Package: haldensify (via r-universe)

September 17, 2024

Title Highly Adaptive Lasso Conditional Density Estimation

Version 0.2.6

Maintainer Nima Hejazi <nh@nimahejazi.org>

Description An algorithm for flexible conditional density estimation based on application of pooled hazard regression to an artificial repeated measures dataset constructed by discretizing the support of the outcome variable. To facilitate non/semi-parametric estimation of the conditional density, the highly adaptive lasso, a nonparametric regression function shown to reliably estimate a large class of functions at a fast convergence rate, is utilized. The pooled hazards formulation implemented was first described by Díaz and van der Laan (2011) <doi:10.2202/1557-4679.1356>. To complement the conditional density estimation utilities, nonparametric inverse probability weighted (IPW) estimators of the causal effects of additive modified treatment policies are implemented, using the conditional density estimation procedure to estimate the generalized propensity score. Per Hejazi, Benkeser, Díaz, and van der Laan <>10.48550/arXiv.2205.05777>, these nonparametric IPW estimators can be coupled with sieve estimation (undersmoothing) of the generalized propensity score estimators to attain the non/semi-parametric efficiency bound.

Depends R (>= 3.2.0)

Imports stats, utils, dplyr, tibble, ggplot2, data.table, matrixStats, future.apply, assertthat, hal9001 (>= 0.4.6), origami (>= 1.0.7), stringr, rlang, scales, Rdpack

Suggests testthat, knitr, rmarkdown, covr, future

License MIT + file LICENSE

URL https://github.com/nhejazi/haldensify

BugReports https://github.com/nhejazi/haldensify/issues

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.2.3

RdMacros Rdpack

Repository https://nhejazi.r-universe.dev

RemoteUrl https://github.com/nhejazi/haldensify

RemoteRef HEAD

RemoteSha 7a553e43ca64d97d29bcf0f5954470e8beeb8c20

Contents

`confint.ipw_haldensify` *Confidence Intervals for IPW Estimates of the Causal Effects of Stochatic Shift Interventions*

Description

Confidence Intervals for IPW Estimates of the Causal Effects of Stochatic Shift Interventions

Usage

```
## S3 method for class 'ipw_haldensify'
confint(object, parm = seq_len(object$psi), level = 0.95, ...)
```

Arguments

`object`	An object of class `ipw_haldensify`, produced by invoking the function `ipw_shift`, for which a confidence interval is to be computed.
`parm`	A `numeric` vector indicating indices of `object$est` for which to return confidence intervals.
`level`	A `numeric` indicating the nominal level of the confidence interval to be computed.
`...`	Other arguments. Not currently used.

Details

Compute confidence intervals for estimates produced by ipw_shift.

Value

A named numeric vector containing the parameter estimate from a ipw_haldensify object, alongside lower/upper Wald-style confidence intervals at a specified coverage level.

Examples

```
# simulate data
n_obs <- 50
W1 <- rbinom(n_obs, 1, 0.6)
W2 <- rbinom(n_obs, 1, 0.2)
W3 <- rpois(n_obs, 3)
A <- rpois(n_obs, 3 * W1 - W2 + 2 * W1 * W2 + 4)
Y <- rbinom(n_obs, 1, plogis(A + W1 + W2 - W3 - W1 * W3))

# fit the IPW estimator
est_ipw <- ipw_shift(
  W = cbind(W1, W2, W3), A = A, Y = Y,
  delta = 0.5, cv_folds = 2L,
  n_bins = 5L, bin_type = "equal_range",
  lambda_seq = exp(seq(-1, -10, length = 100L)),
  # arguments passed to hal9001::fit_hal()
  max_degree = 3,
  smoothness_orders = 0,
  num_knots = NULL,
  reduce_basis = 1 / sqrt(n_obs)
)
confint(est_ipw)
```

---

cv_haldensify *HAL Conditional Density Estimation in a Cross-validation Fold*

Description

HAL Conditional Density Estimation in a Cross-validation Fold

Usage

```
cv_haldensify(
  fold,
  long_data,
  wts = rep(1, nrow(long_data)),
  lambda_seq = exp(seq(-1, -13, length = 1000L)),
  smoothness_orders = 0L,
  ...
)
```

Arguments

fold
: Object specifying cross-validation folds as generated by a call to `make_folds`.

