Package 'PLUCR'

May 7, 2026

Title Policy Learning Under Constraint

Version 0.1.0

Description Estimating treatment probabilities that maximize a primary clinical outcome while controlling for the occurrence of adverse events. The 'PLUCR' package implements methods for constrained policy learning.

License AGPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr, ggplot2, grf, gridExtra, magrittr, R.utils, scales, stats, SuperLearner, tibble, tidyr, tidyselect, glue

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Laura Fuentes [aut, cre]

Maintainer Laura Fuentes <laura.fuentes-vicente@inria.fr>

Repository CRAN

Date/Publication 2026-03-30 09:10:02 UTC

Contents

`binary_S_p` *Constraint function for binary policy*

Description

Computes the constraint function S_p, which ensures that the learned policy satisfies a constraint. This function enforces a limit on the expected impact of treatment via `delta_Z`.

Usage

```
binary_S_p(pi_X, X, alpha, delta_Nu)
```

Arguments

`pi_X`	A numeric vector of binary decisions of length n.
`X`	A matrix of covariates of size n x d (input data in `[0,1]`).
`alpha`	A numeric scalar representing the constraint tolerance (in `[0,1/2]`, 0.1 by default).
`delta_Nu`	A function of `X` that determines the contrast between adverse event outcomes.

Value

A numeric scalar representing the constraint function value.

`check_data` *Check input data for validity*

Description

Performs quality control checks on the input data to ensure it meets expected formats and conditions.

Usage

```
check_data(Y, Xi, A, X, folds)
```

Arguments

Y	A numeric vector or matrix of length n representing primary outcomes (in [0,1]).
Xi	A numeric vector or matrix of length n indicating adverse events (0 or 1).
A	A binary vector or matrix of length n indicating treatment assignment (0 or 1).
X	A matrix or data frame of covariates of size n x d (input data in [0,1]).
folds	A list of cross-validation folds (e.g., a list of indices for each fold).

Value

A list with elements: ok (logical), and diagnoses (character vector of issues).

CVFolds *CVFolds (from SuperLearner package)*

Description

Creates a list of row numbers for the V-fold cross validation.

Usage

```
CVFolds(N, id, Y, cvControl)
```

Arguments

N	Sample size.
id	Optional cluster id variable. If present, all observations in the same cluster will always be in the same split.
Y	outcome.
cvControl	Control parameters for the cross-validation step. See SuperLearner.CV.control for details.

Value

A list of length V where each element in the list is a vector with the row numbers of the corresponding validation sample.

delta_mu_constant *Constant Conditional Average Treatment Effect estimator for Y*

Description

Computes the difference in expected Y outcomes under treatment and control.

Usage

```
delta_mu_constant(X)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).

Value

A numeric vector that represents the contrast between primary outcomes for given X.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
delta_mu_constant(X)
```

delta_mu_linear *Linear-shaped Conditional Average Treatment Effect estimator for Y*

Description

Computes the difference in expected Y outcomes under treatment and control, using h_Y.

Usage

```
delta_mu_linear(X)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).

Value

A numeric vector that represents the contrast between primary outcomes for given X.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
delta_mu_linear(X)
```

delta_mu_mix *Mixed-shape Conditional Average Treatment Effect estimator for Y*

Description

Computes the difference in expected Y outcomes under treatment and control, using `h_Y`.

Usage

```
delta_mu_mix(X)
```

Arguments

`X`	A matrix of covariates of size n x d (input data in `[0,1]`).

Value

A numeric vector that represents the contrast between primary outcomes for given `X`.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
delta_mu_mix(X)
```

delta_mu_null *Null Conditional Average Treatment Effect estimator for Y*

Description

Computes the difference in expected Y outcomes under treatment and control.

Usage

```
delta_mu_null(X)
```

Arguments

`X`	A matrix of covariates of size n x d (input data in `[0,1]`).

Value

A numeric vector that represents the contrast between primary outcomes for given `X`.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
delta_mu_null(X)
```

delta_mu_realistic *Realistic Conditional Average Treatment Effect estimator for Y*

Description

Computes the difference in expected Y outcomes under treatment and control, using `h_Y`.

Usage

```
delta_mu_realistic(X)
```

Arguments

`X`	A matrix of covariates of size n x d (input data).

Value

A numeric vector that represents the contrast between primary outcomes for given `X`.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
delta_mu_realistic(X)
```

delta_mu_threshold *Thresholded-shaped Conditional Average Treatment Effect estimator for Y*

Description

Computes the difference in expected Y outcomes under treatment and control, using `h_Y`.

Usage

```
delta_mu_threshold(X)
```

Arguments

`X`	A matrix of covariates of size n x d (input data in `[0,1]`).

Value

A numeric vector that represents the contrast between primary outcomes for given `X`.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
delta_mu_threshold(X)
```

delta_nu_linear *Linear-shaped Conditional Average Treatment Effect estimator for Xi*

Description

Computes the difference in expected outcomes under treatment and control.

Usage

```
delta_nu_linear(X)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).

Value

A numeric vector that represents the contrast between adverse event outcomes for given X.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
delta_nu_linear(X)
```

delta_nu_mix *Mixed-shaped Conditional Average Treatment Effect estimator for Xi*

Description

Computes the difference in expected outcomes under treatment and control.

Usage

```
delta_nu_mix(X)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).

Value

A numeric vector that represents the contrast between adverse event outcomes for given X.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
delta_nu_mix(X)
```

delta_nu_realistic *Realistic Conditional Average Treatment Effect estimator for Xi*

Description

Computes the difference in expected outcomes under treatment and control.

Usage

```
delta_nu_realistic(X)
```

Arguments

X	A matrix of covariates of size n x d (input data).

Value

A numeric vector that represents the contrast between adverse event outcomes for given X.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
delta_nu_realistic(X)
```

delta_nu_satisfied *Computes the difference in expected outcomes under treatment and control.*

Description

Computes the difference in expected outcomes under treatment and control.

Usage

```
delta_nu_satisfied(X)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).

Value

A numeric vector that represents the contrast between adverse event outcomes for given X.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
delta_nu_satisfied(X)
```

delta_nu_threshold *Thresholded Conditional Average Treatment Effect estimator for Xi*

Description

Computes the difference in expected outcomes under treatment and control.

Usage

```
delta_nu_threshold(X)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).

