Tutorial 1 - Estimating Heterogeneous Treatment Effects in Randomized Data with Machine Learning Techniques

Victor Hugo C. Alexandrino da Silva

8/11/2021

1. Goal

This tutorial discusses and implements different methods to estimate heterogeneous treatment effects (HTE) in **randomized data**:

- OLS with interaction terms
- Post-selection LASSO
- Honest Causal Tree
- Causal Forest

We will compare the heterogeneity in each of these methods and then compare the **conditional average treatment effect (CATE)** in each of these methods.

Then, we compare the causal models by the mean square error (MSE).

2. Set-up

2.1. Packages

First, let's define our directory and install required packages:

```
# Directory
setwd('~/Google Drive/PhD Insper/Thesis/Paper 3/Empirics/Tutorials/HTE Randomized')

# Packages
library(glmnet)              # LASSO
library(rpart)
library(rpart.plot)
library(randomForest)        # Random Forest
library(devtools)            # GitHub installation
library(tidyverse)
library(ggplot2)
library(dplyr)               # Data manipulation
library(grf)                 # Generalized Random Forests
#install_github('susanathey/causalTree')
```

```r
library(causalTree)           # Causal Tree
#install_github('swager/randomForestCI')
library(randomForestCI)
# install_github('swager/balanceHD')
library(balanceHD)            # Approximate residual balancing
library(SuperLearner)
library(caret)
library(xgboost)
library(sandwich)             # Robust SEs
library(ggthemes)             # Updated ggplot2 themes
library(iml)                  # For Shapley Values
```

2.2. Data

We used data from the article "Social Pressure and Voter Turnout: Evidence from a Large-Scale Field Experiment" by Gerber, Green and Larimer (2008). In a random experiment, the article study the effect of letters encouraging voters in the amount of voter turnout. The paper estimated that there is an average of 8 p.p. increase in turnout.

Data is splitted in our set of variables $Z = \{Y_i, X_i, W_i\}$, where Y is the outcome variable (voted or not), X is the treatment variable (received letter or not) and W the covariates.

```r
# Load data
my_data <- readRDS('social_voting.rds')

#Restrict the sample size
n_obs <- 33000 # Change this number depending on the speed of your computer. 6000 is also fine.
my_data <- my_data[sample(nrow(my_data), n_obs), ]

# Split data into 3 samples
folds = createFolds(1:nrow(my_data), k=2)

Y_train <- my_data[folds[[1]],1]
Y_test <- my_data[folds[[2]],1]

X_train <- my_data[folds[[1]],2]
X_test <- my_data[folds[[2]],2]

W_train <- my_data[folds[[1]],3:ncol(my_data)]
W_test <- my_data[folds[[2]],3:ncol(my_data)]

### Creates a vector of 0s and a vector of 1s of length n (hack for later usage)
zeros <- function(n) {
  return(integer(n))
}
ones <- function(n) {
  return(integer(n)+1)
}

summary(W_train)
```

```
##       g2000             g2002             p2000             p2002
```

```
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:1.0000   1st Qu.:1.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :1.0000   Median :1.0000   Median :0.0000   Median :0.0000
##  Mean   :0.8615   Mean   :0.8338   Mean   :0.2645   Mean   :0.4142
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##      p2004             sex                 yob                city
##  Min.   :0.0000   Min.   :-1.0074151   Min.   :-3.830877   Min.   :-0.55571
##  1st Qu.:0.0000   1st Qu.:-1.0074151   1st Qu.:-0.607789   1st Qu.:-0.51573
##  Median :0.0000   Median : 0.9926313   Median : 0.022815   Median :-0.47575
##  Mean   :0.4169   Mean   : 0.0004871   Mean   : 0.008339   Mean   :-0.00723
##  3rd Qu.:1.0000   3rd Qu.: 0.9926313   3rd Qu.: 0.653419   3rd Qu.:-0.07596
##  Max.   :1.0000   Max.   : 0.9926313   Max.   : 2.194896   Max.   : 3.44217
##     hh_size          totalpopulation_estimate  percent_male
##  Min.   :-1.263928   Min.   :-1.552666        Min.   :-4.316129
##  1st Qu.:-1.263928   1st Qu.:-0.894310        1st Qu.:-0.570991
##  Median : 0.121946   Median :-0.076001        Median :-0.115500
##  Mean   :-0.000766   Mean   :-0.001098        Mean   :-0.002874
##  3rd Qu.: 0.121946   3rd Qu.: 0.711651        3rd Qu.: 0.491820
##  Max.   : 7.051318   Max.   : 3.319466        Max.   :13.802246
##    median_age        percent_62yearsandover percent_white
##  Min.   :-4.149665   Min.   :-2.85640       Min.   :-7.04757
##  1st Qu.:-0.567389   1st Qu.:-0.65986       1st Qu.:-0.38908
##  Median : 0.014731   Median :-0.07711       Median : 0.39648
##  Mean   : 0.007075   Mean   : 0.01184       Mean   : 0.01434
##  3rd Qu.: 0.574461   3rd Qu.: 0.48324       3rd Qu.: 0.65833
##  Max.   : 4.671688   Max.   : 6.62458       Max.   : 0.93265
##  percent_black      percent_asian      median_income       employ_20to64
##  Min.   :-0.70036   Min.   :-0.68689   Min.   :-1.853603   Min.   :-8.349088
##  1st Qu.:-0.55341   1st Qu.:-0.52751   1st Qu.:-0.709571   1st Qu.:-0.535593
##  Median :-0.35748   Median :-0.34157   Median :-0.229037   Median : 0.208549
##  Mean   :-0.01044   Mean   :-0.01323   Mean   :-0.002571   Mean   : 0.004058
##  3rd Qu.: 0.11602   3rd Qu.: 0.05688   3rd Qu.: 0.504361   3rd Qu.: 0.645329
##  Max.   : 9.78190   Max.   : 6.72429   Max.   : 4.360707   Max.   : 2.424800
##    highschool       bach_orhigher       percent_hispanicorlatino
##  Min.   :-2.64877   Min.   :-1.848546   Min.   :-0.939459
##  1st Qu.:-0.61295   1st Qu.:-0.777360   1st Qu.:-0.491137
##  Median : 0.02464   Median :-0.116564   Median :-0.291882
##  Mean   :-0.00716   Mean   : 0.002439   Mean   : 0.001262
##  3rd Qu.: 0.75171   3rd Qu.: 0.509454   3rd Qu.: 0.081720
##  Max.   : 2.72042   Max.   : 3.347401   Max.   : 8.425500
```

3. CATE, Causal trees and causal forests

3.1. OLS with interaction terms

We first estimate our CATE using the standard OLS. Let's start with a simple OLS with interaction terms. This is a simple way to estimate differential effects of X on Y, where we only need to include interaction terms in an OLS regression. The algorithm follows:

- i) Regress Y on X and W
- ii) Interact X and W in order to find heterogeneous effects of X on Y depending on W.

Thus, we model an OLS model with interactions as the following:

$$Y = \beta_0 + \beta_1 X + \beta_2 W + \beta_3 (W \times X) + \varepsilon$$

Where X is the treatment vector and W the covariates.

We will use the R package `SuperLearner` to implement some of our ML algorithms:

```
# Estimate a linear model algorithm
sl_lm = SuperLearner(Y = Y_train,
                     X = data.frame(X=X_train, W_train , W_train*X_train),
                     family = binomial(),              # Distribution of errors
                     SL.library = "SL.lm",             # Linear model
                     cvControl = list(V=0))            # Method for cross-validation

summary(sl_lm$fitLibrary$SL.lm_All$object)
```

```
##
## Call:
## stats::lm(formula = Y ~ ., data = X, weights = obsWeights, model = model)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7502 -0.3337 -0.2162  0.5324  0.9911
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                0.1159017  0.0130087   8.910  < 2e-16 ***
## X                          0.0937039  0.0314905   2.976 0.002928 **
## g2000                     -0.0079468  0.0122491  -0.649 0.516501
## g2002                      0.0742642  0.0113419   6.548 6.01e-11 ***
## p2000                      0.0803529  0.0088812   9.048  < 2e-16 ***
## p2002                      0.1171870  0.0080620  14.536  < 2e-16 ***
## p2004                      0.1472991  0.0079381  18.556  < 2e-16 ***
## sex                       -0.0008433  0.0038217  -0.221 0.825362
## yob                       -0.0241342  0.0042466  -5.683 1.34e-08 ***
## city                       0.0390611  0.0040888   9.553  < 2e-16 ***
## hh_size                    0.0114570  0.0040762   2.811 0.004949 **
## totalpopulation_estimate   0.0133083  0.0049946   2.665 0.007717 **
## percent_male              -0.0009902  0.0046300  -0.214 0.830657
## median_age                -0.0135194  0.0090993  -1.486 0.137360
## percent_62yearsandover     0.0254651  0.0089952   2.831 0.004646 **
## percent_white              0.0774998  0.0221581   3.498 0.000471 ***
## percent_black              0.0528264  0.0163273   3.235 0.001217 **
## percent_asian              0.0335540  0.0114494   2.931 0.003387 **
## median_income              0.0335975  0.0087292   3.849 0.000119 ***
## employ_20to64             -0.0174929  0.0054288  -3.222 0.001274 **
## highschool                 0.0459277  0.0139517   3.292 0.000997 ***
## bach_orhigher              0.0127597  0.0151862   0.840 0.400799
## percent_hispanicorlatino   0.0120913  0.0064752   1.867 0.061873 .
## g2000.1                   -0.0525300  0.0293052  -1.793 0.073068 .
## g2002.1                    0.0355025  0.0275901   1.287 0.198187
## p2000.1                   -0.0401641  0.0217183  -1.849 0.064429 .
## p2002.1                    0.0072293  0.0199228   0.363 0.716710
```

```
## p2004.1                      0.0491967  0.0195445   2.517 0.011840 *
## sex.1                       -0.0169138  0.0094063  -1.798 0.072172 .
## yob.1                       -0.0046595  0.0102981  -0.452 0.650945
## city.1                       0.0115392  0.0102406   1.127 0.259839
## hh_size.1                   -0.0036811  0.0099984  -0.368 0.712750
## totalpopulation_estimate.1  0.0066776  0.0120675   0.553 0.580025
## percent_male.1               0.0035270  0.0113922   0.310 0.756873
## median_age.1                -0.0024973  0.0234107  -0.107 0.915049
## percent_62yearsandover.1   -0.0083971  0.0228161  -0.368 0.712853
## percent_white.1             -0.0067575  0.0487593  -0.139 0.889776
## percent_black.1              0.0081093  0.0365158   0.222 0.824257
## percent_asian.1              0.0137792  0.0257635   0.535 0.592771
## median_income.1             -0.0127917  0.0222992  -0.574 0.566221
## employ_20to64.1             -0.0004990  0.0135748  -0.037 0.970679
## highschool.1                -0.0579221  0.0345309  -1.677 0.093483 .
## bach_orhigher.1             -0.0690033  0.0377846  -1.826 0.067834 .
## percent_hispanicorlatino.1 -0.0008732  0.0149472  -0.058 0.953415
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4475 on 16456 degrees of freedom
## Multiple R-squared:  0.07718,    Adjusted R-squared:  0.07477
## F-statistic: 32.01 on 43 and 16456 DF,  p-value: < 2.2e-16
```

