NULL if returnModels = FALSE.

grnMod The fitted object for the reduced-dimension regression that guards against misspecification of the propensity score. Returns NULL if returnModels = FALSE.

a_0 The treatment levels that were requested for computation of covariate-adjusted means.

Examples

```
# load super learner
library(SuperLearner)
# simulate data
set.seed(123456)
n <- 100
W <- data.frame(W1 = runif(n), W2 = rnorm(n))
A <- rbinom(n, 1, plogis(W$W1 - W$W2))
Y <- rbinom(n, 1, plogis(W$W1 * W$W2 * A))
# A quick example of drtmle:
# We note that more flexible super learner libraries
# are available, and that we recommend the user use more flexible
# libraries for SL_Qr and SL_gr for general use.
fit1 <- drtmle(
```

```
  W = W, A = A, Y = Y, a_0 = c(1, 0),
  family = binomial(),
  stratify = FALSE,
  SL_Q = c("SL.glm", "SL.mean", "SL.glm.interaction"),
  SL_g = c("SL.glm", "SL.mean", "SL.glm.interaction"),
  SL_Qr = "SL.glm",
  SL_gr = "SL.glm", maxIter = 1
)
```

---

estimateG *estimateG*

Description

Function to estimate propensity score

Usage

```
estimateG(A, W, DeltaY, DeltaA, SL_g, glm_g, a_0, tolg, stratify = FALSE,
  validRows = NULL, verbose = FALSE, returnModels = FALSE, Qn = NULL,
  adapt_g = FALSE, se_cv = "none", se_cvFolds = 10)
```

Arguments

A	A vector of binary treatment assignment (assumed to be equal to 0 or 1)
W	A data.frame of named covariates
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
SL_g	A vector of characters describing the super learner library to be used for each of the regression (DeltaA, A, and DeltaY). To use the same regression for each of the regressions (or if there is no missing data in A nor Y), a single library may be input.
glm_g	A character describing a formula to be used in the call to glm for the propensity score.
a_0	A vector of fixed treatment values at which to return marginal mean estimates.
tolg	A numeric indicating the minimum value for estimates of the propensity score.
stratify	A boolean indicating whether to estimate the missing outcome regression separately for observations with A equal to 0/1 (if TRUE) or to pool across A (if FALSE).
validRows	A list of length cvFolds containing the row indexes of observations to include in validation fold.
verbose	A boolean indicating whether to print status updates.
returnModels	A boolean indicating whether to return model fits for the outcome regression, propensity score, and reduced-dimension regressions.

Qn	A list of estimates of the outcome regression for each value in a_0. Only needed if adapt_g = TRUE.
adapt_g	A boolean indicating whether propensity score is adaptive to outcome regression.
se_cv	Should cross-validated nuisance parameter estimates be used for computing standard errors? Options are "none" = no cross-validation is performed; "partial" = only applicable if Super Learner is used for nuisance parameter estimates; "full" = full cross-validation is performed. See vignette for further details. Ignored if cvFolds > 1, since then cross-validated nuisance parameter estimates are used by default and it is assumed that you want full cross-validated standard errors.
se_cvFolds	If cross-validated nuisance parameter estimates are used to compute standard errors, how many folds should be used in this computation. If se_cv = "partial", then this option sets the number of folds used by the SuperLearner fitting procedure.

`estimategrn` *estimategrn*

Description

Estimates the reduced dimension regressions necessary for the additional fluctuations.

Usage

```
estimategrn(Y, A, W, DeltaA, DeltaY, Qn, gn, SL_gr, tolg, glm_gr, a_0,
  reduction, returnModels, validRows)
```

Arguments

Y	A vector of continuous or binary outcomes.
A	A vector of binary treatment assignment (assumed to be equal to 0 or 1).
W	A data.frame of named covariates.
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed).
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed).
Qn	A list of outcome regression estimates evaluated on observed data.
gn	A list of propensity regression estimates evaluated on observed data.
SL_gr	A vector of characters or a list describing the Super Learner library to be used for the reduced-dimension propensity score.
tolg	A numeric indicating the minimum value for estimates of the propensity score.
glm_gr	A character describing a formula to be used in the call to glm for the second reduced-dimension regression. Ignored if SL_gr!=NULL.

a_0	A list of fixed treatment values .
reduction	A character equal to 'univariate' for a univariate misspecification correction or 'bivariate' for the bivariate version.
returnModels	A boolean indicating whether to return model fits for the outcome regression, propensity score, and reduced-dimension regressions.
validRows	A list of length cvFolds containing the row indexes of observations to include in validation fold.