Value

A numeric vector that represents the contrast between adverse event outcomes for given X.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
delta_nu_threshold(X)
```

estimate_mu *Estimate mu*

Description

This function trains conditional mean of primary outcome models for treated and control groups using SuperLearner, applying cross-validation to compute out-of-fold estimates.

Usage

```
estimate_mu(
  Y,
  A,
  X,
  folds,
  SL.library = c("SL.glm", "SL.mean"),
  V = 2L,
  threshold = 0.01
)
```

Arguments

Y	A numeric vector or matrix of length n representing primary outcomes (in [0,1]).
A	A binary vector or matrix of length n indicating treatment assignment (0 or 1).
X	A matrix or data frame of covariates of size n x d (input data in [0,1]).
folds	A list of cross-validation folds (e.g., a list of indices for each fold).
SL.library	Vector of libraries for training SuperLearner (c("SL.glm", "SL.mean") by default).
V	Number of folds inside the SuperLearner (2L by default).
threshold	A numeric scalar that sets the minimum allowed value for upper and lower bound estimations (1e-2 by default). Constrains estimation to [threshold, 1 - threshold].

Value

A fold-specific function predicting primary outcome (Y) given treatment (A) and covariates (X)

---

estimate_nu *Estimate nu*

Description

This function trains conditional mean of adverse event outcome models for treated and control groups using SuperLearner, applying cross-validation to compute out-of-fold estimates.

Usage

```
estimate_nu(
  Xi,
  A,
  X,
  folds,
  SL.library = c("SL.glm", "SL.mean"),
  V = 2L,
  threshold = 0.01
)
```

Arguments

Xi	A numeric vector or matrix of adverse events outcomes.
A	A binary vector or matrix of length n indicating treatment assignment (0 or 1).
X	A matrix or data frame of covariates of size n x d (input data in [0,1]).
folds	A list of cross-validation folds (e.g., a list of indices for each fold).
SL.library	Vector of libraries for training SuperLearner (c("SL.glm", "SL.mean") by default).
V	Number of folds inside the SuperLearner (2L by default).
threshold	A numeric scalar that sets the minimum allowed value for upper and lower bound estimations (1e-2 by default). Constrains estimation to [threshold, 1 - threshold].

Value

A fold-specific function predicting adverse event outcome (Xi) given treatment (A) and covariates (X)

`estimate_ps` *Estimate propensity score*

Description

This function trains the propensity score models using `SuperLearner`, applying cross-validation to compute out-of-fold estimates.

Usage

```
estimate_ps(
  A,
  X,
  folds,
  SL.library = c("SL.glm", "SL.mean"),
  V = 2L,
  threshold = 0.01
)
```

Arguments

`A`	A binary vector or matrix of length n indicating treatment assignment (0 or 1).
`X`	A matrix or data frame of covariates of size n x d (input data in `[0,1]`).
`folds`	A list of cross-validation folds (e.g., a list of indices for each fold).
`SL.library`	Vector of libraries for training SuperLearner (c("SL.glm", "SL.mean") by default).
`V`	Number of folds inside the SuperLearner (2L by default).
`threshold`	A numeric scalar that sets the minimum allowed value for upper and lower bound estimations (1e-2 by default). Constrains estimation to `[threshold, 1 - threshold]`.

Value

A fold-specific function predicting propensity score given treatment (A) and covariates (X)

`estimate_real_valued_mu` *Estimate real-valued mu*

Description

This function trains conditional mean of primary outcome models for treated and control groups using `SuperLearner`, applying cross-validation to compute out-of-fold estimates.

Usage

```
estimate_real_valued_mu(
  Y,
  A,
  X,
  folds,
  SL.library = c("SL.glm", "SL.mean"),
  V = 2L
)
```

Arguments

`Y`	A numeric vector or matrix of length n representing primary outcomes (in R).
`A`	A binary vector or matrix of length n indicating treatment assignment (0 or 1).
`X`	A matrix or data frame of covariates of size n x d (input data in R).
`folds`	A list of cross-validation folds (e.g., a list of indices for each fold).
`SL.library`	Vector of libraries for training SuperLearner (c("SL.glm", "SL.mean") by default).
`V`	Number of folds inside the SuperLearner (2L by default).

Value

A fold-specific function predicting primary outcome (Y) given treatment (A) and covariates (X)

`FW` *Frank-Wolfe algorithm*

Description

Implements the Frank-Wolfe optimization algorithm to iteratively refine a convex combination function `psi`. At each iteration, a new solution `theta` is computed via stochastic gradient descent (SGD) and added to the convex combination in the form $2 \cdot \text{expit}(X\theta) - 1$.

Usage

```
FW(
  X,
  delta_Mu,
  delta_Nu,
  lambda,
  alpha = 0.1,
  beta = 0.05,
  centered = FALSE,
  precision = 0.05,
  verbose = TRUE
)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).
delta_Mu	A function of X that determines the contrast between primary outcomes.
delta_Nu	A function of X that determines the contrast between adverse event outcomes.
lambda	A non-negative numeric scalar controlling the penalty for violating the constraint.
alpha	A numeric scalar representing the constraint tolerance (in [0,1/2], 0.1 by default).
beta	A non-negative numeric scalar controlling the sharpness of the probability function (0.05 by default).
centered	A logical (FALSE by default) indicating whether to center the policy.
precision	A numeric scalar defining the desired convergence precision (0.05 by default). The number of Frank-Wolfe iterations (K) is inversely proportional to this value, calculated as 1/precision.
verbose	A logical value indicating whether to print progress updates. Default is TRUE.

Value

A numeric matrix containing the optimized parameter theta, where each row represents the k-th theta solution at iteration k.

---

generate_data *Synthetic data generator and functions generator*

Description

Generates a dataset simulating treatment assignment, covariates, and potential outcomes.

Usage

```
generate_data(
  n,
  ncov = 10L,
  scenario_mu = c("Linear", "Threshold", "Mix", "Null", "Linear2"),
  scenario_nu = c("Linear", "Threshold", "Mix", "Satisfied"),
  is_RCT = FALSE,
  seed = NA
)
```

Arguments

n	Number of observations to generate.
ncov	Number of baseline covariates (at least 2L and 10L by default).
scenario_mu	String indicating the type of scenario for delta_Mu ("Linear", "Threshold", "Mix", "Null", "Constant").
scenario_nu	String indicating the type of scenario for delta_Nu ("Linear", "Threshold", "Mix", "Satisfied").
is_RCT	Logical value indicating whether the scenario is an RCT (FALSE by default).
seed	Integer or NA (NA by default).

Value

A list containing two data frames (`df_complete` with all potential outcomes and treatment assignments and `df_obs` with observed outcomes based on treatment) and the oracular functions delta_Mu and delta_Nu.

Examples

```
data <- generate_data(100)
head(data[[1]])  # complete data
head(data[[2]])  # observed data
```

---

generate_realistic_data

Realistic synthetic data generator and functions generator

Description

Generates a realistic dataset simulating treatment assignment, covariates, and potential outcomes.

Usage

```
generate_realistic_data(
  n,
  ncov = 5L,
  scenario_mu = "Realistic",
  scenario_nu = "Realistic",
  is_RCT = FALSE,
  seed = NA
)
```

Arguments

n	Number of observations to generate.
ncov	Number of baseline covariates (at least 2L and 10L by default).
scenario_mu	String indicating the type of scenario for delta_Mu ("Linear", "Threshold", "Mix", "Null", "Constant").
scenario_nu	String indicating the type of scenario for delta_Nu ("Linear", "Threshold", "Mix", "Satisfied").
is_RCT	Logical value indicating whether the scenario is an RCT (FALSE by default).
seed	Integer or NA (NA by default).

