ECON293-Final_Project

Eric Zhao

5/16/2021

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Load Packages

```r
if (!require("pacman")) install.packages("pacman")
```

```
## Warning: package 'pacman' was built under R version 4.0.5
```

```r
pacman::p_load(causalTree)
pacman::p_load(cobalt)
pacman::p_load(data.table)
pacman::p_load(grf)
pacman::p_load(ggplot2)
pacman::p_load(glmnet)
pacman::p_load(haven)
pacman::p_load(lmtest)
pacman::p_load(matchMulti)
pacman::p_load(sandwich)
pacman::p_load(splines)
pacman::p_load(statar)
pacman::p_load(sqldf)
pacman::p_load(tidyverse)
pacman::p_load(WeightIt)

rm(list = ls())
```

Create Helper Functions

```r
generate_X_Y_W_C <- function(outcome, covariates) {
  # -------------------------------------------------------
  # Generates the X, Y, W and cluster components for a given outcome
  # and a set of covariates by eliminating missing observations
  # from the SC_Data dataset. Assumes the treatment column is
  # named 'treatment', and cluster column is 'b_schoolid.
  # Inputs:
  # - outcome: (str) outcome column name in SC_Data
  # - covariates: (vector of str) covariate column names
  # Returns:
  # A list with X, Y, W, C components each including a matrix
  # or data.table
  # -------------------------------------------------------

  # Filter missing entries
  selected_cols <- c(outcome, covariates, 'treatment', 'b_schoolid')
  data <- SC_Data[, selected_cols, with=FALSE]
  data <- data[complete.cases(data)]

  # Setup formula
  fmla <- formula(paste0(outcome, '~', paste(covariates, collapse='+')))
  X <- model.matrix(fmla, data)
  W <- data[, .(treatment)]
  Y <- data[, outcome, with=FALSE]
  C <- data[, .(b_schoolid)]

  # Format Y, W, C as numeric vectors
  W <- as.numeric(W[[1]])
  Y <- as.numeric(Y[[1]])
  C <- as.numeric(C[[1]])

  list(X = X,Y = Y, W = W, C = C)
}


generate_X_Y_W_C_raw <- function(outcome, covariates) {
  # -------------------------------------------------------
  # Generates the X, Y, W and cluster components for a given outcome
  # and a set of covariates by eliminating missing observations
  # from the SC_Data dataset. Assumes the treatment column is
  # named 'treatment', and cluster column is 'b_schoolid.
  # Inputs:
  # - outcome: (str) outcome column name in SC_Data
  # - covariates: (vector of str) covariate column names
  # Returns:
  # A list with X, Y, W, C components each including a matrix
  # or data.table
  # -------------------------------------------------------

  # Filter missing entries
  selected_cols <- c(outcome, covariates, 'treatment', 'b_schoolid')
  data <- SC_Data_Raw[, selected_cols, with=FALSE]
  data <- data[complete.cases(data)]
  # Setup formula
  fmla <- formula(paste0(outcome, '~', paste(covariates, collapse='+')))
  X <- model.matrix(fmla, data)
  W <- data[, .(treatment)]
  Y <- data[, outcome, with=FALSE]
  C <- data[, .(b_schoolid)]

  # Format Y, W, C as numeric vectors
  W <- as.numeric(W[[1]])
  Y <- as.numeric(Y[[1]])
  C <- as.numeric(C[[1]])

  list(X = X,Y = Y, W = W, C = C)
}

generate_X_Y_W_C_df <- function(outcome, covariates, df, treatment) {
  # ______________________________________________________
  # Generates the X, Y, W and cluster components for a given outcome
  # and a set of covariates by eliminating missing observations
  # from the SC_Data dataset. Assumes the treatment column is
  # named 'treatment', and cluster column is 'b_schoolid.
  # Inputs:
  # - outcome: (str) outcome column name in SC_Data
  # - covariates: (vector of str) covariate column names
  # Returns:
  # A list with X, Y, W, C components each including a matrix
  # or data.table
  # ______________________________________________________
  # Filter missing entries
  SC_table <- as.data.table(df)
  selected_cols <- c(outcome, covariates, 'treatment')
  data <- SC_table[, selected_cols, with=FALSE]
  #data<-SC_table[,selected_cols]
  data <- na.omit(data)
  write.csv(data,"unfactored.csv")
  data$b_schoolid = factor(data$b_schoolid)
  data$bstrata = factor(data$bstrata)
  #data <- data[, b_schoolid = factor(b_schoolid)]
  #data <- data[, bstrata = factor(bstrata)]
  #print(data[, outcome])
  #write.csv(data,"fraction_donated.csv")

  # Setup formula
  fmla <- formula(paste0(outcome, '~', paste(covariates, collapse='+')))
  X <- model.matrix(fmla, data)
  #print("X is coming")
  #print(X)
  W <- data[, 'treatment']
  #Y <- data[, outcome]
  Y <- data[, outcome, with=FALSE]
  C <- data[, 'b_schoolid']

  # Format Y, W, C as numeric vectors
```

```r
  W <- as.numeric(W[[1]])
  Y <- as.numeric(Y[[1]])
  C <- as.numeric(C[[1]])
  #print("here")
  #print(Y)
  list(X = X,Y = Y, W = W, C = C)
}

get_AIPW_scores <- function(var_list, cf) {
  # Get forest predictions.
  m.hat <- cf$Y.hat
  e.hat <- cf$W.hat
  tau.hat <- cf$predictions

  # Predicting mu.hat(X[i], 1) and mu.hat(X[i], 0) for obs in held-out sample
  # Note: to understand this, read equations 6-8 in this vignette
  # https://grf-labs.github.io/grf/articles/muhats.html
  mu.hat.0 <- m.hat - e.hat * tau.hat        # E[Y|X,W=0] = E[Y|X] - e(X)*tau(X)
  mu.hat.1 <- m.hat + (1 - e.hat) * tau.hat  # E[Y|X,W=1] = E[Y|X] + (1 - e(X))*tau(X)

  # Compute AIPW scores
  aipw.scores <- tau.hat + var_list$W / e.hat * (var_list$Y -  mu.hat.1) -
    (1 - var_list$W) / (1 - e.hat) * (var_list$Y -  mu.hat.0)
  aipw.scores
}

run_AIPW <- function(outcome,income,covariates,treatment,df) {
  # Get forest predictions.
  # _______________________________________________________
  # Runs AIPW based on grf on a dataframe.
  # Inputs:
  # - outcome: (str) outcome column name in SC_Data
  # - income: (str) baseline var name in SC_Data
  # - covariates: (vector of str) covariate column names
  # - treatment (str) treatment var name
  # - df: dataframe
  # Returns:
  # AIPW estimate and std.err
  # _______________________________________________________
  covariates <- c(covariates,income)
  list_data <- generate_X_Y_W_C_df(outcome,covariates,df,treatment)
  #print("new X")
  #print(list_data)
  forest <- causal_forest(
    X=list_data$X,
    W=list_data$W,
    Y=list_data$Y,
    clusters = list_data$C,
    W.hat=.5,  # In randomized settings, set W.hat to the (known) probability of assignment
    num.trees = 100)
  forest.ate <- average_treatment_effect(forest,target.sample="overlap")
  print(forest.ate)
  print("95% CI lower")
  print(forest.ate["estimate"]-qnorm(.975)*forest.ate["std.err"])
  print("95% CI upper")
  print(forest.ate["estimate"]+qnorm(.975)*forest.ate["std.err"])
}
run_AIPW_without_baseline <- function(outcome,income,covariates,treatment,df) {
  # Get forest predictions.
  # _______________________________________________________
  # Runs AIPW based on grf on a dataframe.
  # Inputs:
  # - outcome: (str) outcome column name in SC_Data
  # - income: (str) baseline var name in SC_Data
  # - covariates: (vector of str) covariate column names
  # - treatment (str) treatment var name
  # - df: dataframe
  # Returns:
  # AIPW estimate and std.err
  # _______________________________________________________
  #covariates <- c(covariates,income)
  list_data <- generate_X_Y_W_C_df(outcome,covariates,df,treatment)
  #print("new X")
  #print(list_data)
  forest <- causal_forest(
    X=list_data$X,
    W=list_data$W,
    Y=list_data$Y,
    clusters = list_data$C,
    W.hat=.5,  # In randomized settings, set W.hat to the (known) probability of assignment
    num.trees = 100)
  forest.ate <- average_treatment_effect(forest,target.sample="overlap")
  print(forest.ate)
  print("95% CI lower")
  print(forest.ate["estimate"]-qnorm(.975)*forest.ate["std.err"])
  print("95% CI upper")
  print(forest.ate["estimate"]+qnorm(.975)*forest.ate["std.err"])
}
run_AIPW_without_cluster <- function(outcome,income,covariates,treatment,df) {
  # Get forest predictions.
  # _______________________________________________________
  # Runs AIPW based on grf on a dataframe.
  # Inputs:
  # - outcome: (str) outcome column name in SC_Data
  # - income: (str) baseline var name in SC_Data
  # - covariates: (vector of str) covariate column names
  # - treatment (str) treatment var name
  # - df: dataframe
  # Returns:
  # AIPW estimate and std.err
  # _______________________________________________________
  covariates <- c(covariates,income)
  list_data <- generate_X_Y_W_C_df(outcome,covariates,df,treatment)
  #print("new X")
  #print(list_data)
  forest <- causal_forest(
```

```r
    X=list_data$X,
    W=list_data$W,
    Y=list_data$Y,
    W.hat=.5,  # In randomized settings, set W.hat to the (known) probability of assignment
    num.trees = 100)
  forest.ate <- average_treatment_effect(forest,target.sample="overlap")
  print(forest.ate)
  print("95% CI lower")
  print(forest.ate["estimate"]-qnorm(.975)*forest.ate["std.err"])
  print("95% CI upper")
  print(forest.ate["estimate"]+qnorm(.975)*forest.ate["std.err"])
}

partial_dependence_single <- function(selected.covariate, covariates, type, X,
                                      causal.forest, grid_size=0){
  # Get data and define other covariates
  data <- as.data.frame(X)
  other.covariates <- covariates[which(covariates != selected.covariate)]

  # Define grid
  if (type == 'binary') {
    grid.size <- 2
    covariate.grid <- c(0, 1)
  } else {
    grid.size <- grid_size
    covariate.grid <- seq(min(data[,selected.covariate]),
                          max(data[,selected.covariate]), length.out=grid.size)
  }

  # Take median of other covariates
  medians <- apply(data[, other.covariates, F], 2, median)

  # Construct a dataset
  data.grid <- data.frame(sapply(medians, function(x) rep(x, grid.size)), covariate.grid)
  colnames(data.grid) <- c(other.covariates, selected.covariate)

  # Expand the data
  fmla <- formula(paste0('~  ', paste(covariates, collapse = '+')))
  X.grid <- model.matrix(fmla, data.grid)

  # Point predictions of the CATE and standard errors
  forest.pred <- predict(causal.forest, newdata = X.grid, estimate.variance=TRUE)
  tau.hat <- forest.pred$predictions
  tau.hat.se <- sqrt(forest.pred$variance.estimates)

  # Plot predictions for each group and 95% confidence intervals around them.
  data.pred <- transform(data.grid, tau.hat=tau.hat,
                                    ci.low = tau.hat - 2*tau.hat.se,
                                    ci.high = tau.hat + 2*tau.hat.se)
  ggplot(data.pred) +
    geom_line(aes_string(x=selected.covariate, y="tau.hat", group = 1), color="black") +
    geom_errorbar(aes_string(x=selected.covariate, ymin="ci.low",
                             ymax="ci.high", width=.2), color="blue") +
    ylab("") +
    ggtitle(paste0("Predicted treatment effect varying '",
                   selected.covariate, "' (other variables fixed at median)")) +
    scale_x_continuous(selected.covariate, breaks=covariate.grid,
                       labels=signif(covariate.grid, 2)) +
    theme_minimal() +
    theme(plot.title = element_text(size = 11, face = "bold"))
}

school_level_heterogeneity <- function(var_list, covariates, tau.hat, cf){
  school.mat <-
    model.matrix(~ b_schoolid + 0,
                 data = data.frame(var_list$X, b_schoolid = factor(var_list$C)))
  school.size <- colSums(school.mat)

  school.X <- (t(school.mat) %*%
                       as.matrix(var_list$X[, covariates])) /
    school.size
  school.X <- data.frame(school.X)
  colnames(school.X) <- covariates

  # Compute doubly robust treatment estimates
  dr.score = tau.hat + var_list$W / cf$W.hat *
    (var_list$Y - cf$Y.hat - (1 - cf$W.hat) * tau.hat) -
    (1 - var_list$W) / (1 - cf$W.hat) *
    (var_list$Y - cf$Y.hat + cf$W.hat * tau.hat)
  score <- t(school.mat) %*% dr.score / school.size

  # Regression forest analysis
  school.forest <- regression_forest(school.X, score)
  school.pred <- predict(school.forest)$predictions
  print(test_calibration(school.forest))

  # OLS
  school.DF <- data.frame(school.X, school.score=score)
  print(coeftest(lm(school.score ~ ., data = school.DF), vcov = vcovHC))

}


love.plot.new = function(covar, treat, group = rep(1, length(treat)),sd.pre=NULL)
{
  plot.data = data.frame(group = character(), covar = character()
                         , diff = numeric(), stringsAsFactors = FALSE)
  n.groups = length(unique(group[group!=0])) # ignore block_0
  n.covar = ncol(covar)
  for(i in 1:n.groups)# for each subgroup
  {
    covar.g = covar[group == i,]
    treat.g = treat[group == i]
    # calculate standardized difference in means
    n = length(treat.g)
    n.treat = sum(treat.g)
    n.control = n - n.treat

    treat.means = colSums(covar.g[treat.g == 1,])/n.treat
    control.means = colSums(covar.g[treat.g == 0,])/n.control

    if(is.null(sd.pre)){
      sd = sapply(covar.g, sd)
      sd[sd==0]=1
    }else sd=sd.pre

    diff = (treat.means - control.means)/sd

    # now append this information
    temp.data = cbind(rep(i, n.covar), names(covar), diff)
    plot.data = rbind.data.frame(plot.data, temp.data, stringsAsFactors = FALSE)
  }

  colnames(plot.data) = c('group', 'covariate', 'diff')
  range = max(abs(as.numeric(plot.data$diff)))

  # produce plot
  ggplot(plot.data) +
    geom_point(aes(x = as.numeric(diff),
                   y = covariate, color = "red")) +
    geom_vline(xintercept = 0) +
    xlim(-1, 1) +
    labs(x = 'Standardized Difference in Means')
}
```