Thus, note that the treatment X itself has a positive statistically significant effect in the vote turnout. However, we observe statistically significant heterogeneous effect when we interact with some of the covariates. It seems that the effect is heterogeneous for our variables. Moreover, a bunch of features seems to be relevant for our linear model, which reinforces the need to estimate the CATE instead of the ATE. That is why we need one step forward.

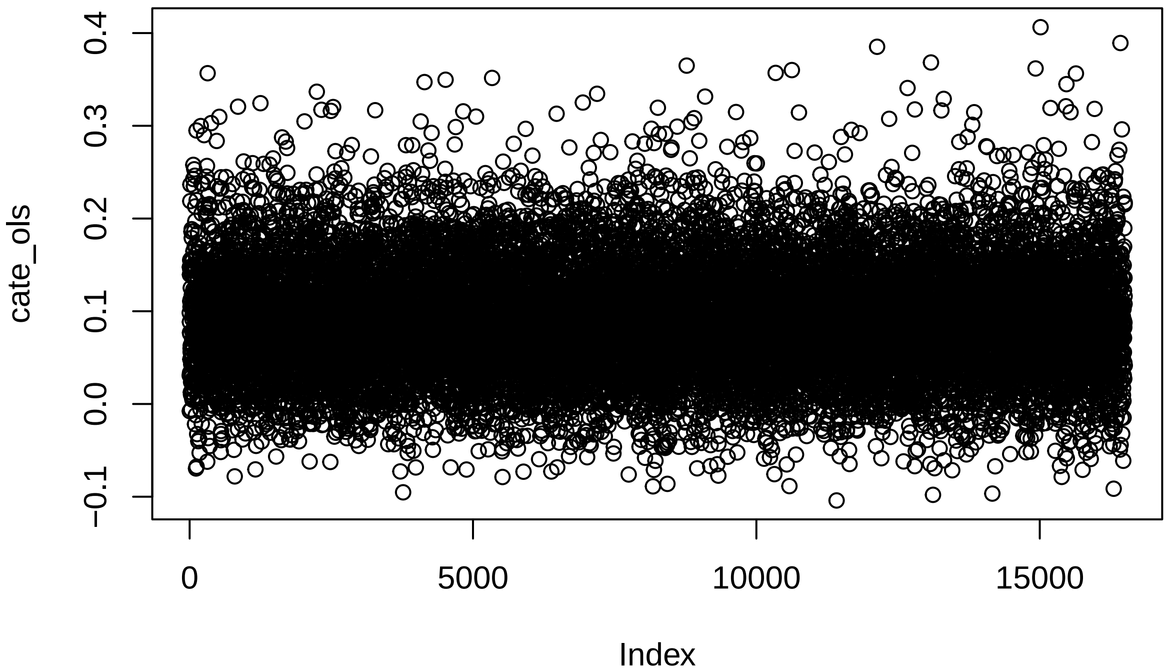
3.1.1. CATE for OLS

We can simply predict our outcome for both treated and non-treated groups in order to estimate the CATE:

$$CATE = \hat{\tau} = E(Y|X=1,W) - E(Y|X=0,W)$$

For this, we use the `predict` function in our `sl_lm` model for each group, using our test sample:

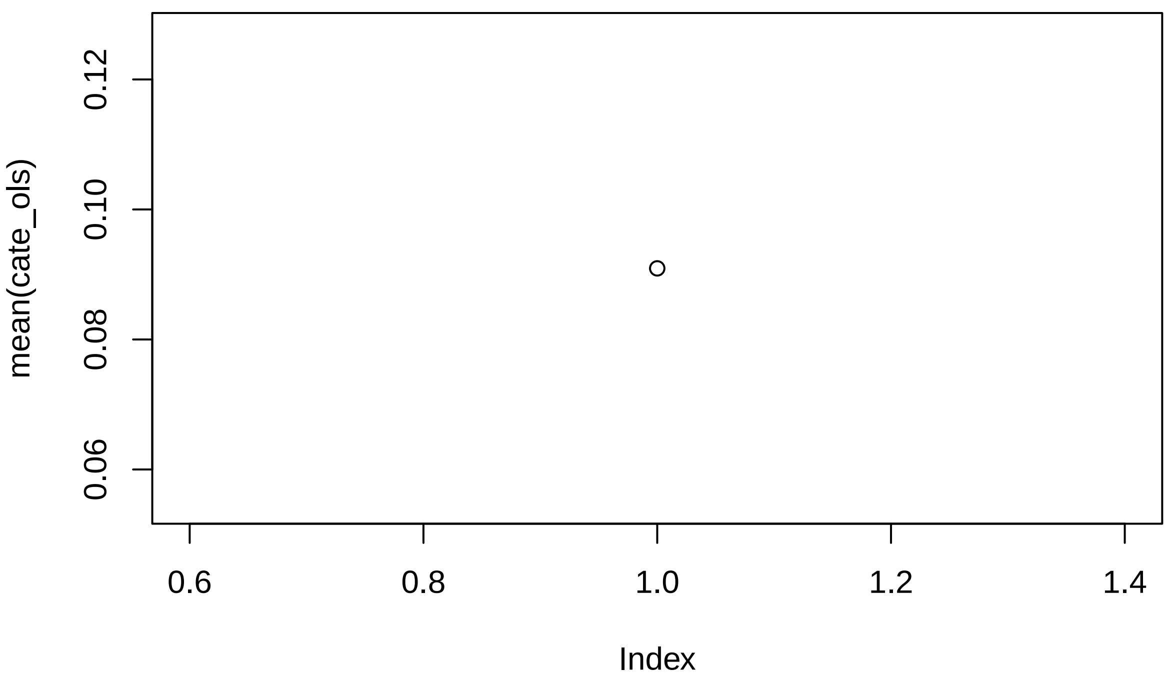
```
# Prediction on control group (X = 0)
ols_pred_control <- predict(sl_lm, data.frame(X = zeros(nrow(W_test)), W_test, W_test*zeros(nrow(W_test)

# Prediction on treated group (X = 1)
ols_pred_treated <- predict(sl_lm, data.frame(X = ones(nrow(W_test)), W_test, W_test*ones(nrow(W_test))

# Calculate CATE
cate_ols <- ols_pred_treated$pred - ols_pred_control$pred

plot(cate_ols)
```




```
# Calculate ATE
mean(cate_ols)
```

```
## [1] 0.09093511
```

```
plot(mean(cate_ols))
```



3.2. Post-selection LASSO

Now, we use LASSO to estimate the heterogeneous effects. However, before estimating the CATE, we use it as a screening algorithm. What does it mean? That in order to reduce the number of variables, we can use LASSO to select the relevant variables. We will use the SuperLearner library again:

```
# Defining LASSO
lasso = create.Learner("SL.glmnet",
                       params = list(alpha = 1),
                       name_prefix = "lasso")


# Getting coefficients by LASSO
get_lasso_coeffs <- function(sl_lasso) {
  return(coef(sl_lasso$fitLibrary$lasso_1_All$object, se = "lambda.min")[-1,])
}


SL.library <- lasso$names

predict_y_lasso <- SuperLearner(Y = Y_train,
                                X = data.frame(X = X_train, W_train ,W_train*X_train),
                                family =  binomial(),
                                SL.library = SL.library,
                                cvControl = list(V=0))

kept_variables <- which(get_lasso_coeffs(predict_y_lasso)!=0)


predict_x_lasso <- SuperLearner(Y = X_train,
                                X = data.frame(W_train),
                                family = binomial(),
                                SL.library = lasso$names,
                                cvControl = list(V=0))


kept_variables2 <- which(get_lasso_coeffs(predict_x_lasso)!=0) + 1
```

