Package ‘SMMAL’

August 28, 2025

Title Semi-Supervised Estimation of Average Treatment Effects

Version 0.0.5

Description Provides a pipeline for estimating the average treatment effect via semi-supervised learning. Outcome regression is fit with cross-fitting using various machine learning method or user customized function. Doubly robust ATE estimation leverages both labeled and unlabeled data under a semi-supervised missing-data framework. For more details see Hou et al. (2021) <doi:10.48550/arxiv.2110.12336>. A detailed vignette is included.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports glmnet,randomForest,splines2,xgboost,stats,utils

Suggests knitr,rmarkdown,testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Jue Hou [aut, cre],
Yuming Zhang [aut],
Shuheng Kong [aut]

Maintainer Jue Hou <hou00123@umn.edu>

Repository CRAN

Date/Publication 2025-08-28 07:30:07 UTC

Contents

ate.SSL *Estimate Average Treatment Effect (ATE) via Semi-Supervised Learning*

Description

Estimate Average Treatment Effect (ATE) via Semi-Supervised Learning

Usage

```
ate.SSL(
  Y,
  A,
  R,
  mu1,
  mu0,
  pi1,
  pi0,
  imp.A,
  imp.A1Y1,
  imp.A0Y1,
  min.pi = 0.05,
  max.pi = 0.95
)
```

Arguments

`Y` Numeric vector. Observed outcomes for labeled data (with missing values for unlabelled).

`A` Numeric vector. Treatment indicator (1 for treated, 0 for control).

`R` Logical or binary vector. Indicator for labeled data (1 if labeled, 0 if not).

`mu1` Numeric vector. Estimated outcome regression $E[Y \mid A = 1, X]$.

`mu0` Numeric vector. Estimated outcome regression $E[Y \mid A = 0, X]$.

`pi1` Numeric vector. Estimated propensity scores $P(A = 1 \mid X)$.

`pi0` Numeric vector. Estimated propensity scores $P(A = 0 \mid X)$.

`imp.A` Numeric vector. Estimated treatment probabilities using surrogate covariates `W`.

`imp.A1Y1` Numeric vector. Imputed $E[Y \mid A = 1, W]$ using surrogate variables.

`imp.A0Y1` Numeric vector. Imputed $E[Y \mid A = 0, W]$ using surrogate variables.

`min.pi` Numeric. Lower bound to truncate estimated propensity scores (default = 0.05).

`max.pi` Numeric. Upper bound to truncate estimated propensity scores (default = 0.95).

Details

This function estimates the ATE in a semi-supervised setting, where outcomes are only observed for a subset of the sample. Surrogate variables and imputed models are used to leverage information from unlabelled data.

Value

A list containing:

est Estimated ATE.

se Estimated standard error of ATE.

Examples

```
set.seed(123)
N <- 400
n <- 200  # Number of labeled observations
labeled_indices <- sample(1:N, n)

# Generate covariates and treatment
X <- rnorm(N)
A <- rbinom(N, 1, plogis(X))

# True potential outcomes
Y0_true <- X + rnorm(N)
Y1_true <- X + 1 + rnorm(N)

# Observed outcomes
Y_full <- ifelse(A == 1, Y1_true, Y0_true)

# Only labeled samples have observed Y
Y <- rep(NA, N)
Y[labeled_indices] <- Y_full[labeled_indices]
R <- rep(0, N); R[labeled_indices] <- 1

# Nuisance parameter estimates (can be replaced by actual model predictions)
mu1 <- X + 0.5
mu0 <- X - 0.5
pi1 <- plogis(X)
pi0 <- 1 - pi1
imp.A <- plogis(X)
imp.A1Y1 <- plogis(X) * (X + 0.5)
imp.A0Y1 <- (1 - plogis(X)) * (X - 0.5)

# Estimate ATE
result <- ate.SSL(
  Y = Y,
  A = A,
  R = R,
  mu1 = mu1,
  mu0 = mu0,
  pi1 = pi1,
  pi0 = pi0,
  imp.A = imp.A,
  imp.A1Y1 = imp.A1Y1,
  imp.A0Y1 = imp.A0Y1
)

print(result$est)
print(result$se)
```

cf *Cross-Fitting with Model Selection and Log Loss Evaluation*

Description

Trains and evaluates predictive models using cross-fitting across `nfold` folds, supporting multiple learner types. Outputs out-of-fold predictions and computes `log_loss` for each hyperparameter tuning round to select the best-performing model.

