cate — Conditional average treatment-effects estimation

Description

`cate` estimates conditional average treatment effects (CATEs), which are average treatment effects (ATEs) conditional on a set of variables for which the treatment effects may vary. Estimating CATEs allows us to study treatment-effect heterogeneity and evaluate treatment-assignment policies.

`cate` provides three different CATE estimates: individualized average treatment effects (IATEs), group average treatment effects (GATEs), and sorted group average treatment effects (GATESs). IATEs are treatment effects conditional on observation-level characteristics; there is one IATE for each observation in the data. GATEs are treatment effects conditional on prespecified groups; there is a treatment effect for each group. GATESs are treatment effects for a prespecified number of groups, where the groups are determined by the quantiles of the IATEs.

To estimate CATEs, `cate` fits an outcome model and a treatment-assignment model. These models can be fit using cross-fitting via lasso, random forest, or parametric regression. The CATEs themselves can be estimated using a partialing-out (PO) estimator or an augmented inverse-probability weighting (AIPW) estimator, either via random forest or linear regression.

Quick start

Estimate the IATE function for outcome `y` and treatment `treat`, conditioning on covariates `x1-x5` and `i.group1`, using the PO estimator, and report the ATE

```
cate po (y x1-x5 i.group1) (treat)
```

Same as above, but add variables `w1-w100` as control variables in the outcome and treatment models

```
cate po (y x1-x5 i.group1) (treat), controls(w1-w100)
```

Same as above, but use the AIPW estimator

```
cate aipw (y x1-x5 i.group1) (treat), controls(w1-w100)
```

Estimate the GATEs for the groups defined by variable `group2`

```
cate aipw (y x1-x5 i.group1) (treat), controls(w1-w100) group(group2)
```

Same as above, but reestimate the GATEs for groups defined by variable `group1` without refitting the IATE function

```
cate, reestimate group(group1)
```

Divide the data into five groups based on quintiles of the IATE estimates, and estimate the GATESs for those groups

```
cate aipw (y x1-x5 i.group1) (treat), controls(w1-w100) group(5)
```

Same as above, but divide the data into four groups (quartiles of the IATE estimates), and reestimate the GATESs for these groups

```
cate, reestimate group(4)
```

Perform cross-fitting with five folds instead of the default ten folds

```
cate aipw (y x1-x5 i.group1) (treat), controls(w1-w100) xfolds(5)
```

Perform random forest for the outcome and treatment models

```
cate aipw (y x1-x5 i.group1) (treat), controls(w1-w100)      ///
    omethod(rforest) tmethod(rforest)
```

Same as above, but use the out-of-bag prediction-based algorithm instead of cross-fitting

```
cate aipw (y x1-x5 i.group1) (treat), controls(w1-w100)      ///
    omethod(rforest) tmethod(rforest) oob
```

Use linear regression to fit the outcome model, logit regression to fit the treatment model, and linear regression to fit the CATE model

```
cate aipw (y x1-x5 i.group1) (treat),                        ///
    omethod(regress) tmethod(logit) cmethod(regress)
```

Menu

Statistics > Causal inference/treatment effects > Continuous outcomes > Conditional average treatment effects

Syntax

Partialing-out estimator

```
cate po (ovar catevarlist) (tvar) [ if ] [ in ] [ , options ]
```

Augmented inverse-probability weighting estimator

```
cate aipw (ovar catevarlist) (tvar) [ if ] [ in ] [ , options ]
```

ovar is a continuous outcome of interest.

catevarlist specifies the covariates of the CATE model—the conditioning variables for the treatment effects. *catevarlist* may contain factor variables; see **[U] 11.4.3 Factor variables**.

tvar must be a binary variable representing the treatment levels.

options	Description
Model	
`controls(`*varlist*`)`	specify the control variables for the outcome and treatment models
* `group(`*varname*`)`	compute the GATE for each group defined by *varname*
* `group(#)`	divide the data into # groups based on IATEs and compute the GATES for each group
`rseed(#)`	set random-number seed
`xfolds(#)`	use cross-fitting algorithm with # folds; default is `xfolds(10)`
Method	
`omethod(`*om_spec*`)`	specify estimation method for outcome model; default is `omethod(lasso)`
`tmethod(`*tm_spec*`)`	specify estimation method for treatment model; default is `tmethod(lasso)`
`cmethod(`*cm_spec*`)`	specify estimation method for CATE model; default is `cmethod(rforest)`
Advanced	
† `reestimate`	reestimate GATEs or GATESs with a new specification in `group()` and without refitting IATE function
‡ `oob`	use out-of-bag prediction-based algorithm instead of cross-fitting
`treatcontrols(`*varlist*`)`	use variables in *varlist* as controls for treatment model instead of variables specified in `controls()` (AIPW estimator only)
`pstolerance(#)`	set tolerance for overlap assumption; default is `pstolerance(1e-5)`
`osample(`*newvar*`)`	generate *newvar* to identify observations that violate the overlap assumption
`rflistwise`	omit observations with missing covariate values when random forest is used for all models
Reporting	
`level(#)`	set confidence level; default is `level(95)`
[`no`]`log`	suppress iteration log
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling

*Only one of `group(`*varname*`)` or `group(#)` may be specified.

†`reestimate` may be specified with `group(#)` only if `group(#)` has been specified in the previous `cate` estimation. If `reestimate` is specified with `group(`*new_varname*`)`, where *new_varname* is different than the previous `group(`*varname*`)`, then *new_varname* must have been a factor variable in *catevarlist* in the previous `cate` estimation.

‡`oob` may not be specified with `group(#)` or `xfolds()`. `oob` is only allowed if the random forest method has been specified for both the outcome and the treatment models (`omethod(rforest)` and `tmethod(rforest)`).

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

om_spec	Description
`lasso`[, *lasso_options*]	use linear lasso to fit the outcome model; the default
`sqrtlasso`[, *lasso_options*]	use square-root lasso to fit the outcome model
`rforest`[, *rforest_options*]	use random forest to fit the outcome model
`regress`	use linear regression to fit the outcome model

tm_spec	Description
`lasso`[, *lasso_options*]	use logit lasso to fit the treatment model; the default
`rforest`[, *rforest_options*]	use random forest to fit the treatment model
`logit`	use logit regression to fit the treatment model
`probit`	use probit regression to fit the treatment model

cm_spec	Description
`rforest`[, *rforest_options*]	use random forest to fit the CATE model; the default
`regress`	use linear regression to fit the CATE model

rforest_options	Description
`samprate(#)`	specify sampling rate for observations; default is `samprate(0.5)`
`ntrees(#)`	specify number of trees in the forest; default is `ntrees(2000)`
`cintrees(#)`	specify number of trees in each group to compute the confidence intervals; default is `cintrees(2)`
`splitminobs(#)`	specify minimum number of observations to split a node; default is `splitminobs(6)`
`splitmeanvars(#)`	specify mean number of variables to be split in each node; default is `splitmeanvars(ceil(sqrt(`*p*`) + 20))` with *p* as the dimension of *catevarlist*
`nohonest`	do not use an honest tree
`honestrate(#)`	set sampling rate for honest tree; default is `honestrate(0.5)`

Options

Model

`controls(`*varlist*`)` specifies the control variables for the outcome and treatment models. *catevarlist* and the specified control variables are the covariates in the outcome and the treatment models. If no control variables are specified, then the variables specified in *catevarlist* will be the only covariates for both models.

`group(`*varname*`)` computes the GATE for each group defined by the levels of *varname*. The ATE for each level in the group variable will be estimated. The grouping variable will be added as a factor variable in *catevarlist*. Only one of `group(`*varname*`)` or `group(#)` may be specified.

`group(#)` computes the GATESs by dividing the observations into # groups. The groups are generated from the quantiles of the estimates of the IATEs. The observations are sorted based on the IATEs and grouped into # levels. For example, if we specify `group(4)`, the data would be divided into four groups. The first group will contain observations with IATE estimates greater than the 75th percentile of the overall IATE estimates, the second group will contain observations that lie between the 50th and the 75th percentiles, the third group will contain those between the 25th and 50th percentiles, and the last group will contain those below the 25th percentile. Once the groups are formed, `cate` computes the ATE for each group. Only one of `group(#)` or `group(`*varname*`)` may be specified.

`rseed(#)` sets the random-number seed. `rseed(#)` is equivalent to typing `set seed #` prior to running `cate`. Random numbers are used to produce split samples for cross-fitting. To reproduce results, you must either use this option or use `set seed`. See [R] **set seed**.

`xfolds(#)` specifies the number of folds for cross-fitting. The default is `xfolds(10)`; that is, cross-fitting is done by randomly dividing the original data into 10 folds.

Method

`omethod(`*om_spec*`)` specifies the estimation method for the outcome model. *om_spec* may be `lasso[ ,` *lasso_options* `]`; `sqrtlasso[ ,` *lasso_options* `]`; `rforest[ ,` *rforest_options* `]`; or `regress`. The default is `omethod(lasso)`.

`lasso[ ,` *lasso_options* `]` specifies that linear lasso be used to fit the outcome model. *lasso_options* are `selection()`, `grid()`, `stop()`, `cvtolerance()`, `bictolerance()`, `tolerance()`, and `dtolerance()`; see [LASSO] **lasso options**. If `selection()` is not specified, then `selection(plugin)` is assumed; that is, the plugin penalty parameter is used.

`sqrtlasso[ ,` *lasso_options* `]` specifies that square-root lasso be used to fit the outcome model. *lasso_options* are `selection()`, `grid()`, `stop()`, `cvtolerance()`, `bictolerance()`, `tolerance()`, and `dtolerance()`; see [LASSO] **lasso options**. If `selection()` is not specified, then `selection(plugin)` is assumed; that is, the plugin penalty parameter is used.

`rforest[ ,` *rforest_options* `]` specifies that random forest be used to fit the outcome model. *rforest_options* are `samprate()`, `ntrees()`, `cintrees()`, `splitminobs()`, `splitmeanvars()`, `nohonest`, and `honestrate()`.

`samprate(#)` sets the sampling rate for observations when drawing the random sample for each tree. The sampling is without replacement. The sampling rate must be in the range (0, 1). The default is `samprate(0.5)`, meaning that half of the estimation sample is used to construct each tree. Using a random sample to construct each tree makes the random forest more robust to the overfitting issues.

`ntrees(#)` sets the number of trees in the random forest. The default is `ntrees(2000)`. Using more trees in the random forest usually implies more stable estimates, but it also requires longer computational time.

`cintrees(#)` sets the number of trees in each group or bag when using the bootstrap of little bags to compute the confidence intervals of the random forest's predictions. Each tree in the same bag draws a random sample from the same half-size sample, which allows us to estimate the variance of the random forest's prediction. The default is `cintrees(2)`, two trees in each bag.

`splitminobs(#)` sets the minimum number of observations to perform a split in a node. A node must have at least # observations to be split. The default is `splitminobs(6)`.

`splitmeanvars(#)` sets the mean number of variables to be split in each node. In each tree node in a random forest, only a random subset of variables is searched to find the best splitting variable and value. The number of variables in this subset is also random and equals $\max(\min(m, p), 1)$, where p is the dimension of *catevarlist* and m follows a Poisson distribution with mean #. The default is `splitmeanvars(ceil(sqrt(`p`) + 20))`.

`nohonest` specifies not to use an honest tree. Honest splitting in an honest tree is the critical feature that allows us to make inferences on the random forest's prediction. Confidence intervals and standard errors for the random forest's prediction cannot be estimated when `nohonest` is specified.

`honestrate(#)` specifies the fraction of the sample used for splitting the honest tree. For a random sample S drawn to create a tree, an honest tree divides sample S into two disjoint parts A and B. Part A is used to split the tree and part B is used to label the tree. `honestrate()`

specifies the fraction of the sample S to be used as part A. Honest splitting is the key feature that allows inference on the random forest's prediction. The default is `honestrate(0.5)`, where half of the data in S is used to split the tree and the other half is used to label the tree.

`regress` specifies that linear regression be used to fit the outcome model. Thus, this option imposes a parametric assumption on the outcome model.

`tmethod(`*tm_spec*`)` specifies the estimation method for the treatment model. *tm_spec* may be `lasso[ ,` *lasso_options* `]`, `rforest[ ,` *rforest_options* `]`, `logit`, or `probit`. The default is `tmethod(lasso)`.

`lasso[ ,` *lasso_options* `]` specifies that logit lasso be used to fit the treatment model. *lasso_options* are `selection()`, `grid()`, `stop()`, `cvtolerance()`, `bictolerance()`, `tolerance()`, and `dtolerance()`; see [LASSO] **lasso options**. If `selection()` is not specified, then `selection(plugin)` is assumed; that is, the plugin penalty parameter is used.

`rforest[ ,` *rforest_options* `]` specifies that random forest be used to fit the treatment model. *rforest_options* are `samprate()`, `ntrees()`, `cintrees()`, `splitminobs()`, `splitmeanvars()`, `nohonest`, and `honestrate()`; see *rforest_options*.

`logit` specifies that a logit model be used to fit the treatment model. Thus, this option imposes a parametric assumption on the treatment model.

`probit` specifies that a probit model be used to fit the treatment model. Thus, this option imposes a parametric assumption on the treatment model.

`cmethod(`*cm_spec*`)` specifies the estimation method for the CATE model. *cm_spec* may be `rforest[ ,` *rforest_options* `]` or `regress`. The default is `cmethod(rforest)`.

`rforest[ ,` *rforest_options* `]` specifies that random forest be used to fit the CATE model. *rforest_options* are `samprate()`, `ntrees()`, `cintrees()`, `splitminobs()`, `splitmeanvars()`, `nohonest`, and `honestrate()`; see *rforest_options*.

`regress` specifies that linear regression be used to fit the CATE model. Thus, this option assumes a parametric assumption on the CATE model.

Advanced

`reestimate` reestimates GATEs or GATESs with a new specification in `group()`. It is much faster than estimating GATEs or GATESs from scratch because it uses existing results from the previous `cate` estimation for the estimates of the IATE function. The typical usages of this option are the following:

`reestimate` with the `group(`*new_varname*`)` option reestimates the GATEs with the group variable *new_varname*. This syntax requires that the group variable *new_varname* was specified as a factor variable in *catevarlist* in the previous `cate` estimation. For example, after using `cate` to estimate the GATE for each level of group variable `group1`,

```
cate (y x1 x2 i.group2) (treat), group(group1)
```

we can estimate the GATE for each level of group variable `group2`:

```
cate, reestimate group(group2)
```

Notice above that `group2` is specified as a factor variable in *catevarlist*.

reestimate with the group(#) option reestimates the GATESs with # data-driven groups. This syntax requires that option group(#) has been specified in the previous cate estimation. For example, after using cate to estimate the GATESs with 4 data-driven groups,

```
cate (y x1 x2) (treat), group(4)
```

we can estimate the GATESs with 5 data-driven groups:

```
cate, reestimate group(5)
```

Notice that we specified option group(#) in the previous cate specification.

oob uses the out-of-bag prediction-based algorithm instead of cross-fitting. It requires that the random forest is used to fit the outcome model and the treatment model; that is, both omethod(rforest) and tmethod(rforest) must be specified with oob. The out-of-bag prediction-based algorithm is generally faster than cross-fitting under the same setup. However, this algorithm does not allow computing the GATESs; thus, oob may not be combined with group(#).

treatcontrols(*varlist*) specifies that *varlist* be used as controls in the treatment model instead of the variables specified in controls(). This option is only allowed for the AIPW estimator. If treatcontrols() is not specified, then the variables in *catevarlist* and the variables specified in controls() are used as covariates in both the outcome and the treatment models. If treatcontrols(*varlist*) is specified, then the treatment model instead uses *varlist* as covariates.

pstolerance(#) specifies the tolerance used to check the overlap assumption. The default value is pstolerance(1e-5). cate will exit with an error if an observation has an estimated propensity score smaller than that specified by pstolerance().

osample(*newvar*) generates indicator variable *newvar* identifying observations that violate the overlap assumption.

rflistwise specifies that listwise deletion be used when the random forest method is used for all models. By default, when omethod(rforest), tmethod(rforest), and cmethod(rforest) are all specified, observations with missing covariate values will be used because the random forest method can use missing covariate values in estimation. See *Generalized random forest* in *Methods and formulas*.

When rflistwise is specified, observations with missing covariate values are not used to estimate the CATEs. If one of omethod(), tmethod(), or cmethod() does not use rforest, then observations with missing covariate values will not be used for estimating the CATEs.

Reporting

level(#); see [R] **Estimation options**.

[no]log displays or suppresses a log showing the progress of the estimation. The log is displayed by default unless you used set iterlog off to suppress it; see set iterlog in [R] ***set iter***.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

Remarks and examples

Remarks are presented under the following headings:

Introduction

Treatment effects estimate the causal effect of a treatment on an outcome. This effect may be constant or it may vary across different subpopulations. For example, a labor economist may want to know if the earnings of immigrants and nonimmigrants are affected differently by a job training program and, if so, by how much. An online shopping company may want to know the effect of a price discount on purchasing behavior for customers with different demographic characteristics such as age or income. A medical team may want to measure the effect of smoking on stress levels for individuals in different age groups.

The ATE is a popular way to summarize the treatment effects by taking the mean of the effects over the population. The ATE characterizes the whole distribution of treatment effects when the treatment effect is constant across the population. However, when the treatment effects are heterogeneous and the units react differently to the same treatment, estimating only the mean of treatment effects may mask the underlying mechanism of how the treatment affects different units. For example, the estimated ATE may be close to zero when some groups experience positive effects while other groups experience adverse effects.

In contrast to the ATE, the CATEs help us better understand the heterogeneous nature of treatment effects. Like the ATE, the CATEs are averages of treatment effects, but unlike the ATE, the averages are taken over population subgroups. Imagine that we have a microscope to observe the treatment effects. The ATE only allows us to look at the effects at the most coarse precision, but the CATEs allow us to zoom in on particular parts of the population. Furthermore, once we understand the heterogeneity of treatment effects, we can evaluate different treatment-assignment policies that may shed light on which policy would result in better overall outcomes for different groups in the population.

In summary, the advantage of studying the CATEs is, at least, two-fold:

1. It improves understanding of the treatment-effect heterogeneity.
2. It builds a foundation to optimize the assignment to treatment.

What is a CATE?

So what is a CATE precisely? Under the potential outcome framework, we define $y_i(1)$ to be the potential outcome if unit i is treated and define $y_i(0)$ to be the potential outcome if unit i is not treated. $\mathbf{x}_i$ is a vector of characteristics for unit i. The CATE is defined as

$$\text{CATE} \equiv \tau(\mathbf{x}) = E\{y_i(1) - y_i(0)|\mathbf{x}_i = \mathbf{x}\}$$

That is, the CATE is the expectation of the difference between the treated and untreated potential outcomes conditional on the characteristics, $\mathbf{x}_i$, being equal to $\mathbf{x}$.

We can identify $\tau(\mathbf{x})$ via either a partial linear or a fully interactive model. Here we build intuition by focusing on identification of the CATE via the partial linear model. For notational simplicity, we drop the subscript i indicating the ith observation to refer to a random variable. We refer to the observed outcome as y and the binary treatment indicator as d.