---

estimategrn_loop *estimategrn_loop*

Description

Helper function to clean up the internal code of drtmle

Usage

```
estimategrn_loop(validRows, Y, A, W, DeltaA, DeltaY, tolg, Qn, gn, glm_gr,
  SL_gr, a_0, reduction, returnModels, use_future)
```

Arguments

validRows	A list of length cvFolds containing the row indexes of observations to include in validation fold.
Y	A vector of continuous or binary outcomes.
A	A vector of binary treatment assignment (assumed to be equal to 0 or 1).
W	A data.frame of named covariates.
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed).
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed).
tolg	A numeric indicating the minimum value for estimates of the propensity score.
Qn	A list of outcome regression estimates evaluated on observed data.
gn	A list of propensity regression estimates evaluated on observed data.
glm_gr	A character describing a formula to be used in the call to glm for the second reduced-dimension regression. Ignored if SL_gr!=NULL.
SL_gr	A vector of characters or a list describing the Super Learner library to be used for the reduced-dimension propensity score.
a_0	A list of fixed treatment values .
reduction	A character equal to 'univariate' for a univariate misspecification correction or 'bivariate' for the bivariate version.
returnModels	A boolean indicating whether to return model fits for the outcome regression, propensity score, and reduced-dimension regressions.
use_future	Should future be used to parallelize?

estimateG_loop *estimateG_loop*

Description

Helper function to clean up internals of drtmle function

Usage

```
estimateG_loop(validRows, A, W, DeltaA, DeltaY, tolg, verbose, stratify,
  returnModels, SL_g, glm_g, a_0, Qn, adapt_g, use_future, se_cv = "none",
  se_cvFolds = 10)
```

Arguments

validRows	A list of length cvFolds containing the row indexes of observations to include in validation fold.
A	A vector of binary treatment assignment (assumed to be equal to 0 or 1)
W	A data.frame of named covariates
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
tolg	A numeric indicating the minimum value for estimates of the propensity score.
verbose	A boolean indicating whether to print status updates.
stratify	A boolean indicating whether to estimate the missing outcome regression separately for observations with A equal to 0/1 (if TRUE) or to pool across A (if FALSE).
returnModels	A boolean indicating whether to return model fits for the outcome regression, propensity score, and reduced-dimension regressions.
SL_g	A vector of characters describing the super learner library to be used for each of the regression (DeltaA, A, and DeltaY). To use the same regression for each of the regressions (or if there is no missing data in A nor Y), a single library may be input.
glm_g	A character describing a formula to be used in the call to glm for the propensity score.
a_0	A vector of fixed treatment values at which to return marginal mean estimates.
Qn	A list of estimates of the outcome regression for each value in a_0. Only needed if adapt_g = TRUE.
adapt_g	A boolean indicating whether propensity score is adaptive to outcome regression.
use_future	Should future be used for parallelization?

se_cv	Should cross-validated nuisance parameter estimates be used for computing standard errors? Options are "none" = no cross-validation is performed; "partial" = only applicable if Super Learner is used for nuisance parameter estimates; "full" = full cross-validation is performed. See vignette for further details. Ignored if cvFolds > 1, since then cross-validated nuisance parameter estimates are used by default and it is assumed that you want full cross-validated standard errors.
se_cvFolds	If cross-validated nuisance parameter estimates are used to compute standard errors, how many folds should be used in this computation. If se_cv = "partial", then this option sets the number of folds used by the SuperLearner fitting procedure.

`estimateQ` *estimateQ*

Description

Function to estimate initial outcome regression

Usage

```
estimateQ(Y, A, W, DeltaA, DeltaY, SL_Q, glm_Q, a_0, stratify, family,
  verbose = FALSE, returnModels = FALSE, se_cv = "none",
  se_cvFolds = 10, validRows = NULL, ...)
```