Usage

```
cf(
  Y,
  X,
  nfold,
  R,
  foldid,
  cf_model,
  sub_set = rep(TRUE, length(Y)),
  custom_model_fun = NULL
)
```

Arguments

`Y` Numeric or factor vector. The response variable, either binary (0/1) or continuous. Only labelled observations (where `R = 1`) are used.

`X` Matrix or data frame. Predictor variables used for model training.

`nfold` Integer. Number of cross-fitting folds.

`R` Binary vector. Indicator of labelled data: 1 = labelled, 0 = unlabelled.

`foldid` Integer vector. Fold assignments for cross-fitting (length equal to the full dataset).

`cf_model` Character string. Specifies the model type. Must be one of `"xgboost"`, `"bspline"`, or `"randomforest"`.

`sub_set` Logical vector. Indicates which labelled samples to include in training.

`custom_model_fun`
A logical or function. If `NULL` or `FALSE`, bypasses adaptive-LASSO feature selection. Otherwise, enables two-stage tuning inside `compute_parameter()`. Defaults to all `TRUE`.

Details

The function supports three learner types:

- **xgboost**: Gradient-boosted trees, tuning `gamma` across rounds.
- **bspline**: Logistic regression using B-spline basis expansions, tuning the number of knots.
- **randomforest**: Random forests, tuning `nodesize`.

Cross-fitting ensures that model evaluation is based on out-of-fold predictions, reducing overfitting. `log_loss` is used as the evaluation metric to identify the best hyperparameter setting.

Value

A list containing:

models (Currently a placeholder) List of trained models per fold and tuning round.

predictions List of out-of-fold predictions for each of the 5 tuning rounds.

log_losses Numeric vector of log loss values for each tuning round.

best_rounds_index Integer index (1–5) of the round achieving the lowest `log_loss`.

best_rounds_log_losses Minimum `log_loss` value achieved across rounds.

best_rounds_prediction Vector of out-of-fold predictions from the best tuning round.

Examples

```
set.seed(123)
N <- 200
X <- matrix(rnorm(N * 5), nrow = N, ncol = 5)

# Simulate treatment assignment
A <- rbinom(N, 1, plogis(X[, 1] - 0.5 * X[, 2]))

# Simulate outcome
Y_full <- rbinom(N, 1, plogis(0.5 * X[, 1] - 0.25 * X[, 3]))

# Introduce some missingness to simulate semi-supervised data
Y <- Y_full
Y[sample(1:N, size = N/4)] <- NA  # 25% missing

# Create R vector (labelled = 1, unlabelled = 0)
R <- ifelse(!is.na(Y), 1, 0)

# Cross-validation fold assignment
foldid <- sample(rep(1:5, length.out = N))

# Run cf with glm model
result <- cf(Y = Y, X = X, nfold = 5, R = R, foldid = foldid, cf_model = "glm")

# Examine output
print(result$log_losses)
print(result$best_rounds_index)
```

`compute_parameter` *Estimate Nuisance Parameters for Semi-Supervised ATE Estimation*

Description

Computes nuisance functions including conditional expectations and propensity scores using cross-fitting, separately for labelled and unlabelled data. These estimates are essential inputs for doubly robust or semi-supervised average treatment effect (ATE) estimators.

Usage

```
compute_parameter(nfold, Y, A, X, S, W, foldid, R, cf_model, custom_model_fun)
```

Arguments

`nfold`	Integer. Number of cross-fitting folds.
`Y`	Numeric vector. Outcome variable. Can contain `NA`s for unlabelled observations.
`A`	Numeric vector. Treatment assignment indicator (0 or 1). Can contain `NA`s.
`X`	Matrix or data frame. Covariates used for outcome and propensity score models.
`S`	Matrix or data frame. Additional covariates used only in imputation models.
`W`	Matrix or data frame. Combined set of covariates (typically `cbind(X, S)`).
`foldid`	Integer vector. Fold assignments for cross-fitting.
`R`	Binary vector. Label indicator: 1 = labelled (observed `A` and `Y`), 0 = unlabelled.
`cf_model`	Function. A user-supplied cross-fitting wrapper function (e.g., based on Super Learner or other learners).
`custom_model_fun`	A logical or function. If `NULL` or `FALSE`, bypasses adaptive-LASSO feature selection. Otherwise, enables two-stage tuning inside `compute_parameter()`.