long_data
: A `data.table` or `data.frame` object containing the data in long format, as given in Díaz I, van der Laan MJ (2011). "Super learner based conditional density estimation with application to marginal structural models." *International Journal of Biostatistics*, **7**(1), 1–20. doi:10.2202/15574679.1356., as produced by `format_long_hazards`.

wts
: A `numeric` vector of observation-level weights, matching in its length the number of records present in the long format data. Default is to weight all observations equally.

lambda_seq
: A `numeric` sequence of values of the regularization parameter of Lasso regression; passed to `fit_hal`.

smoothness_orders
: A `integer` indicating the smoothness of the HAL basis functions; passed to `fit_hal`. The default is set to zero, for indicator basis functions.

...
: Additional (optional) arguments of `fit_hal` that may be used to control fitting of the HAL regression model. Possible choices include `use_min`, `reduce_basis`, `return_lasso`, and `return_x_basis`, but this list is not exhaustive. Consult the documentation of `fit_hal` for complete details.

Details

Estimates the conditional density of A|W for a subset of the full set of observations based on the inputted structure of the cross-validation folds. This is a helper function intended to be used to select the optimal value of the penalization parameter for the highly adaptive lasso estimates of the conditional hazard (via `cross_validate`). The

Value

A `list`, containing density predictions, observations IDs, observation-level weights, and cross-validation indices for conditional density estimation on a single fold of the overall data.

`fit_haldensify` *Fit Conditional Density Estimation for a Sequence of HAL Models*

Description

Fit Conditional Density Estimation for a Sequence of HAL Models

Usage

```
fit_haldensify(
  A,
  W,
  wts = rep(1, length(A)),
```

```
  grid_type = "equal_range",
  n_bins = round(c(0.5, 1, 1.5, 2) * sqrt(length(A))),
  cv_folds = 5L,
  lambda_seq = exp(seq(-1, -13, length = 1000L)),
  smoothness_orders = 0L,
  ...
)
```

Arguments

- `A` — The `numeric` vector of observed values.
- `W` — A `data.frame`, `matrix`, or similar giving the values of baseline covariates (potential confounders) for the observed units. These make up the conditioning set for the conditional density estimate.
- `wts` — A `numeric` vector of observation-level weights. The default is to weight all observations equally.
- `grid_type` — A `character` indicating the strategy to be used in creating bins along the observed support of `A`. For bins of equal range, use `"equal_range"`; consult the documentation of `cut_interval` for more information. To ensure each bin has the same number of observations, use `"equal_mass"`; consult the documentation of `cut_number` for details.
- `n_bins` — This `numeric` value indicates the number(s) of bins into which the support of `A` is to be divided. As with `grid_type`, multiple values may be specified, in which case cross-validation will be used to choose the optimal number of bins. The default sets the candidate choices of the number of bins based on heuristics tested in simulation.
- `cv_folds` — A `numeric` indicating the number of cross-validation folds to be used in fitting the sequence of HAL conditional density models.
- `lambda_seq` — A `numeric` sequence of values of the regularization parameter of Lasso regression; passed to `fit_hal`.
- `smoothness_orders` — A `integer` indicating the smoothness of the HAL basis functions; passed to `fit_hal`. The default is set to zero, for indicator basis functions.
- `...` — Additional (optional) arguments of `fit_hal` that may be used to control fitting of the HAL regression model. Possible choices include `use_min`, `reduce_basis`, `return_lasso`, and `return_x_basis`, but this list is not exhaustive. Consult the documentation of `fit_hal` for complete details.

Details

Estimation of the conditional density of A|W via a cross-validated highly adaptive lasso, used to estimate the conditional hazard of failure in a given bin over the support of A.

Value

A `list`, containing density predictions for the sequence of fitted HAL models; the index and value of the L1 regularization parameter minimizing the density loss; and the sequence of empirical risks for the sequence of fitted HAL models.