Data Setup

```
# Global student-level controls defined by authors
# Please write below formula without spaces
controls <- 'ageinm+male+refugee+astudent+b_schoolsize+braven_sd+beyes_sd+f_csize'
controls_vec <- strsplit(controls, split='\\+')[[1]]

# Define controls used for each outcome (this adds any controls that are specific to an outcome)
# Outcome 1: Student and Teacher Reports of Violence and Antisocial Behavior
violence.covariates <- c('bsbully_c', 'bstrata', 'b_districtid', controls_vec)

# Outcome 2: Social Exclusion
# ffriend
social.outcome <- 'ffriend'
social.covariates <- c('bfriend', 'bstrata', 'b_districtid', controls_vec)

# Host Emotional support
social.outcome <- 'fhostsupportself'
social.covariates <- c('bhostsupportself', 'bstrata', 'b_districtid', controls_vec)

# Outcome 3: Prosocial Behavior: Trust, Reciprocity and Cooperation
prosocial.covariates <- c('bstrata', 'b_districtid', controls_vec)
```

```r
# Outcome 4: Altruism
altruism.covariates <- c('bdonation_sd', 'a2', 'bstrata', 'b_districtid', controls_vec)

# Outcome 5: Achievement Tests
achievement.covariates <- c('bturk_sd', 'bstrata', 'b_districtid', controls_vec)
```

```r
# 1. Load processed data ------------------------------------
SC_Data <- haven::read_dta("C:/Users/ezhao/Downloads/JS_Stata_Processed.dta")
SC_Data <- as.data.table(SC_Data)
SC_Data <- SC_Data[, b_schoolid := factor(b_schoolid)]

SC_Data_Raw <- haven::read_dta("C:/Users/ezhao/Downloads/AlanBaysanGumrenKubilay_NO_IMPUTE.dta")
SC_Data_Raw <- as.data.table(SC_Data_Raw)
SC_Data_Raw <- SC_Data_Raw[, b_schoolid := factor(b_schoolid)]
```

```r
SC_Data_schools <- SC_Data[, .(
  perpetrator = head(perpetrator, 1),
  victim = head(victim, 1),
  events = head(events, 1),
  treatment = head(treatment, 1),
  bstudentnum_2 = max(bstudentnum_2, na.rm = TRUE),
  bstudentnum_3 = max(bstudentnum_3, na.rm = TRUE),
  n_class = uniqueN(b_classid),
  bactive_syrian_2 = max(bactive_syrian_2, na.rm = TRUE),
  bactive_syrian_3 = max(bactive_syrian_3, na.rm = TRUE),
  b_provinceid = head(b_provinceid, 1),
  b_districtid = head(b_districtid, 1),
  bstrata = max(bstrata),
  refugee = sum(refugee),
  fhostsupportself = mean(fhostsupportself, na.rm = TRUE),
  bhostsupportself = mean(bhostsupportself, na.rm = TRUE)
), by = .(b_schoolid)]

# School size and Syrian/Refugee percentages
SC_Data_schools <- SC_Data_schools[, b_schoolsize := bstudentnum_2 + bstudentnum_3]
SC_Data_schools <- SC_Data_schools[, srefshare := (bactive_syrian_2 + bactive_syrian_3) /
                                     b_schoolsize]
SC_Data_schools <- SC_Data_schools[, refugee_share := refugee / b_schoolsize]

# Now for data with missing values

SC_Data_schools_raw <- SC_Data_Raw[, .(
  perpetrator = head(perpetrator, 1),
  victim = head(victim, 1),
  events = head(events, 1),
  treatment = head(treatment, 1),
  bstudentnum_2 = max(bstudentnum_2, na.rm = FALSE),
  bstudentnum_3 = max(bstudentnum_3, na.rm = FALSE),
  n_class = uniqueN(b_classid),
  bactive_syrian_2 = max(bactive_syrian_2, na.rm = FALSE),
  bactive_syrian_3 = max(bactive_syrian_3, na.rm = FALSE),
  b_provinceid = head(b_provinceid, 1),
  b_districtid = head(b_districtid, 1),
  bstrata = max(bstrata),
  refugee = sum(refugee),
  fhostsupportself = mean(fhostsupportself, na.rm = FALSE),
  bhostsupportself = mean(bhostsupportself, na.rm = FALSE)
), by = .(b_schoolid)]

# School size and Syrian/Refugee percentages
SC_Data_schools_raw <- SC_Data_schools_raw[, b_schoolsize := bstudentnum_2 + bstudentnum_3]
SC_Data_schools_raw <- SC_Data_schools_raw[, srefshare := (bactive_syrian_2 + bactive_syrian_3) /
                                     b_schoolsize]
SC_Data_schools_raw <- SC_Data_schools_raw[, refugee_share := refugee / b_schoolsize]
```

