Package 'InfluenceBorrowing'

May 7, 2026

Type Package

Title Adaptive Influence-Based Borrowing for Hybrid Control Trials

Version 0.1.0

Description Implements the adaptive influence-based borrowing framework proposed by Qinwei Yang, Jingyi Li, Peng Wu, and Shu Yang (2026+) in the paper ``Improving Treatment Effect Estimation in Trials through Adaptive Borrowing of External Controls" <doi:10.48550/arXiv.2604.13973> for augmenting Randomized Controlled Trials (RCTs) with External Control (EC) data. This package provides a comprehensive workflow to: (1) quantify the comparability of external control samples using influence scores approximated via the influence function of the M-estimator; (2) construct candidate borrowing subsets and select the optimal subset that minimizes the Mean Squared Error (MSE); and (3) calibrate systematic differences in external outcomes using R-learner methods implemented via Ordinary Least Squares or Kernel Ridge Regression.

License GPL-3

Encoding UTF-8

Imports KRLS, stats

RoxygenNote 7.3.3

NeedsCompilation no

Author Jile Chaoge [aut, cre],
Peng Wu [aut],
Shu Yang [aut]

Maintainer Jile Chaoge <chogjill@126.com>

Repository CRAN

Date/Publication 2026-04-23 20:20:13 UTC

Contents

`compute_influences` *Calculate Influence Scores for External Controls*

Description

This function quantifies the comparability of external control samples by assessing how much each individual external sample perturbs the outcome model fitted on the RCT control data. A smaller influence score indicates that the external sample is more compatible with the RCT controls.

Usage

```
compute_influences(model, testdata = NULL, type = "observed")
```

Arguments

`model`	A fitted `glm` object representing the outcome model on RCT controls.
`testdata`	A `data.frame` containing the external control samples. Must include the outcome variable Y and covariates X matching the model.
`type`	Character string specifying the type of Hessian matrix: `"observed"` (default) or `"expected"`.

Details

The influence score is approximated using the influence function of the M-estimator. It measures the standardized change in model parameters if a specific external sample were added to the training set, without the computational cost of refitting.

Value

Vector of influence scores corresponding to each row in `testdata`.

`estimate_rct` *Estimate ATE using RCT Data Only*

Description

This function estimates the ATE using only the provided RCT data. It calculates two estimators: the direct estimator and the AIPW estimator.

Usage

```
estimate_rct(
  X,
  A,
  Y,
  trim = 0.01,
  outcome_family = gaussian(),
  ps_hat = NULL,
  mu0_hat = NULL,
  mu1_hat = NULL
)
```

Arguments

`X`	Covariate matrix.
`A`	Treatment assignment vector (binary: 0 or 1).
`Y`	Outcome vector.
`trim`	Numeric value for trimming propensity scores (default 0.01).
`outcome_family`	GLM family for outcome models (default gaussian()).
`ps_hat`	Optional pre-calculated propensity scores.
`mu0_hat`	Optional pre-calculated outcome predictions E[Y|X,A=0].
`mu1_hat`	Optional pre-calculated outcome predictions E[Y|X,A=1].

Value

A list containing:

- `estimate`: A named vector containing point estimates for Direct and AIPW methods.
- `se`: A named vector containing standard errors for both estimators.
- `psi`: Vector of influence function values for the AIPW estimator.
- `mu0_hat`: Fitted values for the control outcome model.
- `mu1_hat`: Fitted values for the treated outcome model.
- `ps_hat`: Fitted propensity scores.

Examples

```
n <- 200
X <- runif(n, 0, 2)
A <- rbinom(n, size = 1, prob = 1/2)
Y1 <- 3 - 2*X + rnorm(n, sd = 0.2)
Y0 <- 2*X + rnorm(n, sd = 0.2)
Y <- (1 - A)*Y0 + A*Y1
result <- estimate_rct(X, A, Y)
print(result$estimate)
print(result$se)
```

`estimate_selected` *Estimate ATE for a Selected Data Subset (with GLM support)*

Description

Estimate ATE for a Selected Data Subset (with GLM support)

Usage

```
estimate_selected(
  X,
  A,
  Y,
  reference_value = NULL,
  trim = 0.01,
  outcome_family = gaussian(),
  ps_hat = NULL,
  mu0_hat = NULL,
  mu1_hat = NULL
)
```

Arguments

`X`	Covariate matrix.
`A`	Treatment assignment vector (binary: 0 or 1).
`Y`	Outcome vector.
`reference_value`	A value representing the "true" treatment effect or a reference estimate used to calculate bias and MSE. If NULL, MSE is returned as NULL.
`trim`	Value for trimming propensity scores (default 0.01).
`outcome_family`	GLM family for outcome models (default gaussian()).
`ps_hat`	Optional vector of estimated propensity scores P(A=1|X).
`mu0_hat`	Optional vector of estimated conditional means E[Y|X,A=0].
`mu1_hat`	Optional vector of estimated conditional means E[Y|X,A=1].