Value

A list containing two data frames (`df_complete` with all potential outcomes and treatment assignments and `df_obs` with observed outcomes based on treatment) and the oracular functions delta_Mu and delta_Nu.

Examples

```
data <- generate_realistic_data(100)
head(data[[1]])  # complete data
head(data[[2]])  # observed data
```

---

get_opt_beta_lambda *Select Optimal Beta and Lambda Combination*

Description

This function loads intermediate results corresponding to lambda-optimal solutions for each beta value. It identifies and returns the beta-lambda combination that minimizes the objective function.

Usage

```
get_opt_beta_lambda(combinations, root.path)
```

Arguments

combinations	A matrix or data frame where each row corresponds to a beta and optimal-lambda pair.
root.path	Path to the folder where all results are to be saved.

Value

A vector of intermediate results including the optimal: lambda, beta, risk, constraint, obj.

---

grad_Lagrangian_p *Gradient of the objective function*

Description

Computes the gradient of the objective function with respect to psi at X. The gradient is used in optimization algorithms like Stochastic Gradient Descent (SGD).

Usage

```
grad_Lagrangian_p(
  psi,
  X,
  delta_Mu,
  delta_Nu,
  lambda,
  alpha = 0.1,
  beta = 0.05,
  centered = FALSE
)

grad_Lagrangian_p_X(
  psi_X,
  delta_Mu_X,
  delta_Nu_X,
  lambda,
  alpha = 0.1,
  beta = 0.05,
  centered = FALSE
)
```

Arguments

psi	A function that takes an input X and returns a numeric vector with values in the range [-1,1].
X	A matrix of covariates of size n x d (input data in [0,1]).
delta_Mu	A function of X that determines the contrast between primary outcomes.

delta_Nu	A function of X that determines the contrast between adverse event outcomes.
lambda	A non-negative numeric scalar controlling the penalty for violating the constraint.
alpha	A numeric scalar representing the constraint tolerance (in [0,1/2], 0.1 by default).
beta	A non-negative numeric scalar controlling the sharpness of the probability function (0.05 by default).
centered	A logical value indicating whether to apply centering in sigma_beta (FALSE by default).
psi_X	The function psi evaluated at X (numeric vector).
delta_Mu_X	The function delta_Mu evaluated at X (numeric vector).
delta_Nu_X	The function delta_Nu evaluated at X (numeric vector).

Value

A numeric vector representing the gradient of the objective function with respect to psi(X).

HX *Compute the Inverse Propensity Score Weight (IPW)*

Description

This function computes the inverse propensity score weight based on treatment assignment and a propensity score model.

Usage

```
HX(A, X, prop_score)
```

Arguments

A	A binary vector or matrix of length n indicating treatment assignment (0 or 1).
X	A matrix or data frame of covariates of size n x d (input data in [0,1]).
prop_score	A function that estimates the propensity score given treatment (A) and covariates (X).

Value

A numeric value representing the inverse propensity score weight.

Examples

```
# Example usage:
prop_model <- function(A, X) { 0.5 }  # Constant propensity score for illustration
HX(1, data.frame(x1 = 1, x2 = 2), prop_model)
```

Lagrangian_p *Objective function taking the form of a Lagrangian*

Description

Computes the objective function, which balances the risk function R_p and the constraint function S_p using a parameter lambda.

Usage

```
Lagrangian_p(
  psi,
  X,
  delta_Mu,
  delta_Nu,
  lambda,
  alpha = 0.1,
  beta = 0.05,
  centered = FALSE
)
```

Arguments

psi	A function that takes an input X and returns a numeric vector with values in the range [-1, 1].
X	A matrix of covariates of size n x d (input data in [0,1]).
delta_Mu	A function of X that determines the contrast between primary outcomes.
delta_Nu	A function of X that determines the contrast between adverse event outcomes.
lambda	A non-negative numeric scalar controlling the penalty for violating the constraint.
alpha	A numeric scalar representing the constraint tolerance (in [0,1/2], 0.1 by default).
beta	A non-negative numeric scalar controlling the sharpness of the probability function (0.05 by default).
centered	A logical value indicating whether to apply centering in sigma_beta (FALSE by default).

Value

A numeric scalar representing the objective function value.

learn_threshold *Learn Optimal Decision Threshold*

Description

This function estimates the optimal decision threshold for treatment assignment using cross-fitted nuisance function estimates and targeted maximum likelihood estimation (TMLE). The procedure evaluates candidate thresholds based on empirical performance criteria, selecting the threshold that maximizes the policy value under constraint satisfaction.

Usage

```
learn_threshold(theta, X, A, Y, Xi, folds, alpha)
```

Arguments

theta	A numeric matrix (k x d). Each row is from FW inner minimization, used to recover an extremal point for convex function construction.
X	A matrix or data frame of covariates of size n x d (input data in [0,1]).
A	A binary vector or matrix of length n indicating treatment assignment (0 or 1).
Y	A numeric vector or matrix of length n representing primary outcomes (in [0,1]).
Xi	A numeric vector or matrix of adverse events outcomes.
folds	A list of cross-validation folds (e.g., a list of indices for each fold).
alpha	A numeric scalar representing the constraint tolerance (in [0,1/2], 0.1 by default).

Value

A numeric value corresponding to the optimal threshold chosen from the candidate sequence.

lwr_upper_bound_estimators *Lower and upper bound estimators for policy value and constraint*

Description

This function computes lower and upper confidence bounds for the policy value and constraint, respectively, based on targeted maximum likelihood estimation (TMLE) updates.

Usage

```
lwr_upper_bound_estimators(mu0, nu0, prop_score, pi, X, A, Y, Xi, alpha)
```

Arguments

mu0	A fold-specific function predicting primary outcome (Y) given treatment (A) and covariates (X).
nu0	A fold-specific function predicting adverse event outcome (Xi) given treatment (A) and covariates (X).
prop_score	A function that estimates the propensity score given treatment (A) and covariates (X).
pi	A binary treatment rule vector indicating treatment assignment under the learned decision rule.
X	A matrix or data frame of covariates of size n x d (input data in [0,1]).
A	A binary vector or matrix of length n indicating treatment assignment (0 or 1).
Y	A numeric vector or matrix of length n representing primary outcomes (in [0,1]).
Xi	A numeric vector or matrix of adverse events outcomes.
alpha	A numeric scalar representing the constraint tolerance (in [0,1/2], 0.1 by default).

Value

A list containing a numeric vector of length 2:

- [1]: Lower bound for the primary outcome effect.
- [2]: Upper bound for the adverse event effect.