Data Section Results

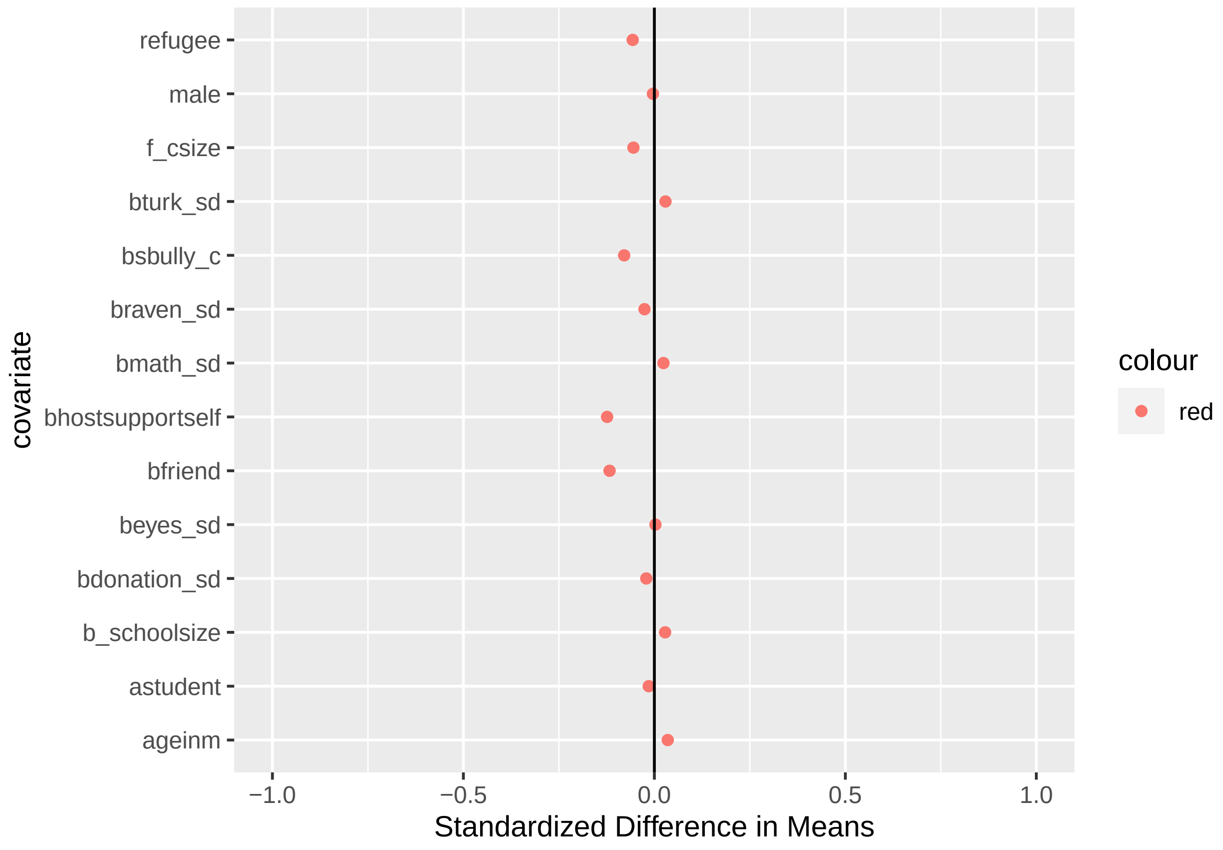
```
# Percent of total refugees
sum(SC_Data$refugee) / nrow(SC_Data)
```

```
## [1] 0.1869808
```

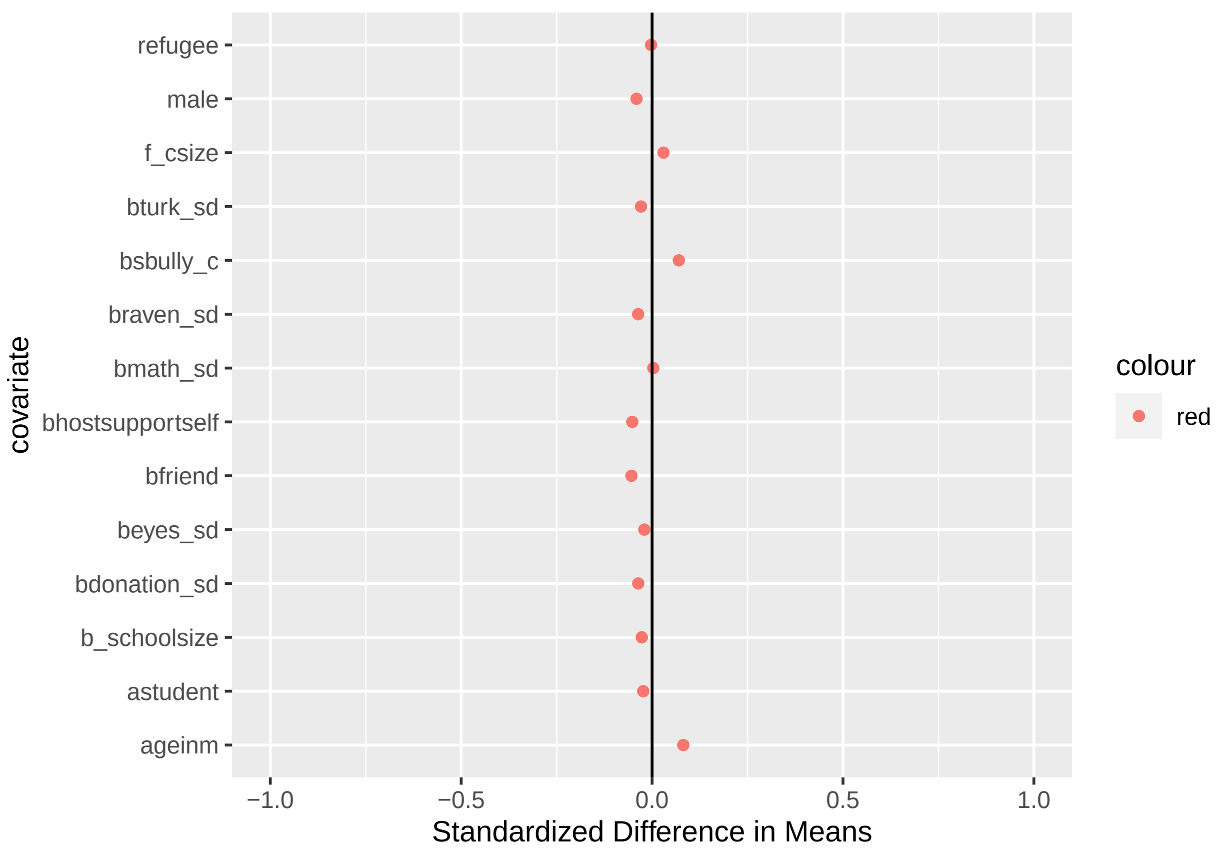
```
# Number of missing data
sqldf("SELECT count(*)
      FROM SC_Data_Raw
      WHERE bhostsupportself is NULL")
```

```
##   count(*)
## 1     2652
```

```
# Love Plot Generation
covariates <- c('ageinm' , 'male' ,
                'refugee' , 'astudent' , 'b_schoolsize', 'braven_sd' , 'beyes_sd',
                'f_csize', 'bfriend','bsbully_c','bhostsupportself'
                ,'bdonation_sd','bmath_sd','bturk_sd')

selected_cols <- c(covariates, 'treatment')
data_fill <- SC_Data[, selected_cols, with=FALSE]
data_fill <- as.data.frame(data_fill[complete.cases(data_fill)])
cov <- data_fill[,1:14]
treat <- data_fill$treatment
love.plot.new(cov,treat)
```



```r
data_fill_raw <- SC_Data_Raw[, selected_cols, with=FALSE]
data_fill_raw <- as.data.frame(data_fill_raw[complete.cases(data_fill_raw)])
cov_raw <- data_fill_raw[,1:14]
treat_raw <- data_fill_raw$treatment
love.plot.new(cov_raw, treat_raw)
```