After selecting the variables by LASSO, we can use the OLS to estimate treatment heterogeneity in the relevant variables selected by the algorithm above. But first, let's formulate our post selection OLS:

```
sl_post_lasso <- SuperLearner(Y = Y_train,
                              X = data.frame(X = X_train, W_train, W_train*X_train)[,c(kept_variables,k
                              family = binomial(),
                              SL.library = "SL.lm",
                              cvControl = list(V=0))

summary(sl_post_lasso$fitLibrary$SL.lm_All$object)
```

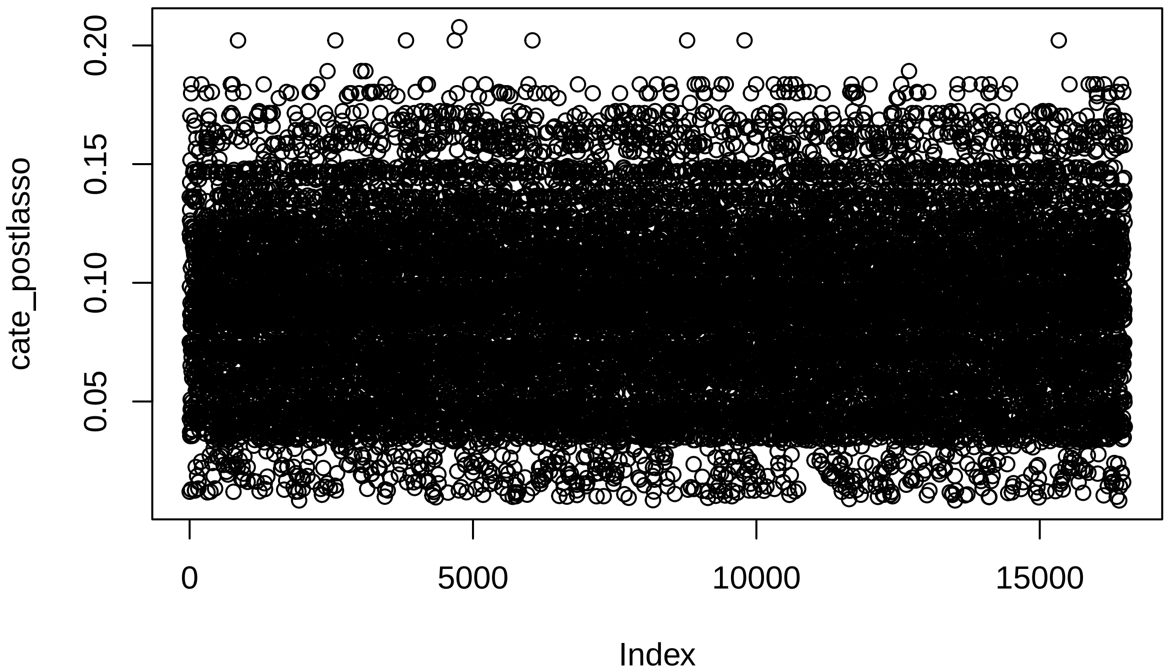
```
##
## Call:
## stats::lm(formula = Y ~ ., data = X, weights = obsWeights, model = model)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7126 -0.3346 -0.2204  0.5411  0.9804
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 0.117735   0.010400  11.321  < 2e-16 ***
## X                           0.047167   0.024097   1.957  0.05032 .
## g2002                       0.069894   0.010765   6.493 8.66e-11 ***
## p2000                       0.070808   0.008037   8.810  < 2e-16 ***
## p2002                       0.117171   0.007352  15.937  < 2e-16 ***
## p2004                       0.141296   0.007827  18.053  < 2e-16 ***
## yob                        -0.021578   0.003687  -5.853 4.93e-09 ***
## city                        0.039112   0.003578  10.932  < 2e-16 ***
## percent_62yearsandover      0.007354   0.003712   1.981  0.04760 *
## employ_20to64              -0.018177   0.003787  -4.799 1.61e-06 ***
## g2002.1                     0.024474   0.024863   0.984  0.32495
## p2004.1                     0.052418   0.019050   2.752  0.00594 **
## totalpopulation_estimate.1  0.025194   0.008681   2.902  0.00371 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4483 on 16487 degrees of freedom
## Multiple R-squared:  0.07228,    Adjusted R-squared:  0.0716
## F-statistic:   107 on 12 and 16487 DF,  p-value: < 2.2e-16
```

That is, now, after selecting by LASSO our most relevant variables, we note heterogeneity in some of our variables. However, we have a set of relevant variables different from the treatment that are also statistically significant.

What is remaining is our estimation of CATE for post-selection LASSO. We can code it as the following, using the test sample:

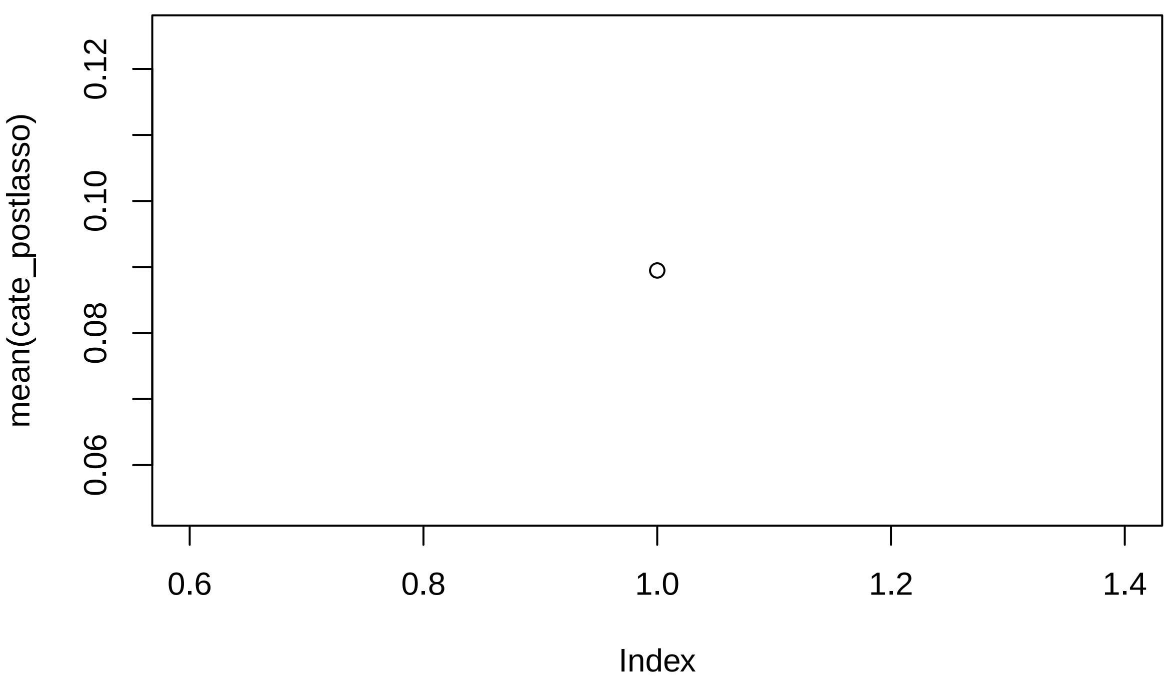
```r
# Prediction on control group (X = 0)
postlasso_pred_control <- predict(sl_post_lasso, data.frame(X = zeros(nrow(W_test)), W_test, W_test*zero

# Prediction on control group (X = 1)
postlasso_pred_treated <- predict(sl_post_lasso, data.frame(X = ones(nrow(W_test)), W_test, W_test*ones

# Estimating CATE with post-selection LASSO
cate_postlasso <- postlasso_pred_treated$pred - postlasso_pred_control$pred

# Plot cate_postlasso
plot(cate_postlasso)
```




```r
# ATE
mean(cate_postlasso)
```

```
## [1] 0.08947195
```

```r
# Plot ATE
plot(mean(cate_postlasso))
```



3.3. Causal Trees

Now we are going to predict the CATE from tree-based algorithms. There are a few packages that do this, but I like Susan Athey's `causalTree` and `grf`. The first is a general algorithm that builds a regression model and returns an *rpart* object, implementing ideas from the CART (Classification and Regression Trees), by Breiman et al. The second implements the generalized random forest algorithm for causal forest, which uses a splitting tree-based rule to divide covariates based in the heterogeneity of treatment effects and, moreover, assumes honesty as one of main assumptions.