---

main_algorithm *Main algorithm*

Description

Executes the full estimation pipeline for learning an optimal treatment policy under constraints. This function performs the complete procedure implemented in the PLUCR framework. It begins by validating and preprocessing the input data, followed by estimating nuisance parameters using the SuperLearner ensemble library. Subsequently, it estimates the optimal treatment policy via the Frank-Wolfe optimization algorithm. The procedure includes an inner grid search over candidate values of lambda and beta to identify the policy that maximizes the expected primary outcome (policy value) while satisfying a constraint on the expected rate of adverse events.

Usage

```
main_algorithm(
  X,
  A,
  Y,
  Xi,
  Lambdas = seq(1, 8, by = 1),
```

```
  alpha = 0.1,
  precision = 0.05,
  B = c(0, 0.05, 0.1, 0.25, 0.5),
  centered = FALSE,
  Jfold = 3L,
  V = 2L,
  SL.library = c("SL.mean", "SL.glm", "SL.ranger", "SL.grf"),
  tol = 0.025,
  max_iter = 5,
  root.path
)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).
A	A binary vector of size n indicating treatment assignment (0 or 1).
Y	A numeric vector or matrix of length n representing primary outcomes (in [0,1]).
Xi	A numeric vector or matrix of length n indicating adverse events (0 or 1).
Lambdas	A sequence of non-negative numeric scalars controlling the penalty for violating the constraint (seq(1,8,by=1) by default).
alpha	A numeric scalar representing the constraint tolerance (in [0,1/2], 0.1 by default).
precision	A numeric scalar defining the desired convergence precision (0.05 by default). The number of Frank-Wolfe iterations (K) is inversely proportional to this value, calculated as 1/precision.
B	A vector of non-negative scalars controlling the sharpness of the treatment probability function (c(0, 0.05, 0.1, 0.25, 0.5) by default).
centered	A logical value indicating whether to apply centering in sigma_beta (FALSE by default).
Jfold	Number of folds for the main algorithm, needs to be set to 3L.
V	Number of folds inside the SuperLearner (2L by default).
SL.library	Vector of libraries for training Super Learner (c("SL.mean","SL.glm","SL.ranger","SL.grf") by default).
tol	A numeric scalar used as an early stopping criterion based on the RMSE between consecutive solutions (0.025 by default).
max_iter	A numeric scalar specifying the maximum number of iterations (5 by default).
root.path	Path to the folder where all results are to be saved.

Value

A list of matrices (theta_0 and theta_final), or theta_0 alone. These matrices are used to construct the optimal treatment rule in two steps. First, build psi using the make_psi function and evaluate it at X (i.e., psi(X)). Then, obtain the optimal treatment rule by applying sigma_beta to the selected beta attribute.

make_psi *Generate psi function*

Description

Constructs a convex combination function `psi` based on the sequence of solutions obtained from the Frank-Wolfe (FW) algorithm. Each new solution `theta` contributes to `psi` in the form $2 \cdot \text{expit}(X\theta) - 1$.

Usage

```
make_psi(Theta)
```

Arguments

`Theta`	A numeric matrix (k x d). Each row `theta` is from FW inner minimization, used to recover an extremal point for convex function construction.

Value

A function `psi` that takes an input `X` and returns a numeric vector with values in the range `[-1,1]`, using a convex combination of past `theta` solutions.

model_Xi_linear *Linear treatment effect on Xi Component Function*

Description

Computes a linear interaction term between covariates and treatment.

Usage

```
model_Xi_linear(X)
```

Arguments

`X`	A matrix of covariates of size n x d (input data in `[0,1]`).

Value

A numeric vector with the transformed values based on covariates and treatment.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
model_Xi_linear(X)
```

model_Xi_mix *Mixed treatment effect on Xi component function*

Description

Computes a threshold-based interaction term between covariates and treatment.

Usage

```
model_Xi_mix(X)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).

Value

A numeric vector with the transformed values based on covariates and treatment.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
model_Xi_mix(X)
```

model_Xi_realistic *Realistic treatment effect on Xi Component Function*

Description

Computes a realistic interaction term between covariates and treatment.

Usage

```
model_Xi_realistic(X)
```

Arguments

X	A matrix of covariates of size n x d (input data).

Value

A numeric vector with the transformed values based on covariates and treatment.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
model_Xi_realistic(X)
```

model_Xi_satisfied *Low treatment effect on Xi*

Description

Computes a close to zero interaction term between covariates and treatment.

Usage

```
model_Xi_satisfied(X)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).

Value

A numeric vector with the transformed values based on covariates and treatment.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
model_Xi_satisfied(X)
```

model_Xi_threshold *Thresholded treatment effect on Xi component function*

Description

Computes a threshold-based interaction term between covariates and treatment.

Usage

```
model_Xi_threshold(X)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).

Value

A numeric vector with the transformed values based on covariates and treatment.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
model_Xi_threshold(X)
```

model_Y_constant *Constant treatment effect on Y component function*

Description

Computes a null treatment effect for everyone

Usage

```
model_Y_constant(X, A)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).
A	A binary vector or matrix of length n indicating treatment assignment (-1 or 1).

Value

A numeric vector with the transformed values based on covariates and treatment.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
A <- rep(1, 10)
model_Y_constant(X, A)
```

model_Y_linear *Linear treatment effect on Y component function*

Description

Computes a linear interaction term between covariates and treatment.

Usage

```
model_Y_linear(X, A)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).
A	A vector indicating treatment assignment (+1 or -1) for each observation.

Value

A numeric vector with the transformed values based on covariates and treatment.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
A <- rep(1, 10)
model_Y_linear(X, A)
```

---

model_Y_mix *Mixed treatment effect on Y component function*

Description

Computes a mix of a linear and threshold shaped interaction term between covariates and treatment.

Usage

```
model_Y_mix(X, A)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).
A	A binary vector or matrix of length n indicating treatment assignment (-1 or 1).

Value

A numeric vector with the transformed values based on covariates and treatment.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
A <- rep(1, 10)
model_Y_mix(X, A)
```

---

model_Y_null *No treatment effect on Y component function*

Description

Computes a null treatment effect for everyone

Usage

```
model_Y_null(X, A)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).
A	A binary vector or matrix of length n indicating treatment assignment (-1 or 1).

Value

A numeric vector with the transformed values based on covariates and treatment.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
A <- rep(1, 10)
model_Y_null(X, A)
```

`model_Y_realistic` *Realistic treatment effect on Y component function*

Description

Computes a realistic interaction term between covariates and treatment.

Usage

```
model_Y_realistic(X, A)
```

Arguments

`X`	A matrix of covariates of size n x d (input data).
`A`	A vector indicating treatment assignment (+1 or -1) for each observation.

Value

A numeric vector with the transformed values based on covariates and treatment.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
A <- rep(1, 10)
model_Y_realistic(X, A)
```

model_Y_threshold *Thresholded treatment effect on Y component function*

Description

Computes a thresholded-shaped interaction term between covariates and treatment.