In the simplest case, when the treatment effects are constant or when we are interested in estimating the ATE, the partial linear model is

$$\begin{aligned} y &= d * \tau + g(\mathbf{x}, \mathbf{w}) + \epsilon \\ d &= f(\mathbf{x}, \mathbf{w}) + u \end{aligned}$$

Here we divide the variables into two groups: $\mathbf{x}$ and $\mathbf{w}$. We will differentiate the two shortly. The outcome model is partial linear because the observed outcome is a sum of the treatment effects $d * \tau$, a nonparametric function $g(\mathbf{x}, \mathbf{w})$, and the error term ϵ. The treatment assignment is modeled by the function $f(\mathbf{x}, \mathbf{w})$ and an additive error term u. By definition, we can write the potential outcome models as

$$\begin{aligned} y(1) &= \tau + g(\mathbf{x}, \mathbf{w}) + \epsilon \\ y(0) &= g(\mathbf{x}, \mathbf{w}) + \epsilon \end{aligned}$$

Thus, τ characterizes the ATE.

$$\text{ATE} \equiv E\{y(1) - y(0)\} = \tau$$

Now let's go one step further. Suppose the treatment effects are heterogeneous and depend on $\mathbf{x}$. We can rewrite the outcome model as

$$y = d * \tau(\mathbf{x}) + g(\mathbf{x}, \mathbf{w}) + \epsilon$$

where $\mathbf{x}$ is a vector of conditioning variables for the treatment effects. $\tau(\mathbf{x})$ is a function of $\mathbf{x}$ that interacts with the treatment d. Notice that $\tau(\mathbf{x})$ is a function of $\mathbf{x}$ but not of $\mathbf{w}$. $\mathbf{w}$ is an optional vector of additional control variables for the outcome and treatment-assignment models, which can potentially be high-dimensional. This model is flexible and general because it does not impose parametric assumptions on $\tau(\mathbf{x})$ or $g(\mathbf{x}, \mathbf{w})$. The potential outcomes now become

$$\begin{aligned} y(1) &= \tau(\mathbf{x}) + g(\mathbf{x}, \mathbf{w}) + \epsilon \\ y(0) &= g(\mathbf{x}, \mathbf{w}) + \epsilon \end{aligned}$$

Thus, the CATE is $\tau(\mathbf{x})$.

$$E\{y(1) - y(0)|\mathbf{x}\} = \tau(\mathbf{x})$$

If we impose parametric assumptions such as $\tau(\mathbf{x}) = \mathbf{x}'\beta$ and $g(\mathbf{x}, \mathbf{w}) = \mathbf{x}'\gamma_1 + \mathbf{w}'\gamma_2$, we can estimate this special model by running a regression of y on $\mathbf{x}$, $\mathbf{w}$, and the interaction between d and $\mathbf{x}$. However, this parametric assumption may be too strong and hard to satisfy with our data. In the implementation of `cate`, we focus on a more general case that does not impose a parametric form on $\tau(\mathbf{x})$ or the nuisance parameters $g(\mathbf{x}, \mathbf{w})$ and $f(\mathbf{x}, \mathbf{w})$. The parameter of interest, $\tau(\mathbf{x})$, can be estimated nonparametrically via the generalized random forest proposed in Athey, Tibshirani, and Wager (2019). The nuisance parameters can be estimated using lasso or random forest. However, if we want to impose a parametric assumption on either $\tau(\mathbf{x})$ or the nuisance parameters, `cate` can also fit such models.

Different versions of the CATE

The granularity of the conditional set $\mathbf{x}_i$ determines the subpopulation. Thus, the CATE has different versions or names depending on the definition of the conditional set $\mathbf{x}_i$.

IATE: At the finest level, when $\mathbf{x}_i$ refers to the characteristics of a specific observation, the CATE measures the expected treatment effect for individuals with the same characteristics as this observation. This version of CATE is also called IATE. In literature, people often refer to IATE as CATE, even though IATE is a special version of CATE. In our terminology, we use IATE when we refer to the finest level of CATE.

Under the unconfoundedness assumption, it is possible to identify and estimate $\tau(\mathbf{x})$ if we are willing to impose some restrictions on $\tau(\mathbf{x})$. In particular, when $\mathbf{x}$ is a low-dimensional vector and $\tau(\mathbf{x})$ is smooth enough as defined in Athey, Tibshirani, and Wager (2019), we can nonparametrically estimate the IATE and provide confidence intervals using the generalized random forest proposed in Athey, Tibshirani, and Wager (2019).

GATE: If $\mathbf{x}_i$ is a prespecified grouping, denoted by G_i, the CATE measures the ATE for each group. This version of CATE is also called GATE.

In particular, the GATE is defined as

$$\tau(g) = E\{y_i(1) - y_i(0) | G_i = g\}$$

where G_i is a prespecified grouping and g is a specific group. The GATEs are coarser than the IATEs because they focus on group effects instead of individual effects.

For the GATEs, we must specify the group variable, G_i, before analyzing the data to avoid p-value hacking, as discussed in Head et al. (2015).

GATES: Sometimes, we do not have a group variable to specify but still want to understand the underlying treatment-effect heterogeneity. In such cases, we can discover the groups in a data-driven way by using the sorted IATEs. This version of CATE is known as GATES.

The groups are generated by the quantiles of the IATE estimates. For example, let's say we want to divide the data into four groups. The first group will consist of the observations with IATE estimates greater than the 75th percentile of the overall IATE estimates, the second group will include observations whose estimates lie between the 50th and 75th percentiles, the third group will contain observations with estimates between the 25th and 50th percentile, and the last group will contain observations with estimates below the 25th percentile.

Once we have the groups as above, we can estimate the GATEs as usual.

The cate command estimates IATEs, GATEs with prespecified groups, and GATESs, for which groups are determined in a data-driven way. cate helps us answer questions about treatment heterogeneity, such as the following:

1. Are the treatment effects heterogeneous?
2. How do the treatment effects vary with some variables?
3. Do the treatment effects vary between prespecified groups?
4. Do the data discover groups where treatment effects are different?

Additionally, we can use the estimates obtained from cate to measure the effect of counterfactual treatment-assignment policies on the outcome by using postestimation commands such as estat policyeval and estat ate. Suppose a hypothetical treatment-assignment policy assigns some individuals in the sample to be treated and some others not to be treated. Using estat policyeval and estat ate postestimation commands, we may answer questions such as the following:

1. If we implement such a policy, how would the average outcome in the population change?
2. Which policy is better among a candidate set of policies?

Overview of the cate suite

Here we outline the Stata commands to estimate, predict, visualize, and make inferences about the CATEs. In particular, these Stata commands can be grouped into the following categories:

Estimation: cate po estimates the IATE function by using the PO estimator discussed in Nie and Wager (2021) via the generalized random forest proposed in Athey, Tibshirani, and Wager (2019). This method is the default and is also known as causal forest.

cate aipw estimates the IATE function by using the AIPW estimator discussed in Knaus (2022) and Kennedy (2023) via the honest regression random forest proposed in Wager and Athey (2018) by default.

cate with option group(*varname*) estimates the GATEs by taking the means of the AIPW scores implied by the model (the estimates of the individual-level treatment effects) over the group variable *varname*. This method is discussed in Semenova and Chernozhukov (2021) and Knaus (2022).

cate with option group(#) estimates the GATESs for # groups with levels based on the rankings of the IATE estimates. This method is discussed in Chernozhukov et al. (2006). Once the groups are discovered, the GATESs are estimated as the GATEs.

Prediction: predict predicts the IATE function $\tau(\mathbf{x})$, its standard errors, and the lower and upper bounds of the pointwise confidence intervals. The estimates of standard errors and confidence intervals are computed using the bootstrap of little bags proposed in Athey, Tibshirani, and Wager (2019).

Visualization: `categraph histogram` plots the histogram of estimated IATEs and shows how the IATEs are distributed. This histogram can serve as a preliminary visualization of the treatment-effect heterogeneity.

`categraph gateplot` plots the estimate of the GATE or GATES and its confidence interval for each group. It visualizes the trend of the GATE or GATES function.

`categraph iateplot` plots the IATE function with varying values of **x**. If **x** is a vector, we can allow one variable to vary, fix the values of the other variables, and then use `categraph iateplot` to plot the function.

Inference: `estat heterogeneity` tests whether the treatment effects are heterogeneous using the method proposed by Chernozhukov et al. (2006).

`estat gatetest` tests whether the estimated GATEs or GATESs are equal across the groups.

`estat classification` compares the means of a variable in the group with the largest treatment effect and the group with the smallest treatment effect. It is used to compare the properties of the subpopulations with the largest and smallest effects.

`estat ate` computes the ATE for a subpopulation. This command can be useful for policy evaluations.

`estat projection` fits a linear regression of the estimated IATE on a vector of variables. It provides a linear approximation of the IATE function.

`estat series` fits a nonparametric series regression of the estimated IATE on a vector of variables using B-spline, piecewise polynomial spline, or polynomial basis. It provides a nonparametric approximation of the IATE function, as discussed in Semenova and Chernozhukov (2021).

`estat policyeval` evaluates and compares the prespecified treatment-assignment policy. In particular, it computes the value of a treatment-assignment policy or compares the difference of two policies' values if specified.

Workflows

Here we provide possible workflows that may be useful, depending on the question of interest. Workflows 1 to 3 help us to answer questions regarding the treatment-effects heterogeneity, and workflows 4 and 5 help us to evaluate treatment-assignment policies. Below, we list the questions of interest for each workflow. Then, we will discuss the details of the workflows.

1. Understand treatment-effect heterogeneity:

   - **Workflow 1: Exploiting the IATE function heterogeneity**

     Given an estimate of the IATE function, are the treatment effects heterogeneous?

   - **Workflow 2: Prespecified group hypothesis testing**

     We have some prespecified groups and we want to test whether the treatment effects are the same across these groups or study how the effects differ across them.

- **Workflow 3: Data-driven group hypothesis testing**

  We do not know the groups for which the treatment effects may vary, so we ask the data to discover these groups and study whether certain variables may be correlated with the treatment-effect heterogeneity.

2. Policy evaluation:

   - **Workflow 4: Evaluation of counterfactual policies**

     We want to evaluate some prespecified treatment-assignment policies and compare them.
   - **Workflow 5: Evaluating policies designed using the IATE estimates**

     We want to evaluate a policy that is designed based on the IATE estimates. For example, we want to treat all units with an IATE greater than a fixed cost and evaluate the effect of such a policy on the outcome.

Workflow 1: Exploiting the IATE function heterogeneity

1. Suppose the outcome variable is `y`, the CATE covariates are `x1-x5`, the treatment variable is `treat`, and the control variables are `w1-w100`. Estimate the IATE function. (We demonstrate using the PO estimator but could also use the AIPW estimator.)

   ```
   cate po (y x1-x5) (treat), controls(w1-w100)
   ```

2. Plot the histogram of IATE estimates.

   ```
   categraph histogram
   ```

3. Test whether the effects are heterogeneous.

   ```
   estat heterogeneity
   ```

4. Regress the estimated IATE function on variables that may impact treatment effects to understand the mechanism underlying the treatment-effect heterogeneity.

   ```
   estat projection x1-x5
   ```

5. Estimate the ATE for a subpopulation of interest. Suppose we suspect that the variable `x1` positively affects the treatment effects; we can estimate the ATE for the subpopulation where `x1` is greater than 0.8.

   ```
   estat ate if x1 > 0.8
   ```

6. Plot the IATE function for `x1` while the other variables are fixed at specific values, such as their means.

   ```
   estat iateplot x1, at((mean) x2-x5)
   ```

Workflow 2: Prespecified group hypothesis testing

1. Estimate the GATE function over the levels of group variable gvar. (We demonstrate using the AIPW estimator but could also use the PO estimator.)

```
cate aipw (y x1-x5) (treat), controls(w1-w100) group(gvar)
```

2. Visualize the GATE estimates and their confidence intervals.

```
categraph gateplot
```

3. Test whether the GATEs are the same across groups.

```
estat gatetest
```

Workflow 3: Data-driven group hypothesis testing

1. Estimate the GATES function for five groups created by dividing data into groups based on rankings of the IATE estimates. (We demonstrate using the AIPW estimator but could also use the PO estimator.)

```
cate aipw (y x1-x5) (treat), controls(w1-w100) group(5)
```

2. Visualize the GATES estimates and their confidence intervals.

```
categraph gateplot
```

3. Test whether the GATESs are the same across groups.

```
estat gatetest
```

4. Compare the mean of x1 in the groups with the smallest and largest treatment effects.

```
estat classification x1
```

Workflow 4: Evaluation of counterfactual policies

1. Estimate the IATE function. (We demonstrate using the PO estimator but could also use the AIPW estimator.)

```
cate po (y x1-x5) (treat), controls(w1-w100)
```

2. Estimate the average outcome in the population for a potential policy. Suppose the policy1 variable stores the treatment assignments for each observation under a counterfactual policy; we estimate the average outcome for policy1.

```
estat policyeval policy1
```

3. Compare the average outcomes in the population for multiple potential policies. Suppose the policy2 variable stores the treatment assignments for an alternative policy; we compare average outcomes for policy1 and policy2.

```
estat policyeval policy1 policy2
```

Workflow 5: Evaluating policies designed using the IATE estimates

1. Split the sample into training and testing data.

```
splitsample, split(0.6 0.4) generate(group)
```

2. Estimate the IATE function using the training data, `group` = 1. (We demonstrate using the PO estimator but could also use the AIPW estimator.)

```
cate po (y x1-x5) (treat) if group == 1 , controls(w1-w100)
```

3. Predict the IATE function in the testing data, `group` = 2; the predicted IATE function is used to construct the policy rule.

```
predict tauhat if group == 2
```

4. Estimate the IATE function using the testing data; the AIPW scores will be to compute the ATE in the testing sample.

```
cate po (y x1-x5) (treat) if group == 2, controls(w1-w100)
```

5. Estimate the ATE for the entire testing sample.

```
estat ate if group == 2
```

6. Estimate the ATE for a subset of units in the testing sample based on the IATE predictions. For instance, estimate the ATE for units with predicted IATEs greater than 50 in the testing sample.

```
estat ate if group == 2 & tauhat >= 50
```

Examples

In the following examples, we illustrate how to use `cate` to study treatment-effect heterogeneity and to evaluate treatment-assignment policies. In particular, in examples 1 to 7, we demonstrate commands to evaluate the effects of 401(k) program eligibility on net financial wealth. Suppose that we want to answer the following questions:

1. Are the effects of 401(k) eligibility on net wealth heterogeneous? In other words, do the treatment effects vary across individuals or groups?

2. If the treatment effects are heterogeneous, how do they vary across levels of prespecified group variables, such as income category, home ownership, or education level?

3. Do the data discover groups in which the treatment effects are particularly high or low?

In example 8, we demonstrate commands to evaluate the effects of two types of lung transplants on patients' health outcomes. Supposing a doctor has a treatment-assignment recommendation rule, we want to evaluate the overall outcomes if this treatment-assignment rule is implemented.

Example 1: Explore treatment-effect heterogeneity

Suppose we want to estimate the effect of 401(k) eligibility (e401k) on net financial assets (asset) using data reported by Chernozhukov and Hansen (2004). These data are from a sample of households in the 1990 Survey of Income and Program Participation (SIPP). The data contain information on the head of the household: income level category (incomecat), age (age), years of education (educ), whether they receive pension benefits (pension), marital status (married), whether they participate in an IRA (ira), whether they own a home (ownhome), and whether there are two earners in the same household (twoearn).

We believe that the treatment effects of e401k on asset could vary based on incomecat, age, educ, pension, married, ira, ownhome, and twoearn, which we denote as **x**. asset(1) represents the potential outcomes (net financial assets) of being eligible for a 401(k), and asset(0) represents the potential outcomes of not being eligible for a 401(k). We want to estimate the effects of 401(k) eligibility on assets conditional on the variables **x**. In other words, we are interested in estimating the effects as a function of **x**. Precisely, we want to estimate

$$E\{\texttt{asset}(1) - \texttt{asset}(0)|\mathbf{x}\}$$

This version of CATEs is also known as individualized average treatment effects (IATEs) because **x** refers to individual characteristics. In the syntax of cate, **x** is referred to as the *catevarlist*.

In this example, we use the PO estimator in the partial linear model to estimate the IATE function. Without assuming any additional control variables, the partial linear model for asset is

$$\texttt{asset} = \texttt{e401k} * \tau(\mathbf{x}) + g(\mathbf{x}) + \epsilon$$

where $\tau(\mathbf{x})$ is a function of **x** that interacts with the treatment e401k, $g(\mathbf{x})$ is a nonparametric nuisance function, and ϵ is the error term for the outcome. The treatment-assignment model for the treatment e401k is

$$\texttt{e401k} = f(\mathbf{x}) + u$$

where $f(\mathbf{x})$ is a nonparametric nuisance function and u is an error term for the treatment.

The potential outcomes are

$$\texttt{asset}(1) = \tau(\mathbf{x}) + g(\mathbf{x}) + \epsilon$$
$$\texttt{asset}(0) = g(\mathbf{x}) + \epsilon$$

Thus, the function $\tau(\mathbf{x})$ identifies the IATE function.

$$\tau(\mathbf{x}) = E\{\texttt{asset}(1) - \texttt{asset}(0)|\mathbf{x}\}$$

Notice that we do not assume any functional form of $\tau(\mathbf{x})$, and it can be as simple as a linear model or any arbitrary function of **x**. Here we want the data to tell us what this function $\tau(\mathbf{x})$ looks like instead of assuming a specific functional form. We can use cate to estimate the function $\tau(\mathbf{x})$ nonparametrically via the generalized random forest proposed in Athey, Tibshirani, and Wager (2019); this method is also known as causal forest and is the default method used by cate.

First, we open the assets3 data. To save some typing later, we define a global macro, catecovars, to represent the IATE conditioning variables **x**.

```
. use https://www.stata-press.com/data/r19/assets3
(Excerpt from Chernozhukov and Hansen (2004))
. global catecovars age educ i.(incomecat pension married twoearn ira ownhome)
```

We are ready to fit the model using `cate`. We specify `po` to use the partialing-out estimator. We specify the outcome variable `asset` and *catevarlist* (the **x** variables) in the first set of parentheses and the treatment-assignment variable `e401k` in the second set. We also specify the `rseed()` option to make the results reproducible. The first portion of the output is

```
. cate po (assets $catecovars) (e401k), rseed(12345671)
Cross-fit fold 1 of 10 ...
Performing lasso for outcome assets ...
Performing lasso for treatment e401k ...
  (output omitted)
Cross-fit fold 10 of 10 ...
Performing lasso for outcome assets ...
Performing lasso for treatment e401k ...
```

This iteration log corresponds to the cross-fitting process that is used to fit the outcome model for `assets` and the treatment model for `e401k`. To estimate the IATE function $\tau(\mathbf{x})$, the PO estimator needs to partial out the nuisance functions $g(\mathbf{x})$ and $f(\mathbf{x})$. To do this, `cate` needs to estimate the expectation of the outcome and the treatment variable conditional on **x**. By default, the lasso for the linear model is used to estimate the outcome `assets`, and the lasso for the logit model is used to estimate the treatment `e401k`. To guard against errors when fitting the nuisance functions (variable selection errors when using lasso and prediction errors when using random forest), `cate` uses cross-fitting. By default, ten-fold cross-fitting is used. See *Methods and formulas* for details.

