An Introduction to Recursive Partitioning for Heterogeneous Causal Effects Estimation Using `causalTree` package

Susan Athey
Guido Imbens
Yanyang Kong

April 30, 2016

Contents

1 Introduction

This document is a brief introduction of `causalTree` package, which is intended to give a short overview of the `causalTree` function and the `honest.causalTree` function, which implement the methods from *Recursive Partitioning for Heterogeneous Causal Effects* [1].

The `causalTree` function builds a regression model and returns an `rpart` object, which is the object derived from `rpart` package, implemneting many ideas in the CART (Classification and Regression Trees), written by Breiman, Friedman, Olshen and Stone [2]. Like `rpart`, `causalTree` builds a binary regression tree model in two stages, but focuses on estimating heterogeneous causal effect.

Following `rpart`, in the first stage, the tree is grown from the root node based on a specified splitting rule. In each node, the data in a leaf will be split into two groups to best minimize the risk function. Next, in the left sub-node and right sub-node, the splitting routine will be applied separately and so on recursively until no improvements can made, or until some limits are reached (e.g. the routine will stop if it cannot make splits that have at least `minsize` of treated observations and `minsize` control observations in each terminal node.)

In the second stage, the tree will be pruned using a specified cross-validation method, where the cross-validation penalty parameter penalizes the number of nodes in the tree. The leaves to be pruned are selected according to the risk function calculated while the tree is built.

The `causalTree` package incorporates an additional function not included in `rpart`, which is honest re-estimation `honest.causalTree` of causal effects. Honest here means that we estimate causal effects in the leaves of a given tree on an independent estimation sample rather than the data used to build and cross-validate the tree. The user first builds the tree with `causalTree`, specifying the training data for building the tree, and then passes the tree object as well as the estimation sample data into `honest.causalTree`, which replaces the leaf estimates from the input tree with new estimates in each leaf, calculated on the estimation sample.

2 Notation

X_i	$i = 1, 2, ..., N$	observed variables or feature matrix for observation i.
Y_i	$i = 1, 2, ..., N$	observed outcome of observation i.
W_i	$i = 1, 2, ..., N$	binary indicator for the treatment, with $W_i = 0$ indicating that observation i received the control treatment, and $W_i = 1$ indicating that observation i received the active treatment.

$\mathcal{S}$	a data sample drawn from data sample population, $\mathcal{S}^{\text{tr}}$ denotes a training sample, $\mathcal{S}^{\text{te}}$ denotes a test sample, $\mathcal{S}^{\text{est}}$ denotes an estimation sample. $\mathcal{S}_{\text{treat}}$ and $\mathcal{S}_{\text{control}}$ denote the subsamples of treated and control units.
N	N^{tr} denotes the number of observations in training sample, N^{te} denotes the number of observations in testing sample, N^{est} denotes the number of observations in estimation sample.
Π	a partitioning tree $\Pi = \{\ell_1, \dots, \ell_{\#(\Pi)}\}$ with $\cup_{j=1}^{\#(\Pi)} \ell_j = \mathbb{X}$ corresponds to a partitioning of the feature space the feature sapce $\mathbb{X}$, with $\#(\Pi)$ the number of elements in the partition.
$\ell(x;\Pi)$	the leaf $\ell \in \Pi$ such that $x \in \ell$.
$\tau(\ell)$	$l = 1, 2, ..., k$ causal effect or treatment effect in leaf ℓ.
p	marginal treatment probability, $p = \text{pr}(W_i = 1)$.

3 Building Causal Trees

3.1 Splitting rules

`causalTree` function offers four different splitting rules for user to choose. Each splitting rule corresponds to a specific risk function, and each split at a node aims to minimize the risk function. For each observation $(Y_i^{\text{obs}}, X_i, W_i)$, given a tree Π, the population average outcome is

$$\mu(w, x; \Pi) \equiv \mathbb{E}\left[Y_i(w) \middle| X_i \in \ell(x;\Pi) \right],$$

and its average causal effect is

$$\tau(x; \Pi) \equiv \mathbb{E}\left[Y_i(1) - Y_i(0) \middle| X_i \in \ell(x;\Pi) \right].$$

the estimated outcome is

$$\hat{\mu}(w, x; \mathcal{S}, \Pi) \equiv \frac{1}{\#(\{i \in \mathcal{S}_w : X_i \in \ell(x;\Pi)\})} \sum_{i \in \mathcal{S}_w : X_i \in \ell(x;\Pi)} Y_i^{\text{obs}},$$

the estimated causal effect is the difference of treated mean and control mean in the leaf l where it belongs,

$$\hat{\tau}(x; \mathcal{S}, \Pi) \equiv \tau(\ell) = \hat{\mu}(1, x; \mathcal{S}, \Pi) - \hat{\mu}(0, x; \mathcal{S}, \Pi).$$