In summary, we want to model

$$Y_i = W_i + \theta X_i + \varepsilon$$

Where c are possible covariates that brings hetrogeneity on the treatment X_i for the outcome Y_i. The algorithm finds

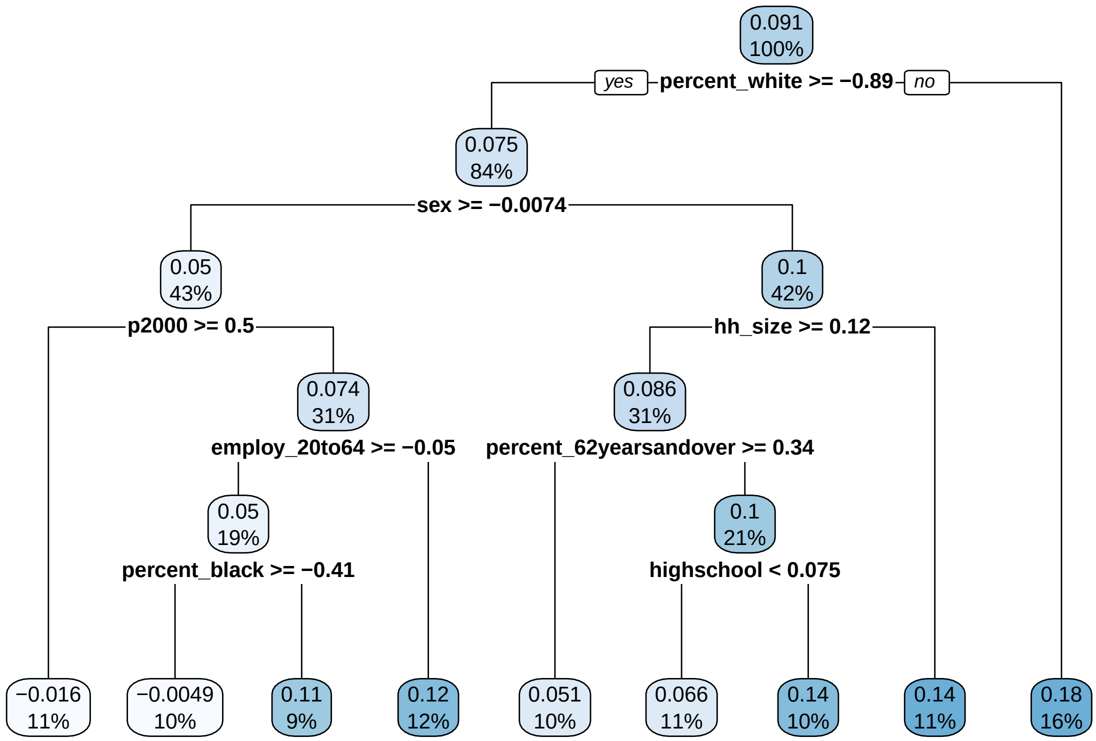
$$\hat{\tau}(W) = argmin_{\tau} \alpha_i(W) W_i (Y_i - \tau X_i)^2$$

Where the weights α_i are estimated by a random forest algorithm. Let's begin by building our causal tree:

```
# Witting the regression formula (to facilitate later)
tree_fml <- as.formula(paste("Y", paste(names(W_train), collapse = ' + '), sep = " ~ "))

# Building causal tree
causal_tree <- causalTree(formula = tree_fml,
                          data = data.frame(Y = Y_train,W_train),
                          treatment = X_train,
                          split.Rule = "CT",          # Causal Tree
                          split.Honest = FALSE,       # So far, we are not assuming honesty
                          split.alpha = 1,
                          cv.option = "CT",
                          cv.Honest = FALSE,
                          split.Bucket = TRUE,
                          bucketNum = 5,
                          bucketMax = 100,
                          minsize = 250               # Number of obs in treatment and control on leaf
                          )
```

```
## [1] 6
## [1] "CTD"
```

```r
rpart.plot(causal_tree, roundint = FALSE)
```



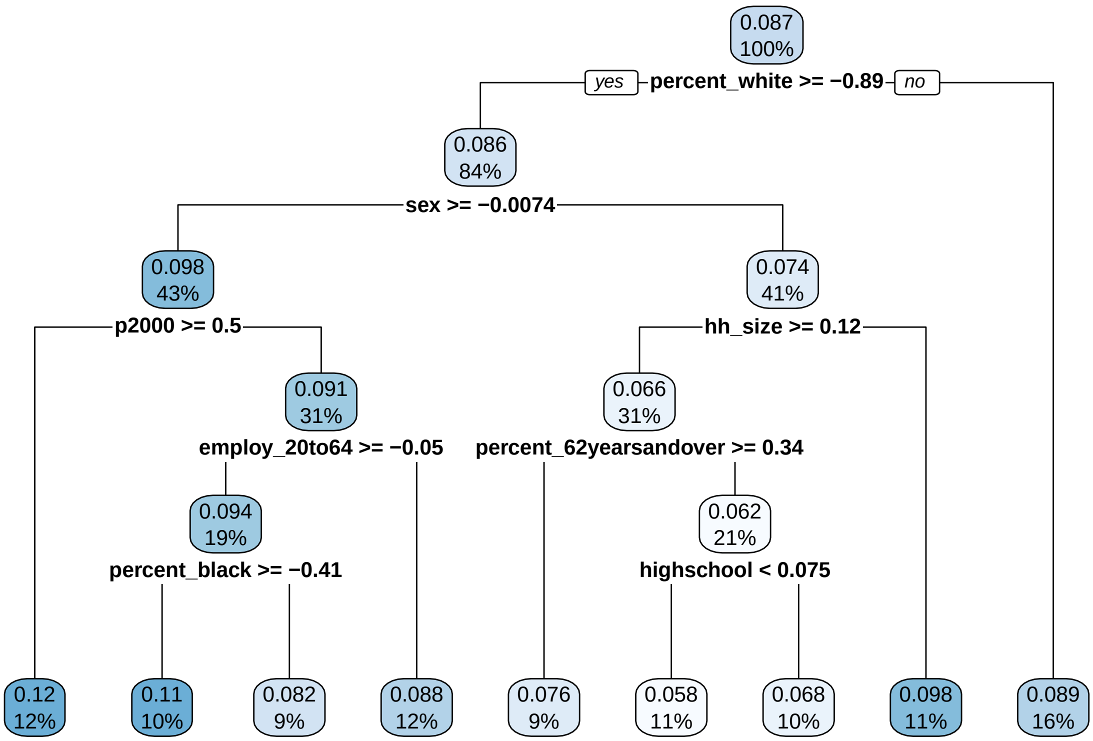
3.3.1. Honest Causal Trees

Honest trees can be obtained when we use part of the sample (train) for building the leafs and part (test) to calculate the heterogeneity of treatment effects. The function `honest.causalTree` does the job:

```r
honest_tree <- honest.causalTree(formula = tree_fml,
                                 data = data.frame(Y=Y_train, W_train),
                                 treatment = X_train,
                                 est_data = data.frame(Y=Y_test, W_test),
                                 est_treatment = X_test,
                                 split.alpha = 0.5,
                                 split.Rule = "CT",
                                 split.Honest = TRUE,
                                 cv.alpha = 0.5,
                                 cv.option = "CT",
                                 cv.Honest = TRUE,
                                 split.Bucket = TRUE,
                                 bucketNum = 5,
                                 bucketMax = 100, # maximum number of buckets
                                 minsize = 250) # number of observations in treatment and control on le
```

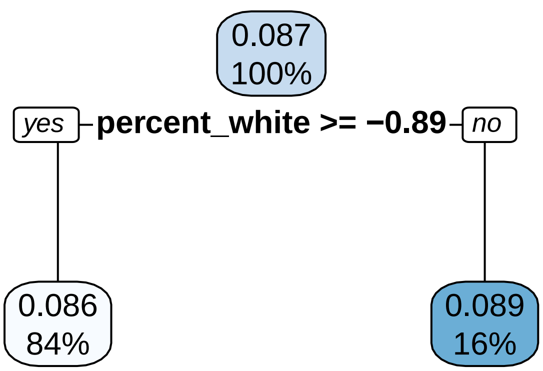
```
## [1] 6
## [1] "CTD"
```

```r
rpart.plot(honest_tree, roundint = F)
```



Then, we prune the tree with cross validation, choosing the simplest tree that minimizes the objective function in a left-out sample:

```r
opcpid <- which.min(honest_tree$cp[, 4])
opcp <- honest_tree$cp[opcpid, 1]
honest_tree_prune <- prune(honest_tree, cp = opcp)

rpart.plot(honest_tree_prune, roundint = F)
```



That is, we have spitted our sample in order to maximize the heterogeneity in each node. That is the way these algorithms work: By splitting in sub-samples by heterogeneity, we are able to estimate with more accuracy the treatment effect.

To estimate the standard errors on the leaves, we can use an OLS. The linear regression is specified such that the coefficients on the leaves are the treatment effects.