We can also use other alternatives, such as a random forest or a parametric model, to estimate the outcome and the treatment models. Here, we use the default lasso for both models.

The remaining output is

```
Performing random forest for IATE ...
Estimating AIPW scores ...
Estimating ATE ...
Conditional average treatment effects     Number of observations       = 9,913
Estimator:       Partialing out           Number of folds in cross-fit =    10
Outcome model:   Linear lasso             Number of outcome controls   =    17
Treatment model: Logit lasso              Number of treatment controls =    17
CATE model:      Random forest            Number of CATE variables     =    17
```

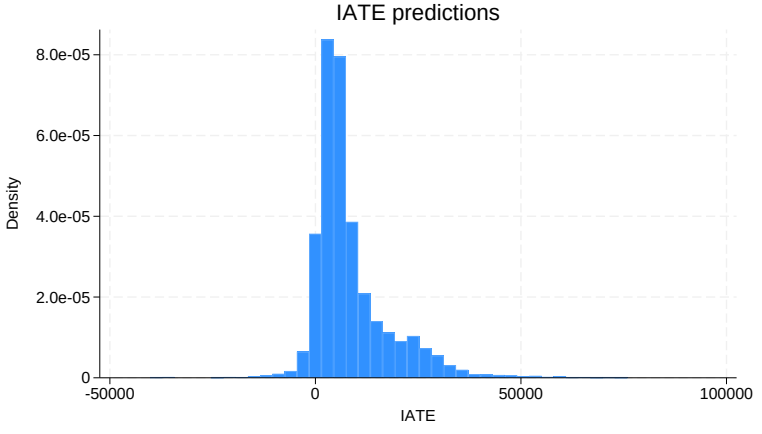
assets	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
e401k (Eligible vs Not elig..)	7937.182	1153.017	6.88	0.000	5677.309	10197.05
POmean						
e401k Not eligi..	14016.38	833.4423	16.82	0.000	12382.87	15649.9

We see that a random forest is used to estimate the IATE function once the cross-fitting is finished. It then estimates the AIPW scores, not to be confused with the AIPW estimator. The AIPW scores are doubly robust estimates of individual-level treatment effects. The average of these AIPW scores is the ATE. The estimated ATE indicates that if everyone in the population is eligible for a 401(k), the net financial assets

will, on average, be \$7,937 larger than the net financial assets if no one is eligible for a 401(k). The potential outcome mean indicates that net financial assets are expected to be \$14,016 if no one is eligible for a 401(k).

In addition to the ATE that we see in the output, `cate` also estimates the IATE function $\tau(\mathbf{x})$, and we can use it to predict the treatment effects for each observation. We can use `categraph histogram` to draw a histogram of the predicted $\tau(\mathbf{x})$ function and see its distribution.

```
. categraph histogram
(bin=39, start=-40204.13, width=2975.4332)
```



The graph shows that treatment effects are mostly positive but have a fat right tail. Thus, the ATE may underestimate the effect of 401(k) eligibility on assets for some groups.

Although the histogram above allows us to inspect the distribution of treatment effects visually, we should not use it as conclusive evidence to support treatment-effect heterogeneity. For example, when the number of observations in the sample is small or when the number of the CATE conditioning variables $\mathbf{x}$ is very large, we will likely see a well-spread histogram of IATE predictions due to the estimation noise even if the actual CATE function is constant.

To statistically test whether the treatment effects are heterogeneous, we use `estat heterogeneity`.

```
. estat heterogeneity

Treatment-effects heterogeneity test
H0: Treatment effects are homogeneous

    chi2(1) =   4.11
Prob > chi2 = 0.0427
```

We find evidence against the null hypothesis that the treatment effects are homogeneous.

To further explore the heterogeneity of treatment effects, we want to know whether a variable positively or negatively affects the treatment effects. One way to do this is by linearly projecting the AIPW scores, the individual-level treatment effects estimated by `cate`, on the variables of interest. We use `estat projection`.

```
. estat projection
Treatment-effects linear projection                Number of obs =      9,913
                                                   F(11, 9901)   =       4.90
                                                   Prob > F      =     0.0000
                                                   R-squared     =     0.0045
                                                   Adj R-squared =     0.0034
                                                   Root MSE      = 1.146e+05
```

	Coefficient	Robust std. err.	t	P>|t|	[95% conf.	interval]
age	205.1206	117.9809	1.74	0.082	-26.14605	436.3873
educ	-442.4583	488.4721	-0.91	0.365	-1399.963	515.0466
incomecat						
1	-2439.222	2013.522	-1.21	0.226	-6386.136	1507.692
2	1874.817	2295.155	0.82	0.414	-2624.154	6373.788
3	5707.689	3298.341	1.73	0.084	-757.7313	12173.11
4	18194.6	5398.391	3.37	0.001	7612.651	28776.54
pension Receives ..	3817.355	2454.437	1.56	0.120	-993.8419	8628.553
married Married	-2399.333	3403.066	-0.71	0.481	-9070.035	4271.37
twoearn Yes	-1428.041	4347.025	-0.33	0.743	-9949.094	7093.013
ira Yes	-2438.404	3619.217	-0.67	0.500	-9532.807	4656
ownhome Yes	3162.649	1669.587	1.89	0.058	-110.081	6435.379
_cons	232.7251	8072.023	0.03	0.977	-15590.08	16055.53

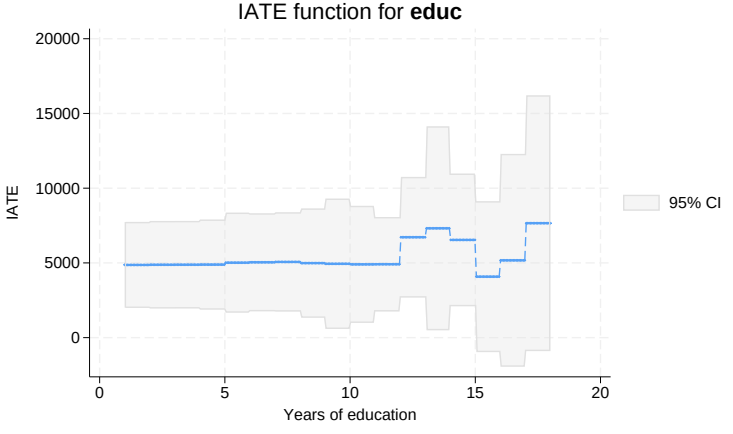
Without specifying any variables, `estat projection` projects the AIPW scores on all the variables defined in **x**. In other words, it performs a regression of $\tau(\mathbf{x})$ on the conditioning variables in our model. We can interpret the coefficients as the effects of variables on the linear approximation of treatment effects. For example, the coefficient for `4.incomecat` is 18,195. We can say that being in the highest income category increases the 401(k) eligibility effects on assets by $18,195 over being in the lowest income category if the treatment effects are linearly approximated by the variables defined in **x**.

We can also plot the estimated IATE function by allowing one variable to vary and fixing the other variables to set values. For example, we can plot the function $\tau(\mathbf{x})$ by allowing `educ` to vary and fixing the values of the other variables.

We use `categraph iateplot`. We specify the variable `educ` to allow the IATE function to vary with `educ`. By default, the continuous variables, such as `age`, are fixed at their sample means, and the factor variables are fixed at their base levels.

```
. categraph iateplot educ
Note: IATE estimated at fixed values of covariates other than educ.
```

Variable	Statistic	Value	Type
age	mean	41.05891	continuous
incomecat	base	0	factor
ira	base	0	factor
married	base	0	factor
ownhome	base	0	factor
pension	base	0	factor
twoearn	base	0	factor



`categraph iateplot` plots the prediction of the IATE function at different education levels while holding other variables fixed. It also plots the 95% pointwise confidence interval for each prediction. Below 10 years of education, the effects seem constant. The treatment effects are larger for people with 12 to 14 years of education and then vary for 15 or more years of education. However, with the wide confidence intervals for the IATEs, especially at the higher education levels, we cannot conclude that the IATEs vary across education levels when the other variables are fixed at these levels.

Example 2: Add high-dimensional controls

In the previous example, the control variables for the outcome and the treatment-assignment model coincide with *catevarlist* or the **x** variables. In this example, we want to allow more flexible models for the outcome and the treatment-assignment by adding high-dimensional controls.

The partial linear model with high-dimensional controls is defined as

$$\texttt{asset} = \texttt{e401k} * \tau(\mathbf{x}) + g(\mathbf{x}, \mathbf{w}) + \epsilon$$
$$\texttt{e401k} = f(\mathbf{x}, \mathbf{w}) + u$$

where **w** is a vector of additional control variables. The nuisance functions $g(\mathbf{x}, \mathbf{w})$ and $f(\mathbf{x}, \mathbf{w})$ now depend on both **x** and **w**.

The potential outcomes are now

$$\texttt{asset}(1) = \tau(\mathbf{x}) + g(\mathbf{x}, \mathbf{w}) + \epsilon$$
$$\texttt{asset}(0) = g(\mathbf{x}, \mathbf{w}) + \epsilon$$

Thus, the function $\tau(\mathbf{x})$ identifies the IATE function.

$$\tau(\mathbf{x}) = E\{\texttt{asset}(1) - \texttt{asset}(0)|\mathbf{x}\}$$

Here we want to include the interactions between the continuous variables (`age` and `educ`) and the factor variables (`incomecat`, `pension`, `married`, `twoearn`, `ira`, and `ownhome`) as controls in the outcome and treatment-assignment models. We define a global macro, `controls`, to represent the interaction terms.

```
. global fvars incomecat pension married twoearn ira ownhome
. global controls c.(educ age)#i.($fvars)
```

We add the `controls()` option to our `cate` command to include the additional control variables in both the outcome and the treatment model. We also specify option `nolog` to suppress the iteration log.

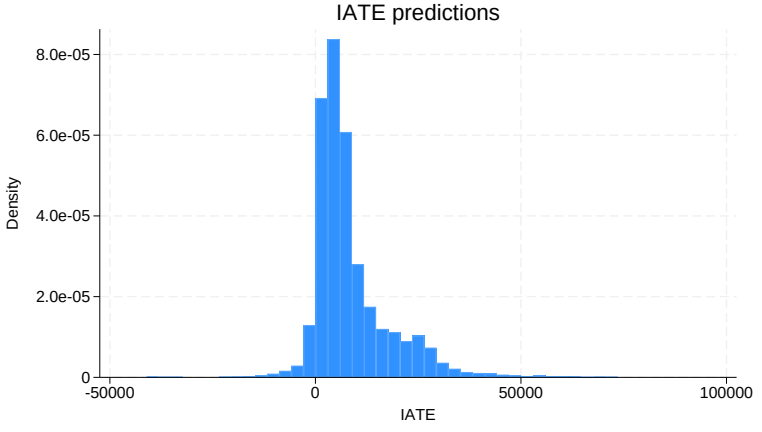
```
. cate po (assets $catecovars) (e401k), rseed(12345671) controls($controls)
> nolog
Conditional average treatment effects     Number of observations       = 9,913
Estimator:       Partialing out           Number of folds in cross-fit =    10
Outcome model:   Linear lasso             Number of outcome controls   =    47
Treatment model: Logit lasso              Number of treatment controls =    47
CATE model:      Random forest            Number of CATE variables     =    17
```

assets	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
e401k (Eligible vs Not elig..)	8107.563	1144.817	7.08	0.000	5863.763	10351.36
POmean						
e401k Not eligi..	13902.88	838.5924	16.58	0.000	12259.27	15546.49

The header shows that there are now 47 control variables for both the outcome and the treatment model. After accounting for these controls, the ATE indicates that if everyone in the population were eligible for a 401(k), the net financial assets would be \$8,108 more than if no one in the population were eligible. This is a larger estimated effect of 401(k) eligibility on financial assets than the ATE of \$7,937 estimated in example 1.

We can again use `categraph histogram` to visualize the distribution of the predicted IATE function.

```
. categraph histogram
(bin=39, start=-41089.941, width=2938.3059)
```



The IATE predictions are primarily positive, and the distribution has a fat right tail. This may imply that the ATE underestimates the treatment effects for some groups in the population. To statistically test if the treatment effects are heterogeneous, we use `estat heterogeneity`.

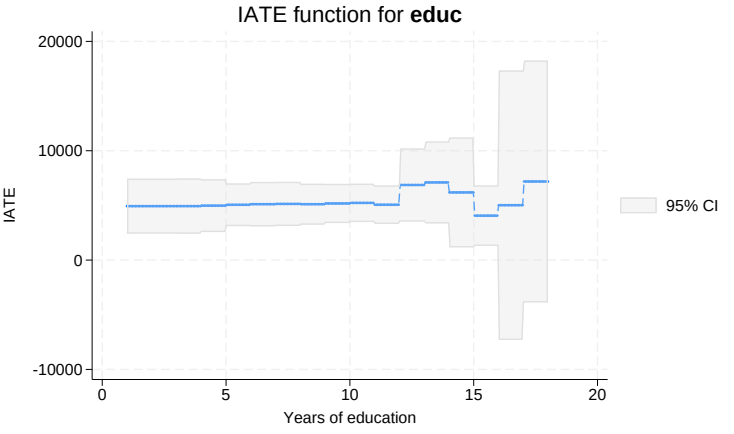
```
. estat heterogeneity

Treatment-effects heterogeneity test
H0: Treatment effects are homogeneous

    chi2(1) =   4.19
Prob > chi2 = 0.0406
```

We find evidence against the null hypothesis that the treatment effects are homogeneous, which is the same conclusion as in example 1.

Finally, we can use `categraph iateplot` to plot the IATE function by allowing one variable to vary and fixing the other variables at some values. We plot the IATE function with respect to the level of education (`educ`).

```
. categraph iateplot educ
Note: IATE estimated at fixed values of covariates other than educ.
```

Variable	Statistic	Value	Type
age	mean	41.05891	continuous
incomecat	base	0	factor
ira	base	0	factor
married	base	0	factor
ownhome	base	0	factor
pension	base	0	factor
twoearn	base	0	factor



The graph shows that the treatment effects are larger for people with 12 to 14 years of education while holding other variables fixed. Again, the confidence intervals for the IATEs are wide, especially at the highest education levels.

Example 3: Estimate the ATEs over prespecified groups

In examples 1 and 2, we learned that the treatment effects of 401(k) eligibility on financial assets are heterogeneous. To further characterize this heterogeneity, we want to know how the ATEs vary across population groups defined by variables such as income category or home ownership.

In general, we refer to a group variable as G and a specific level of the group variable as g. We want to estimate the ATE conditional on belonging to group g, that is, $G = g$. We are interested in estimating

$$\tau(g) = E\{\texttt{asset}(1) - \texttt{asset}(0)|G = g\}$$

The function $\tau(g)$ is referred to as the GATE function. In our case, the first G variable of interest is the income category (`incomecat`). We will consider the home ownership indicator (`ownhome`) later.

We use `table` to report the minimum, maximum, and median income for the five income categories.

```
. table incomecat, stat(min income) stat(max income) stat(median income) nototal
```

	Minimum value	Maximum value	Median
Income category			
0	0	17196	12240
1	17214	26523	21735
2	26526	37275	31482
3	37296	53841	44379
4	53844	242124	69612

Levels 0 and 1 are low-income groups, levels 2 and 3 are middle-income groups, and level 4 is the high-income group.

The GATEs are summaries of the IATE function over the groups defined by variable G. Because we already estimated the IATE function in example 2, there is no need to estimate it again. By specifying the option `reestimate`, we can reuse the IATE function and only reestimate the effects reported, here the GATEs for `incomecat`. With this option, `cate` will require less computational time than estimating the GATEs from scratch. We specify `group(incomecat)` to estimate GATEs for the income categories.

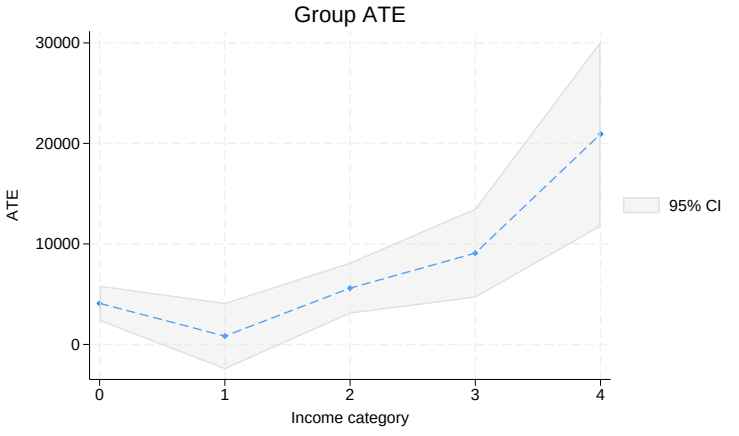
```
. cate, group(incomecat) reestimate
Estimating GATE ...
Conditional average treatment effects     Number of observations       = 9,913
Estimator:       Partialing out           Number of folds in cross-fit =    10
Outcome model:   Linear lasso             Number of outcome controls   =    47
Treatment model: Logit lasso              Number of treatment controls =    47
CATE model:      Random forest            Number of CATE variables     =    17
```

assets	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
GATE						
incomecat						
0	4089.228	900.535	4.54	0.000	2324.212	5854.244
1	830.3422	1687.517	0.49	0.623	-2477.13	4137.815
2	5602.296	1300.555	4.31	0.000	3053.256	8151.336
3	9084.531	2265.143	4.01	0.000	4644.933	13524.13
4	20929.77	4706.377	4.45	0.000	11705.44	30154.1
ATE						
e401k (Eligible vs Not elig..)	8107.563	1144.817	7.08	0.000	5863.763	10351.36
POmean						
e401k Not eligi..	13902.88	838.5924	16.58	0.000	12259.27	15546.49

The results show both the ATE and the GATEs. For example, the GATE estimate for the high-income group (level 4) is \$20,930. For those in the high-income group, being eligible for a 401(k) is expected to increase net financial assets by \$20,930 compared with the net financial assets if not eligible for a 401(k). In contrast, the GATE estimate for the lowest income group (level 0) is only \$4,089. In other words, people who earn more benefit more from working for a company with a 401(k) plan. The ATE estimate indicates that the treatment effects for the population are expected to be \$8,108. The variation in the estimated GATEs across income categories indicates that using the ATE alone does not fully characterize the treatment effects.

We can use `categraph gateplot` to visualize the GATE estimates and see if there is any trend.

```
. categraph gateplot
```



The graph illustrates the upward trend between the income group and the treatment effects.

To further test whether the treatment effects are heterogeneous across income groups, we use `estat gatetest`.

```
. estat gatetest

Group treatment-effects heterogeneity test
H0: Group average treatment effects are homogeneous

 ( 1)  [GATE]0bn.incomecat - [GATE]1.incomecat = 0
 ( 2)  [GATE]0bn.incomecat - [GATE]2.incomecat = 0
 ( 3)  [GATE]0bn.incomecat - [GATE]3.incomecat = 0
 ( 4)  [GATE]0bn.incomecat - [GATE]4.incomecat = 0

    chi2(4) =  21.84
Prob > chi2 = 0.0002
```

We find evidence that the group treatment effects are not homogeneous.