3.1.1 Transformed Outcome Trees (TOT)

We first define the transformed outcome as

$$Y_i^* = Y_i \cdot \frac{W_i - p}{p \cdot (1-p)}$$

where $p = N_{\text{treat}}/N$ is the reatment probability, and

$$Y_i^* = \begin{cases} Y_i/p & W_i = 1 \\ -Y_i/(1-p) & W_i = 0 \end{cases}$$

In **TOT** splitting rule, the risk function is given by

$$\widehat{\text{MSE}}(\mathcal{S}^{\text{tr}}, \mathcal{S}^{\text{tr}}, \Pi) = \frac{1}{N^{\text{tr}}} \sum_{i \in \mathcal{S}^{\text{tr}}} \left\{ (Y_i^* - \hat{\tau}(X_i; \mathcal{S}^{\text{tr}}, \Pi))^2 - Y_i^{*2} \right\}$$

Note that the paper [1] envisions that treatment effects would be estimated by taking the mean of Y_i^* within a leaf, but points out that this is inefficient because the treated fraction in a leaf may differ from the population proportion due to sampling variation. Thus, our package uses $\hat{\tau}$ instead. The `rpart` package can be used off-the-shelf (applied with Y_i^* as the outcome) to implement the method precisely as described in [1].

3.1.2 Causal Trees (CT)

In causal trees splitting rule, we have two versions, adaptive verison, denoted as **CT-A**, and honest version, **CT-H**. You can switch honest version by setting `split.Honest = TRUE` in `causalTree` function.
For **CT-A**, we use $\widehat{\text{MSE}}_\tau(\mathcal{S}^{\text{tr}}, \mathcal{S}^{\text{tr}}, \Pi)$ as the objective risk function, and

$$-\widehat{\text{MSE}}_\tau(\mathcal{S}^{\text{tr}}, \mathcal{S}^{\text{tr}}, \Pi) = \frac{1}{N^{\text{tr}}} \sum_{i \in \mathcal{S}^{\text{tr}}} \hat{\tau}^2(X_i; \mathcal{S}^{\text{tr}}, \Pi).$$

For **CT-H**, the honest version, the splitting objective risk function is $\widehat{\text{EMSE}}_\tau(\mathcal{S}^{\text{tr}}, N^{\text{est}}, \Pi)$, and

$$\begin{aligned} -\widehat{\text{EMSE}}_\tau(\mathcal{S}^{\text{tr}}, N^{\text{est}}, \Pi) = {} & \frac{1}{N^{\text{tr}}} \sum_{i \in \mathcal{S}^{\text{tr}}} \hat{\tau}^2(X_i; \mathcal{S}^{\text{tr}}, \Pi) \\ & - \left(\frac{1}{N^{\text{tr}}} + \frac{1}{N^{\text{est}}} \right) \cdot \sum_{\ell \in \Pi} \left(\frac{S^2_{\mathcal{S}^{\text{tr}}_{\text{treat}}}(\ell)}{p} + \frac{S^2_{\mathcal{S}^{\text{tr}}_{\text{control}}}(\ell)}{1-p} \right). \end{aligned}$$

where $S^2_{\mathcal{S}^{\text{tr}}_{\text{control}}}(\ell)$ is the within-leaf variance on outcome Y for $\mathcal{S}^{\text{tr}}_{\text{control}}$ in leaf ℓ, and $S^2_{\mathcal{S}^{\text{tr}}_{\text{treat}}}(\ell)$ is the counter part for $\mathcal{S}^{\text{tr}}_{\text{treat}}$. N^{est} (number of observations in re-estimation sample) is specified as `HonestSampleSize` in `causalTree` function, and the default value is N^{tr}.
In our package we incorporate an additional parameter `split.alpha` $= \alpha \in (0,1)$ as a parameter to adjust the proportion of $\widehat{\text{MSE}}$ and the varaince term in $\widehat{\text{EMSE}}$.