AIPW

```
covariates <- c('ageinm' , 'male' , 'bstrata',
                'refugee' , 'astudent' , 'braven_sd' , 'beyes_sd' , 'b_schoolsize',
                'f_csize','b_schoolid')
run_AIPW('ffriend','bfriend',covariates,'treatment',SC_Data_Raw)
```

```
##     estimate      std.err
## -0.003340690  0.005272661
## [1] "95% CI lower"
##    estimate
## -0.01367492
## [1] "95% CI upper"
##    estimate
## 0.006993536
```

```
run_AIPW('fsupportself','bsupportself',covariates,'treatment',SC_Data_Raw)
```

```
##   estimate    std.err
## 0.03468431 0.01573329
## [1] "95% CI lower"
##    estimate
## 0.003847626
## [1] "95% CI upper"
##   estimate
## 0.06552099
```

```r
run_AIPW('fsbully_c','bsbully_c',covariates,'treatment',SC_Data_Raw)
```

```
##   estimate    std.err
## 0.00342971 0.02209427
## [1] "95% CI lower"
##    estimate
## -0.03987426
## [1] "95% CI upper"
##   estimate
## 0.04673368
```

```r
run_AIPW('fmath_sd','bmath_sd',covariates,'treatment',SC_Data_Raw)
```

```
##    estimate     std.err
## 0.009819223 0.079565236
## [1] "95% CI lower"
##   estimate
## -0.1461258
## [1] "95% CI upper"
##  estimate
## 0.1657642
```

```r
run_AIPW('fturk_sd','bturk_sd',covariates,'treatment',SC_Data_Raw)
```

```
##   estimate    std.err
## 0.02638368 0.05734376
## [1] "95% CI lower"
##    estimate
## -0.08600802
## [1] "95% CI upper"
##  estimate
## 0.1387754
```

```r
run_AIPW_without_baseline('fs_decision_out','',covariates,'treatment',SC_Data_Raw)
```

```
##   estimate    std.err
## 0.25597495 0.06563883
## [1] "95% CI lower"
##  estimate
## 0.1273252
## [1] "95% CI upper"
##  estimate
## 0.3846247
```

```
covariates <- c('ageinm' , 'male' , 'bstrata',
                'refugee' , 'astudent' , 'braven_sd' , 'beyes_sd' , 'b_schoolsize',
                'f_csize','b_schoolid','a2','inter1')
run_AIPW('fdonate','bdonation_sd',covariates,'treatment',SC_Data_Raw)
```

```
##   estimate    std.err
## 0.08730612 0.02836965
## [1] "95% CI lower"
##   estimate
## 0.03170264
## [1] "95% CI upper"
##  estimate
## 0.1429096
```

```
run_AIPW('fdonation_perc','bdonation_sd',covariates,'treatment',SC_Data_Raw)
```

```
##   estimate    std.err
## 0.05806833 0.01870488
## [1] "95% CI lower"
##   estimate
## 0.02140744
## [1] "95% CI upper"
##   estimate
## 0.09472923
```

```
all_columns_for_schools <- c('ageinm' , 'male' , 'bstrata',
                'refugee' , 'astudent' , 'braven_sd' , 'beyes_sd' , 'b_schoolsize',
                'f_csize','ffriend','bfriend','fsupportself','bsupportself',
                'fsbully_c','bsbully_c','fmath_sd','bmath_sd','fturk_sd','bturk_sd','a2','inter1','trea
school_level <- students2schools(SC_Data_Raw,all_columns_for_schools,'b_schoolid')
covariates <- c('ageinm' , 'male' , 'bstrata',
                'refugee' , 'astudent' , 'braven_sd' , 'beyes_sd' , 'b_schoolsize',
                'f_csize','b_schoolid')
run_AIPW('ffriend','bfriend',covariates,'treatment',school_level)
```

```
##    estimate     std.err
## 0.008520145 0.006720651
## [1] "95% CI lower"
##    estimate
## -0.00465209
## [1] "95% CI upper"
##   estimate
## 0.02169238
```

```
run_AIPW('fsupportself','bsupportself',covariates,'treatment',school_level)
```

```
##   estimate    std.err
## 0.03950970 0.01841041
## [1] "95% CI lower"
##    estimate
## 0.003425948
## [1] "95% CI upper"
##   estimate
## 0.07559344
```

```
run_AIPW('fsbully_c','bsbully_c',covariates,'treatment',school_level)
```

```
##   estimate    std.err
## 0.01840889 0.02176682
## [1] "95% CI lower"
##    estimate
## -0.02425329
## [1] "95% CI upper"
##   estimate
## 0.06107107
```

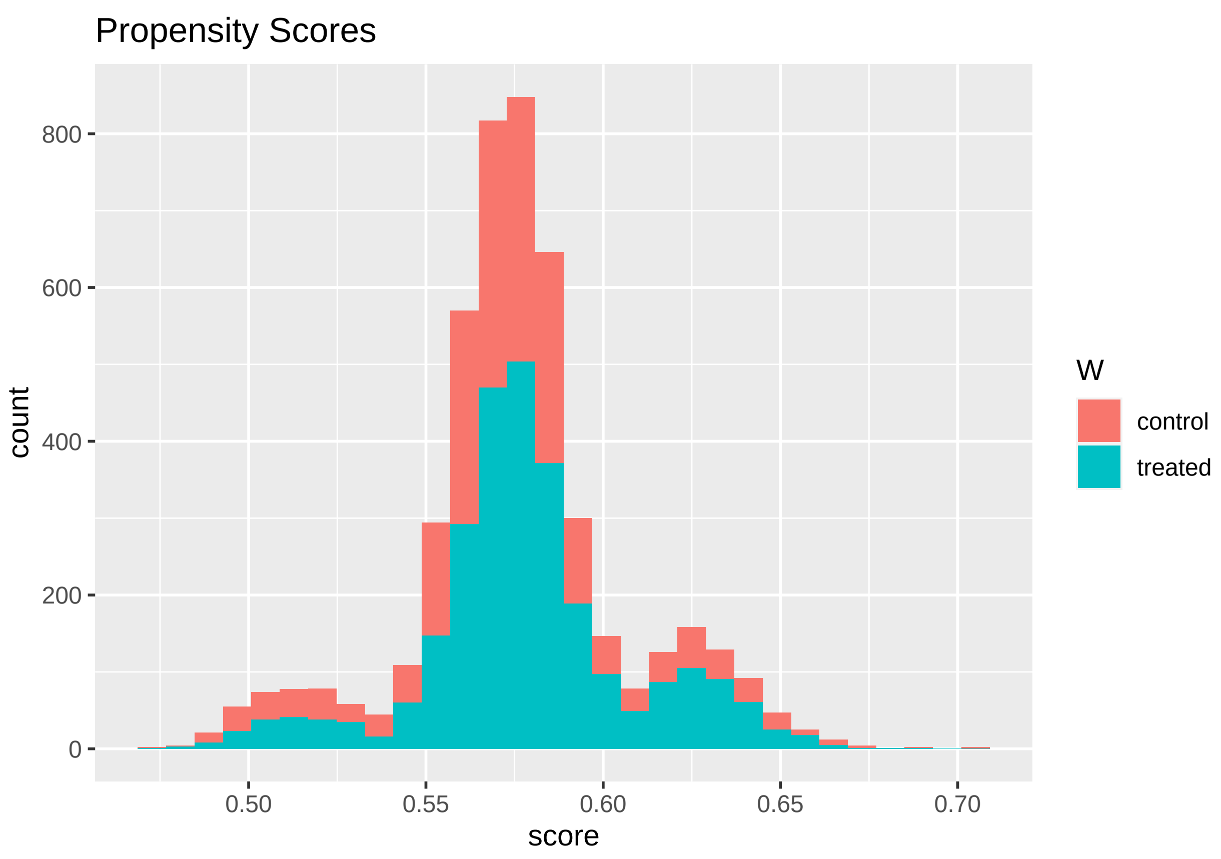
```
run_AIPW('fmath_sd','bmath_sd',covariates,'treatment',school_level)
```

```
##    estimate     std.err
## 0.008716151 0.087592515
## [1] "95% CI lower"
##  estimate
## -0.162962
## [1] "95% CI upper"
##  estimate
## 0.1803943
```

```
run_AIPW('fturk_sd','bturk_sd',covariates,'treatment',school_level)
```

```
##   estimate    std.err
## 0.08920179 0.06171079
## [1] "95% CI lower"
##    estimate
## -0.03174913
## [1] "95% CI upper"
##  estimate
## 0.2101527
```

```
set.seed(123)
covariates <- c('ageinm' , 'male' , 'bstrata',
                'refugee' , 'astudent' , 'braven_sd' , 'beyes_sd' , 'b_schoolsize',
                'f_csize','bfriend')
list_data <- generate_X_Y_W_C_df('ffriend', covariates, SC_Data_Raw, 'treatment')
logit <- cv.glmnet(x=list_data$X, y=list_data$W, family="binomial")
e.hat <- predict(logit, list_data$X, s = "lambda.min", type="response")
e.hat.1 <- e.hat[list_data$W==1]
e.hat.0 <-e.hat[list_data$W==0]
e.hat.1.df = data.frame(prop= e.hat.1)
e.hat.1.df$W = "treated"
e.hat.0.df = data.frame(prop = e.hat.0)
e.hat.0.df$W = "control"
e_df <- rbind(e.hat.1.df,e.hat.0.df)
qplot(prop, data = e_df, geom="histogram",fill=W,main="Propensity Scores",xlab="score",ylab="count")
```