To compare the treatment-effects difference between groups, we use `contrast`.

```
. contrast r.incomecat
warning: cannot perform check for estimable functions.

Contrasts of marginal linear predictions

Margins: asbalanced
```

	df	chi2	P>chi2
GATE			
incomecat			
(1 vs 0)	1	2.90	0.0884
(2 vs 0)	1	0.91	0.3388
(3 vs 0)	1	4.20	0.0404
(4 vs 0)	1	12.35	0.0004
Joint	4	21.84	0.0002

	Contrast	Std. err.	[95% conf.	interval]
GATE				
incomecat				
(1 vs 0)	-3258.886	1912.767	-7007.84	490.0682
(2 vs 0)	1513.068	1581.899	-1587.397	4613.534
(3 vs 0)	4995.303	2437.588	217.7184	9772.887
(4 vs 0)	16840.54	4791.758	7448.869	26232.22

The output shows the difference in each group's ATE compared with the lowest income group (level 0). Except for income group 1, we see that the difference in GATEs increases as income level increases, which corresponds with our conjecture that people who earn more benefit more from being eligible for a 401(k).

Similarly, we can estimate the GATEs for home ownership. We again specify option `reestimate` to reuse the `cate` estimation results and specify option `group(ownhome)` to estimate the GATEs for home ownership categories.

```
. cate, group(ownhome) reestimate
Estimating GATE ...
Conditional average treatment effects     Number of observations       = 9,913
Estimator:       Partialing out           Number of folds in cross-fit =    10
Outcome model:   Linear lasso             Number of outcome controls   =    47
Treatment model: Logit lasso              Number of treatment controls =    47
CATE model:      Random forest            Number of CATE variables     =    17
```

assets	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
GATE						
ownhome						
0	3319.503	795.4385	4.17	0.000	1760.472	4878.534
1	10858.26	1742.672	6.23	0.000	7442.69	14273.84
ATE						
e401k (Eligible vs Not elig..)	8107.563	1144.817	7.08	0.000	5863.763	10351.36
POmean						
e401k Not eligi..	13902.88	838.5924	16.58	0.000	12259.27	15546.49

Among people who own a home, being eligible for a 401(k) is expected to increase their net financial assets by $10,858 compared with the net financial assets if not eligible for a 401(k). This effect is substantially larger than $8,108, the ATE in the population.

We use `estat gatetest` to test whether the GATEs are heterogeneous.

```
. estat gatetest
Group treatment-effects heterogeneity test
H0: Group average treatment effects are homogeneous
 ( 1)  [GATE]0bn.ownhome - [GATE]1.ownhome = 0
    chi2(1) =  15.49
Prob > chi2 = 0.0001
```

We find evidence that the GATEs are not homogeneous.

Finally, we use `contrast` to further quantify the difference of the effects between the groups.

```
. contrast r.ownhome
warning: cannot perform check for estimable functions.
Contrasts of marginal linear predictions
Margins: asbalanced
```

	df	chi2	P>chi2
GATE			
ownhome	1	15.49	0.0001

	Contrast	Std. err.	[95% conf.	interval]
GATE				
ownhome				
(Yes vs No)	7538.76	1915.627	3784.201	11293.32

For homeowners, the GATE of 401(k) eligibility on financial assets is $7,539 more than that for the nonhomeowners.

Example 4: Estimate the ATEs over values of a continuous variable

In example 3, we estimated the ATEs over levels of categorical variables. Sometimes, however, we want to estimate the ATEs over values of a continuous variable. For example, we may want to know how the effects of 401(k) eligibility vary with income (not with income categories).

For a continuous variable Z and a specific value z of variable Z, we are interested in estimating

$$\tau(z) = E\{\texttt{asset}(1) - \texttt{asset}(0)|Z = z\}$$

Semenova and Chernozhukov (2021) proposed the use of nonparametric series regression to approximate the function $\tau(z)$. Specifically, they suggest running a series regression of the AIPW scores estimated in `cate` on the variable Z. To do this after `cate`, we use `estat series`. We specify `income` after `estat series` to indicate that we want to estimate the ATEs over values of `income`. We also specify the `graph` option to plot the estimated function. To reduce the impact of outliers (very high incomes) on estimation, we restrict the sample to incomes less than or equal to $150,000, which is the 99th percentile of incomes in the sample. We also specify the option `knots(5)` to choose five knots in the generated B-spline terms.

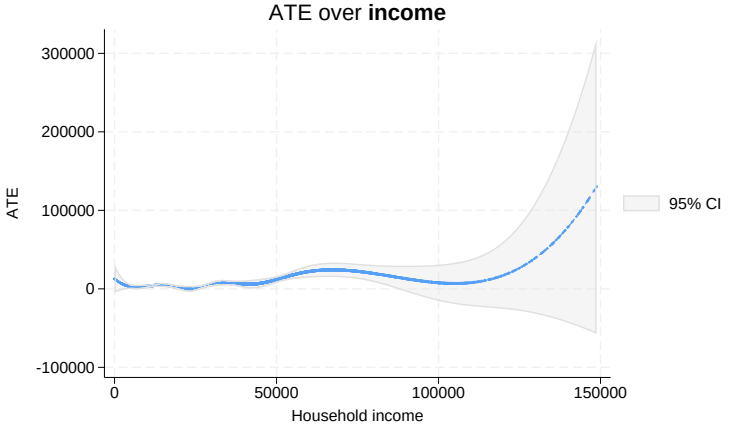
```
. estat series income if income <= 150000, graph knots(5)
Computing approximating function

Computing average derivatives

Nonparametric series regression for IATE
Cubic B-spline estimation                 Number of obs      =          9,884
                                          Number of knots    =              5
```

	Effect	Robust std. err.	z	P>|z|	[95% conf.	interval]
income	.1966117	.0521898	3.77	0.000	.0943216	.2989018

Note: Effect estimates are averages of derivatives.

The estimate shows the marginal effect of income on the 401(k) eligibility treatment effects on the net financial assets. Thus, the average marginal effect of a change in income on the treatment effects is $0.20, indicating that people who earn more benefit more from working in a company with a 401(k) plan.

Because we specified the graph option, we obtain the following graph that illustrates how the treatment effect changes with income.



Each point in the plot corresponds to the estimated ATE at a given income level. It also plots the 95% confidence interval. The graph shows an upward trend between income level and the treatment effects, especially at higher income levels. However, the confidence intervals for the ATEs are also wide at the higher income levels. Compared with the `categraph gateplot` we used for the income category in example 3, the series graph reveals a more nuanced and nonlinear relationship between income and the treatment effects.

Example 5: Use the AIPW estimator

In addition to the PO estimator in a partial linear model, we can also estimate the IATE function in a fully interactive model by using the AIPW estimator. For example, let's estimate the IATE function as in example 2 but using the AIPW estimator.

The IATE function we want to estimate is

$$\tau(\mathbf{x}) = E\{\texttt{asset}(1) - \texttt{asset}(0)|\mathbf{x}\}$$

where $\texttt{asset}(1)$ and $\texttt{asset}(0)$ are the treated and untreated potential outcomes, respectively. $\mathbf{x}$ are the treatment-effects covariates.

The fully interactive model is

$$\begin{aligned}\texttt{asset}(1) &= g_1(\mathbf{x},\mathbf{w}) + \epsilon_1\\ \texttt{asset}(0) &= g_0(\mathbf{x},\mathbf{w}) + \epsilon_2\\ \texttt{e401k} &= f(\mathbf{x},\mathbf{w}_2) + u\end{aligned}$$

where $\mathbf{w}$ and $\mathbf{w}_2$ are vectors of additional control variables for the outcome and treatment models, respectively. By default, $\mathbf{w}_2$ is the same as $\mathbf{w}$, but it can be different if specified. $g_1(\mathbf{x},\mathbf{w})$ and $g_0(\mathbf{x},\mathbf{w})$ are the nonparametric models for the treated and untreated potential outcomes, respectively. ϵ_1 and ϵ_2 are the error terms. $f(\mathbf{x},\mathbf{w}_2)$ is a nonparametric model for the treatment, and u is the error term. The fully interactive model allows the treatment effect to interact with both $\mathbf{x}$ and $\mathbf{w}$; thus, it is more general than the partial linear model. Note that estimating $\tau(\mathbf{x})$ now requires that we also estimate three nuisance functions: $g_1(\mathbf{x},\mathbf{w})$, $g_0(\mathbf{x},\mathbf{w})$, and $f(\mathbf{x},\mathbf{w}_2)$.

We specify aipw after cate to invoke the AIPW estimator. The first portion of the output is

```
. cate aipw (assets $catecovars) (e401k), rseed(12345671) controls($controls)
Cross-fit fold 1 of 10 ...
Estimating lasso for outcome assets if e401k = 0 ...
Estimating lasso for outcome assets if e401k = 1 ...
Performing lasso for treatment e401k ...
        ⋮
  (output omitted)
        ⋮
Cross-fit fold 10 of 10 ...
Estimating lasso for outcome assets if e401k = 0 ...
Estimating lasso for outcome assets if e401k = 1 ...
Performing lasso for treatment e401k ...
```

The iteration log shows that two potential outcome models and a treatment model are fit using ten-fold cross-fitting. By default, the potential outcome models are estimated using a linear lasso and the treatment model is estimated with a logit lasso.

The remaining portion of the output is

```
Estimating AIPW scores ...
Estimating random forest for IATE ...
Estimating ATE ...
Conditional average treatment effects     Number of observations       = 9,913
Estimator:       Augmented IPW            Number of folds in cross-fit =    10
Outcome model:   Linear lasso             Number of outcome controls   =    47
Treatment model: Logit lasso              Number of treatment controls =    47
CATE model:      Random forest            Number of CATE variables     =    17
```

assets	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
e401k (Eligible vs Not elig..)	8164.364	1151.125	7.09	0.000	5908.2	10420.53
POmean						
e401k Not eligi..	13910.87	842.0945	16.52	0.000	12260.39	15561.34

We see that random forest is used to estimate the IATE function once the cross-fitting is finished. Then the AIPW scores implied by the fully interactive model are computed, and the ATE is an average of the AIPW scores. The estimated ATE indicates that if everyone in the population is eligible for a 401(k), the net financial assets will, on average, be $8,164 more than if no one in the population is eligible for a 401(k). The $8,164 ATE estimate is similar to the $8,108 result in example 2.

Both the PO and the AIPW estimators are Neyman orthogonal, implying that the ATE estimation results are robust in response to the machine learning estimation errors in the outcome and the treatment model. However, the AIPW estimator is asymptotically more efficient than the PO estimator (see Kennedy [2023]). In addition, the AIPW estimator enjoys a doubly robust property, meaning that only one of the outcome model or the treatment-assignment model needs to be correctly specified to consistently estimate the ATE (see Chernozhukov et al. [2018]).

Another advantage of the AIPW estimator over the PO estimator is that it allows us to use different control variables in the outcome and the treatment model. For example, suppose we want to add the square of age as an additional control in the treatment-assignment model. In `cate`, we specify the option `treatcontrols($controls c.age#c.age)` to use `c.age#c.age` as the additional control variable in the treatment model.

```
. cate aipw (assets $catecovars) (e401k), rseed(12345671) controls($controls)
> treatcontrols($controls c.age#c.age) nolog
Conditional average treatment effects     Number of observations       = 9,913
Estimator:       Augmented IPW            Number of folds in cross-fit =    10
Outcome model:   Linear lasso             Number of outcome controls   =    47
Treatment model: Logit lasso              Number of treatment controls =    48
CATE model:      Random forest            Number of CATE variables     =    17
```

assets	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
e401k (Eligible vs Not elig..)	8164.364	1151.125	7.09	0.000	5908.2	10420.53
POmean						
e401k Not eligi..	13910.87	842.0945	16.52	0.000	12260.39	15561.34

The ATE estimate does not change in this case even though the outcome model has one extra control variable. The results in this example imply that our results are robust to a different control specification.

We can use the same set of postestimation commands to explore, summarize, and visualize the treatment effects after AIPW estimation as we did after PO estimation. The conclusions for our example would be similar; thus, we will not repeat those postestimation commands here.

Example 6: Data-driven group hypothesis testing

In example 3, we summarized the heterogeneous treatment effects by estimating the ATEs for the prespecified groups of income category and home ownership. We emphasize that these group variables must be prespecified before the data collection and analysis to avoid p-value hacking as discussed in Head et al. (2015). This scenario is suitable when researchers know a priori across which groups they would like to explore treatment-effect heterogeneity.

However, we sometimes do not know which variable is linked to the heterogeneity of treatment effects; we want the data to discover these variables. Chernozhukov et al. (2006) suggest ranking the treatment effects first and then performing a classification analysis based on the groups induced by the treatment-effects ranking.

For example, suppose we estimate the IATE of 401(k) eligibility on net financial assets for each observation, and we divide the data into four groups based on the IATE prediction's ranking. The first group is the people in the top 25% of the treatment effects in the data, and the last group is the people in the bottom 25% of the treatment effects in the data. We want to know whether the mean income differs for the groups with the largest and the smallest treatment effects.

In cate, we specify the group(#) option to rank the data based on the estimated IATEs. In the following example, we specify group(4) to divide the data into four groups based on the IATE ranking. We also specify the option xfolds(5) to use five-fold cross-fitting. The first portion of the output is

```
. cate po (assets $catecovars) (e401k), rseed(12345671) controls($controls)
> group(4) xfolds(5)
Cross-fit fold 1 of 5 ...
Performing lasso for outcome assets ...
Performing lasso for treatment e401k ...
Estimating IATE rankings ...
Estimating AIPW scores ...
        ⋮
  (output omitted)
        ⋮
Cross-fit fold 5 of 5 ...
Performing lasso for outcome assets ...
Performing lasso for treatment e401k ...
Estimating IATE rankings ...
Estimating AIPW scores ...
```

The iteration logs show that the IATE rankings are computed using cross-fitting. This procedure is necessary because it avoids the overfitting issues by using one sample to estimate the IATE function and a different sample to predict the IATE function.

The remaining output is as follows:

```
Performing random forest for IATE ...
Estimating AIPW scores ...
Estimating sorted GATE ...
Conditional average treatment effects     Number of observations       = 9,913
Estimator:       Partialing out           Number of folds in cross-fit =     5
Outcome model:   Linear lasso             Number of outcome controls   =    47
Treatment model: Logit lasso              Number of treatment controls =    47
CATE model:      Random forest            Number of CATE variables     =    17
```

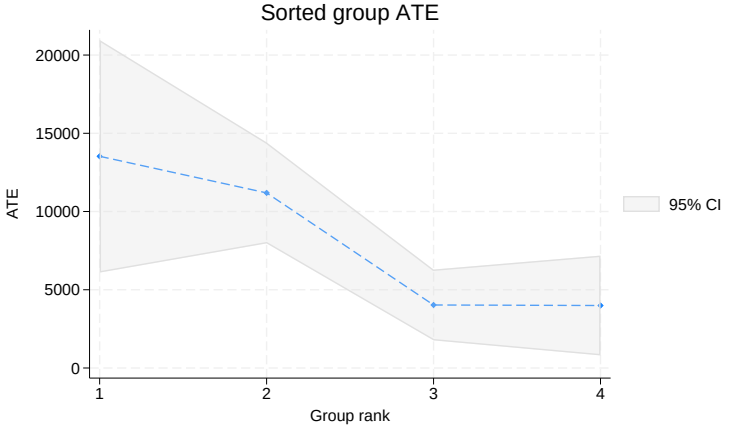
assets	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
GATES						
rank						
1	13529.88	3792.728	3.57	0.000	6096.266	20963.49
2	11190.14	1646.548	6.80	0.000	7962.962	14417.31
3	4026.967	1154.876	3.49	0.000	1763.452	6290.481
4	3993.897	1627.408	2.45	0.014	804.236	7183.558
ATE						
e401k (Eligible vs Not elig..)	8183.327	1148.204	7.13	0.000	5932.888	10433.77
POmean						
e401k Not eligi..	13881.65	840.706	16.51	0.000	12233.89	15529.4

The remaining output shows the ATEs for the groups sorted by IATE predictions, also known as the GATESs. The GATES for the group with the largest treatment effects is $13,530. The results show a substantial difference of GATESs estimates between the most and the least affected groups.

In some cases, however, the highest-ranking group does not necessarily have a greater GATES estimate than the lowest-ranking group because the rankings are generated using cross-fitting to avoid overfitting. The rankings depend on the IATE estimates' quantiles in each cross-fitting fold but not the whole sample. Thus, an observation with a higher ranking implies only that this observation has greater IATE estimates compared with other observations in a particular fold; it does not necessarily mean that it has greater IATE estimates compared with the full sample. If the treatment effects are genuinely homogeneous, we would observe GATES estimates that are similar across ranking levels. See *Methods and formulas* for details.

We can visualize the GATES estimates using `categraph gateplot`.

```
. categraph gateplot
```



To test whether the treatment effects are homogeneous across the group, we use `estat gatetest`. We specify levels 1 and 4 to compare the groups with the largest and smallest effects.

```
. estat gatetest 1 4

Sorted group treatment-effects heterogeneity test
H0: Sorted group average treatment effects are homogeneous

 ( 1)  [GATES]1bn.rank - [GATES]4.rank = 0

    chi2(1) =   5.34
Prob > chi2 = 0.0209
```

We find evidence that the GATESs are not homogeneous across the groups with the largest and smallest effects.

We can use `estat classification` to test whether the means of some variables are different between the groups with the largest and smallest effects. For example, we can test whether the income level has different means in groups ranked 1 and 4.

```
. estat classification income
Classification t test with equal variances
```

Group	Obs	Mean	Std. err.	Std. dev.	[95% conf.	interval]
1	2,480	62522.61	513.3117	25562.71	61516.05	63529.17
4	2,475	26420.44	367.1063	18263.31	25700.57	27140.31
Combined	4,955	44489.74	406.6722	28626.37	43692.48	45287
diff		36102.17	631.2817		34864.58	37339.76

```
    diff = mean(1) - mean(4)                                      t =  57.1887
H0: diff = 0                                     Degrees of freedom =     4953

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 1.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 0.0000
```

The income levels are higher in the group with the largest effects than in the group with the smallest effects.

We can do a similar classification analysis for `age` and `ownhome`.

```
. estat classification ownhome
Classification t test with equal variances
```

Group	Obs	Mean	Std. err.	Std. dev.	[95% conf.	interval]
1	2,480	.8584677	.0070009	.3486401	.8447396	.8721959
4	2,475	.4056566	.0098718	.4911179	.3862986	.4250145
Combined	4,955	.6322906	.0068507	.4822304	.6188603	.645721
diff		.4528112	.0120983		.4290932	.4765291

```
    diff = mean(1) - mean(4)                                      t =  37.4278
H0: diff = 0                                     Degrees of freedom =     4953

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 1.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 0.0000

. estat classification age
Classification t test with equal variances
```

Group	Obs	Mean	Std. err.	Std. dev.	[95% conf.	interval]
1	2,480	45.16815	.1808494	9.006225	44.81351	45.52278
4	2,475	35.26747	.2175367	10.82231	34.8409	35.69405
Combined	4,955	40.22281	.1579318	11.1171	39.91319	40.53242
diff		9.90067	.2828416		9.346176	10.45517

```
    diff = mean(1) - mean(4)                                      t =  35.0043
H0: diff = 0                                     Degrees of freedom =     4953

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 1.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 0.0000
```

More people in group 1 own homes than in group 4, and people are older in group 1 than people in group 4.