Usage

```
model_Y_threshold(X, A)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).
A	A binary vector or matrix of length n indicating treatment assignment (-1 or 1).

Value

A numeric vector with the transformed values based on covariates and treatment.

Examples

```
X <- matrix(stats::runif(10*5), 10, 5)
A <- rep(1, 10)
model_Y_threshold(X, A)
```

naive_approach_algorithm *Naive approach main algorithm*

Description

Learning an optimal treatment policy under constraints. This function begins by validating and preprocessing the input data, and assumes that the nuisance components mu0, nu0, and prop_score (train and test), have already been estimated. Subsequently, it estimates the optimal treatment policy via the Frank-Wolfe optimization algorithm. The procedure includes an inner grid search over candidate values of lambda and beta to identify the policy that maximizes the expected primary outcome (policy value) while satisfying a constraint on the expected rate of adverse events.

Usage

```
naive_approach_algorithm(
  X,
  A,
  Y,
  Xi,
  folds,
  mu0_train,
  mu0_test,
  nu0_train,
  nu0_test,
  prop_score_train,
  prop_score_test,
  Lambdas = seq(1, 8, by = 1),
  alpha = 0.1,
  precision = 0.05,
  B = c(0.05, 0.1, 0.25, 0.5),
  centered = FALSE,
  root.path
)
```

Arguments

X	A matrix of covariates of size n x d (input data in [0,1]).
A	A binary vector of size n indicating treatment assignment (0 or 1).
Y	A numeric vector or matrix of length n representing primary outcomes (in [0,1]).
Xi	A numeric vector or matrix of length n indicating adverse events (0 or 1).
folds	A list of cross-validation folds (e.g., a list of indices for each fold).
mu0_train	A function predicting primary outcome (Y) given treatment (A) and covariates (X) for training.
mu0_test	A fold-specific function predicting primary outcome (Y) given treatment (A) and covariates (X) for testing.
nu0_train	A function predicting adverse event outcome (Xi) given treatment (A) and covariates (X) for training.
nu0_test	A function predicting adverse event outcome (Xi) given treatment (A) and covariates (X) for testing.
prop_score_train	A function that estimates the propensity score given treatment (A) and covariates (X) for training.
prop_score_test	A function that estimates the propensity score given treatment (A) and covariates (X) for testing.
Lambdas	A sequence of non-negative numeric scalars controlling the penalty for violating the constraint (seq(1,8,by=1) by default).

alpha	A numeric scalar representing the constraint tolerance (in [0,1/2], 0.1 by default).
precision	A numeric scalar defining the desired convergence precision (0.05 by default). The number of Frank-Wolfe iterations (K) is inversely proportional to this value, calculated as 1/precision.
B	A vector of non-negative scalars controlling the sharpness of the treatment probability function (c(0.05, 0.1, 0.25, 0.5) by default).
centered	A logical value indicating whether to apply centering in sigma_beta (FALSE by default).
root.path	Path to the folder where all results are to be saved.

Value

A list of matrices (theta_0 and theta_final), or theta_0 alone. These matrices are used to construct the optimal treatment rule in two steps. First, build psi using the make_psi function and evaluate it at X (i.e., psi(X)). Then, obtain the optimal treatment rule by applying sigma_beta to the selected beta attribute (sigma_beta(psi(X), beta)).

Optimization_Estimation

Iterative optimization procedure

Description

This function performs an iterative optimization routine to correct and minimize the objective function. It iteratively finds a solution and corrects the objective function for such optimal solution, until two consecutive solutions do not change much.

Usage

```
Optimization_Estimation(
  mu0,
  nu0,
  prop_score,
  X,
  A,
  Y,
  Xi,
  lambda,
  alpha = 0.1,
  precision = 0.05,
  beta = 0.05,
  centered = FALSE,
  tol = 2.5 * 0.01,
  max_iter = 5
)
```

Arguments

mu0	A fold-specific function predicting primary outcome (Y) given treatment (A) and covariates (X).
nu0	A fold-specific function predicting adverse event outcome (Xi) given treatment (A) and covariates (X).
prop_score	A function that estimates the propensity score given treatment (A) and covariates (X).
X	A matrix of covariates of size n x d (input data in [0,1]).
A	A binary vector or matrix of length n indicating treatment assignment (0 or 1).
Y	A numeric vector or matrix of length n representing primary outcomes (in [0,1]).
Xi	A numeric vector or matrix of length n indicating adverse events (0 or 1).
lambda	A non-negative numeric scalar controlling the penalty for violating the constraint.
alpha	A numeric scalar representing the constraint tolerance (0.1 by default).
precision	A numeric scalar defining the desired convergence precision (0.05 by default). The number of Frank-Wolfe iterations (K) is inversely proportional to this value, calculated as 1/precision.
beta	A non-negative numeric scalar controlling the sharpness of the probability function (0.05 by default).
centered	A logical value indicating whether to apply centering in sigma_beta (FALSE by default).
tol	A numeric scalar used as an early stopping criterion based on the RMSE between consecutive solutions (0.025 by default).
max_iter	A numeric scalar specifying the maximum number of iterations (5 by default).

Details

This function saves intermediate results to files in order to recover progress or inspect iteration-level behavior. If the optimization converges or the maximum number of iterations is reached, the final parameter vector theta_init is saved.

Value

A list containing:

iter	The number of completed iterations.
offset_mu	Initial logit-transformed outcome predictions.
offset_nu	Initial logit-transformed auxiliary predictions.
psi_collection	Matrix of covariate projections across iterations.
sigma_psi_collection	Matrix of transformed projections across iterations.
epsilon1	GLM coefficients from the outcome model.
epsilon2	GLM coefficients from the auxiliary model.
theta_collection	List of parameter vectors from each iteration of the functional weight estimation.

`oracular_approach_algorithm` *Oracular approach main algorithm*

Description

Learns an optimal treatment policy under constraints using true data structures. It begins by validating and preprocessing the input data. Subsequently, it approximates the optimal treatment policy via the Frank-Wolfe optimization algorithm. The procedure includes an inner grid search over candidate values of `lambda` and `beta` to identify the policy that maximizes the expected primary outcome (policy value) while satisfying a constraint on the expected rate of adverse events.

Usage