Arguments

Y	A vector of continuous or binary outcomes.
A	A vector of binary treatment assignment (assumed to be equal to 0 or 1).
W	A data.frame of named covariates.
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed).
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed).
SL_Q	A vector of characters or a list describing the Super Learner library to be used for the outcome regression.
glm_Q	A character describing a formula to be used in the call to glm for the outcome regression.
a_0	A list of fixed treatment values
stratify	A boolean indicating whether to estimate the outcome regression separately for observations with A equal to 0/1 (if TRUE) or to pool across A (if FALSE).
family	A character passed to SuperLearner
verbose	A boolean indicating whether to print status updates.
returnModels	A boolean indicating whether to return model fits for the outcome regression, propensity score, and reduced-dimension regressions.

se_cv	Should cross-validated nuisance parameter estimates be used for computing standard errors? Options are "none" = no cross-validation is performed; "partial" = only applicable if Super Learner is used for nuisance parameter estimates; "full" = full cross-validation is performed. See vignette for further details. Ignored if cvFolds > 1, since then cross-validated nuisance parameter estimates are used by default and it is assumed that you want full cross-validated standard errors.
se_cvFolds	If cross-validated nuisance parameter estimates are used to compute standard errors, how many folds should be used in this computation. If se_cv = "partial", then this option sets the number of folds used by the SuperLearner fitting procedure.
validRows	A list of length cvFolds containing the row indexes of observations to include in validation fold.
...	Additional arguments (not currently used)

---

estimateQrn *estimateQrn*

Description

Estimates the reduced dimension regressions necessary for the fluctuations of g

Usage

```
estimateQrn(Y, A, W, DeltaA, DeltaY, Qn, gn, glm_Qr, SL_Qr,
  family = stats::gaussian(), a_0, returnModels, validRows = NULL)
```

Arguments

Y	A vector of continuous or binary outcomes.
A	A vector of binary treatment assignment (assumed to be equal to 0 or 1)
W	A data.frame of named covariates
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
Qn	A list of outcome regression estimates evaluated on observed data. If NULL then 0 is used for all Qn (as is needed to estimate reduced dimension regression for adaptive_iptw)
gn	A list of propensity regression estimates evaluated on observed data
glm_Qr	A character describing a formula to be used in the call to glm for the first reduced-dimension regression. Ignored if SL_gr!=NULL.
SL_Qr	A vector of characters or a list describing the Super Learner library to be used for the first reduced-dimension regression.
family	Should be gaussian() unless called from adaptive_iptw with binary Y.

a_0	A list of fixed treatment values.
returnModels	A boolean indicating whether to return model fits for the outcome regression, propensity score, and reduced-dimension regressions.
validRows	A list of length cvFolds containing the row indexes of observations to include in validation fold.

---

estimateQrn_loop *estimateQrn_loop*

Description

Helper function to clean up internal code of drtmle function.

Usage

```
estimateQrn_loop(validRows, Y, A, W, DeltaA, DeltaY, Qn, gn, SL_Qr, glm_Qr,
  family, a_0, returnModels, use_future)
```

Arguments

validRows	A list of length cvFolds containing the row indexes of observations to include in validation fold.
Y	A vector of continuous or binary outcomes.
A	A vector of binary treatment assignment (assumed to be equal to 0 or 1)
W	A data.frame of named covariates
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
Qn	A list of outcome regression estimates evaluated on observed data. If NULL then 0 is used for all Qn (as is needed to estimate reduced dimension regression for adaptive_iptw)
gn	A list of propensity regression estimates evaluated on observed data
SL_Qr	A vector of characters or a list describing the Super Learner library to be used for the first reduced-dimension regression.
glm_Qr	A character describing a formula to be used in the call to glm for the first reduced-dimension regression. Ignored if SL_gr!=NULL.
family	Should be gaussian() unless called from adaptive_iptw with binary Y.
a_0	A list of fixed treatment values.
returnModels	A boolean indicating whether to return model fits for the outcome regression, propensity score, and reduced-dimension regressions.
use_future	Should future be used in the fitting process.

estimateQ_loop *estimateQ_loop*

Description

A helper loop function to clean up the internals of drtmle function.