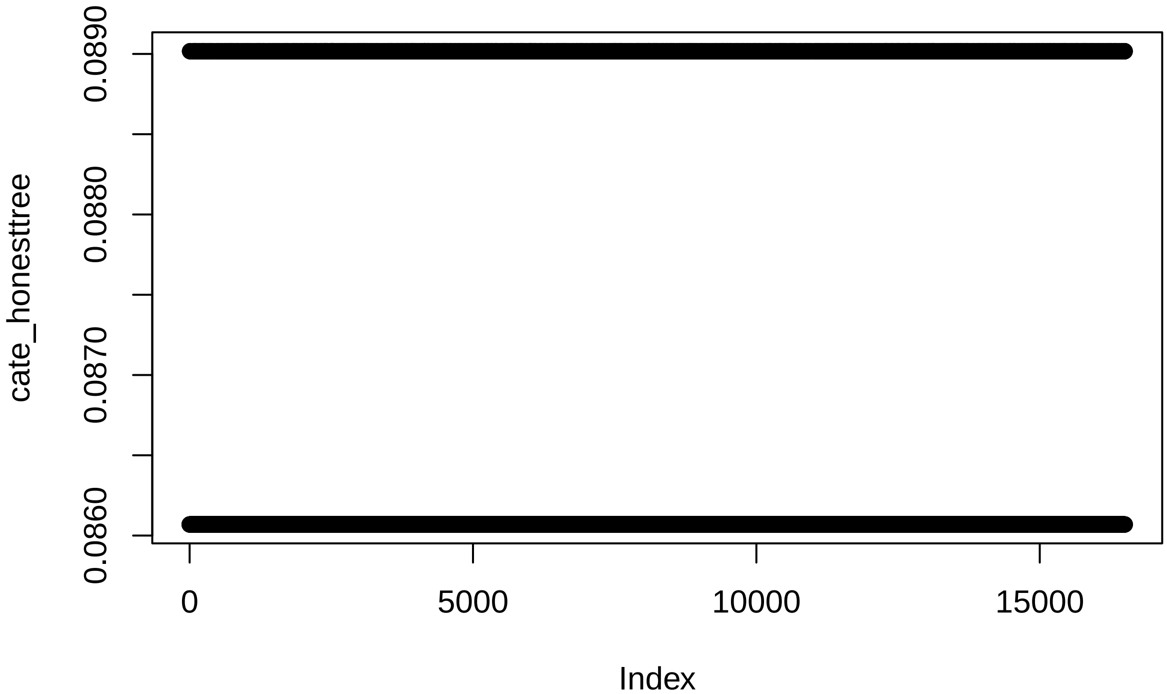
```r
# Constructing factors variables for the leaves
leaf_test <- as.factor(round(predict(honest_tree_prune,
                                     newdata = data.frame(Y = Y_test, W_test),
                                     type = "vector"), 4))

# Run an OLS that estimate the treatment effect magnitudes and standard errors
honest_ols_test <- lm(Y ~ leaf + X * leaf - X -1, data = data.frame(Y = Y_test, X = X_test, leaf = leaf_

summary(honest_ols_test)
```

```
##
## Call:
## lm(formula = Y ~ leaf + X * leaf - X - 1, data = data.frame(Y = Y_test,
##     X = X_test, leaf = leaf_test, W_test))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.3914 -0.3054 -0.3054  0.6946  0.7026
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## leaf0.0861   0.305349   0.004317  70.728  < 2e-16 ***
## leaf0.089    0.297347   0.009941  29.912  < 2e-16 ***
## leaf0.0861:X 0.086069   0.010640   8.089 6.44e-16 ***
## leaf0.089:X  0.089017   0.024285   3.665 0.000248 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4648 on 16496 degrees of freedom
## Multiple R-squared:  0.3216, Adjusted R-squared:  0.3215
## F-statistic:  1955 on 4 and 16496 DF,  p-value: < 2.2e-16
```

That is, all the heterogeneities in our pruned tree are relevant in our honest tree.

But we still need to predict our CATE from our tree. This is easily done by the function predict:

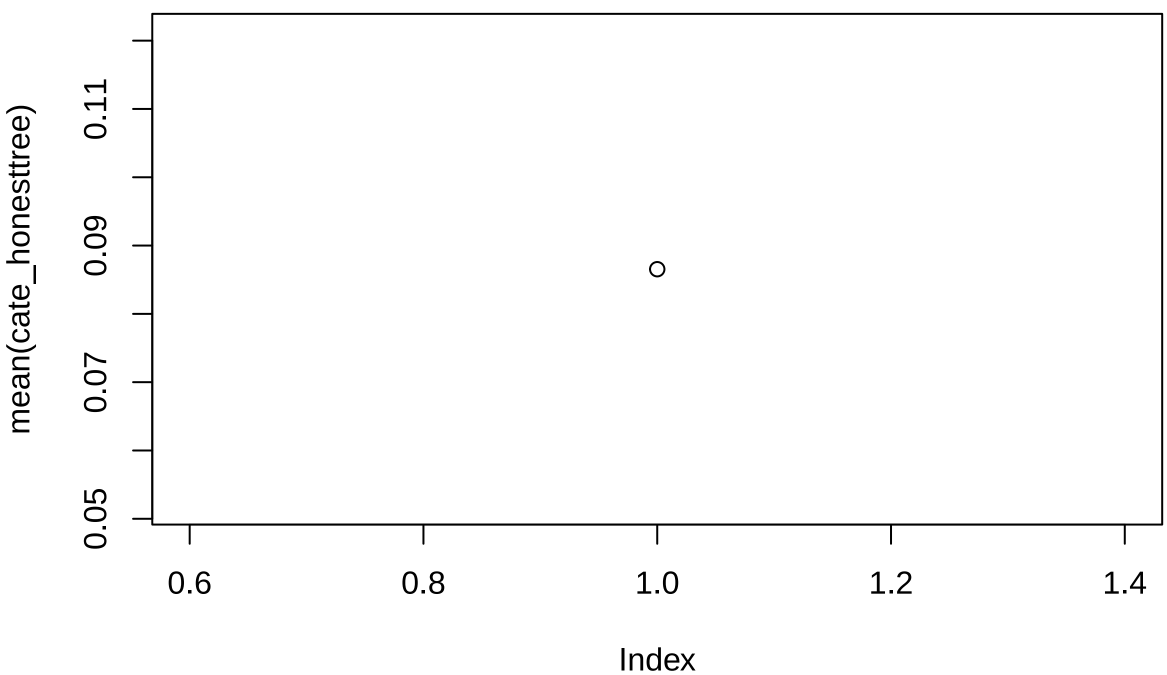
```r
# Estimate CATE
cate_honesttree <- predict(honest_tree_prune, newdata = data.frame(Y = Y_test, W_test), type = "vector"

# Plot CATE
plot(cate_honesttree)
```




```r
# ATE
mean(cate_honesttree)
```

```
## [1] 0.08653827
```

```r
# Plot ATE
plot(mean(cate_honesttree))
```



3.4. Causal Forests

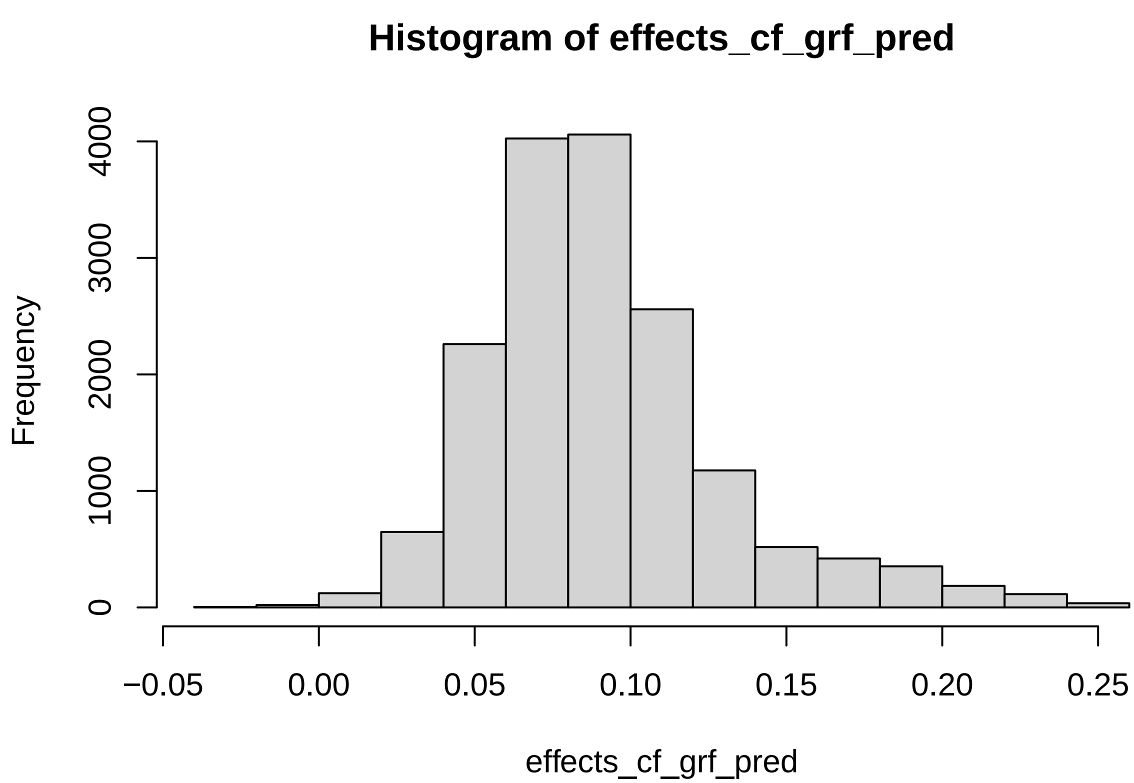
Finally, we estimate the CATE with the causal forest algorithm. The method is similar to the R-learner, but with a splitting procedure using a tree-based algorithm. It uses a residual-residual approach do estimate the propensity score in order to find the conditional average treatment effect. For more information, check Jacob (2021).

Let's do in two ways. The first one will be using the `grf` package that estimates the CATE directly in the function. The second uses the `causalForest` package, estimating a random forest with honest causal trees.