$$
\begin{aligned}
-\widehat{\text{EMSE}}_\tau(\mathcal{S}^{\text{tr}}, N^{\text{est}}, \Pi, \alpha) = {} & \alpha \cdot \frac{1}{N^{\text{tr}}} \sum_{i \in \mathcal{S}^{\text{tr}}} \hat{\tau}^2(X_i; \mathcal{S}^{\text{tr}}, \Pi) \\
& - (1-\alpha) \cdot \left(\frac{1}{N^{\text{tr}}} + \frac{1}{N^{\text{est}}} \right) \cdot \sum_{\ell \in \Pi} \left(\frac{S^2_{\mathcal{S}^{\text{tr}}_{\text{treat}}}(\ell)}{p} + \frac{S^2_{\mathcal{S}^{\text{tr}}_{\text{control}}}(\ell)}{1-p} \right)
\end{aligned}
$$

3.1.3 Fit-based Trees (fit)

In fit-based splitting rule, we decide at what value of the feature to split based on the goodness-of-fit of the outcome rather than the treatment effect. As **CT**, there are two versions of **fit**, namely adaptive version **fit-A** and honest version **fit-H**.
For **fit-A**, the objective risk function in splitting is

$$
\widehat{\text{MSE}}_{\mu,W}(\mathcal{S}^{\text{tr}}, \mathcal{S}^{\text{tr}}, \Pi) = \sum_{i \in \mathcal{S}^{\text{tr}}} \left\{ (Y_i - \hat{\mu}_w(W_i, X_i; \mathcal{S}^{\text{tr}}, \Pi))^2 - Y_i^2 \right\}
$$

where $\hat{\mu}_w$ is the mean of outcome in treatment/control group.

For **fit-H**, the honest version, the risk function is $\widehat{\text{EMSE}}_{\mu,W}(\mathcal{S}^{\text{tr}}, N^{\text{est}}, \Pi)$,

$$
\begin{aligned}
-\widehat{\text{EMSE}}_{\mu,W}(\mathcal{S}^{\text{tr}}, N^{\text{est}}, \Pi) = {} & \frac{1}{N^{\text{tr}}} \sum_{i \in \mathcal{S}^{\text{tr}}} \hat{\mu}_w^2(W_i, X_i; \mathcal{S}^{\text{tr}}, \Pi) \\
& - \left(\frac{1}{N^{\text{tr}}} + \frac{1}{N^{\text{est}}} \right) \cdot \sum_{\ell \in \Pi} \left(S^2_{\mathcal{S}^{\text{tr}}_{\text{treat}}}(\ell) + S^2_{\mathcal{S}^{\text{tr}}_{\text{control}}}(\ell) \right),
\end{aligned}
$$

where $S^2_{\mathcal{S}^{\text{tr}}_{\text{control}}}(\ell)$ is the within-leaf variance on outcome Y for $\mathcal{S}^{\text{tr}}_{\text{control}}$ in leaf ℓ, and $S^2_{\mathcal{S}^{\text{tr}}_{\text{treat}}}(\ell)$ is the counter part for $\mathcal{S}^{\text{tr}}_{\text{treat}}$. N^{est} (number of observations in re-estimation sample) is specified as `HonestSampleSize` in `causalTree` function, and the default value is N^{tr}.
Also like **CT**, we have adjusted honest verison for $\widehat{\text{EMSE}}_{\mu,W}$ using `split.alpha`,

$$
\begin{aligned}
-\widehat{\text{EMSE}}_{\mu,W}(\mathcal{S}^{\text{tr}}, N^{\text{est}}, \Pi, \alpha) = {} & \alpha \cdot \frac{1}{N^{\text{tr}}} \sum_{i \in \mathcal{S}^{\text{tr}}} \hat{\mu}_w^2(W_i, X_i; \mathcal{S}^{\text{tr}}, \Pi) \\
& - (1-\alpha) \cdot \left(\frac{1}{N^{\text{tr}}} + \frac{1}{N^{\text{est}}} \right) \cdot \sum_{\ell \in \Pi} \left(S^2_{\mathcal{S}^{\text{tr}}_{\text{treat}}}(\ell) + S^2_{\mathcal{S}^{\text{tr}}_{\text{control}}}(\ell) \right),
\end{aligned}
$$

3.1.4 Squared T-statistic Trees (tstats)

In sqaured t-statistic trees, we consider the splits with the largest value for square of the t-statistic for testing the null hypothesis that the average treatment effect is the same in the two potential leaves. Denote the left leaf as L and right leaf as R, the square of the t-statistic is

$$T^2 \equiv \frac{((\overline{Y}_{L1} - \overline{Y}_{L0}) - (\overline{Y}_{R1} - \overline{Y}_{R0}))^2}{S^2_{L1}/N_{L1} + S^2_{L0}/N_{L0} + S^2_{R1}/N_{R1} + S^2_{R0}/N_{R0}},$$

where $S^2_{\ell,w}$ is the conditional within treatment group sample variance given the split.