# HTE - Pre-Specified Hypothesis

```
m.table9.1 <- lm(formula = paste0(
  'fsbully_c ~ refugee* treatment + bsbully_c  + factor(bstrata) + factor(b_districtid) +', controls),
  data = SC_Data_Raw)

# Outcome 2: Social Exclusion
m.table10.1 <- lm(formula = paste0(
  'ffriend ~ refugee* treatment + bfriend  + factor(bstrata) + factor(b_districtid) +', controls),
  data = SC_Data_Raw)

m.table10.3 <- lm(formula = paste0(
  'fhostsupportself ~ refugee*treatment + bhostsupportself  + factor(bstrata) + factor(b_districtid) +'
  data = SC_Data_Raw)

# Outcome 3: Prosocial Behavior: Trust, Reciprocity and Cooperation
m.table11.2 <- lm(formula = paste0(
  'fs_decision_out ~ refugee* treatment  + factor(bstrata) + factor(b_districtid) +', controls),
  data = SC_Data_Raw)

# Outcome 4: Altruism
m.table12.1 <- lm(formula = paste0(
  'fdonate ~ a2*treatment + factor(bstrata) + factor(b_districtid) + bdonation_sd +', controls),
  data = SC_Data_Raw, subset = SC_Data_Raw$refugee == 0)

m.table12.2 <- lm(formula = paste0(
  'fdonate ~ a2*treatment + factor(bstrata) + factor(b_districtid)  + bdonation_sd +', controls),
```

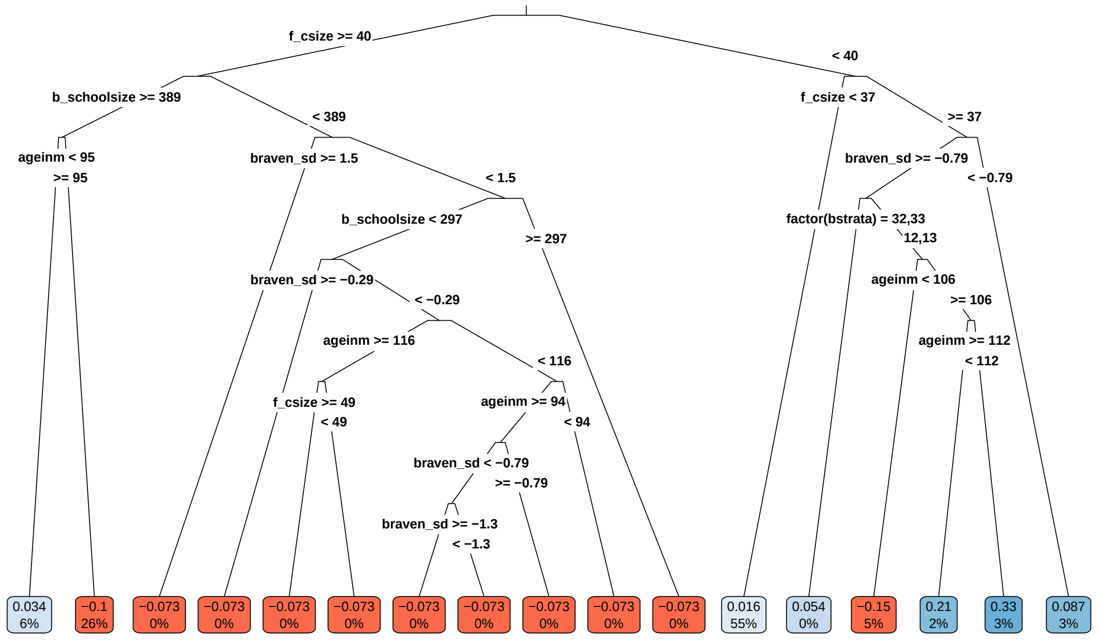
```r
  data = SC_Data_Raw, subset = SC_Data_Raw$refugee == 1)

m.table12.3 <- lm(formula = paste0(
  'fdonation_perc ~ a2*treatment + factor(bstrata) + factor(b_districtid) + bdonation_sd +', controls),
  data = SC_Data_Raw, subset = SC_Data_Raw$refugee == 0)

m.table12.4 <- lm(formula = paste0(
  'fdonation_perc ~ a2*treatment + factor(bstrata) + factor(b_districtid) + bdonation_sd +', controls),
  data = SC_Data_Raw, subset = SC_Data_Raw$refugee == 1)

# Outcome 5: Achievement Tests
m.table13.1 <- lm(formula = paste0(
  'fturk_sd ~ treatment + bturk_sd + factor(bstrata) + factor(b_districtid) +', controls),
  data = SC_Data_Raw, subset = SC_Data_Raw$refugee == 0)

m.table13.2 <- lm(formula = paste0(
  'fturk_sd ~ treatment + bturk_sd + factor(bstrata) + factor(b_districtid) +', controls),
  data = SC_Data_Raw, subset = SC_Data_Raw$refugee == 1)

m.table13.3 <- lm(formula = paste0(
  'fmath_sd ~ treatment + bmath_sd + factor(bstrata) + factor(b_districtid) +', controls),
  data = SC_Data_Raw, subset = SC_Data_Raw$refugee == 0)

m.table13.4 <- lm(formula = paste0(
  'fmath_sd ~ treatment + bmath_sd + factor(bstrata) + factor(b_districtid) +', controls),
  data = SC_Data_Raw, subset = SC_Data_Raw$refugee == 1)
```

HTE - Causal Trees

```r
fmla <- paste0(
  'fsbully_c ~ bsbully_c  + factor(bstrata) +', controls)

# Dividing data into three subsets
indices <- split(seq(nrow(SC_Data_Raw)), sort(seq(nrow(SC_Data_Raw)) %% 3))
names(indices) <- c('split', 'est', 'test')

# Fitting the forest
ct.unpruned <- honest.causalTree(
  formula=fmla,             # Define the model
  data=SC_Data_Raw[indices$split,],
  treatment=SC_Data_Raw[indices$split, treatment],
  est_data=SC_Data_Raw[indices$est,],
  est_treatment=SC_Data_Raw[indices$est, treatment],
  minsize=1,                  # Min. number of treatment and control cases in each leaf
  HonestSampleSize=length(indices$est), #  Num obs used in estimation after splitting
  # We recommend not changing the parameters below
  split.Rule="CT",            # Define the splitting option
  cv.option="TOT",            # Cross validation options
  cp=0,                       # Complexity parameter
  split.Honest=TRUE,          # Use honesty when splitting
  cv.Honest=TRUE                # Use honesty when performing cross-validation
)
```

```
## [1] 2
## [1] "CT"
```