Usage

```
estimateQ_loop(validRows, Y, A, W, DeltaA, DeltaY, verbose, returnModels, SL_Q,
  a_0, stratify, glm_Q, family, use_future, se_cv, se_cvFolds)
```

Arguments

validRows	A list of length cvFolds containing the row indexes of observations to include in validation fold.
Y	A vector of continuous or binary outcomes.
A	A vector of binary treatment assignment (assumed to be equal to 0 or 1)
W	A data.frame of named covariates
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
verbose	A boolean indicating whether to print status updates.
returnModels	A boolean indicating whether to return model fits for the outcome regression, propensity score, and reduced-dimension regressions.
SL_Q	A vector of characters or a list describing the Super Learner library to be used for the outcome regression. See SuperLearner for details.
a_0	A list of fixed treatment values.
stratify	A boolean indicating whether to estimate the outcome regression separately for different values of A (if TRUE) or to pool across A (if FALSE).
glm_Q	A character describing a formula to be used in the call to glm for the outcome regression. Ignored if SL_Q!=NULL.
family	Should be gaussian() unless called from adaptive_iptw with binary Y.
use_future	Boolean indicating whether to use future_lapply or instead to just use lapply. The latter can be easier to run down errors.
se_cv	Should cross-validated nuisance parameter estimates be used for computing standard errors? Options are "none" = no cross-validation is performed; "partial" = only applicable if Super Learner is used for nuisance parameter estimates; "full" = full cross-validation is performed. See vignette for further details. Ignored if cvFolds > 1, since then cross-validated nuisance parameter estimates are used by default and it is assumed that you want full cross-validated standard errors.
se_cvFolds	If cross-validated nuisance parameter estimates are used to compute standard errors, how many folds should be used in this computation. If se_cv = "partial", then this option sets the number of folds used by the SuperLearner fitting procedure.

eval_Diptw — *Evaluate usual influence function of IPTW*

Description

Evaluate usual influence function of IPTW

Usage

```
eval_Diptw(A, Y, DeltaA, DeltaY, gn, psi_n, a_0)
```

Arguments

A	A vector of binary treatment assignment (assumed to be equal to 0 or 1)
Y	A numeric of continuous or binary outcomes.
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
gn	List of estimated propensity scores evaluated at observations
psi_n	List of estimated ATEs
a_0	Vector of values to return marginal mean

eval_Diptw_g — *Evaluate extra piece of the influence function for the IPTW*

Description

Evaluate extra piece of the influence function for the IPTW

Usage

```
eval_Diptw_g(A, DeltaA, DeltaY, Qrn, gn, a_0)
```

Arguments

A	A vector of binary treatment assignment (assumed to be equal to 0 or 1)
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
Qrn	List of estimated reduced-dimension outcome regression evaluated at observations
gn	List of estimated propensity scores evaluated at observations
a_0	Vector of values to return marginal mean

eval_Dstar — *Evaluate usual efficient influence function*

Description

Evaluate usual efficient influence function

Usage

```
eval_Dstar(A, Y, DeltaY, DeltaA, Qn, gn, psi_n, a_0)
```

Arguments

A	A vector of binary treatment assignment (assumed to be equal to 0 or 1)
Y	A numeric of continuous or binary outcomes.
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
Qn	List of estimated outcome regression evaluated at observations
gn	List of estimated propensity scores evaluated at observations
psi_n	List of estimated ATEs
a_0	Vector of values to return marginal mean

eval_Dstar_g — *Evaluate extra piece of efficient influence function resulting from misspecification of outcome regression*

Description

Evaluate extra piece of efficient influence function resulting from misspecification of outcome regression

Usage

```
eval_Dstar_g(A, DeltaY, DeltaA, Qrn, gn, a_0)
```

Arguments

A	A vector of binary treatment assignment (assumed to be equal to 0 or 1)
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
Qrn	List of estimated reduced-dimension outcome regression evaluated at observations
gn	List of estimated propensity scores evaluated at observations
a_0	Vector of values to return marginal mean

eval_Dstar_Q — *Evaluate extra piece of efficient influence function resulting from misspecification of propensity score*

Description

Evaluate extra piece of efficient influence function resulting from misspecification of propensity score

Usage

```
eval_Dstar_Q(A, Y, DeltaY, DeltaA, Qn, gn, grn, a_0, reduction)
```

Arguments

A	A vector of binary treatment assignment (assumed to be equal to 0 or 1)
Y	A numeric of continuous or binary outcomes.
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
Qn	List of estimated outcome regression evaluated at observations
gn	List of estimated propensity scores evaluated at observations
grn	List of estimated reduced-dimension propensity scores evaluated at observations
a_0	Vector of values to return marginal mean
reduction	A character equal to "univariate" for a univariate misspecification correction or "bivariate" for the bivariate version.