```
oracular_approach_algorithm(
  X,
  A,
  Y,
  Xi,
  folds,
  ncov,
  delta_Mu,
  delta_Nu,
  scenario_mu,
  scenario_nu,
  Lambdas = seq(1, 8, by = 1),
  alpha = 0.1,
  precision = 0.05,
  B = c(0.05, 0.1, 0.25, 0.5),
  centered = FALSE,
  root.path
)
```

Arguments

`X`	A matrix of covariates of size n x d (input data in `[0,1]`).
`A`	A binary vector of size n indicating treatment assignment (0 or 1).
`Y`	A numeric vector or matrix of length n representing primary outcomes (in `[0,1]`).
`Xi`	A numeric vector or matrix of length n indicating adverse events (0 or 1).
`folds`	A list of cross-validation folds (e.g., a list of indices for each fold).
`ncov`	An integer indicating the number of covariates in synthetic setting.
`delta_Mu`	A function that computes the treatment effect (mu difference) from covariates.
`delta_Nu`	A function that computes the selection effect (nu difference) from covariates.
`scenario_mu`	String indicating the type of scenario for delta_Mu ("Linear", "Threshold", "Mix").

scenario_nu	String indicating the type of scenario for delta_Nu ("Linear", "Threshold", "Mix").
Lambdas	A sequence of non-negative numeric scalars controlling the penalty for violating the constraint (seq(1,8,by=1) by default).
alpha	A numeric scalar representing the constraint tolerance (in [0,1/2], 0.1 by default).
precision	A numeric scalar defining the desired convergence precision (0.05 by default). The number of Frank-Wolfe iterations (K) is inversely proportional to this value, calculated as 1/precision.
B	A vector of non-negative scalars controlling the sharpness of the treatment probability function (c(0.05, 0.1, 0.25, 0.5) by default).
centered	A logical value indicating whether to apply centering in sigma_beta (FALSE by default).
root.path	Path to the folder where all results are to be saved.

Value

A list of matrices (theta_0 and theta_final), or theta_0 alone. These matrices are used to construct the optimal treatment rule in two steps. First, build psi using the make_psi function and evaluate it at X (i.e., psi(X)). Then, obtain the optimal treatment rule by applying sigma_beta to the selected beta attribute (sigma_beta(psi(X), beta)).

---

oracular_process_results

Oracular evaluation of a policy

Description

This function evaluates the optimal policy derived from theta. This enables the approximation of the objective functions: risk, constraint, and the main objective and policy value.

Usage

```
oracular_process_results(
  theta,
  ncov = 10L,
  scenario_mu = c("Linear", "Threshold", "Mix", "Null", "Linear2", "Realistic"),
  scenario_nu = c("Linear", "Threshold", "Mix", "Satisfied", "Realistic"),
  lambda,
  alpha = 0.1,
  beta = 0.05,
  centered = FALSE
)
```

Arguments

theta	A numeric matrix (k x d). Each row is from FW inner minimization, used to recover an extremal point for convex function construction.
ncov	Number of baseline covariates (at least 2L and 10L by default).
scenario_mu	String indicating the type of scenario for delta_Mu ("Linear", "Threshold", "Mix", "Linear2", "Realistic").
scenario_nu	String indicating the type of scenario for delta_Nu ("Linear", "Threshold", "Mix","Satisfied", "Realistic").
lambda	A non-negative numeric scalar controlling the penalty for violating the constraint.
alpha	A numeric scalar representing the constraint tolerance (in [0,1/2], 0.1 by default).
beta	A non-negative numeric scalar controlling the sharpness of the probability function.
centered	A logical value indicating whether to apply centering in sigma_beta (FALSE by default).

Value

A vector of optimized policy parameters (theta) trained across folds.

---

phi *Normalize a Matrix by Column Min-Max Scaling*

Description

This function performs feature-wise min-max normalization on a matrix or data frame. Each column is rescaled to the range [0,1] using its minimum and maximum values. The minimum and maximum values are stored as attributes for later inverse transformation.

Usage

```
phi(u)
```

Arguments

u	A numeric matrix or data frame. Each column will be normalized independently.

Value

A numeric matrix of the same dimensions as u, with all values rescaled to [0,1]. The returned matrix has two attributes:

- "min_X": A matrix containing the column-wise minima.
- "max_X": A matrix containing the column-wise maxima.

Examples

```
# Example matrix
X <- matrix(c(1, 2, 3, 4, 5, 6), ncol = 2)
X_norm <- phi(X)
X_norm
attributes(X_norm)$min_X
attributes(X_norm)$max_X
```

---

phi_inv *Inverse Min-Max Normalization*

Description

This function reverses the min-max normalization applied by phi. It reconstructs the original scale of the data given a normalized matrix and the stored attributes "min_X" and "max_X".

Usage

```
phi_inv(t)
```

Arguments

t	A normalized numeric matrix produced by phi, containing attributes "min_X" and "max_X".

Value

A numeric matrix with the same dimensions as t, rescaled back to the original values.

Examples

```
# Normalize and then invert
X <- matrix(c(1, 2, 3, 4, 5, 6), ncol = 2)
X_norm <- phi(X)
X_recovered <- phi_inv(X_norm)
all.equal(X, X_recovered)
```

plot_metric_comparison *Plot metric values for comparison*

Description

Creates a comparison plot of different metrics across different treatment rule estimation methods.

Usage

```
plot_metric_comparison(
  data,
  metrics = c("policy_value", "constraint"),
  techniques = NULL,
  root.path
)
```

Arguments

data	A tibble or data frame with two columns: **method** A character vector indicating the estimation method (e.g., "Theta_0", "Theta_naive", etc.). **metric1** A numeric vector containing the corresponding metric1 values for each method. **metric2** A numeric vector containing the corresponding metric2 values for each method.
metrics	A vector containing the metrics to be represented.
techniques	A vector containing the names of the techniques to be represented.
root.path	Path to the folder where images are to be saved.

Details

The function takes a data frame with method names and corresponding policy values, constraints, etc. (typically including `Theta_0`, `Theta_naive`, `Theta_final`, and `Theta_oracular`) and generates a visual comparison (e.g., bar plot or point plot).

Value

Saves a plot to "Images/"Comparison_techniques.pdf".

plot_realistic *Plot realistic data setting*

Description

Generates and saves a two-panel plot: one showing the sign of the treatment effect (`delta_Mu`) and the other visualizing the magnitude of selection effect (`delta_Nu`) across covariates X.1 and X.2.

Usage

```
plot_realistic(delta_Mu, delta_Nu, B = 100, root.path, name)
```

Arguments

`delta_Mu`	A function that computes the treatment effect (mu difference) from covariates.
`delta_Nu`	A function that computes the selection effect (nu difference) from covariates.
`B`	Integer, number of Monte Carlo repetitions (1e4 by default).
`root.path`	Path to the folder where images are to be saved.
`name`	A string to add to the end of filename.

Value

Saves a plot to "Images/synthetic_setting.pdf".

predict.SL.grf *predict.SL.grf*

Description

This function trains conditional mean of primary outcome models for treated and control groups using `SuperLearner`, applying cross-validation to compute out-of-fold estimates.

Usage

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

Arguments

`object`	SL.grf object
`newdata`	dataframe to generate predictions
`...`	not used

Value

the requested predictions

process_results *Evaluate a policy*

Description

This function evaluates the optimal policy derived from `theta` and gives the upper bound of the constraint estimator. It updates `mu0` and `nu0` following the estimation step from the alternating optimization procedure. This enables targeted estimation of the objective functions: risk, constraint, and the main objective, providing a consistent upper bound for the constraint estimator.

Usage