Examples

```
# simulate data: W ~ U[-4, 4] and A|W ~ N(mu = W, sd = 0.5)
n_train <- 50
w <- runif(n_train, -4, 4)
a <- rnorm(n_train, w, 0.5)
# fit cross-validated HAL-based density estimator of A|W
haldensify_cvfit <- fit_haldensify(
  A = a, W = w, n_bins = 10L, lambda_seq = exp(seq(-1, -10, length = 100)),
  # the following arguments are passed to hal9001::fit_hal()
  max_degree = 3, reduce_basis = 1 / sqrt(length(a))
)
```

---

format_long_hazards *Generate Augmented Repeated Measures Data for Pooled Hazards Regression*

Description

Generate Augmented Repeated Measures Data for Pooled Hazards Regression

Usage

```
format_long_hazards(
  A,
  W,
  wts = rep(1, length(A)),
  grid_type = c("equal_range", "equal_mass"),
  n_bins = NULL,
  breaks = NULL
)
```

Arguments

A	The `numeric` vector or similar of the observed values of an intervention for a group of observational units of interest.
W	A `data.frame`, `matrix`, or similar giving the values of baseline covariates (potential confounders) for the observed units whose observed intervention values are provided in the previous argument.
wts	A `numeric` vector of observation-level weights. The default is to weight all observations equally.
grid_type	A `character` indicating the strategy (or strategies) to be used in creating bins along the observed support of the intervention `A`. For bins of equal range, use "equal_range"; consult documentation of `cut_interval` for more information. To ensure each bin has the same number of points, use "equal_mass"; consult documentation of `cut_number` for details.
n_bins	Only used if `grid_type` is set to `"equal_range"` or `"equal_mass"`. This `numeric` value indicates the number(s) of bins into which the support of `A` is to be divided.

breaks — A numeric vector of break points to be used in dividing up the support of A. This is passed through the ... argument to cut.default by cut_interval or cut_number.

Details

Generates an augmented (long format, or repeated measures) dataset that includes multiple records for each observation, a single record for each discretized bin up to and including the bin in which a given observed value of A falls. Such bins are derived from selecting break points over the support of A. This repeated measures dataset is suitable for estimating the hazard of failing in a particular bin over A using a highly adaptive lasso (or other) classification model.

Value

A list containing the break points used in dividing the support of A into discrete bins, the length of each bin, and the reformatted data. The reformatted data is a data.table of repeated measures data, with an indicator for which bin an observation fails in, the bin ID, observation ID, values of W for each given observation, and observation-level weights.

---

haldensify — *Cross-validated HAL Conditional Density Estimation*

Description

Cross-validated HAL Conditional Density Estimation

Usage

```
haldensify(
  A,
  W,
  wts = rep(1, length(A)),
  grid_type = "equal_range",
  n_bins = round(c(0.5, 1, 1.5, 2) * sqrt(length(A))),
  cv_folds = 5L,
  lambda_seq = exp(seq(-1, -13, length = 1000L)),
  smoothness_orders = 0L,
  hal_basis_list = NULL,
  ...
)
```

Arguments

A — The numeric vector observed values.

W — A data.frame, matrix, or similar giving the values of baseline covariates (potential confounders) for the observed units. These make up the conditioning set for the density estimate. For estimation of a marginal density, specify a constant numeric vector or NULL.

wts
: A numeric vector of observation-level weights. The default is to weight all observations equally.

grid_type
: A character indicating the strategy to be used in creating bins along the observed support of A. For bins of equal range, use "equal_range"; consult the documentation of cut_interval for more information. To ensure each bin has the same number of observations, use "equal_mass"; consult the documentation of cut_number for details. The default is "equal_range" since this has been found to provide better performance in simulation experiments; however, both types may be specified (i.e., c("equal_range", "equal_mass")) together, in which case cross-validation will be used to select the optimal binning strategy.

n_bins
: This numeric value indicates the number(s) of bins into which the support of A is to be divided. As with grid_type, multiple values may be specified, in which case cross-validation will be used to choose the optimal number of bins. The default sets the candidate choices of the number of bins based on heuristics tested in simulation.

cv_folds
: A numeric indicating the number of cross-validation folds to be used in fitting the sequence of HAL conditional density models.

lambda_seq
: A numeric sequence of values of the regularization parameter of Lasso regression; passed to fit_hal via its argument lambda.

smoothness_orders
: A integer indicating the smoothness of the HAL basis functions; passed to fit_hal. The default is set to zero, for indicator basis functions.

hal_basis_list
: A list consisting of a preconstructed set of HAL basis functions, as produced by fit_hal. The default of NULL results in creating such a set of basis functions. When specified, this is passed directly to the HAL model fitted upon the augmented (repeated measures) data structure, resulting in a much lowered computational cost. This is useful, for example, in fitting HAL conditional density estimates with external cross-validation or bootstrap samples.