3.2 Discrete splitting

In our package, we also support discrete version of `causalTree`, which is more robust when data is big. To use discrete splitting, one should set `split.Bucket = TRUE` and specify`bucketNum`, `bucketMax`. The default value of `bucketNum = 5` and `bucketMax = 100`.

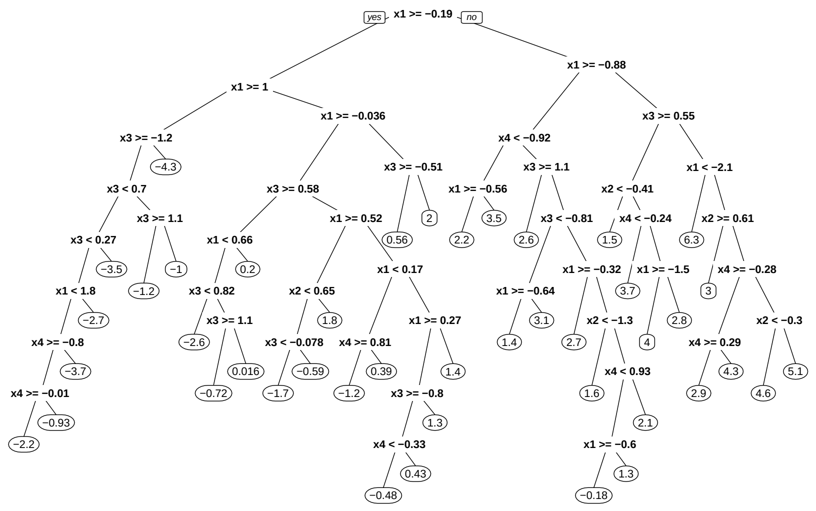
In discrete splitting, the samples in a node will first be sorted by value of a feautre and then get partitioned in to several buckets. Each bucket contains `bucketNum` observations. Then one bucket will be treated as a whole and assigned into left branch or right branch. `bucketMax` is specified as the maximum number of buckets to be used in splitting tree.

3.3 Example

The data we use in this example is a simulated data set called `simulation.1` built in `causalTree` package.

In this model, we choose **TOT** as splitting rule and **fit** as cross validation method by setting `split.Rule = "TOT"` and `cv.option = "fit"`. The propensity score (treatment probability) is set as `propensity = 0.5` for **TOT** splitting rule. We also use discrete splitting version by setting `split.Bucket = T`.

```
> library(causalTree)
> tree <- causalTree(y ~ x1 + x2 + x3 + x4, data = simulation.1,
+                    treatment = simulation.1$treatment, split.Rule = "TOT",
+                    cv.option = "fit", cv.Honest = F, split.Bucket = T,
+                    xval = 10, cv.alpha = 0.5, propensity = 0.5)
> rpart.plot(tree)
```



From the plot we can see, without pruning, the tree we get is quite large and implies overfitting. The following section will talk about different cross validation methods to prune the tree.

4 Cross Validation and Pruning

Adoptting the same idea in `rpart`, we will build cross validation trees to select a complexity parameter corresponding to the minimum cross validaiton error used for pruning. Different from `rpart`, in cross validation, we can choose different evaluation criteria to calculate the error.

4.1 Cross validation options

We offers four criteria for cross validation, **TOT**, **CT**, **fit** and **matching**. Each criterion corresponds to an evaluation function for computing cross validation error. Notice we still use the same splitting rule as before (specified by `split.Rule`) to build cross validation trees, but the cross validation error evaluation function is specified by current criterion.