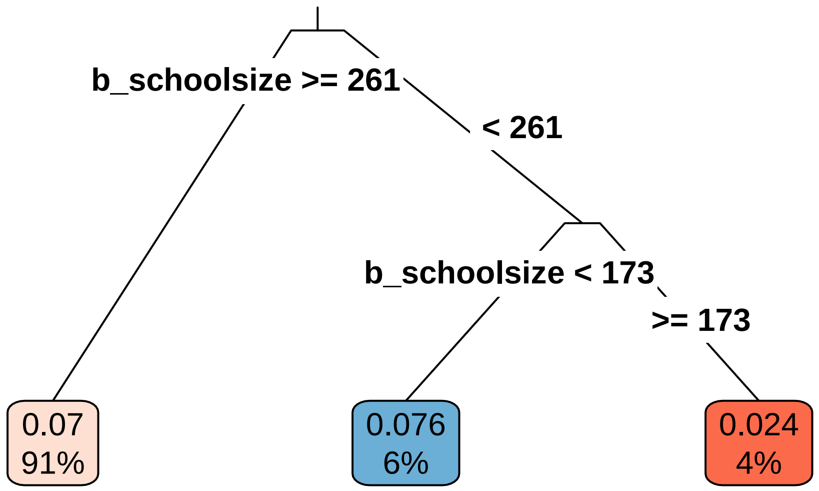
```r
# Table of cross-validated values by tuning parameter.
ct.cptable <- as.data.frame(ct.unpruned$cptable)

# Obtain optimal complexity parameter to prune tree.
cp.selected <- which.min(ct.cptable$xerror)
cp.optimal <- ct.cptable[cp.selected, "CP"]

# Prune the tree at optimal complexity parameter.
ct.pruned <- prune(tree=ct.unpruned, cp=cp.optimal)

# Predict point estimates (on estimation sample)
tau.hat.est <- predict(ct.pruned, newdata=SC_Data_Raw[indices$est,])

# Create a factor column 'leaf' indicating leaf assignment in the estimation set
num.leaves <- length(unique(tau.hat.est))
leaf <- factor(tau.hat.est, levels=sort(unique(tau.hat.est)), labels = seq(num.leaves))

#Plot
rpart.plot(
  x=ct.pruned,        # Pruned tree
  type=3,             # Draw separate split labels for the left and right directions
  fallen=TRUE,        # Position the leaf nodes at the bottom of the graph
  leaf.round=1,       # Rounding of the corners of the leaf node boxes
  extra=100,          # Display the percentage of observations in the node
  branch=.1,          # Shape of the branch lines
  box.palette="RdBu") # Palette for coloring the node
```



```r
fmla <- paste0(
  'fhostsupportself ~ bhostsupportself  + factor(bstrata) +', controls)

# Dividing data into three subsets
indices <- split(seq(nrow(SC_Data_Raw)), sort(seq(nrow(SC_Data_Raw)) %% 3))
names(indices) <- c('split', 'est', 'test')

# Fitting the forest
ct.unpruned <- honest.causalTree(
  formula=fmla,             # Define the model
  data=SC_Data_Raw[indices$split,],
  treatment=SC_Data_Raw[indices$split, treatment],
  est_data=SC_Data_Raw[indices$est,],
  est_treatment=SC_Data_Raw[indices$est, treatment],
  minsize=1,                  # Min. number of treatment and control cases in each leaf
  HonestSampleSize=length(indices$est), #  Num obs used in estimation after splitting
  # We recommend not changing the parameters below
  split.Rule="CT",            # Define the splitting option
  cv.option="TOT",            # Cross validation options
  cp=0,                       # Complexity parameter
  split.Honest=TRUE,          # Use honesty when splitting
  cv.Honest=TRUE              # Use honesty when performing cross-validation
)
```

```
## [1] 2
## [1] "CT"
```

```r
# Table of cross-validated values by tuning parameter.
ct.cptable <- as.data.frame(ct.unpruned$cptable)

# Obtain optimal complexity parameter to prune tree.
cp.selected <- which.min(ct.cptable$xerror)
cp.optimal <- ct.cptable[cp.selected, "CP"]

# Prune the tree at optimal complexity parameter.
ct.pruned <- prune(tree=ct.unpruned, cp=cp.optimal)

# Predict point estimates (on estimation sample)
tau.hat.est <- predict(ct.pruned, newdata=SC_Data_Raw[indices$est,])

# Create a factor column 'leaf' indicating leaf assignment in the estimation set
num.leaves <- length(unique(tau.hat.est))
leaf <- factor(tau.hat.est, levels=sort(unique(tau.hat.est)), labels = seq(num.leaves))

#Plot
rpart.plot(
  x=ct.pruned,        # Pruned tree
  type=3,             # Draw separate split labels for the left and right directions
  fallen=TRUE,        # Position the leaf nodes at the bottom of the graph
  leaf.round=1,       # Rounding of the corners of the leaf node boxes
  extra=100,          # Display the percentage of observations in the node
  branch=.1,          # Shape of the branch lines
  box.palette="RdBu") # Palette for coloring the node
```



```r
fmla <- paste0(
  'fs_decision_in ~ factor(bstrata) +', controls)

# Dividing data into three subsets
indices <- split(seq(nrow(SC_Data_Raw)), sort(seq(nrow(SC_Data_Raw)) %% 3))
names(indices) <- c('split', 'est', 'test')

# Fitting the forest
ct.unpruned <- honest.causalTree(
  formula=fmla,             # Define the model
  data=SC_Data_Raw[indices$split,],
  treatment=SC_Data_Raw[indices$split, treatment],
  est_data=SC_Data_Raw[indices$est,],
  est_treatment=SC_Data_Raw[indices$est, treatment],
  minsize=1,                  # Min. number of treatment and control cases in each leaf
  HonestSampleSize=length(indices$est), #  Num obs used in estimation after splitting
  # We recommend not changing the parameters below
  split.Rule="CT",            # Define the splitting option
  cv.option="TOT",            # Cross validation options
  cp=0,                       # Complexity parameter
  split.Honest=TRUE,          # Use honesty when splitting
  cv.Honest=TRUE              # Use honesty when performing cross-validation
)
```