Details

This function applies cross-fitting to estimate all required nuisance functions for semi-supervised or doubly robust ATE estimators. Separate models are fit for the labelled dataset and the full dataset (for imputation).

Value

A named list of estimated nuisance parameters (each a numeric vector):

pi1.bs Estimated propensity score $P(A = 1 \mid X)$.

pi0.bs Estimated propensity score $P(A = 0 \mid X)$ (computed as `1 - pi1.bs`).

mu1.bs Estimated outcome regression $E[Y \mid A = 1, X]$.

mu0.bs Estimated outcome regression $E[Y \mid A = 0, X]$.

cap_pi1.bs Estimated imputed propensity score $P(A = 1 \mid W)$.

cap_pi0.bs Estimated imputed propensity score $P(A = 0 \mid W)$ (computed as `1 - cap_pi1.bs`).

m1.bs Estimated imputed outcome regression $E[Y \mid A = 1, W]$.

m0.bs Estimated imputed outcome regression $E[Y \mid A = 0, W]$.

See Also

`cf`

`cross_validation` *Assign Cross-Validation Folds for Labelled and Unlabelled Data*

Description

Creates fold assignments for both labelled and unlabelled data using stratified random sampling, ensuring an approximately equal number of samples per fold within each group.

Usage

```
cross_validation(N, nfold, A, Y)
```

Arguments

`N`	Integer. Total number of observations in the dataset.
`nfold`	Integer. Number of folds to assign for cross-validation.
`A`	Numeric vector. Treatment assignment indicator (may contain `NA` for unlabelled samples).
`Y`	Numeric vector. Outcome variable (may contain `NA` for unlabelled samples).

Details

The function first separates observations into labelled and unlabelled groups based on the availability of both treatment (`A`) and outcome (`Y`). Within each group, fold assignments are randomly assigned to ensure approximately balanced sample sizes across folds. This setup supports semi-supervised learning workflows by maintaining structure between labelled and unlabelled data during cross-fitting.

Value

A list containing:

R Binary vector of length `N`, where 1 indicates labelled observations (non-missing `A` and `Y`), and 0 indicates unlabelled observations.

foldid Integer vector of length `N`. Fold assignments (from 1 to `nfold`) for use in cross-validation.

Examples

```
set.seed(123)
N <- 100
A <- sample(c(0, 1, NA), size = N, replace = TRUE, prob = c(0.45, 0.45, 0.10))
Y <- sample(c(0, 1, NA), size = N, replace = TRUE, prob = c(0.45, 0.45, 0.10))

# Assign 5 folds for cross-fitting
result <- cross_validation(N = N, nfold = 5, A = A, Y = Y)

table(result$R)  # Check number of labelled vs unlabelled
table(result$foldid)  # Check how folds are distributed
```

`param_fun` *Parameter grid function*

Description

Returns two list of hyperparameters for model tuning.

Usage

```
param_fun()
```

Value

A list, e.g., list(ridge = ..., lambda = ...)

`SMMAL` *Estimate Average Treatment Effect (ATE) via Semi-Supervised Learning Pipeline*

Description

Executes a full semi-supervised ATE estimation pipeline. This includes cross-validation fold assignment, feature selection via adaptive LASSO, model fitting using a specified learner (e.g., bspline, xgboost, or random forest), and doubly robust ATE estimation that leverages both labelled and unlabelled data.

Usage

```
SMMAL(Y, A, S, X, nfold = 5, cf_model = "bspline", custom_model_fun = NULL)
```

Arguments

`Y` Numeric vector. Outcome variable (may contain `NA` for unlabelled observations).

`A` Numeric vector. Treatment indicator (1 = treated, 0 = control). May contain `NA` for unlabelled observations.

`S` Matrix or data frame. Surrogate variables used only in imputation models.

`X` Matrix or data frame. Main covariates used for outcome and propensity score modeling.

`nfold` Integer. Number of cross-validation folds. Default is 5.

`cf_model` Character string. Modeling method to use in cross-fitting. One of `"bspline"`, `"xgboost"`, or `"randomforest"`. Default is "bspline".

`custom_model_fun`
A logical or function. If `NULL` or `FALSE`, bypasses adaptive-LASSO feature selection. Otherwise, enables two-stage tuning inside `compute_parameter()`.