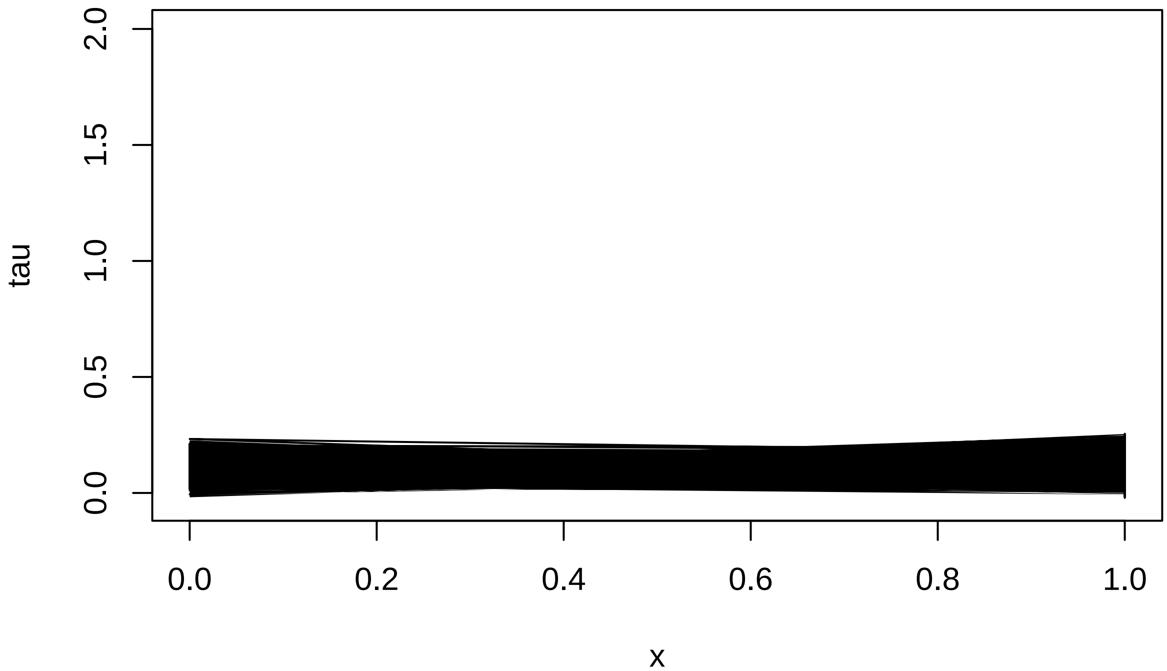
3.4.1. Generalized Random Forest

The `grf` algorithm assumes honesty as the main assumption. We can easily do this by fitting the causal forest with our training sample and predicting with our testing sample. Estimating our causal forest is simply:

```
# Train our causal forest
cf_grf <- causal_forest(X = W_train,
                        Y = Y_train,
                        W = X_train)

# Get predictions from test sample
effects_cf_grf <- predict(cf_grf,W_test)

# Get effects
effects_cf_grf_pred <- predict(cf_grf,W_test)$predictions

# Histogram
hist(effects_cf_grf_pred)
```



```r
# Plot of treatment effects
plot(W_test[, 1], effects_cf_grf_pred, ylim = range(effects_cf_grf_pred, 0, 2), xlab = "x", ylab = "tau
```




```r
# Estimate the CATE for the full sample
cate_grf <- average_treatment_effect(cf_grf, target.sample = "all")

cate_grf
```

```
##   estimate    std.err
## 0.09036008 0.00972967
```

```r
# Estimate the CATE for the treated sample (CATT)
average_treatment_effect(cf_grf, target.sample = "treated")
```

```
##    estimate     std.err
## 0.091763050 0.009707333
```

```r
# Best Linear Projection of the CATE
cate_best_grf <- best_linear_projection(cf_grf)

cate_best_grf
```

```
##
## Best linear projection of the conditional average treatment effect.
## Confidence intervals are cluster- and heteroskedasticity-robust (HC3):
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 0.0903502  0.0098623  9.1612 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

That is, `cate_best_grf` and `cate_grf` match each other, both when we predict the result using the `average_treatment_effect` function or when we use the `best_linear_projection`. Let's see if it holds without the `grf` package.

3.4.2. causalTree package

The same estimation can be obtained by the function `causalForest` inside the `causalTree` package:

```
cf_causalTree <- causalForest(tree_fml,
                              data=data.frame(Y=Y_train, W_train),
                              treatment=X_train,
                              split.Rule="CT",
                              split.Honest=T,
                              split.Bucket=T,
                              bucketNum = 5,
                              bucketMax = 100,
                              cv.option="CT",
                              cv.Honest=T,
                              minsize = 2,
                              split.alpha = 0.5,
                              cv.alpha = 0.5,
                              sample.size.total = floor(nrow(Y_train) / 2),
                              sample.size.train.frac = .5,
                              mtry = ceiling(ncol(W_train)/3),
                              nodesize = 5,
                              num.trees = 10,
                              ncov_sample = ncol(W_train),
                              ncolx = ncol(W_train))
```

```
## [1] "Building trees ..."
## [1] "Tree 1"
## [1] 6
## [1] "CTD"
## [1] "Tree 2"
## [1] 6
## [1] "CTD"
## [1] "Tree 3"
## [1] 6
## [1] "CTD"
## [1] "Tree 4"
## [1] 6
## [1] "CTD"
## [1] "Tree 5"
## [1] 6
## [1] "CTD"
## [1] "Tree 6"
## [1] 6
## [1] "CTD"
## [1] "Tree 7"
## [1] 6
## [1] "CTD"
## [1] "Tree 8"
## [1] 6
```

```
## [1] "CTD"
## [1] "Tree 9"
## [1] 6
## [1] "CTD"
## [1] "Tree 10"
## [1] 6
## [1] "CTD"
```

And, to estimate the CATE:

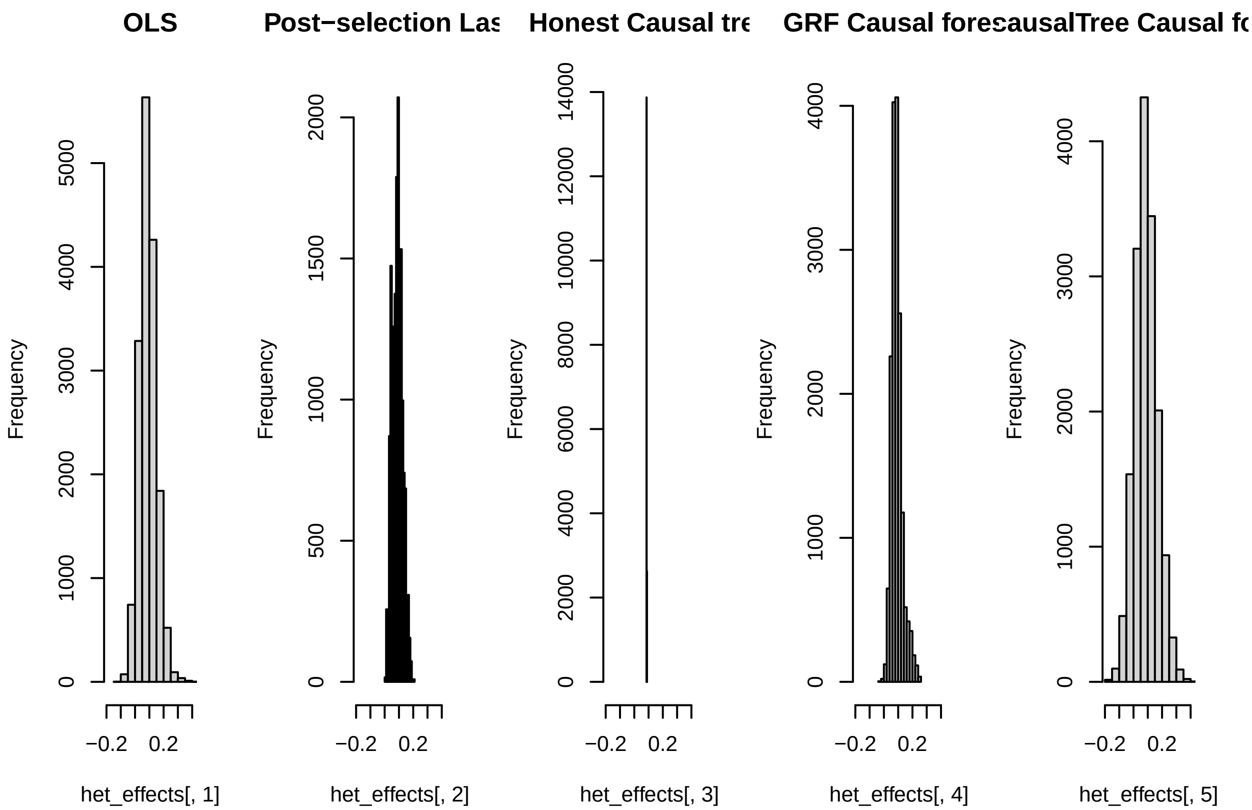
```
cate_causalTree <- predict(cf_causalTree, newdata = data.frame(Y = Y_test, W_test), type = "vector")
```

```
## [1] 16500    10
```

4. Comparing models

Finally, we can compare all of our models (OLS with interaction terms, post-selection LASSO, Causal trees and Causal forest). Let's first see some histograms:

```
# Creating a data frame withh all CATEs
het_effects <- data.frame(ols = cate_ols,
                          post_selec_lasso = cate_postlasso,
                          causal_tree = cate_honesttree,
                          causal_forest_grf = effects_cf_grf,
                          causal_forest_causalTree = cate_causalTree)