Value

A list containing:

- `estimate`: The AIPW point estimate.
- `se`: The standard error of the estimated treatment effect.
- `psi`: Vector of influence function values.
- `mse`: The estimated Mean Squared Error (Variance + Bias^2), if `reference_value` is provided.
- `mu0_hat`, `mu1_hat`, `ps_hat`: Fitted nuisance parameters.

Examples

```
# Generate RCT data
n <- 100
X_rct <- runif(n, 0, 2)
A_rct <- rbinom(n, size = 1, prob = 1/2)
Y1_rct <- 3 - 2*X_rct + rnorm(n, sd = 0.2)
Y0_rct <- 2*X_rct + rnorm(n, sd = 0.2)
Y_rct <- (1 - A_rct)*Y0_rct + A_rct*Y1_rct

# Generate EC data
n <- 500
X_ec <- runif(n, 0, 2)
A_ec <- rep(0, n)
Y_ec <- rep(NA, n)
Y_ec[1:200] <- 2*X_ec[1:200] + rnorm(200, sd = 0.2)
Y_ec[201:n] <- 3*X_ec[201:n] + rnorm(n-200, sd = 0.2)

# Selected EC data
X_selected <- X_ec[1:200]
A_selected <- A_ec[1:200]
Y_selected <- Y_ec[1:200]

result <- estimate_selected(X = c(X_rct, X_ec),
                            A = c(A_rct, A_ec),
                            Y = c(Y_rct, Y_ec))
print(result$estimate)
print(result$se)

result <- estimate_selected(X = c(X_rct, X_selected),
                            A = c(A_rct, A_selected),
                            Y = c(Y_rct, Y_selected))
print(result$estimate)
print(result$se)
```

`find_optimal_k` *Select Optimal Subset of External Controls based on MSE*

Description

This function iterates through a sequence of candidate subset sizes (k), selecting the top-k external controls with the smallest influence scores. For each k, it estimates the treatment effect and calculates the MSE relative to a provided reference value. It returns the results for all k and identifies the optimal k that minimizes MSE.

Usage

```
find_optimal_k(
  dat_rct,
  dat_ec,
  influences,
  reference_value,
  trim = 0.01,
  k_vector = NULL,
  outcome_family = gaussian()
)
```

Arguments

`dat_rct`	A `data.frame` containing the RCT data.
`dat_ec`	A `data.frame` containing the External Control data.
`influences`	Vector of influence scores for the external controls. The length must match the number of rows in `dat_ec`.
`reference_value`	A value representing the "true" treatment effect or a high-quality reference estimate used to calculate bias and MSE.
`trim`	Value for trimming propensity scores (default 0.01).
`k_vector`	Optional integer vector specifying the candidate numbers of external controls to borrow. If NULL, a default sequence is generated (from 0 to N_ec, step 50).
`outcome_family`	GLM family for outcome models (default gaussian()).

Details

Data Structure Requirements: The input data frames (`dat_rct` and `dat_ec`) must follow this column order:

- Covariates (X): The first `ncol-2` columns are covariates.
- Treatment (A): Must contain a column named `A` (binary 0/1).
- Outcome (Y): Must contain a column named `Y`.

The code automatically identifies covariates as the first `ncol-2` columns. Therefore, please ensure `A` and `Y` are placed at the very end of the data frame (e.g., columns order: `X1, X2, ..., Xp, A, Y`).

Value

A list containing:

- `mse_k`: A `data.frame` summarizing the estimate, bias, variance, and MSE for each candidate k.
- `mse_optimal`: The single row from `mse_k` corresponding to the minimum MSE.

---

gen_demo_data *Generate Simulation Data for RCT and External Controls*

Description

This function generates synthetic data for a RCT and an EC arm. It is designed to demonstrate the adaptive borrowing framework, creating a scenario where the external controls have a different outcome mechanism (bias) compared to the RCT controls, along with some outliers.

Usage

```
gen_demo_data(n_rct = 100, n_ec = 200, seed = 123)
```

Arguments

`n_rct`	Integer. Sample size of the randomized controlled trial (default 100).
`n_ec`	Integer. Sample size of the external controls (default 200).
`seed`	Integer. Random seed for reproducibility (default 123).