```
process_results(
  theta,
  X,
  A,
  Y,
  Xi,
  mu0,
  nu0,
  prop_score,
  lambda,
  alpha = 0.1,
  beta = 0.05,
  centered = FALSE
)
```

Arguments

`theta`	A numeric matrix (k x d). Each row is from FW inner minimization, used to recover an extremal point for convex function construction.
`X`	A matrix of covariates of size n x d (input data in `[0,1]`).
`A`	A binary vector or matrix of length n indicating treatment assignment (0 or 1).
`Y`	A numeric vector or matrix of length n representing primary outcomes (in `[0,1]`).
`Xi`	A numeric vector or matrix of length n indicating adverse events (0 or 1).
`mu0`	A fold-specific function predicting primary outcome (Y) given treatment (A) and covariates (X).
`nu0`	A fold-specific function predicting adverse event outcome (Xi) given treatment (A) and covariates (X).
`prop_score`	A function that estimates the propensity score given treatment (A) and covariates (X).
`lambda`	A non-negative numeric scalar controlling the penalty for violating the constraint.
`alpha`	A numeric scalar representing the constraint tolerance (in `[0,1/2]`, 0.1 by default).

beta	A non-negative numeric scalar controlling the sharpness of the probability function.
centered	A logical value indicating whether to apply centering in sigma_beta (FALSE by default).

Value

A vector of results for given theta.

R_p *Risk function for Conditional Average Treatment Effect (CATE)*

Description

Computes the risk function R_p for estimating the Conditional Average Treatment Effect (CATE). The function minimizes the squared error between psi(X) and delta_Y(X).

Usage

```
R_p(psi, X, delta_Mu)
```

Arguments

psi	A function that takes an input X and returns a numeric vector with values in the range [-1, 1].
X	A matrix of covariates of size n x d (input data in [0,1]).
delta_Mu	A function of X that determines the contrast between primary outcomes.

Value

A numeric scalar representing the risk function value.

SGD *Stochastic Gradient Descent (SGD) algorithm*

Description

Performs stochastic gradient descent to optimize the parameters.

Usage

```
SGD(
  theta_current,
  psi,
  X,
  delta_Mu,
  delta_Nu,
  lambda,
  alpha = 0.1,
  beta = 0.05,
  centered = FALSE,
  batch_prop = 1/5,
  max_iter = 1000,
  tol = 0.001,
  lr = 0.01,
  verbose = FALSE
)
```

Arguments

theta_current	A numeric matrix of size 1 x d (intialization for parameter to estimate).
psi	A function that takes an input X and returns a numeric vector with values in the range [-1,1].
X	A matrix of covariates of size n x d (input data in [0,1]).
delta_Mu	A function of X that determines the contrast between primary outcomes.
delta_Nu	A function of X that determines the contrast between adverse event outcomes.
lambda	A non-negative numeric scalar controlling the penalty for violating the constraint.
alpha	A numeric scalar representing the constraint tolerance (in [0,1/2], 0.1 by default).
beta	A non-negative numeric scalar controlling the sharpness of the probability function (0.05 by default).
centered	A logical value indicating whether to apply centering in sigma_beta (FALSE by default).
batch_prop	Proportion of data in a batch (by default 1/5).
max_iter	Maximum number of iterations in the SGD (by default 1e3).
tol	Tolerance parameter (by default 1e-3).
lr	Learning rate parameter (by default 1e-2).
verbose	A logical value indicating whether to print progress updates. Default is FALSE.

Value

A numeric matrix of size 1 x d (optimized parameters).

sigma_beta *Link function*

Description

Link function mapping [-1,1] to [0,1], parametrized by beta with an optional centering.

Usage

```
sigma_beta(t, beta = 0.05, centered = FALSE)
```

Arguments

t	A vector of numerics (in [-1,1]).
beta	A non-negative numeric scalar controlling the sharpness of the probability function (0.05 by default).
centered	A logical value indicating whether to apply centering in sigma_beta (FALSE by default).

Value

A numeric vector of treatment probabilities.

sigma_beta_prime *Derivative of link function*

Description

Computes the derivative of the link function sigma_beta, with respect to t.

Usage

```
sigma_beta_prime(t, beta = 0.05, centered = FALSE)
```

Arguments

t	A vector of numerics (in [-1,1]).
beta	A non-negative numeric scalar controlling the sharpness of the probability function (0.05 by default).
centered	A logical value indicating whether to apply centering in sigma_beta (FALSE by default).

Value

The derivative of sigma_beta evaluated at t.

SL.grf *SL.grf*

Description

This function trains conditional mean of a target variable for treated and control groups using SuperLearner, applying cross-validation to compute out-of-fold estimates.

Usage

```
SL.grf(Y, X, newX, family, obsWeights, ...)
```

Arguments

Y	outcome variable
X	training dataframe
newX	test dataframe
family	gaussian or binomial
obsWeights	observation-level weights
...	not used

Value

a list containing the predictions and the fitted object

SuperLearner.CV.control *SuperLearner.CV.control (from SuperLearner package)*

Description

SuperLearner.CV.control (from SuperLearner package)

Usage

```
SuperLearner.CV.control(
  V = 10L,
  stratifyCV = FALSE,
  shuffle = TRUE,
  validRows = NULL
)
```

Arguments

V	Number of splits for the V-fold cross-validation step. The default is 10. In most cases, between 10 and 20 splits works well.
stratifyCV	Should the data splits be stratified by a binary response? Attempts to maintain the same ratio in each training and validation sample.
shuffle	Should the rows of X be shuffled before creating the splits.
validRows	Use this to pass pre-specified rows for the sample splits. The length of the list should be V and each entry in the list should contain a vector with the row numbers of the corresponding validation sample.

Value

A list containing the control parameters.

---

synthetic_data_plot *Plot synthetic data setting*

Description

Generates and saves a two-panel plot: one showing the sign of the treatment effect (delta_Mu) and the other visualizing the magnitude of selection effect (delta_Nu) across covariates X.1 and X.2.

Usage

```
synthetic_data_plot(delta_Mu, delta_Nu, B = 100, root.path, name)
```

Arguments

delta_Mu	A function that computes the treatment effect (mu difference) from covariates.
delta_Nu	A function that computes the selection effect (nu difference) from covariates.
B	Integer, number of Monte Carlo repetitions (1e4 by default).
root.path	Path to the folder where images are to be saved.
name	A string to add to the end of filename.

Value

Saves a plot to "Images/synthetic_setting.pdf".

S_p *Constraint function*

Description

Computes the constraint function S_p, which ensures that the learned policy satisfies a constraint. This function enforces a limit on the expected impact of treatment via `delta_Z`.

Usage