extract_models — *Help function to extract models from fitted object*

Description

Help function to extract models from fitted object

Usage

```
extract_models(a_list)
```

Arguments

a_list	Structured list of nuisance parameters

`fluctuateG` *fluctuateG*

Description

Function called internally by drtmle to perform the fluctuation of the initial estimator of g (i.e., solves the new estimating eqn that results from misspecification of Q)

Usage

```
fluctuateG(Y, A, W, DeltaY, DeltaA, a_0, gn, Qrn, tolg, coefTol = 1000)
```

Arguments

`Y`	The outcome
`A`	The treatment
`W`	The covariates
`DeltaY`	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
`DeltaA`	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
`a_0`	A list of fixed treatment values
`gn`	A list of propensity regression estimates evaluated on observed data
`Qrn`	A list of reduced-dimension regression estimates evaluated on observed data
`tolg`	The lower bound on propensity score estimates
`coefTol`	A tolerance level on the magnitude of the coefficient that flags the result as potentially the result of numeric instability.

`fluctuateQ` *fluctuateQ*

Description

Function called internally by drtmle to perform simultaneous fluctuation of the initial estimator of Q (i.e., solves both EIF estimating eqn and the new estimating eqn that results from misspecification of g)

Usage

```
fluctuateQ(Y, A, W, DeltaY, DeltaA, Qn, gn, grn, a_0, reduction,
  coefTol = 1000)
```

Arguments

Y	The outcome
A	The treatment
W	The covariates
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
Qn	A list of outcome regression estimates evaluated on observed data
gn	A list of propensity regression estimates evaluated on observed data
grn	A list of reduced-dimension regression estimates evaluated on observed data
a_0	A list of fixed treatment values
reduction	A character indicating what reduced dimension regression was used.
coefTol	A tolerance level on the magnitude of the coefficient that flags the result as potentially the result of numeric instability.

---

fluctuateQ1 *fluctuateQ1*

Description

Function called internally by drtmle to perform the first fluctuation of the initial estimator of Q (i.e., solves the original EIF estimating eqn)

Usage

```
fluctuateQ1(Y, A, W, DeltaA, DeltaY, Qn, gn, a_0, coefTol = 1000)
```

Arguments

Y	The outcome
A	The treatment
W	The covariates
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
Qn	A list of outcome regression estimates evaluated on observed data
gn	A list of propensity regression estimates evaluated on observed data
a_0	A list of fixed treatment values
coefTol	A tolerance level on the magnitude of the coefficient that flags the result as potentially the result of numeric instability.

fluctuateQ2 *fluctuateQ2*

Description

Function called internally by drtmle to perform the second fluctuation of the initial estimator of Q (i.e., solves the new estimating eqn that results from misspecification of g)

Usage

```
fluctuateQ2(Y, A, W, DeltaY, DeltaA, Qn, gn, grn, a_0, reduction,
  coefTol = 1000)
```

Arguments

Y	The outcome
A	The treatment
W	The covariates
DeltaY	Indicator of missing outcome (assumed to be equal to 0 if missing 1 if observed)
DeltaA	Indicator of missing treatment (assumed to be equal to 0 if missing 1 if observed)
Qn	A list of outcome regression estimates evaluated on observed data
gn	A list of propensity regression estimates evaluated on observed data
grn	A list of reduced-dimension regression estimates evaluated on observed data
a_0	A list of fixed treatment values
reduction	A character indicating what reduced dimension regression was used.
coefTol	A tolerance level on the magnitude of the coefficient that flags the result as potentially the result of numeric instability.

make_validRows *Make list of rows in each validation fold.*

Description

Make list of rows in each validation fold.

Usage

```
make_validRows(cvFolds, n, ...)
```

Arguments

cvFolds	Numeric number of cv folds
n	Number of observations
...	Other arguments

partial_cv_preds — *Helper function to properly format partially cross-validated predictions from a fitted super learner.*

Description

Helper function to properly format partially cross-validated predictions from a fitted super learner.