The analysis suggests that individuals who are older, have a home, and have a higher income see a more substantial effect of 401(k) eligibility on net financial assets. Without any ex-ante assumptions about the effect of eligibility, we can learn from the data which subpopulations defined by covariates would benefit more with greater access to a 401(k).

Finally, we can specify the `group(#)` option with `reestimate` if we want to divide the data into a different number of ranking groups but want to avoid recomputing the IATE function. It is much faster than estimating GATESs from scratch. Here, we specify the `group(2)` option to divide the data into two ranking levels.

```
. cate, group(2) reestimate
Estimating sorted GATE ...
Conditional average treatment effects     Number of observations       = 9,913
Estimator:       Partialing out           Number of folds in cross-fit =     5
Outcome model:   Linear lasso             Number of outcome controls   =    47
Treatment model: Logit lasso              Number of treatment controls =    47
CATE model:      Random forest            Number of CATE variables     =    17
```

assets	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
GATES						
rank						
1	12360.48	2067.996	5.98	0.000	8307.28	16413.68
2	4010.449	997.4399	4.02	0.000	2055.502	5965.395
ATE						
e401k (Eligible vs Not elig..)	8183.327	1148.204	7.13	0.000	5932.888	10433.77
POmean						
e401k						
Not eligi..	13881.65	840.706	16.51	0.000	12233.89	15529.4

Example 7: Flexible models

In the CATE estimation, we need to specify the estimation methods in three different models: the outcome model, the treatment-assignment model, and the CATE model. The outcome and treatment models are the nuisance parameters, which we are not interested in making inferences about. The CATE model is the object of interest, and we want to make inferences on IATEs, ATEs, GATEs, and GATESs.

In the previous examples, we used the lasso linear model for the outcome model, the lasso logit model for the treatment model, and the random forest for the IATE function $\tau(\mathbf{x})$, the default in `cate`. Sometimes, however, we want to use different techniques to explore data based on different assumptions. For the outcome and treatment models, we can use a semi-parametric method such as lasso, a nonparametric method such as random forest, or a purely parametric method such as linear or logistic regression. For the IATE function, we can use either a nonparametric method, such as random forest, or a parametric method, such as linear regression. We can try different models to see how sensitive the results are to the modeling methods and assumptions.

In the case of our 401(k) eligibility example, we will try the parametric models, which are estimated using parametric methods, and the nonparametric models, which are estimated using random forest.

Parametric models like linear regression are easier to compute and interpret than nonparametric models such as a random forest. However, the assumptions of a parametric model are less likely to be satisfied. Nevertheless, we can use it as a benchmark. For example, suppose we assume a linear model for the outcome, a logit model for the treatment, and the linear model for the IATE. Under these assumptions, the IATE function is

$$\tau(\mathbf{x}) = \mathbf{x}'\beta$$

Thus, the outcome and treatment models under these parametric assumptions become

$$\texttt{asset} = \texttt{e401k} * (\mathbf{x}'\beta) + \mathbf{x}'\gamma_1 + \epsilon$$

$$\Pr(\texttt{e401k} = 1|\mathbf{x}) = \frac{\exp(\mathbf{x}'\gamma_2)}{1 + \exp(\mathbf{x}'\gamma_2)}$$

In `cate`, we specify the `omethod(regress)` option to use linear regression for the outcome, the `tmethod(logit)` option to use a logit model for the treatment, and the `cmethod(regress)` option to use linear regression for the IATE function.

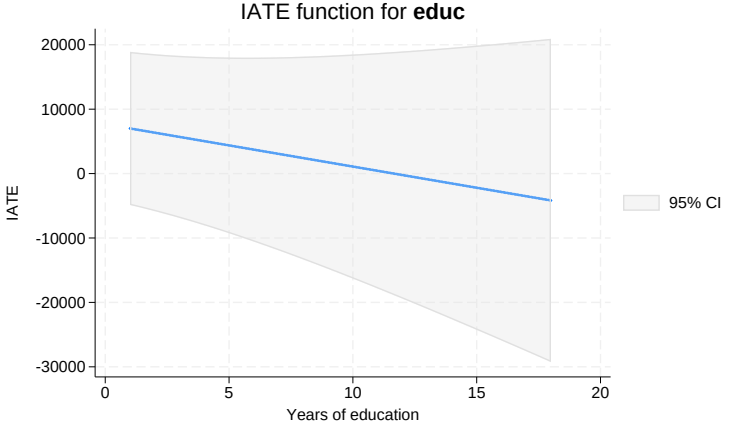
```
. cate po (assets $catecovars) (e401k), rseed(12345671)
> omethod(regress) tmethod(logit) cmethod(regress) nolog

Conditional average treatment effects     Number of observations       = 9,913
Estimator:       Partialing out           Number of folds in cross-fit =    10
Outcome model:   Linear regression        Number of outcome controls   =    17
Treatment model: Logit                    Number of treatment controls =    17
CATE model:      Linear regression        Number of CATE variables     =    17
```

assets	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
e401k (Eligible vs Not elig..)	7904.218	1155.565	6.84	0.000	5639.351	10169.08
POmean						
e401k Not eligi..	13977.45	831.0932	16.82	0.000	12348.54	15606.37

We can interpret the results as before. If we plot the IATE function over one variable, such as educ, we see a straight line, which is expected due to the parametric assumption on the IATE function $\tau(\mathbf{x})$.

```
. categraph iateplot educ
Note: IATE estimated at fixed values of covariates other than educ.
```

Variable	Statistic	Value	Type
age	mean	41.05891	continuous
incomecat	base	0	factor
ira	base	0	factor
married	base	0	factor
ownhome	base	0	factor
pension	base	0	factor
twoearn	base	0	factor



Interestingly, the graph shows a downward trend between the years of education and the IATEs, and the estimates of the IATEs are not different from zero (the confidence intervals include zero for all education levels). This conclusion is very different from that in examples 1 and 2 when we use the random forest to estimate the IATE function. It may imply that the parametric assumptions on the IATE function are too strong.

In contrast to the pure parametric model, we can use the random forest in the outcome, the treatment, and the IATE models. Random forest allows us to model flexibly without imposing restrictive assumptions. Another advantage is that we can use the out-of-bag predictions from the random forest to avoid using cross-fitting, which may be time consuming.

We specify the `omethod(rforest)` and `tmethod(rforest)` options to use the random forest model for both the outcome and the treatment models. The default method for the IATE estimation is already a random forest, so we do not need to specify the `cmethod()` option here. In addition, to use the out-of-bag prediction-based algorithm, we specify the `oob` option.

```
. cate po (assets $catecovars) (e401k), rseed(12345671)
> omethod(rforest) tmethod(rforest) oob

Performing random forest for outcome assets ...
Performing random forest for treatment e401k ...
Performing random forest for IATE ...
Estimating AIPW scores ...
Estimating ATE ...

Conditional average treatment effects     Number of observations       = 9,913
Estimator:       Partialing out           Number of folds in cross-fit =     1
Outcome model:   Random forest            Number of outcome controls   =    17
Treatment model: Random forest            Number of treatment controls =    17
CATE model:      Random forest            Number of CATE variables     =    17
```

assets	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
e401k (Eligible vs Not elig..)	8225.258	1173.862	7.01	0.000	5924.53	10525.99
POmean						
e401k Not eligi..	14016.34	850.1257	16.49	0.000	12350.13	15682.56

The results are a little different from those in examples 1 and 2. This is expected because, in examples 1 and 2, we used the lasso methods for the outcome and treatment models.

We can test whether the treatment effects are heterogeneous using `estat heterogeneity`.

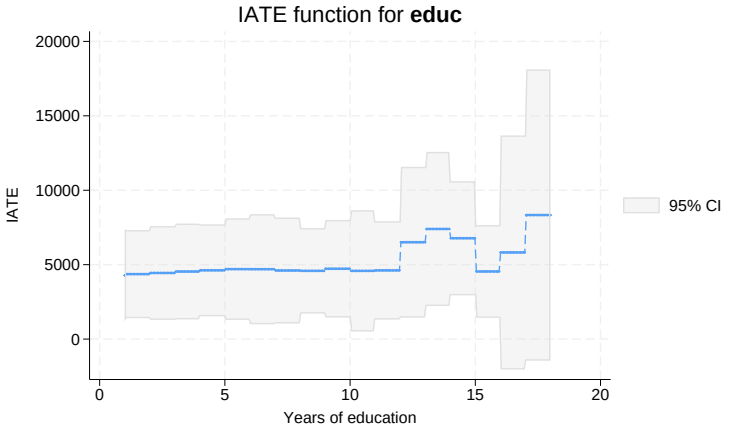
```
. estat heterogeneity

Treatment-effects heterogeneity test
H0: Treatment effects are homogeneous

    chi2(1) =   4.07
Prob > chi2 = 0.0437
```

We find evidence that the treatment effects are not homogeneous.

To compare with the parametric model, we can also plot the IATE function with respect to education (`educ`).

```
. categraph iateplot educ

Note: IATE estimated at fixed values of covariates other than educ.
```

Variable	Statistic	Value	Type
age	mean	41.05891	continuous
incomecat	base	0	factor
ira	base	0	factor
married	base	0	factor
ownhome	base	0	factor
pension	base	0	factor
twoearn	base	0	factor



Compared with the parametric IATE function, the graph shows a nonlinear form, and the treatment effects are more substantial for people with 12 to 15 years of education while holding other variables fixed.

The random forest method puts less restrictive assumptions on the data generating process (DGP) than the lasso method, and thus, random forest may model the real-world data better. However, the caveats are that the random forest can not handle high-dimensional controls like lasso, and it takes much longer to compute when there are many observations in the data. In contrast, lasso can be considered a semiparametric model by approximating a function using a set of basis functions. It can approximate the DGP reasonably well when the underlying function is sparse, meaning only a few terms among high-dimensional controls have nonzero coefficients.

Example 8: Treatment-assignment policy evaluation

In examples 1 to 7, we use `cate` to study the treatment-effects heterogeneity from different perspectives, such as IATEs, GATEs, and GATESs. Sometimes, researchers are not interested in the treatment-effects heterogeneity itself but instead want to use the estimated treatment effects to evaluate a treatment-assignment policy.

We want to evaluate the policy by answering questions such as the following:

1. If we implement such a policy, what is the average outcome of the population?

2. If we have alternative policies, which one is better?

For the first question, we compute the average outcome if the treatment is assigned according to the policy, which is also known as the policy's value. Precisely, the value of the treatment-assignment policy, $\Pi(\pi)$, is defined as

$$\Pi(\pi) = E\{\pi_i y_i(1) + (1 - \pi_i)\, y_i(0)\}$$

where $\pi_i \in [0, 1]$ is a prespecified treatment-assignment probability for the ith observation. Thus, π_i is referred to as the policy. $y_i(1)$ and $y_i(0)$ are the potential outcomes for being treated or not treated, respectively.

For the second question, we compute the value difference between the two policies,

$$\Pi(\pi_A) - \Pi(\pi_B)$$

where π_A and π_B are two different treatment-assignment policies.

We illustrate how to use `cate` to evaluate a hypothetical treatment-assignment policy that assigns patients to two types of lung transplants. Bilateral lung transplant (BLT) is usually associated with a higher death rate in the short term after the operation but with a more significant improvement in the quality of life compared with a single lung transplant (SLT). Suppose a doctor has a simple treatment-assignment rule, which assigns a patient to BLT if the patient's walking distance is greater than 500 meters in six minutes and if the patient does not have diabetes. The doctor wants to evaluate this policy by answering the following two questions:

1. What would be the average outcome if this policy is implemented?

2. Is this policy better than the treatment assignment observed in the data?

We have a fictional dataset (`lung.dta`) inspired by Koch, Vock, and Wolfson (2018). An individual's forced expiratory volume in one second (FEV1) measures a patient's quality of life. The outcome of interest is the percentage of FEV1 that a patient has relative to a healthy person with similar characteristics, FEV1% (`fev1p`), measured one year after the operation. The treatment variable (`transtype`) indicates whether the treatment is BLT or SLT.

To open the dataset and describe it, we type

```
. use https://www.stata-press.com/data/r19/lung, clear
(Fictional data on lung transplant)
. describe *, short
```

Variable name	Storage type	Display format	Value label	Variable label
agep	byte	%10.0g		Patient age (years)
bmip	double	%10.0g		Patient body mass index
diabetesp	byte	%12.0g	lbdiab	Patient diabetes status
heightp	double	%10.0g		Patient height (cm)
o2amt	double	%10.0g		Oxygen delivered
karn	byte	%8.0g	lbyes	Karnofsky score > 60
lungals	double	%10.0g		Lung allocation score
racep	byte	%8.0g	lbrace	Patient race
sexp	byte	%8.0g	lbsex	Patient gender
lifesvent	byte	%8.0g	lbyes	Life support ventilator needed
assisvent	byte	%8.0g	lbyes	Assisted ventilation needed
centervol	double	%10.0g		Center volume
walkdist	double	%10.0g		Walking distance in 6 minutes
o2rest	byte	%8.0g	lbyes	Oxygen needed at rest
aged	byte	%10.0g		Donor age (years)
raced	byte	%8.0g	lbrace	Donor race
bmid	double	%10.0g		Donor body mass index
smoked	byte	%8.0g	lbyes	Donor if has history of smoking
cmv	byte	%8.0g	lbyes	Positive cytomegalovirus test
deathcause	byte	%8.0g	lbyes	Cause of death - traumatic brain injury
diabetesd	byte	%12.0g	lbdiab	Donor diabetes status
expandd	byte	%8.0g	lbyes	Expanded donor needed
heightd	double	%10.0g		Donor height (cm)
sexd	byte	%8.0g	lbsex	Donor gender
distd	int	%10.0g		Donor to treatment center distance
lungpo2	double	%10.0g		Lung PO2
lungalloc	byte	%8.0g	lballo	Lung allocation status
hratio	double	%10.0g		Height ratio
ischemict	double	%10.0g		Ischemic time
genderm	byte	%19.0g	lbgm	Matching gender status
racem	byte	%17.0g	lbrm	Matching race status
transtype	byte	%8.0g	lbtau	Lung transplant type
fev1p	double	%10.0g		Percentage of predicted value of FEV1

Thirty-one variables measure characteristics of the patients and donors. To construct *catevarlist* and the control variables, we want to use these variables and the interactions among them. The following commands create global macro `catecovars` to represent the covariates in the IATE function and global macro `controls` to represent the additional control variables in the outcome and treatment models.

```
. global cvars bmip heightp o2amt lungals centervol walkdist bmid heightd
> distd lungpo2 hratio ischemict
. global fvars diabetesp karn racep sexp lifesvent assisvent o2rest raced
> smoked cmv deathcause diabetesd expandd sexd lungalloc genderm racem
. global catecovars c.($cvars) i.($fvars)
. global controls c.($cvars)#i.($fvars)
```

The hypothetical policy variable policy1 assigns a patient to BLT if the patient's walking distance is more than 500 meters and if the patient does not have diabetes.

```
. generate policy1 = walkdist > 500 & !diabetesp & !missing(walkdist)
```

To evaluate policy1, we first need to use cate and estimate the potential outcomes for each individual.

```
. cate aipw (fev1p $catecovars) (transtype), rseed(12345671) controls($controls)
> nolog
Conditional average treatment effects       Number of observations       = 937
Estimator:       Augmented IPW              Number of folds in cross-fit =  10
Outcome model:   Linear lasso               Number of outcome controls   = 454
Treatment model: Logit lasso                Number of treatment controls = 454
CATE model:      Random forest              Number of CATE variables     =  46
```

fev1p	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
transtype (BLT vs SLT)	37.5243	.1646795	227.86	0.000	37.20153	37.84707
POmean						
transtype SLT	46.49502	.2025403	229.56	0.000	46.09805	46.892

The FEV1% if all the patients were to be assigned to BLT is expected to be 38 percentage points higher than the 46% average expected if all patients were to be assigned to SLT.

The ATE is a special version of policy evaluation. The ATE estimates the difference in average outcomes between the two policies: everyone is treated versus everyone is untreated. Here is an illustration.

We create the variable treatall representing a policy that assigns each patient to BLT treatment. In contrast, variable treatnone represents a policy that assigns each patient to SLT treatment.

```
. generate treatall = 1
. generate treatnone = 0
```

We can use estat policyeval to compare these treatment policies.

```
. estat policyeval treatall treatnone
Treatment-assignment policy evaluation                    Number of obs = 937
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
Value						
policy						
treatall	84.01932	.3085432	272.31	0.000	83.41459	84.62406
treatnone	46.49502	.2025403	229.56	0.000	46.09805	46.892
Contrast						
policy						
(treatall vs treatnone)	37.5243	.1646795	227.86	0.000	37.20153	37.84707

The value of a policy is the average outcome if the policy is implemented. For example, the value of treatnone is 46.5, which means that the expected FEV1% is 46.5% if all patients are assigned to SLT. By definition, the value of treatnone corresponds to the potential outcome mean if the treatment status is SLT, and in the cate output, the POmean for SLT is indeed 46.5. That is,

$$\Pi(\texttt{treatnone}) = E\{0 * \texttt{fev1p}(1) + 1 * \texttt{fev1p}(0)\} = E\{\texttt{fev1p}(0)\}$$

Similarly, the value of treatall is the potential outcome if all patients are assigned to BLT. That is,

$$\Pi(\texttt{treatall}) = E\{1 * \texttt{fev1p}(1) + 0 * \texttt{fev1p}(0)\} = E\{\texttt{fev1p}(1)\}$$

The contrast is the difference in the values of the two policies. The contrast between treatall and treatnone is 37.5, which means that the FEV1% is 37.5% higher if all patients are assigned to BLT over SLT. By definition, the contrast between treatall and treatnone is the ATE, and the ATE estimate in the cate output is indeed 37.5. That is,

$$\Pi(\texttt{treatall}) - \Pi(\texttt{treatnone}) = E\{\texttt{fev1p}(1) - \texttt{fev1p}(0)\} = \text{ATE}$$

Now we are ready to evaluate the hypothetical policy policy1, which assigns a patient to BLT if the patient's walking distance is greater than 500 meters and the patient does not have diabetes.