```
S_p(psi, X, beta, alpha, centered, delta_Nu)
```

Arguments

`psi`	A function that takes an input `X` and returns a numeric vector with values in the range `[-1, 1]`.
`X`	A matrix of covariates of size n x d (input data in `[0,1]`).
`beta`	A non-negative numeric scalar controlling the sharpness of the probability function (0.05 by default).
`alpha`	A numeric scalar representing the constraint tolerance (in `[0,1/2]`, 0.1 by default).
`centered`	A logical value indicating whether to apply centering in `sigma_beta` (FALSE by default).
`delta_Nu`	A function of `X` that determines the contrast between adverse event outcomes.

Value

A numeric scalar representing the constraint function value.

update_mu *Update mu via augmented covariate adjustment*

Description

Computes updated Mu predictions by adding the bias correction for all previous solutions at X `psi(X)` scaled by coefficients `epsilon1_collection`, then transforms back via the logistic function.

Usage

```
update_mu(A, X, mu0, epsilon1, theta_collection, prop_score)
```

Arguments

A	A binary vector or matrix of length n indicating treatment assignment (0 or 1).
X	A matrix of covariates of size n x d (input data in [0,1]).
mu0	A fold-specific function predicting primary outcome (Y) given treatment (A) and covariates (X).
epsilon1	A numeric vector of GLM coefficients for each column in psi_collection.
theta_collection	A list of the optimal theta enabling the reconstruction of optimal psi functions.
prop_score	A function that estimates the propensity score given treatment (A) and covariates (X).

Value

A numeric vector of updated mu on the [0,1] scale.

---

update_mu_XA *Update mu via augmented covariate adjustment for fixed X*

Description

Computes updated Mu predictions by adding the bias correction for fixed psi_collection scaled by coefficients epsilon1_collection, then transforms back via the logistic function.

Usage

```
update_mu_XA(offset_mu_XA, epsilon1, psi_collection, H_XA)
```

Arguments

offset_mu_XA	A numeric vector of logit-scale baseline mu0 predictions.
epsilon1	A numeric vector of GLM coefficients for each column in psi_collection.
psi_collection	A matrix whose columns are optimal psi solutions.
H_XA	A numeric vector of inverse-propensity weights, typically from HX().

Value

A numeric vector of updated mu on the [0,1] scale.

update_nu *Update nu via augmented covariate adjustment*

Description

Computes updated Nu predictions by adding the bias correction for all previous solutions at X using `sigma_beta` scaled by coefficients `epsilon2_collection`, then transforms back via the logistic function.

Usage

```
update_nu(
  A,
  X,
  nu0,
  epsilon2,
  theta_collection,
  prop_score,
  beta = 0.05,
  centered = FALSE
)
```

Arguments

`A`	A binary vector or matrix of length n indicating treatment assignment (0 or 1).
`X`	A matrix of covariates of size n x d (input data in `[0,1]`).
`nu0`	A fold-specific function predicting adverse event outcome (Xi) given treatment (A) and covariates (X).
`epsilon2`	A numeric vector of GLM coefficients for each column in `sigma_psi_collection`.
`theta_collection`	A list of the optimal `theta` enabling the reconstruction of optimal `sigma_beta` applied to `psi` functions.
`prop_score`	A function that estimates the propensity score given treatment (A) and covariates (X).
`beta`	A non-negative numeric scalar controlling the sharpness of the probability function (0.05 by default).
`centered`	A logical value indicating whether to apply centering in `sigma_beta` (FALSE by default).

Value

A numeric vector of updated nu on the `[0,1]` scale.

update_nu_XA *Update nu via augmented covariate adjustment for fixed X*

Description

Computes updated Nu predictions by adding the bias correction for fixed `sigma_psi_collection` scaled by coefficients `epsilon2_collection`, then transforms back via the logistic function.

Usage

```
update_nu_XA(offset_nu_XA, epsilon2, sigma_psi_collection, H_XA)
```

Arguments

`offset_nu_XA`	A numeric vector of logit-scale baseline nu0 predictions.
`epsilon2`	A numeric vector of GLM coefficients for each column in `sigma_psi_collection`.
`sigma_psi_collection`	A matrix whose columns are optimal `psi` solutions composed by sigma.
`H_XA`	A numeric vector of inverse-propensity weights, typically from `HX()`.

Value

A numeric vector of updated nu on the `[0,1]` scale.

visual_treatment_plot *Visualize treatment assignment probability*

Description

Plots the smoothed treatment assignment probability over covariates Var_X_axis and Var_Y_axis

Usage

```
visual_treatment_plot(
  psi_X,
  lambda,
  beta,
  centered,
  Var_X_axis,
  Var_Y_axis,
  root.path,
  name
)
```

Arguments

psi_X	A numeric vector with values in the range [-1,1] representing the output of psi at fixed X.
lambda	Regularization scalar for the constraint (display purposes).
beta	A non-negative numeric scalar controlling the sharpness of the probability function (0.05 by default).
centered	A logical value indicating whether to apply centering in sigma_beta (FALSE by default).
Var_X_axis	Vector of covariates values for x-axis (length n).
Var_Y_axis	Vector of covariates values for y-axis (length n).
root.path	Path to the folder where images are to be saved.
name	A string to add to the end of filename.

Value

A message indicating that the image was saved.

V_p *Oracular approximation of value function*

Description

Computes the expected outcome under a policy determined by the previously optimized psi(X). The policy assigns treatment probabilistically based on sigma_beta(psi(X)), and the expected outcome is calculated using counterfactual outcomes.

Usage

```
V_p(
  psi,
  beta = 0.05,
  centered = FALSE,
  alpha = 0.1,
  B = 1e+06,
  ncov = 10L,
  scenario_mu = c("Linear", "Threshold", "Mix", "Linear2", "Null", "Realistic"),
  scenario_nu = c("Linear", "Threshold", "Mix", "Satisfied", "Realistic"),
  seed = NA
)
```

Arguments

psi	A function that takes an input X and returns a numeric vector with values in the range [-1,1].
beta	A non-negative numeric scalar controlling the sharpness of the probability function (0.05 by default).
centered	A logical value indicating whether to apply centering in sigma_beta (FALSE by default).
alpha	A numeric scalar representing the constraint tolerance (in [0,1/2], 0.1 by default).
B	Integer, number of Monte Carlo repetitions (1e4 by default).
ncov	Number of baseline covariates (at least 2L and 10L by default).
scenario_mu	String indicating the type of scenario for delta_Mu ("Linear", "Threshold", "Mix").
scenario_nu	String indicating the type of scenario for delta_Nu ("Linear", "Threshold", "Mix").
seed	Integer or NA (NA by default).

Value

A numeric scalar representing the expected primary outcome under the policy.

V_Pn *Estimation of policy value*

Description

Computes the expected outcome under a policy determined by the previously optimized psi(X). The policy assigns treatment probabilistically based on sigma_beta(psi(X)), and the expected outcome is calculated using counterfactual outcomes.

Usage

```
V_Pn(policy, y1, y0)
```

Arguments

policy	A numeric vector of treatment probabilities associated with X (length n).
y1	A numeric vector or matrix of length n representing primary outcomes under treatment (in [0, 1]).
y0	A numeric vector or matrix of length n representing primary outcomes under no treatment (in [0, 1]).

Value

A numeric scalar representing the expected primary outcome under the policy.

Index