Usage

```
partial_cv_preds(fit_sl, a_0, W = NULL, family, include = NULL, easy = FALSE)
```

Arguments

fit_sl	A fitted SuperLearner object with control$saveCVFitLibrary = TRUE
a_0	Treatment level to set. If NULL, assume this function is being used to get partially cross-validated propensity score predictions.
W	A data.frame of named covariates.
family	Family of prediction model
include	A boolean vector indicating which observations were actually used to fit the regression.
easy	A boolean indicating whether the predictions can be computed the "easy" way, i.e., based just on the Z matrix from SuperLearner. This is possible for propensity score models when no missing data AND no stratification.

plot.drtmle — *Plot reduced dimension regression fits*

Description

Plot reduced dimension regression fits

Usage

```
## S3 method for class 'drtmle'
plot(x, nPoints = 500, ask = TRUE, a_0 = x$a_0[1], ...)
```

Arguments

x	An object of class "drtmle"
nPoints	Number of points to plot lines (increase for less bumpy plot, decrease for faster evaluation).
ask	Boolean indicating whether R should ask to show each plot
a_0	For what value of a_0 should the plot be made for?
...	More arguments passed to plot

Examples

```
# load super learner
library(SuperLearner)
# simulate data
set.seed(123456)
n <- 100
W <- data.frame(W1 = runif(n), W2 = rnorm(n))
A <- rbinom(n, 1, plogis(W$W1 - W$W2))
Y <- rbinom(n, 1, plogis(W$W1 * W$W2 * A))
# fit drtmle with maxIter = 1 to run fast

fit1 <- drtmle(
  W = W, A = A, Y = Y, a_0 = c(1, 0),
  family = binomial(),
  stratify = FALSE,
  SL_Q = c("SL.glm", "SL.mean", "SL.glm.interaction"),
  SL_g = c("SL.glm", "SL.mean", "SL.glm.interaction"),
  SL_Qr = "SL.npreg", SL_gr = "SL.npreg",
  maxIter = 1, returnModels = TRUE
)
# plot the reduced-dimension regression fits (not run)

plot(fit1)

#
```

`predict.SL.npreg` *Predict method for SL.npreg*

Description

Method for predicting SL.npreg objects.

Usage

```
## S3 method for class 'SL.npreg'
predict(object, newdata, ...)
```

Arguments

`object`	An object of class `"SL.npreg"`.
`newdata`	The new data used to obtain predictions.
`...`	Other arguments passed to predict.

Examples

```
# simulate data
set.seed(1234)
n <- 100
X <- data.frame(X1 = rnorm(n))
Y <- X$X1 + rnorm(n)
# fit npreg
fit <- SL.npreg(Y = Y, X = X, newX = X)
# predict on fit
newX <- data.frame(X1 = c(-1, 0, 1))
pred <- predict(fit$fit, newdata = newX)
#
```

---

print.adaptive_iptw *Print the output of a "adaptive_iptw" object.*

Description

Print the output of a "adaptive_iptw" object.

Usage

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

Arguments

x	A "adaptive_iptw" object.
...	Other arguments (not used)

---

print.ci.adaptive_iptw *Print the output of ci.adaptive_iptw*

Description

Print the output of ci.adaptive_iptw

Usage

```
## S3 method for class 'ci.adaptive_iptw'
print(x, digits = 3, ...)
```

Arguments

x	An object of class ci.adaptive_iptw
digits	Number of digits to round to
...	Other options (not currently used)

print.ci.drtmle *Print the output of ci.drtmle*

Description

Print the output of ci.drtmle

Usage

```
## S3 method for class 'ci.drtmle'
print(x, digits = 3, ...)
```

Arguments

x	An object of class ci.drtmle
digits	Number of digits to round to
...	Other options (not currently used)

print.drtmle *Print the output of a* "drtmle" *object.*

Description

Print the output of a "drtmle" object.

Usage

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

Arguments

x	A "drtmle" object
...	Other arguments (not used)

print.wald_test.adaptive_iptw

Print the output of wald_test.adaptive_iptw

Description

Print the output of wald_test.adaptive_iptw

Usage

```
## S3 method for class 'wald_test.adaptive_iptw'
print(x, digits = 3, ...)
```

Arguments

x	An object of class wald_test.adaptive_iptw
digits	Number of digits to round to
...	Other options (not currently used)

print.wald_test.drtmle

Print the output of wald_test.drtmle

Description

Print the output of wald_test.drtmle

Usage

```
## S3 method for class 'wald_test.drtmle'
print(x, digits = 3, ...)
```

Arguments

x	An object of class wald_test.drtmle
digits	Number of digits to round to
...	Other options (not currently used)

reorder_list *Helper function to reorder lists according to cvFolds*

Description

Helper function to reorder lists according to cvFolds

Usage

```
reorder_list(a_list, a_0, validRows, n_SL = 1, grn_ind = FALSE, n,
  for_se_cv = FALSE)
```

Arguments

a_list	Structured list of nuisance parameters
a_0	Treatment levels
validRows	List of rows of data in validation data for each split.
n_SL	Number of super learners. If >1, then predictions are averaged
grn_ind	Structure of grn call is slightly different
n	Sample size
for_se_cv	Is this being used to average over cross-validated standard errors? Affects index of a_list.