4.1.1 TOT

In **TOT** cross validation method, the evaluation function is

$$\widehat{\text{MSE}}(\mathcal{S}^{\text{tr,cv}}, \mathcal{S}^{\text{tr,tr}}, \Pi) = \frac{1}{N^{\text{tr,cv}}} \sum_{i \in \mathcal{S}^{\text{tr,cv}}} \left\{ (Y_i^* - \hat{\tau}(X_i; \mathcal{S}^{\text{tr,tr}}, \Pi))^2 - Y_i^{*2} \right\}$$

where $\mathcal{S}^{\text{tr,tr}}$ is part of training sample used for building cross validaiton trees and $\mathcal{S}^{\text{tr,cv}}$ is the other part of training sample (here we called validation sample) used for predicting and calculating the error, and $N^{\text{tr,cv}}$ is the number of observations in $\mathcal{S}^{\text{tr,cv}}$.

4.1.2 CT

In **CT** cross validation method, like its splitting rule, we have two versions, adaptive and honest. We also denote them as **CT-A** and **CT-H**.
For **CT-A** cross validation method, the evaluation funciton is

$$\begin{aligned}\widehat{\text{MSE}}_{\tau}(\mathcal{S}^{\text{tr,cv}}, \mathcal{S}^{\text{tr,tr}}, \Pi) = &- \frac{2}{N^{\text{tr,cv}}} \sum_{i \in \mathcal{S}^{\text{tr,cv}}} \hat{\tau}(X_i; \mathcal{S}^{\text{tr,cv}}, \Pi) \hat{\tau}(X_i; \mathcal{S}^{\text{tr,tr}}, \Pi) \\ &+ \frac{1}{N^{\text{tr,cv}}} \sum_{i \in \mathcal{S}^{\text{tr,cv}}} \hat{\tau}^2(X_i; \mathcal{S}^{\text{tr,tr}}, \Pi).\end{aligned}$$

where $\hat{\tau}(X_i; \mathcal{S}^{\text{tr,cv}}, \Pi)$ is the treatment effect calculated through the validation sample and $\hat{\tau}(X_i; \mathcal{S}^{\text{tr,tr}}, \Pi)$ is the treatment effect in the already-built cross validation tree.
For **CT-H** cross validation method, the evaluation function is $\widehat{\text{EMSE}}_{\tau}(\mathcal{S}^{\text{tr,cv}}, N^{\text{est}}, \Pi)$, and

$$\begin{aligned}-\widehat{\text{EMSE}}_{\tau}(\mathcal{S}^{\text{tr,cv}}, N^{\text{est}}, \Pi) = &\frac{1}{N^{\text{tr,cv}}} \sum_{i \in \mathcal{S}^{\text{tr,cv}}} \hat{\tau}^2(X_i; \mathcal{S}^{\text{tr,cv}}, \Pi) \\ &- \left(\frac{1}{N^{\text{tr,cv}}} + \frac{1}{N^{\text{est}}}\right) \cdot \sum_{\ell \in \Pi} \left(\frac{S^2_{\mathcal{S}^{\text{tr,cv}}_{\text{treat}}}(\ell)}{p} + \frac{S^2_{\mathcal{S}^{\text{tr,cv}}_{\text{control}}}(\ell)}{1-p} \right).\end{aligned}$$

Like its splitting method, we also incorporate an additional factor `cv.alpha` for adjustment of two terms in the formula,

$$\begin{aligned}-\widehat{\text{EMSE}}_{\tau}(\mathcal{S}^{\text{tr,cv}}, N^{\text{est}}, \Pi, \alpha) = &\alpha \cdot \frac{1}{N^{\text{tr,cv}}} \sum_{i \in \mathcal{S}^{\text{tr,cv}}} \hat{\tau}^2(X_i; \mathcal{S}^{\text{tr,cv}}, \Pi) \\ &- (1-\alpha) \cdot \left(\frac{1}{N^{\text{tr,cv}}} + \frac{1}{N^{\text{est}}}\right) \cdot \sum_{\ell \in \Pi} \left(\frac{S^2_{\mathcal{S}^{\text{tr,cv}}_{\text{treat}}}(\ell)}{p} + \frac{S^2_{\mathcal{S}^{\text{tr,cv}}_{\text{control}}}(\ell)}{1-p} \right).\end{aligned}$$

4.1.3 fit

Like its splitting counterpart, **fit** cross validation criterion also has adaptive version **fit-A** and honest version **fit-H** to evaluate the model.
For **fit-A** criterion, the evaluation risk function is

$$\widehat{\text{MSE}}_{\mu,W}(\mathcal{S}^{\text{tr,cv}}, \mathcal{S}^{\text{tr,tr}}, \Pi) = \sum_{i \in \mathcal{S}^{\text{tr,cv}}} \left\{ (Y_i - \hat{\mu}_w(W_i, X_i; \mathcal{S}^{\text{tr,tr}}, \Pi))^2 - Y_i^2 \right\}$$

where $\hat{\mu}_w(W_i, X_i; \mathcal{S}^{\text{tr,tr}}, \Pi)$ is the mean of outcome in treatment/control group of the built cross validation tree where validation sample $(X_i, Y_i, W_i) \in \mathcal{S}^{\text{tr,cv}}$ finally be assigned.