...
: Additional (optional) arguments of fit_hal that may be used to control fitting of the HAL regression model. Possible choices include use_min, reduce_basis, return_lasso, and return_x_basis, but this list is not exhaustive. Consult the documentation of fit_hal for complete details.

Details

Estimation of the conditional density A|W through using the highly adaptive lasso to estimate the conditional hazard of failure in a given bin over the support of A. Cross-validation is used to select the optimal value of the penalization parameters, based on minimization of the weighted log-likelihood loss for a density.

Value

Object of class haldensify, containing a fitted hal9001 object; a vector of break points used in binning A over its support W; sizes of the bins used in each fit; the tuning parameters selected by cross-validation; the full sequence (in lambda) of HAL models for the CV-selected number of bins and binning strategy; and the range of A.

Note

Parallel evaluation of the cross-validation procedure to select tuning parameters for density estimation may be invoked via the framework exposed in the **future** ecosystem. Specifically, set `plan` for `future_mapply` to be used internally.

Examples

```
# simulate data: W ~ U[-4, 4] and A|W ~ N(mu = W, sd = 0.5)
set.seed(429153)
n_train <- 50
w <- runif(n_train, -4, 4)
a <- rnorm(n_train, w, 0.5)
# learn relationship A|W using HAL-based density estimation procedure
haldensify_fit <- haldensify(
  A = a, W = w, n_bins = 10L, lambda_seq = exp(seq(-1, -10, length = 100)),
  # the following arguments are passed to hal9001::fit_hal()
  max_degree = 3, reduce_basis = 1 / sqrt(length(a))
)
```

`ipw_shift` *IPW Estimator of the Causal Effects of Additive Modified Treatment Policies*

Description

IPW Estimator of the Causal Effects of Additive Modified Treatment Policies

Usage

```
ipw_shift(
  W,
  A,
  Y,
  delta = 0,
  n_bins = make_bins(A, "hist"),
  cv_folds = 10L,
  lambda_seq,
  ...,
  bin_type = c("equal_range", "equal_mass"),
  selector_type = c("dcar", "plateau", "gcv", "all")
)
```

Arguments

`W`	A `matrix`, `data.frame`, or similar containing a set of baseline covariates.
`A`	A `numeric` vector corresponding to a exposure variable. The parameter of interest is defined as a location shift of this quantity.

Y	A numeric vector of the observed outcomes.
delta	A numeric value indicating the shift in the exposure to be used in defining the target parameter. This is defined with respect to the scale of the exposure (A).
n_bins	A numeric, scalar or vector, indicating the number of bins into which the support of A is to be partitioned for constructing conditional density estimates.
cv_folds	A numeric giving the number of folds to be used for cross-validation. Note that this form of sample splitting is used for the selection of tuning parameters by empirical risk minimization, not for the estimation of nuisance parameters (i.e., to relax regularity conditions).
lambda_seq	A numeric sequence of the regularization parameter (L1 norm of HAL coefficients) to be used in fitting HAL models.
...	Additional arguments for model fitting to be passed directly to haldensify.
bin_type	A character indicating the strategy to be used in creating bins along the observed support of A. For bins of equal range, use "equal_range"; to ensure each bin has the same number of observations, use instead "equal_mass". For more information, see documentation of grid_type in haldensify.
selector_type	A character indicating the selection strategy for identifying an efficent IPW estimator. The choices include "gcv" for global cross-validation, "dcar" for solving the EIF equation, and "plateau" for agnostic approaches (1) balancing changes in the IPW estimate and its standard error (adapting Lepski's method) and (2) a plateau detector for inflection points in the IPW estimator's trajectory. The option "all" runs all three selection strategies while sharing redundant computation between each.