SL.npreg *Super learner wrapper for kernel regression*

Description

Kernel regression based on the np package. Uses leave-one-out cross-validation to fit a kernel regression. See ?npreg for more details.

Usage

```
SL.npreg(Y, X, newX, family = gaussian(), obsWeights = rep(1, length(Y)),
  rangeThresh = 1e-07, ...)
```

Arguments

Y	A vector of outcomes.
X	A matrix or data.frame of training data predictors.
newX	A test set of predictors.
family	Not used by the function directly, but ensures compatibility with SuperLearner.
obsWeights	Not used by the function directly, but ensures compatibility with SuperLearner.
rangeThresh	If the the range of the outcomes is smaller than this number, the method returns the empirical average of the outcomes. Used for computational expediency and stability.
...	Other arguments (not currently used).

Examples

```
# simulate data
set.seed(1234)
n <- 100
X <- data.frame(X1 = rnorm(n))
Y <- X$X1 + rnorm(n)
# fit npreg
fit <- SL.npreg(Y = Y, X = X, newX = X)
#
```

---

tmp_method.CC_LS *Temporary fix for convex combination method mean squared error Relative to existing implementation, we reduce the tolerance at which we declare predictions from a given algorithm the same as another*

Description

Temporary fix for convex combination method mean squared error Relative to existing implementation, we reduce the tolerance at which we declare predictions from a given algorithm the same as another

Usage

```
tmp_method.CC_LS()
```

`tmp_method.CC_nloglik` *Temporary fix for convex combination method negative log-likelihood loss Relative to existing implementation, we reduce the tolerance at which we declare predictions from a given algorithm the same as another. Note that because of the way* `SuperLearner` *is structure, one needs to install the optimization software separately.*

Description

Temporary fix for convex combination method negative log-likelihood loss Relative to existing implementation, we reduce the tolerance at which we declare predictions from a given algorithm the same as another. Note that because of the way `SuperLearner` is structure, one needs to install the optimization software separately.

Usage

```
tmp_method.CC_nloglik()
```

`wald_test` *Wald tests for drtmle and adaptive_iptw objects*

Description

Wald tests for drtmle and adaptive_iptw objects

Usage

```
wald_test(...)
```

Arguments

...	Arguments to be passed to method

`wald_test.adaptive_iptw` *Wald tests for adaptive_iptw objects*

Description

Wald tests for adaptive_iptw objects

Usage

```
## S3 method for class 'adaptive_iptw'
wald_test(object, est = c("iptw_tmle"), null = 0, contrast = NULL, ...)
```

Arguments

`object`	An object of class `"adaptive_iptw"`
`est`	A vector indicating for which estimators to return a confidence interval. Possible estimators include the TMLE IPTW (`"iptw_tmle"`, recommended), the one-step IPTW (`"iptw_os"`, not recommended), the standard IPTW (`"iptw"`, recommended only for comparison to the other two estimators).
`null`	The null hypothesis value(s).
`contrast`	This option specifies what parameter to return confidence intervals for. If `contrast=NULL`, then test the null hypothesis that the covariate-adjusted marginal means equal the value(s) specified in `null`. `contrast` can also be a numeric vector of ones, negative ones, and zeros to define linear combinations of the various means (e.g., to estimate an average treatment effect, see examples). In this case, we test the null hypothesis that the linear combination of means equals the value specified in `null`. `contrast` can also be a list with named functions `f`, `h`, and `fh_grad`. The function `f` takes as input argument `eff` and specifies which transformation of the effect measure to test. The function `h` defines the contrast to be estimated and should take as input `est`, a vector of the same length as `object$a_0`, and output the desired contrast. The function `fh_grad` is the gradient of the function `h(f())`. The function computes a test of the null hypothesis that `h(f(object$est)) = null`. See examples.
`...`	Other options (not currently used).