For **fit-H** criterion, the evaluation function is $\widehat{\text{EMSE}}_{\mu,W}(\mathcal{S}^{\text{tr,cv}}, N^{\text{est}}, \Pi)$,

$$\begin{aligned}-\widehat{\text{EMSE}}_{\mu,W}(\mathcal{S}^{\text{tr,cv}}, N^{\text{est}}, \Pi) = \frac{1}{N^{\text{tr,cv}}} \sum_{i \in \mathcal{S}^{\text{tr,cv}}} \hat{\mu}_w^2(W_i, X_i; \mathcal{S}^{\text{tr,cv}}, \Pi) \\ - \left(\frac{1}{N^{\text{tr,cv}}} + \frac{1}{N^{\text{est}}}\right) \cdot \sum_{\ell \in \Pi} \left(S^2_{\mathcal{S}^{\text{tr,cv}}_{\text{treat}}}(\ell) + S^2_{\mathcal{S}^{\text{tr,cv}}_{\text{control}}}(\ell)\right),\end{aligned}$$

In contrast to the adaptive version, $\hat{\mu}_w(W_i, X_i; \mathcal{S}^{\text{tr,cv}}, \Pi)$ is the mean of outcome derived from the validation group.
Also we incorporate `cv.alpha` for adjustment in **CT-H** evaluation function,

$$\begin{aligned}-\widehat{\text{EMSE}}_{\mu,W}(\mathcal{S}^{\text{tr,cv}}, N^{\text{est}}, \Pi, \alpha) = \alpha \cdot \frac{1}{N^{\text{tr,cv}}} \sum_{i \in \mathcal{S}^{\text{tr,cv}}} \hat{\mu}_w^2(W_i, X_i; \mathcal{S}^{\text{tr,cv}}, \Pi) \\ - (1-\alpha) \cdot \left(\frac{1}{N^{\text{tr,cv}}} + \frac{1}{N^{\text{est}}}\right) \cdot \sum_{\ell \in \Pi} \left(S^2_{\mathcal{S}^{\text{tr,cv}}_{\text{treat}}}(\ell) + S^2_{\mathcal{S}^{\text{tr,cv}}_{\text{control}}}(\ell)\right),\end{aligned}$$

4.1.4 matching

In **matching** method, we first define $n(W_i, X_i; \mathcal{S})$ to be the nearest neighbor of (X_i, Y_i, W_i) in feature space with opposite W.
To be more specific, let $j = n(W_i, X_i; \mathcal{S})$, then

$$\begin{aligned}&(X_j, Y_j, W_j) \in \mathcal{S}, \\ &W_j = 1 - W_i, \\ &d_X((X_j, Y_j, W_j), (X_i, Y_i, W_i)) = \min_{\substack{(X_k, Y_k, W_k) \in \mathcal{S}, \\ W_k = 1 - W_i}} d_X((X_k, Y_k, W_k), (X_i, Y_i, W_i)).\end{aligned}$$

Then we can define the **matching** estimator of treatment effect as

$$\tau^*(X_i, W_i; \mathcal{S}) \equiv (2W_i - 1)(Y_i - Y_{n(W_i, X_i; \mathcal{S})})$$