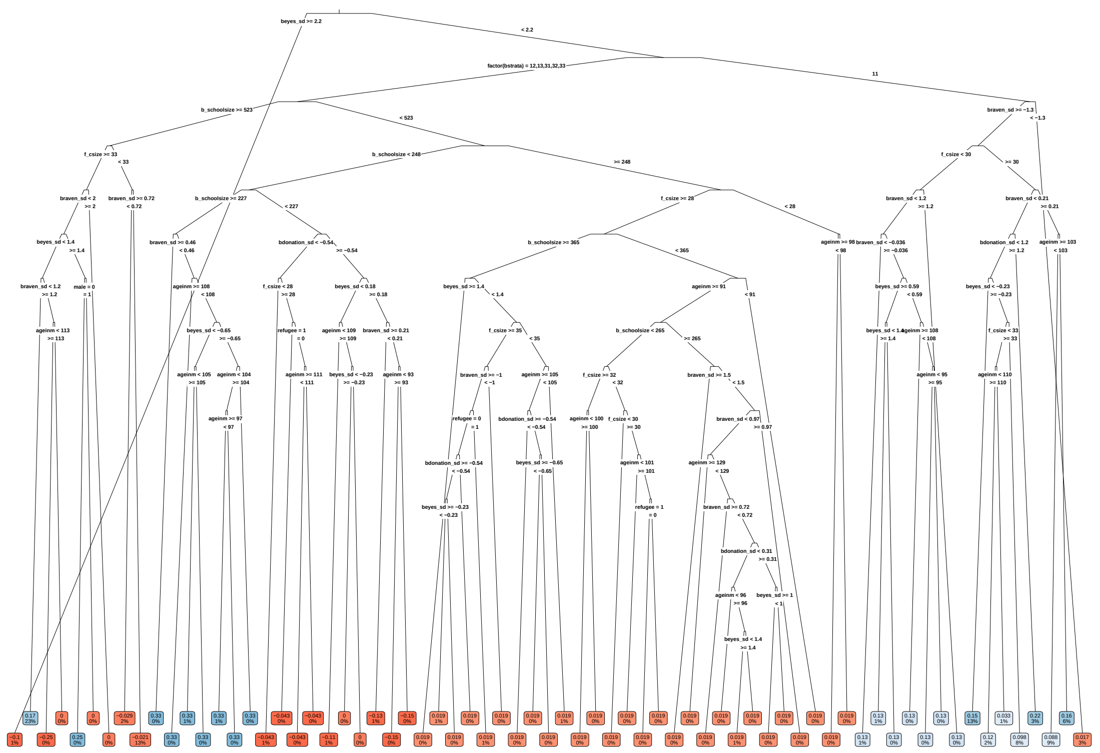
```
## [1] 2
## [1] "CT"
```

```r
# Table of cross-validated values by tuning parameter.
ct.cptable <- as.data.frame(ct.unpruned$cptable)

# Obtain optimal complexity parameter to prune tree.
cp.selected <- which.min(ct.cptable$xerror)
cp.optimal <- ct.cptable[cp.selected, "CP"]

# Prune the tree at optimal complexity parameter.
ct.pruned <- prune(tree=ct.unpruned, cp=cp.optimal)

# Predict point estimates (on estimation sample)
tau.hat.est <- predict(ct.pruned, newdata=SC_Data_Raw[indices$est,])

# Create a factor column 'leaf' indicating leaf assignment in the estimation set
num.leaves <- length(unique(tau.hat.est))
leaf <- factor(tau.hat.est, levels=sort(unique(tau.hat.est)), labels = seq(num.leaves))

#Plot
rpart.plot(
  x=ct.pruned,        # Pruned tree
  type=3,             # Draw separate split labels for the left and right directions
  fallen=TRUE,        # Position the leaf nodes at the bottom of the graph
  leaf.round=1,       # Rounding of the corners of the leaf node boxes
  extra=100,          # Display the percentage of observations in the node
  branch=.1,          # Shape of the branch lines
  box.palette="RdBu") # Palette for coloring the node
```

```
fmla <- paste0(
  'fdonate ~ bdonation_sd  + factor(bstrata) +', controls)

# Dividing data into three subsets
indices <- split(seq(nrow(SC_Data_Raw)), sort(seq(nrow(SC_Data_Raw)) %% 3))
names(indices) <- c('split', 'est', 'test')

# Fitting the forest
ct.unpruned <- honest.causalTree(
  formula=fmla,             # Define the model
  data=SC_Data_Raw[indices$split,],
  treatment=SC_Data_Raw[indices$split, treatment],
  est_data=SC_Data_Raw[indices$est,],
  est_treatment=SC_Data_Raw[indices$est, treatment],
  minsize=1,                  # Min. number of treatment and control cases in each leaf
  HonestSampleSize=length(indices$est), #  Num obs used in estimation after splitting
  # We recommend not changing the parameters below
  split.Rule="CT",            # Define the splitting option
  cv.option="TOT",            # Cross validation options
  cp=0,                       # Complexity parameter
  split.Honest=TRUE,          # Use honesty when splitting
  cv.Honest=TRUE              # Use honesty when performing cross-validation
)
```

```
## [1] 2
## [1] "CT"
```

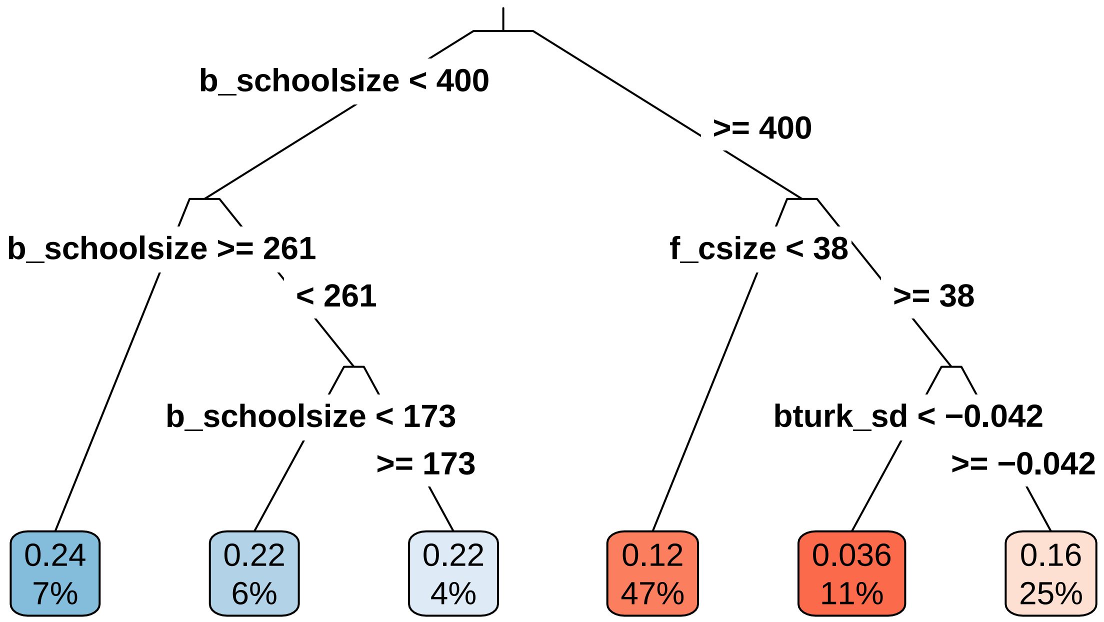
```r
# Table of cross-validated values by tuning parameter.
ct.cptable <- as.data.frame(ct.unpruned$cptable)

# Obtain optimal complexity parameter to prune tree.
cp.selected <- which.min(ct.cptable$xerror)
cp.optimal <- ct.cptable[cp.selected, "CP"]

# Prune the tree at optimal complexity parameter.
ct.pruned <- prune(tree=ct.unpruned, cp=cp.optimal)

# Predict point estimates (on estimation sample)
tau.hat.est <- predict(ct.pruned, newdata=SC_Data_Raw[indices$est,])

# Create a factor column 'leaf' indicating leaf assignment in the estimation set
num.leaves <- length(unique(tau.hat.est))
leaf <- factor(tau.hat.est, levels=sort(unique(tau.hat.est)), labels = seq(num.leaves))

#Plot
rpart.plot(
  x=ct.pruned,        # Pruned tree
  type=3,             # Draw separate split labels for the left and right directions
  fallen=TRUE,        # Position the leaf nodes at the bottom of the graph
  leaf.round=1,       # Rounding of the corners of the leaf node boxes
  extra=100,          # Display the percentage of observations in the node
  branch=.1,          # Shape of the branch lines
  box.palette="RdBu") # Palette for coloring the node
```



```
fmla <- paste0(
  'fturk_sd ~ bturk_sd  + factor(bstrata) +', controls)

# Dividing data into three subsets
indices <- split(seq(nrow(SC_Data_Raw)), sort(seq(nrow(SC_Data_Raw)) %% 3))
names(indices) <- c('split', 'est', 'test')

# Fitting the forest
ct.unpruned <- honest.causalTree(
  formula=fmla,             # Define the model
  data=SC_Data_Raw[indices$split,],
  treatment=SC_Data_Raw[indices$split, treatment],
  est_data=SC_Data_Raw[indices$est,],
  est_treatment=SC_Data_Raw[indices$est, treatment],
  minsize=1,                  # Min. number of treatment and control cases in each leaf
  HonestSampleSize=length(indices$est), #  Num obs used in estimation after splitting
  # We recommend not changing the parameters below
  split.Rule="CT",             # Define the splitting option
  cv.option="TOT",             # Cross validation options
  cp=0,                        # Complexity parameter
  split.Honest=TRUE,           # Use honesty when splitting
  cv.Honest=TRUE               # Use honesty when performing cross-validation
)
```