Details

Output Format: The returned data frames are formatted to be directly compatible with `find_optimal_k`:

- Column Order: Covariates (X) are in the first columns, followed by Treatment (A) and Outcome (Y).
- Bias Mechanism: External controls are generated with a shifted intercept and slope to simulate systematic bias.
- True ATE: The true Average Treatment Effect is fixed at -1.

Value

A list containing:

- `data_rct`: A `data.frame` with columns X (covariate), A (treatment 0/1), and Y (outcome).
- `data_ec`: A `data.frame` with columns X, A (always 0), and Y (outcome).
- `ATE_true`: The true Average Treatment Effect (numeric, fixed at -1).

Examples

```
sim_data <- gen_demo_data(n_rct = 100, n_ec = 200)
head(sim_data$data_rct)
head(sim_data$data_ec)
```

`predict.rlearner_krls` *Predictions for rlearner_krls objects*

Description

Predict estimated treatment effects (tau) for new data using a trained rlearner_krls model.

Usage

```
## S3 method for class 'rlearner_krls'
predict(object, newx = NULL, ...)
```

Arguments

`object`	An object of class `rlearner_krls`.
`newx`	Covariate matrix to make predictions on. If NULL, returns predictions on the training data.
`...`	Additional arguments (currently ignored).

Value

A vector of predicted treatment effects.

`predict.rlearner_lm` *Predictions for rlearner_lm*

Description

Get estimated tau(x) using the trained rlearner_lm model.

Usage

```
## S3 method for class 'rlearner_lm'
predict(object, newx = NULL, ...)
```

Arguments

`object`	An object of class `rlearner_lm`.
`newx`	Covariate matrix to make predictions on. If NULL, returns predictions on the training data.
`...`	Additional arguments (currently ignored).

Value

Vector of predicted treatment effects.

Examples

```
n = 200; p = 5
set.seed(123)
x = matrix(rnorm(n*p), n, p)
r = rbinom(n, 1, 0.5)
y = 0.5*x[,1] + 0.8*x[,2] + 1.2*r*x[,1] + rnorm(n, sd=0.5)
rl_fit = rlearner_lm(x, r, y)
new_data = matrix(rnorm(10*5), 10, 5)
predictions = predict(rl_fit, new_data)
```

---

rlearner_krls *R-learner implemented via kernel ridge regression with a Gaussian kernel*

Description

Implements the R-learner (Nie and Wager, 2017) using kernel ridge regression (via the KRLS package) for nuisance parameter estimation and the final treatment effect model.

Usage

```
rlearner_krls(x, r, y, whichkernel = "gaussian", pi_hat = NULL, m_hat = NULL)
```

Arguments

x	Covariate matrix.
r	Treatment assignment vector (binary: 0 or 1).
y	Outcome vector.
whichkernel	Character string specifying the kernel type (default "gaussian"). Passed to KRLS::krls.
pi_hat	Optional vector of estimated propensity scores E[R|X]. If NULL, estimated using KRLS.
m_hat	Optional vector of estimated conditional means E[Y|X]. If NULL, estimated using KRLS.

Value

An object of class rlearner_krls containing the fitted models and estimates.

`rlearner_lm` *R-learner implemented via Ordinary Least Squares (Linear Model)*

Description

R-learner, as proposed by Nie and Wager (2017), implemented via standard linear regression (lm). It uses linear models (or logistic regression for propensity scores) to estimate nuisance parameters and the final treatment effect model.

Usage

```
rlearner_lm(x, r, y, pi_hat = NULL, m_hat = NULL)
```

Arguments

`x`	Covariate matrix.
`r`	Treatment assignment vector (binary: 0 or 1).
`y`	Outcome vector.
`pi_hat`	Optional vector of estimated propensity scores E[R|X]. If NULL, estimated using logistic regression.
`m_hat`	Optional vector of estimated conditional means E[Y|X]. If NULL, estimated using linear regression.

Value

An object of class `rlearner_lm` containing the fitted models and estimates.

Examples

```
n = 200; p = 5
set.seed(123)
x = matrix(rnorm(n*p), n, p)
r = rbinom(n, 1, 0.5)
y = 0.5*x[,1] + 0.8*x[,2] + 1.2*r*x[,1] + rnorm(n, sd=0.5)
rl_fit = rlearner_lm(x, r, y)
rl_est = predict(rl_fit, x)
```

Index