```
. estat policyeval policy1
Treatment-assignment policy evaluation                    Number of obs = 937
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
Value						
policy						
policy1	72.66426	.714435	101.71	0.000	71.26399	74.06452

The average FEV1% is 72.6% if this policy is implemented. We can compare this with the observed treatment policy (transtype).

```
. estat policyeval policy1 transtype
Treatment-assignment policy evaluation                    Number of obs = 937
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
Value policy						
policy1	72.66426	.714435	101.71	0.000	71.26399	74.06452
transtype	66.53891	.5149955	129.20	0.000	65.52954	67.54828
Contrast policy (policy1 vs transtype)	6.125348	.9130896	6.71	0.000	4.335725	7.91497

The average FEV1% is 6.1% higher if policy1 is implemented compared with the actual treatment-assignment policy transtype.

Stored results

cate stores the following in e():

Scalars

e(N)	number of observations
e(n_xfolds)	number of folds for cross-fitting
e(k_controls_om)	number of controls in the outcome model
e(k_controls_tm)	number of controls in the treatment model
e(k_cate_covars)	number of covariates in the CATE model
e(gateslevel)	number of sub-levels in the data-driven group if group(#) specified
e(samprate_om)	sampling rate for outcome model, if omethod(rforest) specified
e(ntrees_om)	number of trees for outcome model, if omethod(rforest) specified
e(cintrees_om)	number of trees in each group for outcome model, if omethod(rforest) specified
e(splitminobs_om)	minimum number of observations to split a node for outcome model, if omethod(rforest) specified
e(splitmeanvars_om)	mean number of variables to be split in each node for outcome model, if omethod(rforest) specified
e(honestrate_om)	sampling rate for honest tree for outcome model, if omethod(rforest) specified
e(samprate_tm)	sampling rate for treatment model, if tmethod(rforest) specified
e(ntrees_tm)	number of trees for treatment model, if tmethod(rforest) specified
e(cintrees_tm)	number of trees in each group for treatment model, if tmethod(rforest) specified
e(splitminobs_tm)	minimum number of observations to split a node for treatment model, if tmethod(rforest) specified
e(splitmeanvars_tm)	mean number of variables to be split in each node for treatment model, if tmethod(rforest) specified
e(honestrate_tm)	sampling rate for honest tree for treatment model, if tmethod(rforest) specified
e(samprate_cm)	sampling rate for CATE model, if cmethod(rforest) specified
e(ntrees_cm)	number of trees for CATE model, if cmethod(rforest) specified
e(cintrees_cm)	number of trees in each bag for CATE model, if cmethod(rforest) specified
e(splitminobs_cm)	minimum number of observations to split a node for CATE model, if cmethod(rforest) specified
e(splitmeanvars_cm)	mean number of variables to be split in each node for CATE model, if cmethod(rforest) specified
e(honestrate_cm)	sampling rate for honest tree for CATE model, if cmethod(rforest) specified

Macros

e(cmd)	cate
e(cmdline)	command as typed
e(depvar)	name of outcome variable
e(tvar)	name of treatment variable
e(gate_var)	GATE variable
e(cate_covars)	name of CATE covariates
e(covars_om)	covariates for outcome model
e(covars_tm)	covariates for treatment model
e(lasso_selection_om)	lasso selection method for outcome model
e(lasso_selection_tm)	lasso selection method for treatment model
e(estimator)	name of estimator
e(omethod)	estimation method for outcome model
e(tmethod)	estimation method for treatment model
e(cmethod)	estimation method for CATE model
e(title)	title in estimation output
e(predict)	program used to implement predict
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(rngstate)	random-number state used
e(properties)	b V
e(marginsnotok)	predictions disallowed by margins

Matrices

e(b)	coefficient vector
e(V)	variance–covariance matrix of the estimators

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

The methods and formulas are presented under the following headings:

We can group the estimation methods by the outcome model. The outcome model can be expressed as a partial linear or fully interactive model. For the partial linear model, the estimation method is the PO estimator proposed in Nie and Wager (2021). The fully interactive model's estimation methods are

built around the AIPW estimator discussed in Knaus (2022) and Kennedy (2023). We will briefly discuss the features of these two estimators and compare them. For notational simplicity, we drop the subscript i indicating the ith observation to refer to a random variable.

PO for the partial linear model

The partial linear model is

$$y = d * \tau(\mathbf{x}) + g(\mathbf{x}, \mathbf{w}) + \epsilon$$
$$d = f(\mathbf{x}, \mathbf{w}) + u$$

where y is the outcome variable, d is a binary treatment, $\mathbf{x}$ is a vector of covariates for the IATE function, $\mathbf{w}$ are optional controls for both the outcome and the treatment models, and $g(\mathbf{x}, \mathbf{w})$ and $f(\mathbf{x}, \mathbf{w})$ are nuisance functions for the outcome and the treatment, respectively. ϵ and u are the error terms.

Let $y(1)$ be the potential outcome when the unit is treated and $y(0)$ be the potential outcome when the unit is not treated. By definition, the potential outcomes are

$$y(1) = \tau(\mathbf{x}) + g(\mathbf{x}, \mathbf{w}) + \epsilon$$
$$y(0) = g(\mathbf{x}, \mathbf{w}) + \epsilon$$

Thus, the IATE function is $\tau(\mathbf{x}) = E\{y(1) - y(0)|\mathbf{x}\}$.

To estimate $\tau(\mathbf{x})$, we need to partial-out the nuisance functions $g(\mathbf{x}, \mathbf{w})$ and $f(\mathbf{x}, \mathbf{w})$. To do this, we need to construct residuals of the outcome and the treatment that are independent of $\mathbf{w}$ and $\mathbf{x}$. First, we take the expectation of y conditional on $\mathbf{x}$ and $\mathbf{w}$. That is,

$$E(y|\mathbf{x}, \mathbf{w}) = f(\mathbf{x}, \mathbf{w})\tau(\mathbf{x}) + g(\mathbf{x}, \mathbf{w})$$

Note that unconfoundedness implies that $E(\epsilon|\mathbf{x}, \mathbf{w}) = E(u|\mathbf{x}, \mathbf{w}) = 0$, and therefore we have $E(d|\mathbf{x}, \mathbf{w}) = f(\mathbf{x}, \mathbf{w})$. Thus, subtracting $E(y|\mathbf{x}, \mathbf{w})$ from y removes the term $g(\mathbf{x}, \mathbf{w})$.

$$y - E(y|\mathbf{x}, \mathbf{w}) = \{d - f(\mathbf{x}, \mathbf{w})\}\tau(\mathbf{x}) + \epsilon$$

We can estimate $E(y|\mathbf{x}, \mathbf{w})$ and $f(\mathbf{x}, \mathbf{w})$ via lasso, random forest, or parametric regression. Then we use the residuals to estimate $\tau(\mathbf{x})$. Let $h(\mathbf{x}, \mathbf{w}) = E(y|\mathbf{x}, \mathbf{w})$, and let $\hat{h}(\mathbf{x}, \mathbf{w})$ and $\hat{f}(\mathbf{x}, \mathbf{w})$ be the estimates for $h(\mathbf{x}, \mathbf{w})$ and $f(\mathbf{x}, \mathbf{w})$, respectively. We construct the PO version of y and d as

$$\widetilde{y} = y - \hat{h}(\mathbf{x}, \mathbf{w})$$
$$\widetilde{d} = d - \hat{f}(\mathbf{x}, \mathbf{w})$$

One way we can estimate $\tau(\mathbf{x})$ is by solving the local moment condition

$$\sum_{i=1}^{N} \left[\alpha(x_i)\widetilde{d}_i \left\{\widetilde{y}_i - \widetilde{d}_i\tau(\mathbf{x})\right\}\right] = 0$$

where $\alpha(x_i)$ defines the local weights that attach more weight to observations that are close to $\mathbf{x}$. We use the generalized random forest proposed in Athey, Tibshirani, and Wager (2019) to solve this moment condition. For details on causal forest, see *Generalized random forest*.

Another way to estimate $\tau(\mathbf{x})$ is by assuming a linear functional form for the CATE, $\tau(\mathbf{x}) = \mathbf{x}\beta$, and estimating β using linear regression.

Next we discuss the PO estimator for IATEs, GATEs, and GATESs.

PO IATE estimator

In practice, the partialed-out residuals $\tilde{d}$ and $\tilde{y}$ are constructed using the out-of-sample prediction. In particular, $\widehat{h}(\mathbf{x}_i, \mathbf{w}_i) = \widehat{h}^{(-i)}(\mathbf{x}_i, \mathbf{w}_i)$ is the out-of-sample prediction of y_i, with $\widehat{h}^{(-i)}(\mathbf{x}, \mathbf{w})$ estimated using data that exclude observation i. Similarly, $\widehat{f}(\mathbf{x}_i, \mathbf{w}_i) = \widehat{f}^{(-i)}(\mathbf{x}_i, \mathbf{w}_i)$ is the out-of-sample prediction of d_i.

The residuals are based on the out-of-sample prediction.

$$\widetilde{y}_i^{(-i)} = y_i - \widehat{h}^{(-i)}(\mathbf{x}_i, \mathbf{w}_i)$$
$$\widetilde{d}_i^{(-i)} = d_i - \widehat{f}^{(-i)}(\mathbf{x}_i, \mathbf{w}_i)$$

Thus, in practice, we solve the moment condition

$$\sum_{i=1}^{N} \left[\alpha(x_i) \widetilde{d}_i^{(-i)} \left\{ \widetilde{y}_i^{(-i)} - \widetilde{d}_i^{(-i)} \tau(\mathbf{x}) \right\} \right] = 0$$

We can construct the out-of-sample prediction using the cross-fitting technique. We split the data into K folds, define the main sample as the observations in the kth fold, and define the auxiliary sample as the observations not in the kth fold. We estimate all the nuisance functions using the auxiliary sample and get the out-of-sample predictions in the main sample. After circulating through all the folds, we eventually compute the out-of-sample predictions for the full sample. For details of the cross-fitting version of the PO estimator for the IATE, see algorithm 1 below.

Algorithm 1: PO for the IATE using cross-fitting

1. Define the input.
   (a) Set the number of cross-fitting folds K.
   (b) Set the estimation method for the outcome model. It can be one of random forest, lasso, square-root lasso, or linear regression.
   (c) Set the estimation method for the treatment model. It can be one of probability forest, lasso, probit, or logit.
2. Do the cross-fitting to construct the residuals.
   (a) Randomly split the data into K folds.
   (b) For each fold $k = 1$ to K, do the following:
      i. Define the main sample S^M as the observations in the kth fold and the auxiliary sample S^A as the observations not in the kth fold.
      ii. Construct outcome residuals.
         A. Using the auxiliary sample S^A, train the outcome model $\widehat{h}^A(\mathbf{x}, \mathbf{w})$.

B. Based on $\widehat{h}^A(\mathbf{x},\mathbf{w})$, predict the outcome in the main sample S^M. Denote the prediction as $\widehat{y}^M$.

C. Compute outcome residuals in the main sample $\widetilde{y}^M = y - \widehat{y}^M$.

iii. Construct treatment residuals.

A. Using the auxiliary sample S^A, train the treatment model $\widehat{f}^A(\mathbf{x},\mathbf{w})$.

B. Based on $\widehat{f}^A(\mathbf{x},\mathbf{w})$, predict the outcome in the main sample S^M. Denote the prediction as $\widehat{d}^M$.

C. Compute treatment residuals in the main sample $\widetilde{d}^M = d - \widehat{d}^M$.

3. Using the full sample, estimate the function $\tau(\mathbf{x})$ via linear regression or via generalized random forest as in Athey, Tibshirani, and Wager (2019):

$$\sum_{i=1}^{N}\left[\alpha(x_i)\widetilde{d}_i^{(-i)}\left\{\widetilde{y}_i^{(-i)} - \widetilde{d}_i^{(-i)}\tau(\mathbf{x})\right\}\right] = 0$$

Cross-fitting can be applied to generic machine learning techniques. However, by construction, it is computationally demanding.

When the outcome and treatment models are estimated using random forest, we can use a particular case of cross-fitting that saves computational time. That is, we can use the out-of-bag predictions to construct the residuals. The out-of-bag prediction for an observation is constructed using only the trees in the random forest that do not contain this observation. See *Generalized random forest* for more details. This procedure is equivalent to cross-fitting but has a faster computation time. For details of the out-of-bag prediction-based PO estimator for the IATE, see algorithm 2 below.

Algorithm 2: PO for the IATE using out-of-bag prediction

1. Construct outcome residuals.

   (a) Use the full sample, fit a regression forest for the outcome model, and denote it as $\widehat{h}(\mathbf{x},\mathbf{w})$.

   (b) Based on $\widehat{h}(\mathbf{x},\mathbf{w})$, compute the out-of-bag prediction for the full sample, and denote it as $\widehat{y}^{(\text{oob})}$.

   (c) Compute the outcome residuals for the full sample: $\widetilde{y} = y - \widehat{y}^{(\text{oob})}$.

2. Construct treatment residuals.

   (a) Use the full sample, fit a probability forest for the treatment model, and denote it as $\widehat{f}(\mathbf{x},\mathbf{w})$.

   (b) Based on $\widehat{f}(\mathbf{x},\mathbf{w})$, compute the out-of-bag prediction for the full sample, and denote it as $\widehat{d}^{(\text{oob})}$.

   (c) Compute the treatment residuals for the full sample: $\widetilde{d} = d - \widehat{d}^{(\text{oob})}$.

3. Using the full sample, estimate the function $\tau(\mathbf{x})$ via linear regression or via generalized random forest:

$$\sum_{i=1}^{N}\left[\alpha(x_i)\widetilde{d}_i^{(-i)}\left\{\widetilde{y}_i^{(-i)} - \widetilde{d}_i^{(-i)}\tau(\mathbf{x})\right\}\right] = 0$$

`cate po` implements both algorithms 1 and 2, and the cross-fitting-based PO estimator in algorithm 1 is the default. In algorithm 2, the `oob` option specifies to use the out-of-bag prediction, and it also requires that options `omethod()` and `tmethod()` be specified with random forest.

In practice, we need to specify the following points in `cate po` for the IATE estimation:

1. The outcome variable y and the treatment variable d.
2. The treatment-effects conditioning variables $\mathbf{x}$, which correspond to *catevarlist* in the syntax of `cate`.
3. The `controls()` option with the control variables $\mathbf{w}$, which are empty by default.
4. The `cmethod()` option with the estimation method for the IATE function $\tau(\mathbf{x})$, which can be either random forest or linear regression. The default is random forest.
5. The `omethod()` option with the estimation method for $E(y|\mathbf{x},\mathbf{w})$, which can be lasso, random forest, or linear regression. The default is lasso.
6. The `tmethod()` option with the estimation method for the treatment model $f(\mathbf{x},\mathbf{w})$, which can be logit, probit, lasso, or random forest. The default is lasso.

At minimum, we must specify points 1 and 2 and use the default settings for the other points.

Here is a general note on choosing between random forest, lasso, and parametric regression: Random forest is suitable when the number of covariates is low-dimensional relative to the number of observations and the function is smooth enough. Lasso is suitable when the model can be approximated by a sparse function, which can be very useful in the presence of high-dimensional controls. Parametric regression is easy to compute and interpret but imposes strong assumptions that may be too hard to satisfy with real-world data. In terms of computational speed, lasso is generally faster than random forest.

PO GATE estimator with prespecified groups

The GATE estimator is constructed by regressing the AIPW scores implied by the partial linear model on the group dummy variables, following Semenova and Chernozhukov (2021). In the partial linear model, the AIPW score is defined as

$$\Gamma_i = \widehat{\tau}^{(-i)}(\mathbf{x}_i) + \frac{d_i - \widehat{f}^{(-i)}(\mathbf{x}_i,\mathbf{w}_i)}{\widehat{f}^{(-i)}(\mathbf{x}_i,\mathbf{w}_i)\left\{1-\widehat{f}^{(-i)}(\mathbf{x}_i,\mathbf{w}_i)\right\}}\left\{y_i - \widehat{\mu}^{(-i)}(\mathbf{x}_i,\mathbf{w}_i,d_i)\right\}$$

where

1. $\widehat{\tau}^{(-i)}(\mathbf{x}_i)$ is the out-of-bag prediction of IATE produced by `predict` after `cate po`. If CATE is estimated by linear regression, then $\widehat{\tau}^{(-i)}(\mathbf{x}_i)$ is the linear prediction.
2. d_i is the treatment indicator.
3. $\widehat{f}^{(-i)}(\mathbf{x}_i,\mathbf{w}_i)$ is the prediction of propensity scores. These out-of-sample predictions are already produced in the cross-fitting algorithm 1 or the out-of-bag prediction algorithm 2.
4. y_i is the outcome.
5. $\widehat{\mu}^{(-i)}(\mathbf{x}_i,\mathbf{w}_i,d_i)$ is the out-of-sample prediction of the outcome mean conditional on both the controls and the treatment status. It is defined as

   $$\widehat{\mu}^{(-i)}(\mathbf{x}_i,\mathbf{w}_i,d_i) = \widehat{h}^{(-i)}(\mathbf{x}_i,\mathbf{w}_i) + \left\{d_i - \widehat{f}^{(-i)}(\mathbf{x}_i,\mathbf{w}_i)\right\}\widehat{\tau}^{(-i)}(\mathbf{x}_i)$$

   where $\widehat{h}^{(-i)}(\mathbf{x}_i,\mathbf{w}_i)$ is the out-of-sample prediction of $E(y|\mathbf{x},\mathbf{w})$, which are already computed in either algorithm 1 or 2 when estimating the IATE function.

Here we derive $\widehat{\mu}^{(-i)}(\mathbf{x}_i, \mathbf{w}_i, d_i)$. By the definition of the partial linear model, $h(\mathbf{x}, \mathbf{w}) = E(y|\mathbf{x}, \mathbf{w}) = \tau(\mathbf{x})f(\mathbf{x}, \mathbf{w}) + g(\mathbf{x}, \mathbf{w})$. Denote $\mu_0(\mathbf{x}, \mathbf{w})$ as the conditional expected value of the potential outcome when it is not treated. It is defined as

$$\mu_0(\mathbf{x}, \mathbf{w}) = g(\mathbf{x}, \mathbf{w}) = h(\mathbf{x}, \mathbf{w}) - \tau(\mathbf{x})f(\mathbf{x}, \mathbf{w})$$

Denote $\mu_1(\mathbf{x}, \mathbf{w})$ as the conditional expected value of the potential outcome when it is treated. It is defined as

$$\mu_1(\mathbf{x}, \mathbf{w}) = \mu_0(\mathbf{x}, \mathbf{w}) + \tau(\mathbf{x}) = h(\mathbf{x}, \mathbf{w}) + \{1 - f(\mathbf{x}, \mathbf{w})\}\tau(\mathbf{x})$$

Thus, the cross-fitting version of $\mu(\mathbf{x}, \mathbf{w}, d)$ is defined as above.