Examples

```
# simulate data
n_obs <- 50
W1 <- rbinom(n_obs, 1, 0.6)
W2 <- rbinom(n_obs, 1, 0.2)
W3 <- rpois(n_obs, 3)
A <- rpois(n_obs, 3 * W1 - W2 + 2 * W1 * W2 + 4)
Y <- rbinom(n_obs, 1, plogis(A + W1 + W2 - W3 - W1 * W3))

# fit the IPW estimator
est_ipw <- ipw_shift(
  W = cbind(W1, W2, W3), A = A, Y = Y,
  delta = 0.5, cv_folds = 2L,
  n_bins = 5L, bin_type = "equal_range",
  lambda_seq = exp(seq(-1, -10, length = 100L)),
  # arguments passed to hal9001::fit_hal()
  max_degree = 3,
  smoothness_orders = 0,
  num_knots = NULL,
  reduce_basis = 1 / sqrt(n_obs)
)
```

map_hazard_to_density *Map Predicted Hazard to Predicted Density for a Single Observation*

Description

Map Predicted Hazard to Predicted Density for a Single Observation

Usage

```
map_hazard_to_density(hazard_pred_single_obs)
```

Arguments

`hazard_pred_single_obs`
: A `numeric` vector of predicted hazard of failure in a given bin (under a given partitioning of the support) for a single observational unit based on a long format data structure (from `format_long_hazards`). This is the probability that a given value falls in a corresponding bin, given that it has not yet failed (fallen in a preceding bin), as per Díaz I, van der Laan MJ (2011). "Super learner based conditional density estimation with application to marginal structural models." *International Journal of Biostatistics*, **7**(1), 1–20. doi:10.2202/15574679.1356..

Details

For a single observation, map a predicted hazard of failure (as an occurrence in a particular bin, under a given partitioning of the support) to a density.

Value

A `matrix` composed of a single row and a number of columns specified by the grid of penalization parameters used in fitting of the highly adaptive lasso. This is the predicted conditional density for a given observation, re-mapped from the hazard scale.

plot.haldensify *Plot Method for HAL Conditional Density Estimates*

Description

Plot Method for HAL Conditional Density Estimates

Usage

```
## S3 method for class 'haldensify'
plot(x, ..., type = c("risk", "density"))
```

Arguments

x	Object of class haldensify, containing conditional density estimates, as produced by haldensify.
...	Additional arguments to be passed plot, currently ignored.
type	A character indicating the type of plot to be produced. Options include visualizing the empirical risks of the conditional density estimators across a grid of values of the regularization parameter and a plot of the estimated conditional density (based on the estimator selected by cross-validation). The latter has yet to be implemented.

Value

Object of class ggplot containing a plot of the desired type.

Examples

```
# simulate data: W ~ U[-4, 4] and A|W ~ N(mu = W, sd = 0.5)
n_train <- 50
w <- runif(n_train, -4, 4)
a <- rnorm(n_train, w, 0.5)
# learn relationship A|W using HAL-based density estimation procedure
haldensify_fit <- haldensify(
  A = a, W = w, n_bins = 3,
  lambda_seq = exp(seq(-1, -10, length = 50)),
  # the following arguments are passed to hal9001::fit_hal()
  max_degree = 3, reduce_basis = 0.1
)
plot(haldensify_fit)
```

---

predict.haldensify *Prediction Method for HAL Conditional Density Estimation*

Description

Prediction Method for HAL Conditional Density Estimation

Usage

```
## S3 method for class 'haldensify'
predict(
  object,
  ...,
  new_A,
  new_W,
  trim = TRUE,
  trim_min = NULL,
  lambda_select = c("cv", "undersmooth", "all")
)
```

Arguments

- `object` — An object of class `haldensify`, containing the results of fitting the highly adaptive lasso for conditional density estimation, as produced by a call to `haldensify`.
- `...` — Additional arguments passed to `predict` as necessary.
- `new_A` — The `numeric` vector or similar of the observed values for which a conditional density estimate is to be generated.
- `new_W` — A `data.frame`, `matrix`, or similar giving the values of baseline covariates (potential confounders) for the conditioning set of the observed values `A`.
- `trim` — A `logical` indicating whether estimates of the conditional density below the value indicated in `trim_min` should be truncated. The default value of `TRUE` enforces truncation of any values below the cutoff specified in `trim_min` and similarly truncates predictions for any of `new_A` falling outside of the training support.
- `trim_min` — A `numeric` indicating the minimum allowed value of the resultant density predictions. Any predicted density values below this tolerance threshold are set to the indicated minimum. The default is to use a scaled inverse square root of the sample size of the prediction set, i.e., 5/sqrt(n)/log(n) (another notable choice is 1/sqrt(n)). If there are observations in the prediction set with values of `new_A` outside of the support of the training set (i.e., provided in the argument `A` to `haldensify`), their predictions are similarly truncated.
- `lambda_select` — A `character` indicating whether to return the predicted density for the value of the regularization parameter chosen by the global cross-validation selector or whether to return an undersmoothed sequence (which starts with the cross-validation selector's choice but also includes all values in the sequence that are less restrictive). The default is `"cv"` for the global cross-validation selector. Setting the choice to `"undersmooth"` returns a matrix of predicted densities, with each column corresponding to a value of the regularization parameter less than or equal to the choice made by the global cross-validation selector. When `"all"` is set, predictions are returned for the full sequence of the regularization parameter on which the HAL model `object` was fitted.