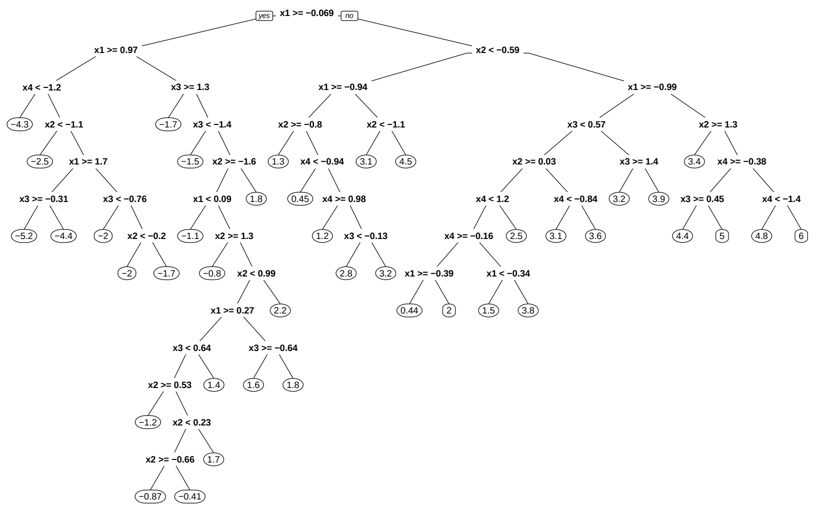
The evaluation risk function in **matching** method is

$$\widehat{\text{MSE}}_\tau(\mathcal{S}^{\text{tr,cv}}, \mathcal{S}^{\text{tr,tr}}, \Pi) = \sum_{i \in \mathcal{S}^{\text{tr,cv}}} \left(\tau^*(X_i, W_i; \mathcal{S}) - \frac{\hat{\tau}(X_i; \mathcal{S}^{\text{tr,tr}}, \Pi) + \hat{\tau}(X_{n(W_i, X_i; \mathcal{S}^{\text{tr,cv}})}; \mathcal{S}^{\text{tr,tr}}, \Pi)}{2}\right)^2$$

4.2 Example

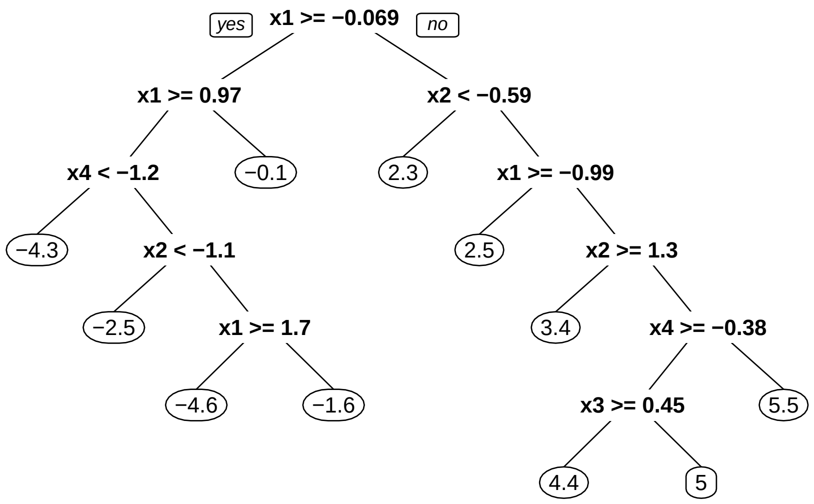
In the following example, we choose honest splitting rule as **CT-H** (`split.Rule ="CT"`, `split.Honest = T`), and cross validation method as **matching** (`cv.option = "matching"` and `cv.Honest = F`). We set 10 folds cross validation (`xval = 10`) and print out the `cptable` to check out the complexity parameter (`cp`) and normalized cross validation error (`xerror`).

```
> tree <- causalTree(y ~ x1 + x2 + x3 + x4, data = simulation.1,
+                    treatment = simulation.1$treatment, split.Rule = "CT",
+                    split.Honest = T, cv.option = "matching", cv.Honest = F,
+                    split.Bucket = F, xval = 10)
> tree$cptable
             CP nsplit rel error    xerror        xstd
1  1.145837e-02      0 1.0000000 1.0000000 0.002544452
2  1.941622e-03      1 0.9885416 0.5141833 0.001487195
3  1.862875e-03      2 0.9866000 0.5129707 0.001411806
4  1.031403e-03      3 0.9847371 0.4762972 0.001260437
5  7.763028e-04      5 0.9826743 0.4471484 0.001116234
6  5.128562e-04      8 0.9803454 0.4203597 0.001078958
7  4.327552e-04     10 0.9793197 0.4197124 0.001068104
8  3.525900e-04     11 0.9788870 0.4226628 0.001067524
9  3.052253e-04     13 0.9781818 0.4302321 0.001095403
10 2.632508e-04     14 0.9778766 0.4621311 0.001207203
11 2.198657e-04     16 0.9773500 0.4899981 0.001275376
12 1.945478e-04     17 0.9771302 0.5248392 0.001358469
13 1.887622e-04     18 0.9769356 0.5248630 0.001364536
14 1.792430e-04     19 0.9767469 0.5342639 0.001361384
15 1.677139e-04     20 0.9765676 0.5296446 0.001341866
16 1.366938e-04     25 0.9756676 0.5243033 0.001334980
17 1.111979e-04     28 0.9752575 0.5361940 0.001354982
18 1.096178e-04     32 0.9747981 0.5363380 0.001346708
19 1.002128e-04     36 0.9743597 0.5360979 0.001347304
20 7.376079e-05     37 0.9742595 0.5436176 0.001345552
21 4.858865e-05     38 0.9741857 0.5597567 0.001375235
22 2.870558e-05     39 0.9741371 0.5652205 0.001389057
23 0.000000e+00     40 0.9741084 0.5682949 0.001396325
> rpart.plot(tree)
```