We already computed all the necessary terms when fitting the IATE function, so it is computationally convenient to compute the AIPW scores. Once we get the AIPW scores, computing the GATE is easy. Just run a linear regression of the AIPW scores on the group indicators. For details on estimating the GATE in the partial linear model, see algorithm 3 below.

Algorithm 3: PO for the GATE with a prespecified group

1. Set the group indicator variable G.
2. Run either algorithm 1 or algorithm 2 to get the following terms:
   (a) $\widehat{y_i}^{(-i)} = \hat{h}^{(-i)}(\mathbf{x}_i, \mathbf{w}_i)$
   (b) $\widehat{d_i}^{(-i)} = \hat{f}^{(-i)}(\mathbf{x}_i, \mathbf{w}_i)$
3. Compute the out-of-bag prediction of the IATE $\hat{\tau}^{(-i)}(\mathbf{x}_i)$.
4. Compute the AIPW scores for the partial linear model:

$$\Gamma_i = \hat{\tau}^{(-i)}(\mathbf{x}_i) + \frac{d_i - \hat{f}^{(-i)}(\mathbf{x}_i, \mathbf{w}_i)}{\hat{f}^{(-i)}(\mathbf{x}_i, \mathbf{w}_i)\left\{1 - \hat{f}^{(-i)}(\mathbf{x}_i, \mathbf{w}_i)\right\}}\left\{y_i - \hat{\mu}^{(-i)}(\mathbf{x}_i, \mathbf{w}_i, d_i)\right\}$$

   with

$$\hat{\mu}^{(-i)}(\mathbf{x}_i, \mathbf{w}_i, d_i) = \hat{h}^{(-i)}(\mathbf{x}_i, \mathbf{w}_i) + \left\{d_i - \hat{f}^{(-i)}(\mathbf{x}_i, \mathbf{w}_i)\right\}\hat{\tau}^{(-i)}(\mathbf{x}_i)$$

5. Run ordinary least-squares (OLS) regression of the AIPW scores Γ_i on the group dummy indicators based on G. The coefficients estimated on these indicators are the GATEs.

PO GATES estimator with data-driven groups

Suppose we do not know which groups we should condition on when computing the GATEs. We can ask the data to discover the groups based on the sorted estimates of the IATEs. The groups are generated by the quantiles of IATE estimates. For example, if we want to divide the IATEs into four groups, the first group will be observations with IATE estimates greater than the 75th percentile of the overall IATE estimates; the second group will be observations with estimates between the 50th and 75th percentiles; the third group will be between the 25th and 50th percentiles; and the last group will be below the 25th percentile.

We use the cross-fitting technique to generate the group's ranking to avoid overfitting. Suppose we split the data into K folds. For each fold, we do the following. Observations in the kth fold are defined as the main sample, and observations in the other folds are defined as the auxiliary sample. First, we train an IATE model using the auxiliary sample. Second, we predict the IATE function $\tau(\mathbf{x})$ in the main sample and denote the prediction as $\hat{\tau}(\mathbf{x})^{(k)}$. The ranking in the main sample depends on the quantile of $\hat{\tau}(\mathbf{x})^{(k)}$. After circulating all the folds, we divide the full sample into the prespecified number of groups.

Because of the nature of cross-fitting, using the data to discover groups is time consuming. In addition, it requires a large sample because we need to fit a separate random forest model for the IATE in each fold.

Once the group is discovered, we run an OLS regression of the AIPW scores, which is generated in the cross-fitting procedure, to the group indicator. For details of the GATES estimator with the data-driven group for the partial linear model, see algorithm 4. For a discussion of using the sorted effects to generate the group, see Chernozhukov et al. (2006, sec. E) and Golub Capital Social Impact Lab (2023, chap. 4).

Algorithm 4: GATES estimator for data-driven group for partial linear model

1. Define the input.
   (a) Set number of cross-fitting folds K.
   (b) Set the estimation method for the outcome model. It can be one of random forest, lasso, square-root lasso, or linear regression.
   (c) Set the estimation method for the treatment model. It can be one of random forest, lasso, logit, or probit.
   (d) Set the number of groups G to divide.
2. Do the cross-fitting to construct the AIPW scores and group ranking.
   (a) Randomly split the data into K folds.
   (b) For each fold $k = 1$ to K, do the following:
      i. Define the main sample S^M as the observations in the kth fold and the auxiliary sample S^A as the observations not in the kth fold.
      ii. Construct the outcome prediction.
         A. Using the auxiliary sample S^A, train the outcome model $\hat{h}^A(\mathbf{x}, \mathbf{w})$.
         B. Based on $\hat{h}^A(\mathbf{x}, \mathbf{w})$, predict the outcome in the main sample S^M. Denote the prediction as $\hat{h}^M(\mathbf{x}_i, \mathbf{w}_i)$.
      iii. Construct the treatment prediction.
         A. Using the auxiliary sample S^A, train the treatment model $\hat{f}^A(\mathbf{x}, \mathbf{w})$.
         B. Based on $\hat{f}^A(\mathbf{x}, \mathbf{w})$, predict the propensity score in the main sample S^M. Denote the prediction as $\hat{f}^M(\mathbf{x}_i, \mathbf{w}_i)$.
      iv. Construct the IATE ranking.
         A. Using the auxiliary sample S^A, fit the IATE model $\hat{\tau}(\mathbf{x})^A$ using algorithm 1.
         B. Based on $\hat{\tau}(\mathbf{x})^A$, predict $\tau(\mathbf{x})$ in the main sample and denote it as $\hat{\tau}(\mathbf{x})^M$.
         C. Generate the ranking in the main sample using the quantiles of $\hat{\tau}(\mathbf{x})^M$.

v. Construct the AIPW scores in the main sample:

$$\Gamma_i^M = \hat{\tau}^M(\mathbf{x}_i) + \frac{d_i - \hat{f}^M(\mathbf{x}_i, \mathbf{w}_i)}{\hat{f}^M(\mathbf{x}_i, \mathbf{w}_i)\left\{1 - \hat{f}^M(\mathbf{x}_i, \mathbf{w}_i)\right\}} \left\{y_i - \hat{\mu}^M(\mathbf{x}_i, \mathbf{w}_i, d_i)\right\}$$

with

$$\hat{\mu}^M(\mathbf{x}_i, \mathbf{w}_i, d_i) = \hat{h}^M(\mathbf{x}_i, \mathbf{w}_i) + \left\{d_i - \hat{f}^M(\mathbf{x}_i, \mathbf{w}_i)\right\} \hat{\tau}^M(\mathbf{x}_i)$$

3. Regress the AIPW scores on the group dummies based on the IATE quantiles to estimate the GATESs.

AIPW for the fully interactive model

The fully interactive model is

$$\begin{aligned} y(1) &= g_1(\mathbf{x}, \mathbf{w}) + \epsilon_1 \\ y(0) &= g_0(\mathbf{x}, \mathbf{w}) + \epsilon_0 \\ d &= f(\mathbf{x}, \mathbf{w}_2) + u \end{aligned}$$

where $g_1(\mathbf{x}, \mathbf{w})$ and $g_0(\mathbf{x}, \mathbf{w})$ are the models for the potential outcomes $y(1)$ and $y(0)$, respectively. ϵ_1 and ϵ_0 are the error terms. $\mathbf{w}_2$ is a vector of control variables for the treatment model. By default, $\mathbf{w}_2$ is equal to $\mathbf{w}$. However, it can be different if specified. The other terms, d, $\mathbf{x}$, $\mathbf{w}$, and $f(\mathbf{x}, \mathbf{w}_2)$, are the same as seen in the partial linear model. The fully interactive model allows the treatment effect to interact with both $\mathbf{x}$ and $\mathbf{w}$; thus, it is more general than the partial linear model.

By definition, the IATE function $\tau(\mathbf{x})$ is

$$\tau(\mathbf{x}) = E\{y(1) - y(0)|\mathbf{x}\} = E\{g_1(\mathbf{x}, \mathbf{w}) - g_0(\mathbf{x}, \mathbf{w})|\mathbf{x}\}$$

Intuitively, we can regress $g_1(\mathbf{x}, \mathbf{w}) - g_0(\mathbf{x}, \mathbf{w})$ on $\mathbf{x}$ to estimate $\tau(\mathbf{x})$. This method is also known as regression adjustment (RA). However, RA is vulnerable to machine learning mistakes made when estimating $g_1(\mathbf{x}, \mathbf{w})$ and $g_0(\mathbf{x}, \mathbf{w})$. Similarly, the inverse-probability weighting (IPW) estimator is also a bad choice. In contrast, the classical AIPW estimator, known as a doubly robust estimator, is Neyman orthogonal (see Chernozhukov et al. [2018] and Knaus [2022]). The AIPW version of the potential outcomes are

$$\begin{aligned} y(1)_{\text{AIPW}} &= g_1(\mathbf{x}, \mathbf{w}) + \frac{I(d=1)\{y - g_1(\mathbf{x}, \mathbf{w})\}}{f(\mathbf{x}, \mathbf{w}_2)} \\ y(0)_{\text{AIPW}} &= g_0(\mathbf{x}, \mathbf{w}) + \frac{I(d=0)\{y - g_0(\mathbf{x}, \mathbf{w})\}}{1 - f(\mathbf{x}, \mathbf{w}_2)} \end{aligned}$$

Thus, $\tau(\mathbf{x})$ can also be written as

$$\tau(\mathbf{x}) = E\left\{y(1)_{\text{AIPW}} - y(0)_{\text{AIPW}}|\mathbf{x}\right\}$$

Given the estimates of $g_1(\mathbf{x}, \mathbf{w})$, $g_0(\mathbf{x}, \mathbf{w})$, and $f(\mathbf{x}, \mathbf{w}_2)$, let $\widehat{y(1)}_{\text{AIPW}}$ and $\widehat{y(0)}_{\text{AIPW}}$ be estimates of $y(1)_{\text{AIPW}}$ and $y(0)_{\text{AIPW}}$, respectively. Let $\hat{\Gamma} = \widehat{y(1)}_{\text{AIPW}} - \widehat{y(0)}_{\text{AIPW}}$ be an estimate of the AIPW scores. We estimate $\tau(\mathbf{x})$ by solving the local moment condition

$$\sum_{i=1}^{N} \left[\alpha(\mathbf{x}_i)\left\{\hat{\Gamma}_i - \tau(\mathbf{x})\right\}\right] = 0$$

where $\alpha(\mathbf{x}_i)$ defines the local weights that attach more weight to observations that are close to $\mathbf{x}$. To solve this moment condition, we use the honest regression forest proposed in Wager and Athey (2018) and implement it using the generalized random forest in Athey, Tibshirani, and Wager (2019). For details on honest regression forest, see *Generalized random forest*.

AIPW IATE estimator

To guard against machine learning estimation bias in the nuisance function, we need to construct the AIPW scores using the out-of-sample prediction. We estimate the IATE function $\tau(\mathbf{x})$ by solving

$$\sum_{i=1}^{N}\left[\alpha(\mathbf{x}_i)\left\{\hat{\Gamma}_i^{(-i)}-\theta(\mathbf{x})\right\}\right]=0$$

where $\hat{\Gamma}_i^{(-i)}$ are predictions of the AIPW scores for the ith observation via estimation of the nuisance function using observations excluding the ith observation.

We can construct the out-of-sample prediction using the cross-fitting technique. We split the data into K folds, define the main sample as the observations in the kth fold, and define the auxiliary sample as the observations not in the kth fold. We estimate all the nuisance functions by using the auxiliary sample and get the out-of-sample predictions in the main sample. After circulating through all the folds, we eventually compute the out-of-sample predictions for the full sample. For details of the cross-fitting version of the AIPW estimator for the IATE, see algorithm 5 below.

Algorithm 5: AIPW estimator for the IATE using cross-fitting

1. Define the input.
   (a) Set the number of cross-fitting folds K.
   (b) Set the estimation method for the outcome model. It can be one of random forest, lasso, square-root lasso, or linear regression.
   (c) Set the estimation method for the treatment model. It can be one of random forest, lasso, logit, or probit.
2. Do the cross-fitting to construct the AIPW scores.
   (a) Randomly split the data into K folds.
   (b) For each fold $k = 1$ to K, do the following:
      i. Define the main sample S^M as the observations in the kth fold and the auxiliary sample S^A as the observations not in the kth fold.
      ii. Construct $g_1(\mathbf{x},\mathbf{w})$.
         A. Using the auxiliary sample S^A and treated observations ($d_i = 1$), train the outcome model $\hat{g}_1^A(\mathbf{x},\mathbf{w})$.
         B. Based on $\hat{g}_1^A(\mathbf{x},\mathbf{w})$, predict the treated potential outcome in the main sample S^M. Denote the prediction as $\widehat{y(1)}^M$.
      iii. Construct $g_0(\mathbf{x},\mathbf{w})$.
         A. Using the auxiliary sample S^A and untreated observations ($d_i = 0$), train the outcome model $\hat{g}_0^A(\mathbf{x},\mathbf{w})$.
         B. Based on $\hat{g}_0^A(\mathbf{x},\mathbf{w})$, predict the untreated potential outcome in the main sample S^M. Denote the prediction as $\widehat{y(0)}^M$.

iv. Construct the propensity score.
   A. Using the auxiliary sample S^A, train the outcome model $\hat{f}^A(\mathbf{x}, \mathbf{w})$.
   B. Based on $\hat{f}^A(\mathbf{x}, \mathbf{w})$, predict the propensity score in the main sample S^M. Denote the prediction as $\hat{d}^M$.

v. Construct the AIPW score in the main sample S^M as

$$\Gamma_i^M = \left[\widehat{y(1)}_i^M + \frac{d_i\left\{y_i - \widehat{y(1)}_i^M\right\}}{\hat{d}_i^M}\right] - \left[\widehat{y(0)}_i^M + \frac{(1-d_i)\left\{y_i - \widehat{y(0)}_i^M\right\}}{1-\hat{d}_i^M}\right]$$

3. Fit an honest random forest regression of Γ on $\mathbf{x}$ using the generalized random forest in Athey, Tibshirani, and Wager (2019) or fit a linear regression of Γ on $\mathbf{x}$.

Cross-fitting can be applied to generic machine learning techniques. However, by construction, it is computationally demanding.

When the outcome and treatment models are estimated using random forest, we can use a particular case of cross-fitting that saves computational time. That is, we can use the out-of-bag predictions to construct the residuals. The out-of-bag prediction for an observation is constructed using only the trees in the random forest that do not contain this observation. See *Generalized random forest* for more details. This procedure is equivalent to cross-fitting but has a faster computation time. For details of the out-of-bag prediction-based PO estimator for the IATE, see algorithm 6 below.

Algorithm 6: AIPW estimator for the IATE using out-of-bag prediction

1. Construct $g_1(\mathbf{x}, \mathbf{w})$.
   (a) Using the full sample and the treated observations ($d_i = 1$), fit a regression forest for the outcome model. Denote it as $\hat{g}_1(\mathbf{x}, \mathbf{w})$.
   (b) Based on $\hat{g}_1(\mathbf{x}, \mathbf{w})$, predict the treated potential outcome for the full sample using the out-of-bag prediction. Denote it as $\widehat{y(1)}^{(\text{oob})}$.
2. Construct $g_0(\mathbf{x}, \mathbf{w})$.
   (a) Using the full sample and the untreated observations ($d_i = 0$), fit a regression forest for the outcome model. Denote it as $\hat{g}_0(\mathbf{x}, \mathbf{w})$.
   (b) Based on $\hat{g}_0(\mathbf{x}, \mathbf{w})$, predict the treated potential outcome for the full sample using the out-of-bag prediction. Denote it as $\widehat{y(0)}^{(\text{oob})}$.
3. Construct the propensity score.
   (a) Using the full sample, fit a probability forest for the treatment model. Denote it as $\hat{f}(\mathbf{x}, \mathbf{w})$.
   (b) Based on $\hat{f}(\mathbf{x}, \mathbf{w})$, predict the propensity score for the full sample using the out-of-bag prediction. Denote it as $\hat{d}^{(\text{oob})}$.
4. Construct the AIPW scores as

$$\Gamma_i = \left[\widehat{y(1)}_i^{(\text{oob})} + \frac{d_i\left\{y_i - \widehat{y(1)}_i^{(\text{oob})}\right\}}{\hat{d}_i^{(\text{oob})}}\right] - \left[\widehat{y(0)}_i^{(\text{oob})} + \frac{(1-d_i)\left\{y_i - \widehat{y(0)}_i^{(\text{oob})}\right\}}{1-\hat{d}_i^{(\text{oob})}}\right]$$

5. Fit a random forest regression of Γ on $\mathbf{x}$ using the generalized random forest in Athey, Tibshirani, and Wager (2019) or fit a linear regression of Γ on $\mathbf{x}$.

`cate aipw` implements both algorithms 5 and 6, and the cross-fitting-based AIPW estimator in algorithm 5 is the default. The `oob` option specifies to use the out-of-bag prediction in algorithm 6, and it requires that options `omethod()` and `tmethod()` be specified with random forest.

In practice, we must specify the following points in `cate aipw` for the IATE estimation:

1. The outcome variable y and the treatment variable d.
2. The treatment-effects conditioning variables $\mathbf{x}$, which correspond to *catevarlist* in the syntax of `cate`.
3. The `controls()` option with the control variables $\mathbf{w}$, which are empty by default.
4. The `cmethod()` option with the estimation method for the IATE function $\tau(\mathbf{x})$, which can be either random forest or linear regression. The default is random forest.
5. The `omethod()` option with the estimation method for $g_1(\mathbf{x}, \mathbf{w})$ and $g_0(\mathbf{x}, \mathbf{w})$ can be lasso, random forest, or linear regression. The default is lasso.
6. The `tmethod()` option with the estimation method for the treatment model $f(\mathbf{x}, \mathbf{w})$, which can be logit, probit, lasso, or random forest. The default is lasso.

At minimum, we must specify points 1 and 2 and use the default settings for the other points.

Both PO and AIPW are consistent estimators of $\tau(\mathbf{x})$ under similar regularity conditions, and they are Neyman orthogonal in the sense that the estimates are robust to the machine learning mistakes made in the nuisance parameters, such as $g(\mathbf{x}, \mathbf{w})$ or $f(\mathbf{x}, \mathbf{w})$.

The AIPW estimator is more efficient than the PO estimator. That is, in large samples, the AIPW estimates of the IATE function are more precise than the PO estimates (see Kennedy [2023]). In addition, the AIPW estimator has double robustness; that is, the estimator is still consistent even if the outcome model or the treatment model is misspecified (see Chernozhukov et al. [2018]).

However, the PO estimator is more robust than the AIPW estimator when there are some nearly perfect predictions of the propensity score $f(\mathbf{x}, \mathbf{w})$. More precisely, the AIPW estimator needs to compute the inverse of propensity score $f(\mathbf{x}, \mathbf{w})$ or $1 - f(\mathbf{x}, \mathbf{w})$, which lies between 0 and 1. Thus, the AIPW estimator may be undefined whenever the propensity score $f(\mathbf{x}, \mathbf{w})$ estimates are close to 0 or 1. In contrast, the PO estimator does not have these issues.

AIPW GATE estimator with prespecified groups

The GATE estimator for the fully interactive model is the OLS estimate for the AIPW scores on the group indicator, following Semenova and Chernozhukov (2021).

In the full interactive model, the AIPW score is defined as

$$\Gamma_i = \left[\widehat{y(1)}_i^{(-i)} + \frac{d_i \left\{ y_i - \widehat{y(1)}_i^{(-i)} \right\}}{\hat{d}_i^{(-i)}} \right] - \left[\widehat{y(0)}_i^{(-i)} + \frac{(1 - d_i) \left\{ y_i - \widehat{y(0)}_i^{(-i)} \right\}}{1 - \hat{d}_i^{(-i)}} \right]$$