# Set range of x-axis
xrange <- range(c(het_effects[,1],het_effects[,2],het_effects[,3],het_effects[,4],het_effects[,5]))

# Set margins (two rows, five columns)
par(mfrow = c(1,5))

hist(het_effects[, 1], main = "OLS", xlim = xrange)
hist(het_effects[, 2], main = "Post-selection Lasso", xlim = xrange)
hist(het_effects[, 3], main = "Honest Causal tree", xlim = xrange)
hist(het_effects[, 4], main = "GRF Causal forest", xlim = xrange)
hist(het_effects[, 5], main = "causalTree Causal forest", xlim = xrange)
```



And to finalize, we can summary a table of results with each of CATEs:

```
summary_stats <- do.call(data.frame,
                         list(mean = apply(het_effects, 2, mean),
                              sd = apply(het_effects, 2, sd),
                              median = apply(het_effects, 2, median),
                              min = apply(het_effects, 2, min),
                              max = apply(het_effects, 2, max)))

summary_stats
```

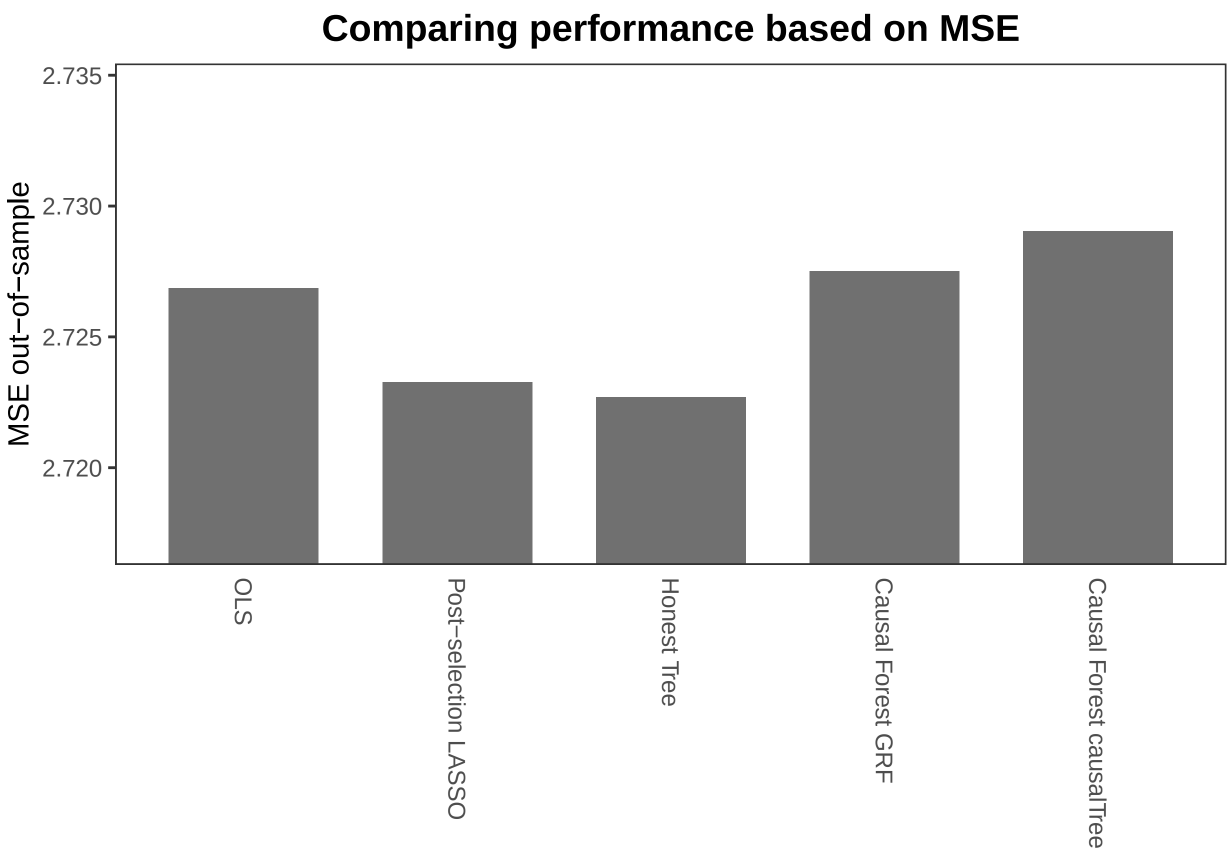
```
##                              mean          sd     median          min
## ols                    0.09093511 0.059320657 0.08804199 -0.104018590
## post_selec_lasso       0.08947195 0.035956936 0.08966228  0.008310022
## causal_tree            0.08653827 0.001078374 0.08606912  0.086069125
## predictions            0.09119050 0.038813514 0.08542703 -0.038330591
## causal_forest_causalTree 0.08711032 0.079410682 0.08349570 -0.192965648
##                              max
## ols                    0.40632133
## post_selec_lasso       0.20768888
## causal_tree            0.08901688
## predictions            0.25528656
## causal_forest_causalTree 0.40673882
```

From the histograms and the summary statistics, it seems that the causal forest from the `grf` package yields most heterogeneity, once we have higher standard deviation among our treatment effects.

We can also compare the methods by looking at the minimum square error (MSE) on a test set using the transformed outcome (which I call Y^*) as a proxy for the true treatment effect. For this, we need to construct the propensity score ($E(X = 1)$) of our treatment in our test sample. Let's code it:

```r
# Construct propensity score from randomized experiment
prop_score <- mean(X_test)

# Construct Y_star in our test sample
Y_star <- X_test * (Y_test / prop_score) - (1 - X_test) * (Y_test / (1 - prop_score))

## MSEs

# OLS with interaction
MSE_ols <- mean((Y_star - cate_ols)^2)

# Post-selection LASSO
MSE_lasso <- mean((Y_star - cate_postlasso)^2)

# Honest Tree
MSE_causalTree <- mean((Y_star - cate_honesttree)^2)

# Causal Forest GRF
MSE_cf_grf <- mean((Y_star - cate_grf)^2)

# Causal Forest causalTree
MSE_cf_causalTree <- mean((Y_star - cate_causalTree)^2)

# Create data frame with all MSEs
performance_MSE <- data.frame(matrix(rep(NA,1), nrow = 1, ncol = 1))
rownames(performance_MSE) <- c("OLS")
colnames(performance_MSE) <- c("MSE")

# Load in results
performance_MSE["OLS","MSE"] <- MSE_ols
performance_MSE["Post-selection LASSO","MSE"] <- MSE_lasso
performance_MSE["Honest Tree","MSE"] <- MSE_causalTree
performance_MSE["Causal Forest GRF","MSE"] <- MSE_cf_grf
performance_MSE["Causal Forest causalTree","MSE"] <- MSE_cf_causalTree

# Setting range
xrange2 <- range(performance_MSE$MSE - 2*sd(performance_MSE$MSE),
                 performance_MSE$MSE,
                 performance_MSE$MSE + 2*sd(performance_MSE$MSE))

# Create plot
MSEplot <- ggplot(performance_MSE) +
  geom_bar(mapping = aes(x = factor(rownames(performance_MSE),
                                    levels = rownames(performance_MSE)),
                         y = MSE),
           stat = "identity", fill = "gray44", width=0.7,
           position = position_dodge(width=0.2)) +
  theme_bw() +
  coord_cartesian(ylim=c(xrange2[1], xrange2[2])) +
  theme(axis.ticks.x = element_blank(), axis.title.x = element_blank(),
        panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
```

```
        plot.background = element_blank(),
        axis.text.x = element_text(angle = -90, hjust = 0, vjust = 0.5)) +
  ylab("MSE out-of-sample") +
  ggtitle("Comparing performance based on MSE") +
  theme(plot.title = element_text(hjust = 0.5, face ="bold",
                                  colour = "black", size = 14))