Details

The pipeline first selects important covariates via adaptive LASSO. Then, it fits nuisance functions (outcome regressions and propensity scores) using cross-fitting with the specified learner. Finally, it applies a doubly robust estimator that integrates information from both labelled and unlabelled observations to estimate the ATE.

Value

A list containing:

est Estimated Average Treatment Effect (ATE).

se Estimated standard error of the ATE.

See Also

`cf`, `compute_parameter`, `cross_validation`, `ate.SSL`

`SMMAL_ada_lasso` *Adaptive LASSO with Cross-Validation*

Description

Performs adaptive LASSO for binary outcomes by first fitting a ridge regression to compute penalty factors, and then running cross-validated lasso fits over a grid of lambda values.

Usage

```
SMMAL_ada_lasso(
  X,
  Y,
  X_full,
  foldid,
  foldid_labelled,
  sub_set,
  labeled_indices,
  nfold,
  log_loss
)
```

Arguments

`X` A numeric matrix of predictors (n observations × p features).

`Y` A numeric or integer vector of binary outcomes (length n).

`X_full` The full matrix of predictors for all observations.

`foldid` A vector assigning each observation (labelled or unlabelled) to a fold.

`foldid_labelled` An integer vector (length n) of fold assignments for labeled observations. Values should run from 1 to `nfold`; other values (e.g., `NA`) indicate unlabeled or held-out rows.

`sub_set` A logical or integer vector indicating which rows of `X`/`Y` are used in supervised CV.

`labeled_indices` An integer or logical vector indicating which rows have non-missing outcomes.

`nfold` A single integer specifying the number of CV folds (e.g., 5 or 10).

`log_loss` A function of the form `function(true_labels, pred_probs)` that returns a single log-loss numeric.

Details

This function expects that a parameter-generating function `param_fun()` is available in the package, returning a list with elements `$ridge` (a vector of ridge penalty values) and `$lambda` (a vector of lasso penalty values). Internally, it:

1. Fits a ridge-penalized logistic regression on all data to obtain coefficients.
2. Computes penalty factors as `1 / (abs(coef) + 1e-4)`.
3. For each ridge value, runs n-fold CV over `lambda` values with `glmnet(..., alpha=1)`.
4. Records predictions on held-out folds, computes log-loss for each `lambda`, and selects the `lambda` with minimum log-loss.
5. Returns a list of CV-predicted probability vectors (one vector per ridge value).

Value

A list of length equal to the number of ridge penalty values provided by param_fun(). Each element is a numeric vector (length = n) containing cross-validated predicted probabilities for the best lambda under that ridge penalty.

Examples

```
# Assume param_fun() is defined elsewhere and returns:
#   list(ridge = c(0.01, 0.1, 1), lambda = exp(seq(log(0.001), log(1), length = 50)))

# Simulate small data:
set.seed(123)
n <- 100; p <- 10
X <- matrix(rnorm(n * p), nrow = n)
true_beta <- c(rep(1.5, 3), rep(0, p - 3))
lin <- X %*% true_beta
probs <- 1 / (1 + exp(-lin))
Y <- rbinom(n, 1, probs)

# Create fold assignments for labeled observations:
labeled <- sample(c(TRUE, FALSE), n, replace = TRUE, prob = c(0.8, 0.2))
foldid_labelled <- rep(NA_integer_, n)
foldid_labelled[labeled] <- sample(1:5, sum(labeled), replace = TRUE)
sub_set         <- labeled
labeled_indices <- which(labeled)

# For simplicity, assign foldid to all observations (labeled & unlabeled)
foldid <- sample(1:5, n, replace = TRUE)

# Define a simple log-loss function:
log_loss_fn <- function(true, pred) {
  eps <- 1e-15
  pred_clipped <- pmin(pmax(pred, eps), 1 - eps)
  -mean(true * log(pred_clipped) + (1 - true) * log(1 - pred_clipped))
}

# Call SMMAL_ada_lasso with all required args:
results <- SMMAL_ada_lasso(
  X = X,
  Y = Y,
  X_full = X,  # Here full data same as X for example
  foldid = foldid,
  foldid_labelled = foldid_labelled,
  sub_set = sub_set,
  labeled_indices = labeled_indices,
  nfold = 5,
  log_loss = log_loss_fn
)

# 'results' is a list (one element per ridge value), each a numeric vector of CV predictions.
```

Index