Value

An object of class `"ci.adaptive_iptw"` with point estimates and confidence intervals of the specified level.

Examples

```
# load super learner
library(SuperLearner)
# fit adaptive_iptw
```

```
set.seed(123456)
n <- 200
W <- data.frame(W1 = runif(n), W2 = rnorm(n))
A <- rbinom(n, 1, plogis(W$W1 - W$W2))
Y <- rbinom(n, 1, plogis(W$W1 * W$W2 * A))

fit1 <- adaptive_iptw(
  W = W, A = A, Y = Y, a_0 = c(1, 0),
  SL_g = c("SL.glm", "SL.mean", "SL.step"),
  SL_Qr = "SL.glm"
)

# get test that each mean = 0.5
test_mean <- wald_test(fit1, null = 0.5)

# get test that the ATE = 0
ci_ATE <- ci(fit1, contrast = c(1, -1), null = 0)

# get test for risk ratio = 1 on log scale
myContrast <- list(
  f = function(eff) {
    log(eff)
  },
  f_inv = function(eff) {
    exp(eff)
  }, # not necessary
  h = function(est) {
    est[1] / est[2]
  },
  fh_grad = function(est) {
    c(1 / est[1], -1 / est[2])
  }
)
ci_RR <- ci(fit1, contrast = myContrast, null = 1)
#
```

`wald_test.drtmle` *Wald tests for drtmle objects*

Description

Wald tests for drtmle objects

Usage

```
## S3 method for class 'drtmle'
wald_test(object, est = c("drtmle"), null = 0, contrast = NULL, ...)
```

Arguments

object	An object of class "drtmle"
est	A vector indicating for which estimators to return a confidence interval. Possible estimators include the TMLE with doubly robust inference ("drtmle", recommended), the AIPTW with additional correction for misspecification ("aiptw_c", not recommended), the standard TMLE ("tmle", recommended only for comparison to "drtmle"), the standard AIPTW ("aiptw", recommended only for comparison to "drtmle"), and G-computation ("gcomp", not recommended).
null	The null hypothesis value.
contrast	This option specifies what parameter to return confidence intervals for. If contrast=NULL, then test the null hypothesis that the covariate-adjusted marginal means equal the value(s) specified in null. contrast can also be a numeric vector of ones, negative ones, and zeros to define linear combinations of the various means (e.g., to estimate an average treatment effect, see examples). In this case, we test the null hypothesis that the linear combination of means equals the value specified in null. contrast can also be a list with named functions f, h, and fh_grad. The function f takes as input argument eff and specifies which transformation of the effect measure to test. The function h defines the contrast to be estimated and should take as input est, a vector of the same length as object$a_0, and output the desired contrast. The function fh_grad is the gradient of the function h(f()). The function computes a test of the null hypothesis that h(f(object$est)) = null. See examples.
...	Other options (not currently used).

Value

An object of class "ci.drtmle" with point estimates and confidence intervals of the specified level.

Examples

```
# load super learner
library(SuperLearner)
# simulate data
set.seed(123456)
n <- 100
W <- data.frame(W1 = runif(n), W2 = rnorm(n))
A <- rbinom(n, 1, plogis(W$W1 - W$W2))
Y <- rbinom(n, 1, plogis(W$W1 * W$W2 * A))
# fit drtmle with maxIter = 1 so runs fast
fit1 <- drtmle(
  W = W, A = A, Y = Y, a_0 = c(1, 0),
  family = binomial(),
  stratify = FALSE,
  SL_Q = c("SL.glm", "SL.mean", "SL.glm.interaction"),
  SL_g = c("SL.glm", "SL.mean", "SL.glm.interaction"),
  SL_Qr = "SL.glm",
  SL_gr = "SL.glm", maxIter = 1
)
# get hypothesis test that each mean = 0.5
```

```
test_mean <- wald_test(fit1, null = 0.5)

# get test that ATE = 0
test_ATE <- wald_test(fit1, null = 0, contrast = c(1, -1))

# get test that risk ratio = 1, computing test on log scale
myContrast <- list(
  f = function(eff) {
    log(eff)
  },
  f_inv = function(eff) {
    exp(eff)
  },
  h = function(est) {
    est[1] / est[2]
  },
  fh_grad = function(est) {
    c(1 / est[1], -1 / est[2])
  }
)
test_RR <- wald_test(fit1, contrast = myContrast, null = 1)
#
```

Index