# Plot
MSEplot
```



From the MSE analysis, it seems that both causal forest models perform worse than simpler models like OLS, post-selection LASS) and honest tree. One explanation could be that since we have little heterogeneity in our model, simpler models do best.

6. Interpretable Machine Learning

Finally, let's do a visual analysis on the feature importance of our variables in the voters' turnout. For this, we will use the `iml`, a nice package that for interpretable machine learning in R.

One of the most interesting features from the `iml` package is the possibility of plotting Shapley Values - a method from coalitional game theory - tells us how to fairly distribute the "payout" among the covariates.

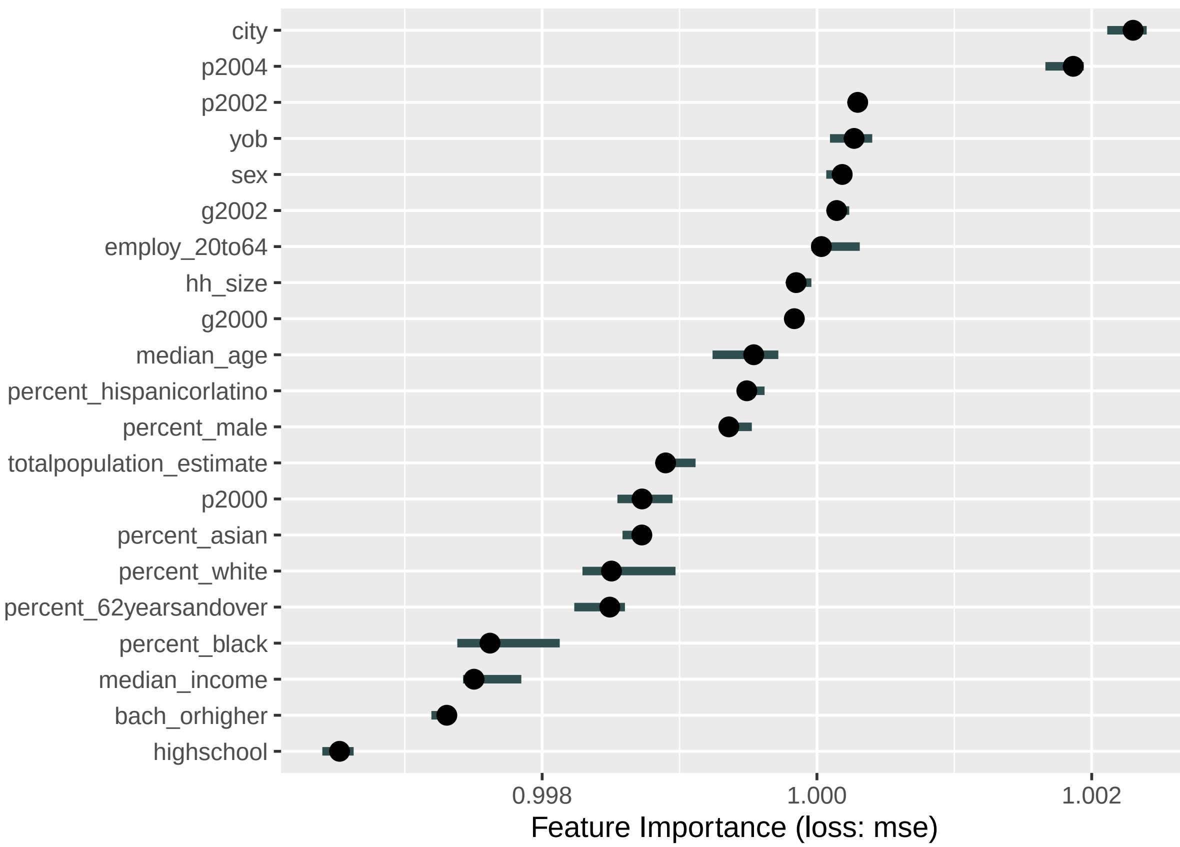
6.1. Feature importance

First, let's create a `Predictor` object. This is necessary since the `iml` package uses R6 classes, where New objects must be created from estimated machine learning models that holds the model itself and the data. This is done as the following:

```
# Using the iml Predictor() container
cf_predictor <- Predictor$new(cf_grf, data = W_train, y = Y_train)
```

We can measure how each feature is important for the prediction of our causal forest with the function `FeatureImp`. The feature importance measure works by shuffling each feature and measuring how much the performance drops.

```
cf_importance <- FeatureImp$new(cf_predictor, loss = "mse")

plot(cf_importance)
```




```
cf_importance$results
```

```
##                    feature importance.05 importance importance.95
## 1                     city     1.0021130  1.0023011     1.0024001
## 2                    p2004     1.0016629  1.0018654     1.0019420
## 3                    p2002     1.0002746  1.0002966     1.0003275
## 4                      yob     1.0000949  1.0002705     1.0004026
## 5                      sex     1.0000680  1.0001840     1.0002548
```

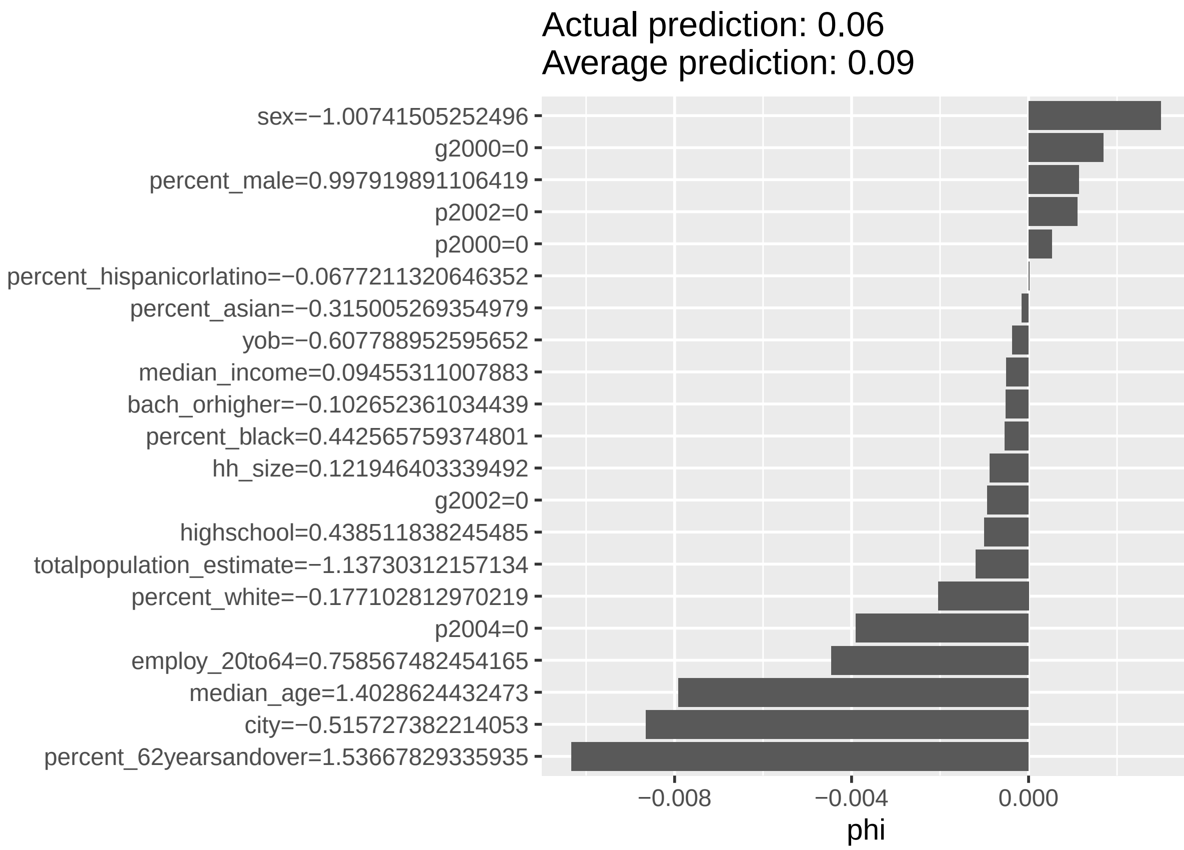
```
## 6                     g2002     1.0000904  1.0001440     1.0002350
## 7              employ_20to64     0.9999594  1.0000324     1.0003118
## 8                    hh_size     0.9998215  0.9998488     0.9999596
## 9                      g2000     0.9997882  0.9998356     0.9998917
## 10                median_age     0.9992402  0.9995400     0.9997189
## 11 percent_hispanicorlatino     0.9994660  0.9994897     0.9996189
## 12              percent_male     0.9992973  0.9993587     0.9995261
## 13 totalpopulation_estimate     0.9988692  0.9988986     0.9991167
## 14                     p2000     0.9985478  0.9987273     0.9989490
## 15             percent_asian     0.9985849  0.9987257     0.9987828
## 16             percent_white     0.9982935  0.9985047     0.9989705
## 17   percent_62yearsandover     0.9982337  0.9984924     0.9986026
## 18             percent_black     0.9973825  0.9976203     0.9981281
## 19             median_income     0.9974247  0.9975043     0.9978483
## 20             bach_orhigher     0.9971938  0.9973063     0.9973462
## 21                highschool     0.9964001  0.9965255     0.9966280
##    permutation.error
## 1          0.2682796
## 2          0.2681630
## 3          0.2677431
## 4          0.2677361
## 5          0.2677130
## 6          0.2677023
## 7          0.2676724
## 8          0.2676233
## 9          0.2676197
## 10         0.2675406
## 11         0.2675271
## 12         0.2674921
## 13         0.2673689
## 14         0.2673231
## 15         0.2673226
## 16         0.2672635
## 17         0.2672602
## 18         0.2670268
## 19         0.2669957
## 20         0.2669427
## 21         0.2667337
```

6.2. Feature effects

Is interesting to know also how the features influence the predicted outcome. This is done by the function `FeatureEffect`.

6.3. Shapley Values

```
cf_shapley <- Shapley$new(cf_predictor, x.interest = W_train[1,])

cf_shapley$plot()
```



There are still a lot of features from the `iml` package. The next version from this PDF will deal with them.

Moreover, *Tutorial 2 - HTE in Randomized Panel Data with Machine Learning Tecniques* brings the `grf` application in randomized panel data, when we account for individual fixed effects. You can find it at my website (https://sites.google.com/view/victor-hugo-alexandrino/) and GitHub (https://github.com/victoralexs)

References

https://cran.r-project.org/web/packages/SuperLearner/vignettes/Guide-to-SuperLearner.html

https://gsbdbi.github.io/ml_tutorial/

https://ml-in-econ.appspot.com

https://github.com/QuantLet/Meta_learner-for-Causal-ML/blob/main/GRF/Causal-Forest.R

https://grf-labs.github.io/grf/

https://www.markhw.com/blog/causalforestintro

https://lost-stats.github.io/Machine_Learning/causal_forest.html