where

1. y_i is the observed outcome.
2. d_i is the observed treatment indicator.
3. $\widehat{y(1)}_i^{(-i)}$ is the out-of-sample prediction of the treated potential outcome.
4. $\widehat{y(0)}_i^{(-i)}$ is the out-of-sample prediction of the untreated potential outcome.
5. $\hat{d}_i^{(-i)}$ is the out-of-sample prediction of the propensity score.

The scores can be obtained using either the cross-fitting AIPW estimator in algorithm 5 or the out-of-bag prediction-based AIPW estimator in algorithm 6. After the AIPW scores are obtained, the GATE estimator is just the OLS estimate of scores on the group indicator. For details of estimating the GATE in the fully interactive model, see algorithm 7 below.

Algorithm 7: GATE estimator with a prespecified group

1. Select the group variable G.
2. Run either algorithm 5 or algorithm 6 to obtain the AIPW scores $\Gamma_i^{(-i)}$.

$$\Gamma_i^{(-i)} = \left[\widehat{y(1)}_i^{(-i)} + \frac{d_i\left\{y_i - \widehat{y(1)}_i^{(-i)}\right\}}{\hat{d}_i^{(-i)}}\right] - \left[\widehat{y(0)}_i^{(-i)} + \frac{(1-d_i)\left\{y_i - \widehat{y(0)}_i^{(-i)}\right\}}{1-\hat{d}_i^{(-i)}}\right]$$

3. Run OLS of Γ on the group indicators based on G.

AIPW GATES estimator with data-driven groups

In the fully interactive model, the procedure to compute the GATESs with data-driven groups is very similar to that for the partial linear model. The only difference is using the AIPW scores implied by the fully interactive model. For details of estimating the GATESs with data-driven groups for the fully interactive model, see algorithm 8.

Algorithm 8: GATES estimator with data-driven groups in fully interactive model

1. Define the input.
   (a) Set the number of cross-fitting folds K.
   (b) Select the estimation method for the outcome and treatment models.
2. Perform cross-fitting to construct the AIPW scores.
   (a) Randomly split the data into K folds.
   (b) For each fold $k = 1$ to K, do the following:
      i. Define the main sample S^M as the observations in folds k and the auxiliary sample S^A as the observations not in folds k.

ii. Construct $g_1(\mathbf{x}, \mathbf{w})$.
   A. Using the auxiliary sample S^A and treated observations ($d_i = 1$), train the outcome model $\hat{g}_1^A(\mathbf{x}, \mathbf{w})$.
   B. Based on $\hat{g}_1^A(\mathbf{x}, \mathbf{w})$, predict the treated potential outcome in the main sample S^M. Denote the prediction as $\widehat{y(1)}^M$.

iii. Construct $g_0(\mathbf{x}, \mathbf{w})$.
   A. Using the auxiliary sample S^A and untreated observations ($d_i = 0$), train the outcome model $\hat{g}_0^A(\mathbf{x}, \mathbf{w})$.
   B. Based on $\hat{g}_0^A(\mathbf{x}, \mathbf{w})$, predict the untreated potential outcome in the main sample S^M. Denote the prediction as $\widehat{y(0)}^M$.

iv. Construct the propensity score.
   A. Using the auxiliary sample S^A, train the outcome model $\hat{f}^A(\mathbf{x}, \mathbf{w})$.
   B. Based on $\hat{f}^A(\mathbf{x}, \mathbf{w})$, predict the propensity score in the main sample S^M. Denote the prediction as $\hat{d}^M$.

v. Construct the AIPW score in the main sample S^M as

$$\Gamma_i^M = \left[\widehat{y(1)}_i^M + \frac{d_i \left\{ y_i - \widehat{y(1)}_i^M \right\}}{\hat{d}_i^M} \right] - \left[\widehat{y(0)}_i^M + \frac{(1 - d_i) \left\{ y_i - \widehat{y(0)}_i^M \right\}}{1 - \hat{d}_i^M} \right]$$

vi. Construct the IATE ranking.
   A. Using the auxiliary sample S^A, train the IATE model $\hat{\tau}(\mathbf{x})^A$ using algorithm 5.
   B. Based on $\hat{\tau}(\mathbf{x})^A$, predict $\tau(\mathbf{x})$ in the main sample, and denote it as $\hat{\tau}(\mathbf{x})^M$.
   C. Generate the ranking in the main sample based on the quantiles of $\hat{\tau}(\mathbf{x})^M$.

3. Run an OLS regression of the AIPW scores on the generated groups' indicator dummies.

Generalized random forest

The generalized random forest, proposed in Athey, Tibshirani, and Wager (2019), solves the moment condition

$$E\left\{ \psi_{\theta(\mathbf{x})}(\mathbf{o}_i) | \mathbf{x}_i = \mathbf{x} \right\} = 0$$

where $\theta(\mathbf{x})$ is a parameter of interest, $\mathbf{o}_i$ is a vector of variables including the outcome variable and some covariates, and $\mathbf{x}_i$ is a vector of covariates in the function $\theta(\mathbf{x})$.

The form of $\psi_{\theta(\mathbf{x})}(\mathbf{o}_i)$ varies depending on the context. For example, in the regression or probability random forest, the parameter of interest is $\theta(\mathbf{x}) = E(y|\mathbf{x})$, and the moment condition is

$$E\left\{ y_i - \theta(\mathbf{x}) | \mathbf{x}_i = \mathbf{x} \right\} = 0$$

In the causal random forest, the parameter of interest is $\theta(\mathbf{x}) = \tau(\mathbf{x})$, and the moment condition is

$$E\left[\tilde{d}_i \left\{ \tilde{y}_i - \theta(\mathbf{x}) \tilde{d}_i \right\} \middle| \mathbf{x}_i = \mathbf{x} \right] = 0$$

where $\tilde{d}$ and $\tilde{y}$ are partialed-out residuals of d and y discussed in the PO estimator.

To estimate $\theta(\mathbf{x})$, generalized random forest solves the empirical moment condition

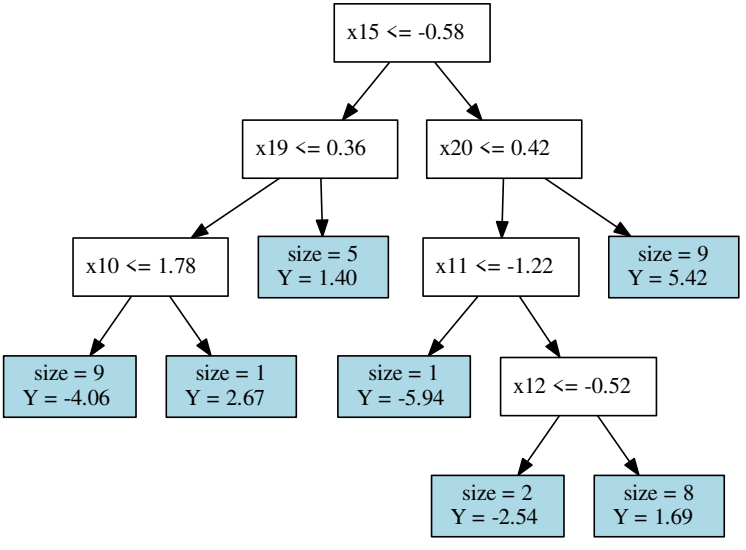
$$\sum_{i=1}^{N} \alpha_i(\mathbf{x}) \psi_{\theta(\mathbf{x})}(\mathbf{o}_i) = 0$$

where $\alpha_i(\mathbf{x})$ defines the local weights that attach more weight to observations close to x. Generalized random forest obtains $\alpha_i(\mathbf{x})$ by averaging the neighborhood implied by a forest. See Athey, Tibshirani, and Wager (2019, sec. 2) for a detailed discussion.

The generalized random forest is an ensemble of honest trees, which we explain next.

Honest tree

A tree is an algorithm that divides the data into different parts, such that each part consists of observations that are as similar as possible. The following graph illustrates a regular regression tree:



For example, in the above graph, we first look at variable $x15$ to determine how to divide the data. If its value is smaller than or equal to -0.58, that observation goes to the left; otherwise, it goes to the right. Each time we split the data, it is represented by a node. We recursively continue this procedure until we hit a "leaf" node, which means we cannot find a variable to split the data.

The leaf nodes in the graph are represented as a blue rectangle. A leaf node defines a part of the tree partition. For example, the leftmost leaf has 9 observations (size $= 9$ and $Y = -4.06$ in the graph). We can travel to this leaf by finding observations satisfying three conditions: $x15 \leq -0.58$, $x19 \leq 0.36$, and $x10 \leq 1.78$. These three conditions correspond to the three nodes from the top to the bottom, enabling us to travel from the top to the particular leaf.

In each leaf node, the mean of the outcome within a leaf is used to label the leaf or attach a value to it. Thus, constructing a tree involves two main steps: splitting the tree and labeling the tree. More precisely, tree construction involves the following:

1. Splitting the tree: For each node, find a splitting variable x_s and a value v_s such that the resulting divisions are as different as possible. In particular, we find x_s and v_s by solving

$$(x_s, v_s) = \text{argmin}_{(x_m, v_m) \in \mathbf{x} \times D(x_m)} \{\text{cost}(x_m \leq v_m) + \text{cost}(x_m > v_m)\}$$

   where cost$(\cdot)$ is a cost function that characterizes the node's homogeneity and its definition varies depending on the context. For example, in the regression forest, the cost function is the mean squared error of the outcome; in the probability forest, the cost function is the Gini index for the outcome; in the causal forest, the cost function is the squared sum of the influence function implied by the causal forest moment condition. For details, see Athey, Tibshirani, and Wager (2019, sec. 2).

2. When there are not enough observations in any node, the algorithm stops searching. The `splitminobs()` option specifies the minimum number of observations needed to perform a split.

3. Labeling the tree: In each leaf, attach a value or a label to the leaf. The labeling formula varies depending on the context. In the regression or probability forest, the label is the outcome's mean within a leaf. In the causal forest, the label consists of the means of the outcome, the treatment, and their interactions. For details, see Athey, Tibshirani, and Wager (2019, sec. 2).

An honest tree differs from a regular tree by dividing the sample into two subsamples. One is used to split the tree, and the other is used to label the leaves. For details, see algorithm 9 below.

Algorithm 9: Honest tree

1. Split the data into two parts: A and B. The `honestrate()` option specifies the fraction of the data used to construct A.

2. Use sample A to split the tree.

3. Use sample B to label the tree.

Honest random forest

The honest random forest is an ensemble of honest trees with some requirements on the randomness of the tree. There are two main requirements: first, each tree must use a random subsample of the data; second, for each node in the tree, only a random subset of the variables may be searched to find the best variable to split. For more details on the honest random forest, see algorithm 10 below.

Algorithm 10: Honest random forest

1. Define B as the number of trees, which can be specified in the `ntrees()` option.

2. For each tree $b = 1$ to B, do the following:

   (a) Draw a random sample S_b of the full data. The `samprate()` option specifies the fraction of the sample.

(b) Based on S_b, construct an honest tree using algorithm 9. In each node of the honest tree, only a random subset of variables is searched to find the best variable to split. The mean number of this variable set follows a Poisson distribution with expectation μ. The `splitmeanvars()` option specifies μ.

3. Use the forest to construct local weights $\alpha_i(\mathbf{x})$, and then estimate $\theta(\mathbf{x})$ by solving the moment condition $\sum_{i=1}^{N} \alpha_i(\mathbf{x})\psi_{\theta(\mathbf{x})}(\mathbf{o}_i) = 0$.

Confidence intervals

The confidence intervals of the estimate $\theta(\mathbf{x})$ in generalized random forest are constructed using the delta method, as discussed in Athey, Tibshirani, and Wager (2019, sec. 4). In particular, the confidence intervals with significance level α are defined as

$$\lim_{n\to\infty} E\left[\theta(\mathbf{x}) \in \left\{\hat{\theta}(\mathbf{x}) \pm \Phi^{-1}(1-\alpha/2)\hat{\sigma}(\mathbf{x})\right\}\right] = \alpha$$

where $\Phi^{-1}(\cdot)$ is the inverse function of the Gaussian cumulative distribution function and $\hat{\sigma}(\mathbf{x})$ is an estimate of the standard errors of $\hat{\theta}(\mathbf{x})$.

The variance of $\theta(\mathbf{x})$ is defined as

$$Var\{\theta(\mathbf{x})\} = \xi' V(\mathbf{x})^{-1} H(\mathbf{x}) V(\mathbf{x})^{-1} \xi$$

where $V(\mathbf{x}) = \frac{\partial E\{\psi_{\theta(\mathbf{x})}(\mathbf{o})|\mathbf{x}_i=\mathbf{x}\}}{\partial\theta(\mathbf{x})}$, ξ is the subvector selector, and $H(\mathbf{x}) = Var\left\{\sum_{i=1}^{N} \alpha_i \psi_{\theta(\mathbf{x})}(\mathbf{o}_i)\right\}$. See Athey, Tibshirani, and Wager (2019, sec. 4) for a detailed discussion on $V(\mathbf{x})$ and ξ. We use the bootstrap of little bags to estimate $H(\mathbf{x})$. See algorithm 11 for details.

Algorithm 11: Bootstrap of little bags

1. Define the number of trees in each bag, l, which can be specified in the `cintrees()` option.
2. Define the number of trees in the forest, B, which can be specified in the `ntrees()` option.
3. For each bag, $g = 1, \ldots, \lceil B/l \rceil$, draw a random half-sample $H_g \subset \{1, \ldots, n\}$ of size $\lceil n/2 \rceil$, where n is the total number of observations.
4. For each tree, $b = 1, \ldots, B$, draw a random sample $I_b \subseteq H_{\lceil b/l \rceil}$, and build an honest tree using I_b. This step implies that the trees in the same bag are drawing samples from the same half-sample H_g.
5. Define $\Psi_b\{\theta(\mathbf{x})\} = \sum_{i=1}^{n} \alpha_{bi}(\mathbf{x})\psi_{\theta(\mathbf{x})}(\mathbf{o}_i)$, where $\alpha_{bi}(\mathbf{x})$ is the tree-level local weight. Similarly, define $\Psi\{\theta(\mathbf{x})\} = \sum_{i=1}^{n} \alpha_i(\mathbf{x})\psi_{\theta(\mathbf{x})}(\mathbf{o}_i)$, where $\alpha_i(\mathbf{x})$ is the forest-level local weight. Estimate $H(\mathbf{x})$ as

$$\widehat{H}(\mathbf{x}) = E\left\{\left(\frac{1}{l}\sum_{b=1}^{l}\Psi_b - \Psi\right)^2\right\} - \frac{1}{l-1}E\left\{\frac{1}{l}\sum_{b=1}^{l}\left(\Psi_b - \frac{1}{l}\sum_{b=1}^{l}\Psi_b\right)^2\right\}$$

where the expectation is taken over the bags.

See Athey, Tibshirani, and Wager (2019, sec. 4.1) for a detailed discussion on the consistency of the bootstrap of little bags.

Missing values

Random forest can handle missing values in covariates. When facing a missing value of a covariate, there are three possible scenarios for splitting a node. All three scenarios are computed for every variable in the random subset of variables to search, and the one with the best cost function is used. For more discussion, see Twala, Jones, and Hand (2008).

1. Scenario 1: All observations that are not missing in a specific covariate are sent to the left node, and all the other observations are sent to the right node.

   Missing values can provide information on the dependent variable, and potentially, a missing value of a covariate can be used to split a node in the tree.

2. We use the optimal nonmissing value of the specific covariate (the value that minimizes the cost) to split observations and then proceed as follows:

   (a) Scenario 2: The observations with missing values are sent to the right node.

   (b) Scenario 3: The observations with missing values are sent to the left node.

In `cate`, the observations with missing covariate values are kept if the random forest is used in all the models, that is, when `omethod()`, `tmethod()`, and `cmethod()` all use `rforest`. Specifying the `rflistwise` option will drop the observations with missing covariate values when all the models are estimated using random forest.

In contrast, if one of `omethod()`, `tmethod()`, or `cmethod()` does not use `rforest`, the observations with missing covariates will be dropped before any computation. This is because estimation methods such as `lasso`, `regress`, and `probit` will predict missing values if one of the covariates is missing, and the predicted missing values make the dependent variable of the IATE estimation missing. Because random forest cannot handle missing dependent variable values, these observations will be dropped eventually.

References

Athey, S., J. Tibshirani, and S. Wager. 2019. Generalized random forests. *Annals of Statistics* 47: 1148–1178. https://doi.org/10.1214/18-AOS1709.

Chernozhukov, V., D. Chetverikov, M. Demirer, E. Duflo, C. B. Hansen, W. K. Newey, and J. M. Robins. 2018. Double/debiased machine learning for treatment and structural parameters. *Econometrics Journal* 21: C1–C68. https://doi.org/10.1111/ectj.12097.

Chernozhukov, V., M. Demirer, E. Duflo, and I. Fernández-Val. 2006. Generic machine learning inference on heterogeneous treatment effects in randomized experiments, with an application to immunization in India. NBER Working Paper 24678, National Bureau of Economic Research. https://doi.org/10.3386/w24678.

Chernozhukov, V., and C. B. Hansen. 2004. The effects of 401(k) participation on the wealth distribution: An instrumental quantile regression analysis. *Review of Economics and Statistics* 86: 735–751. https://doi.org/10.1162/0034653041811734.

Golub Capital Social Impact Lab. 2023. *Machine Learning-based Causal Inference Tutorial*. Stanford, CA, https://bookdown.org/stanfordgsbsilab/ml-ci-tutorial/hte-i-binary-treatment.html.

Head, M. L., L. Holman, R. Lanfear, A. T. Kahn, and M. D. Jennions. 2015. The extent and consequences of p-hacking in science. *PLOS Biology* 13: e1002106. https://doi.org/10.1371/journal.pbio.1002106.

Kennedy, E. H. 2023. Towards optimal doubly robust estimation of heterogeneous causal effects. *Electronic Journal of Statistics* 17: 3008–3049. https://doi.org/10.1214/23-EJS2157.

Knaus, M. C. 2022. Double machine learning–based programme evaluation under unconfoundedness. *Econometrics Journal* 25: 602–627. https://doi.org/10.1093/ectj/utac015.

Koch, B., D. M. Vock, and J. Wolfson. 2018. Covariate selection with group lasso and doubly robust estimation of causal effects. *Biometrics* 74: 8–17. https://doi.org/10.1111/biom.12736.

Nie, X., and S. Wager. 2021. Quasi-oracle estimation of heterogeneous treatment effects. *Biometrika* 108: 299–319. https://doi.org/10.1093/biomet/asaa076.

Semenova, V., and V. Chernozhukov. 2021. Debiased machine learning of conditional average treatment effects and other causal functions. *Econometrics Journal* 24: 264–289. https://doi.org/10.1093/ectj/utaa027.

Twala, B. E. T. H., M. C. Jones, and D. J. Hand. 2008. Good methods for coping with missing data in decision trees. *Pattern Recognition Letters* 29: 950–956. https://doi.org/10.1016/j.patrec.2008.01.010.

Wager, S., and S. Athey. 2018. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association* 113: 1228–1242. https://doi.org/10.1080/01621459.2017.1319839.

Also see

[CAUSAL] **cate postestimation** — Postestimation tools for cate

[CAUSAL] **teffects aipw** — Augmented inverse-probability weighting

[CAUSAL] **telasso** — Treatment-effects estimation using lasso

[U] 20 Estimation and postestimation commands