Details

Method for computing and extracting predictions of the conditional density estimates based on the highly adaptive lasso estimator, returned as an S3 object of class `haldensify` from `haldensify`.

Value

A `numeric` vector of predicted conditional density values from a fitted `haldensify` object.

Examples

```
# simulate data: W ~ U[-4, 4] and A|W ~ N(mu = W, sd = 0.5)
n_train <- 50
w <- runif(n_train, -4, 4)
a <- rnorm(n_train, w, 0.5)
# HAL-based density estimator of A|W
```

```
haldensify_fit <- haldensify(
  A = a, W = w, n_bins = 10L, lambda_seq = exp(seq(-1, -10, length = 100)),
  # the following arguments are passed to hal9001::fit_hal()
  max_degree = 3, reduce_basis = 1 / sqrt(length(a))
)
# predictions to recover conditional density of A|W
new_a <- seq(-4, 4, by = 0.1)
new_w <- rep(0, length(new_a))
pred_dens <- predict(haldensify_fit, new_A = new_a, new_W = new_w)
```

---

print.haldensify *Print: Highly Adaptive Lasso Conditional Density Estimates*

Description

Print: Highly Adaptive Lasso Conditional Density Estimates

Usage

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

Arguments

x	An object of class haldensify.
...	Other options (not currently used).

Details

The print method for objects of class haldensify

Value

None. Called for the side effect of printing an informative summary of slots of objects of class haldensify.

Examples

```
# simulate data: W ~ U[-4, 4] and A|W ~ N(mu = W, sd = 0.5)
set.seed(429153)
n_train <- 50
w <- runif(n_train, -4, 4)
a <- rnorm(n_train, w, 0.5)

# learn relationship A|W using HAL-based density estimation procedure
haldensify_fit <- haldensify(
  A = a, W = w, n_bins = c(3, 5),
  lambda_seq = exp(seq(-1, -15, length = 50L)),
  max_degree = 3, reduce_basis = 0.1
)
print(haldensify_fit)
```

print.ipw_haldensify — *Print: IPW Estimates of the Causal Effects of Stochatic Shift Interventions*

Description

Print: IPW Estimates of the Causal Effects of Stochatic Shift Interventions

Usage

```
## S3 method for class 'ipw_haldensify'
print(x, ..., ci_level = 0.95)
```

Arguments

`x`	An object of class `ipw_haldensify`.
`...`	Other options (not currently used).
`ci_level`	A `numeric` indicating the level of the confidence interval to be computed.

Details

The `print` method for objects of class `ipw_haldensify`

Value

None. Called for the side effect of printing an informative summary of slots of objects of class `ipw_haldensify`.

Examples

```
# simulate data
n_obs <- 50
W1 <- rbinom(n_obs, 1, 0.6)
W2 <- rbinom(n_obs, 1, 0.2)
A <- rnorm(n_obs, (2 * W1 - W2 - W1 * W2), 2)
Y <- rbinom(n_obs, 1, plogis(3 * A + W1 + W2 - W1 * W2))

# fit the IPW estimator
est_ipw_shift <- ipw_shift(
  W = cbind(W1, W2), A = A, Y = Y,
  delta = 0.5, n_bins = 3L, cv_folds = 2L,
  lambda_seq = exp(seq(-1, -10, length = 100L)),
  # arguments passed to hal9001::fit_hal()
  max_degree = 1,
  # ...continue arguments for IPW
  selector_type = "gcv"
)
print(est_ipw_shift)
```

Index