The built tree in the plot is large and deep, which implies overfitting. Like `rpart`, we choose the complexity paramter `opcp` corresponding to the minimum cross validation error (`xerror`) in `tree$cptable`, and use the function `prune()` to trim the tree:

```
> opcp <- tree$cptable[, 1][which.min(tree$cptable[,4])]
> optree <- prune(tree, cp = opcp)
> rpart.plot(optree)
```

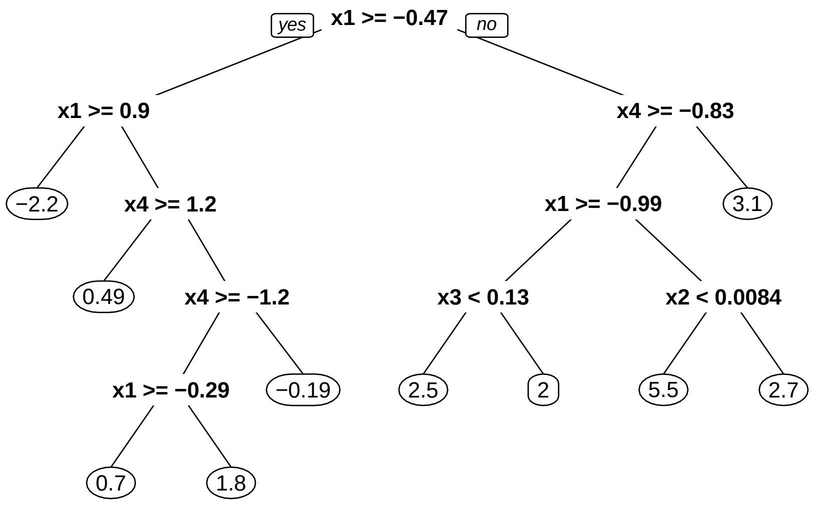


5 Honest Estimation

In addtion to causalTree, we also support one-step honest re-estimation in function honest.causalTree. It can fit a causalTree model and get honest estimation results with tree structre built on training sample (including cross validation) and leaf treatment effect estimates taken from estimation sample.

5.1 Example

```
> n <- nrow(simulation.1)
> trIdx <- which(simulation.1$treatment == 1)
> conIdx <- which(simulation.1$treatment == 0)
> train_idx <- c(sample(trIdx, length(trIdx) / 2),
+                sample(conIdx, length(conIdx) / 2))
> train_data <- simulation.1[train_idx, ]
> est_data <- simulation.1[-train_idx, ]
> honestTree <- honest.causalTree(y ~ x1 + x2 + x3 + x4, data = train_data,
+                                 treatment = train_data$treatment,
+                                 est_data = est_data,
+                                 est_treatment = est_data$treatment,
+                                 split.Rule = "CT", split.Honest = T,
+                                 HonestSampleSize = nrow(est_data),
+                                 split.Bucket = T, cv.option = "fit",
+                                 cv.Honest = F)
> opcp <-  honestTree$cptable[,1][which.min(honestTree$cptable[,4])]
> opTree <- prune(honestTree, opcp)
> rpart.plot(opTree)
```



References

[1] Susan Athey and Guido Imbens. Machine learning methods for estimating heterogeneous causal effects. *arXiv preprint arXiv:1504.01132*, 2015.

[2] L. Breiman, J.H. Friedman, R.A. Olshen, , and C.J Stone. *Classification and Regression Trees*. Wadsworth, Belmont, Ca, 1983.