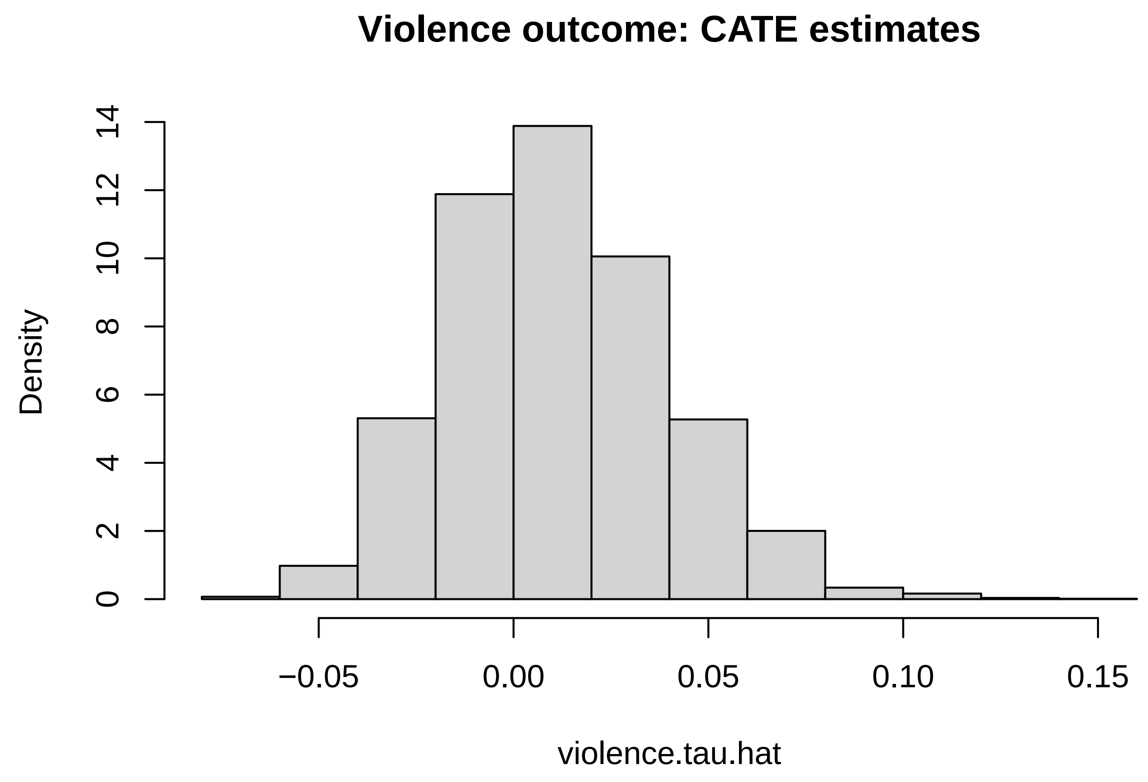
```
## [1] 2
## [1] "CT"
```

```r
# Table of cross-validated values by tuning parameter.
ct.cptable <- as.data.frame(ct.unpruned$cptable)

# Obtain optimal complexity parameter to prune tree.
cp.selected <- which.min(ct.cptable$xerror)
cp.optimal <- ct.cptable[cp.selected, "CP"]

# Prune the tree at optimal complexity parameter.
ct.pruned <- prune(tree=ct.unpruned, cp=cp.optimal)

# Predict point estimates (on estimation sample)
tau.hat.est <- predict(ct.pruned, newdata=SC_Data_Raw[indices$est,])

# Create a factor column 'leaf' indicating leaf assignment in the estimation set
num.leaves <- length(unique(tau.hat.est))
leaf <- factor(tau.hat.est, levels=sort(unique(tau.hat.est)), labels = seq(num.leaves))

#Plot
rpart.plot(
  x=ct.pruned,        # Pruned tree
  type=3,             # Draw separate split labels for the left and right directions
  fallen=TRUE,        # Position the leaf nodes at the bottom of the graph
  leaf.round=1,       # Rounding of the corners of the leaf node boxes
  extra=100,          # Display the percentage of observations in the node
  branch=.1,          # Shape of the branch lines
  box.palette="RdBu") # Palette for coloring the node
```



HTE - Causal Forests

```
violence_list <- generate_X_Y_W_C_raw(outcome = 'fsbully_c', covariates = violence.covariates)
violence.n <- dim(violence_list$X)[1]
violence.cf <- causal_forest(X = violence_list$X, Y = violence_list$Y,
                             W = violence_list$W, clusters = violence_list$C, W.hat = .5)

# CATE histogram
violence.tau.hat <- predict(violence.cf)$predictions
hist(violence.tau.hat, main="Violence outcome: CATE estimates", freq=F)
```



```r
# Variable importance
violence.var_imp <- c(variable_importance(violence.cf))
names(violence.var_imp) <- violence.covariates # TODO What IS NA? The clusters?
violence.sorted_var_imp <- sort(violence.var_imp, decreasing = TRUE)

# Best linear projection
best_linear_projection(violence.cf, violence_list$X)
```

```
##
## 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)  -2.3734e-01  2.4934e-01 -0.9519  0.34122
## bsbully_c    -2.4600e-02  3.8763e-02 -0.6346  0.52571
## bstrata       2.2401e-03  4.7372e-03  0.4729  0.63634
## b_districtid -1.5106e-03  1.4169e-02 -0.1066  0.91510
## ageinm        2.6715e-03  1.8505e-03  1.4437  0.14890
## male          1.2542e-02  2.8149e-02  0.4456  0.65593
## refugee      -7.4950e-02  5.0095e-02 -1.4962  0.13468
## astudent      9.3366e-02  5.5346e-02  1.6870  0.09169 .
## b_schoolsize -2.2448e-05  9.7317e-05 -0.2307  0.81759
## braven_sd    -2.4363e-02  1.3280e-02 -1.8346  0.06664 .
## beyes_sd     -2.7339e-03  1.4507e-02 -0.1885  0.85053
## f_csize      -1.4324e-03  3.8317e-03 -0.3738  0.70856
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

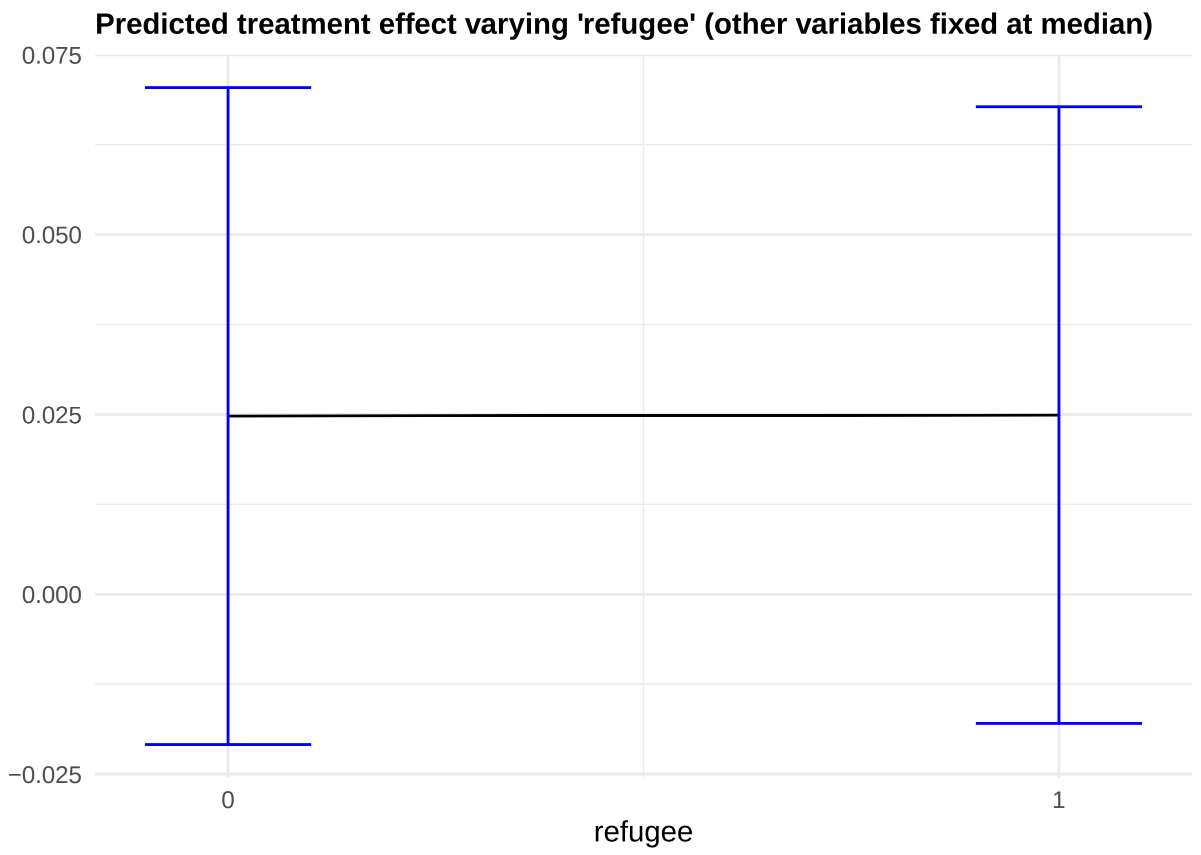
```
# Calibration
test_calibration(violence.cf)
```

```
##
## Best linear fit using forest predictions (on held-out data)
## as well as the mean forest prediction as regressors, along
## with one-sided heteroskedasticity-robust (HC3) SEs:
##
##                                Estimate Std. Error t value Pr(>t)
## mean.forest.prediction           0.77033    1.80873  0.4259 0.3351
## differential.forest.prediction -0.80086    0.58905 -1.3596 0.9130
```

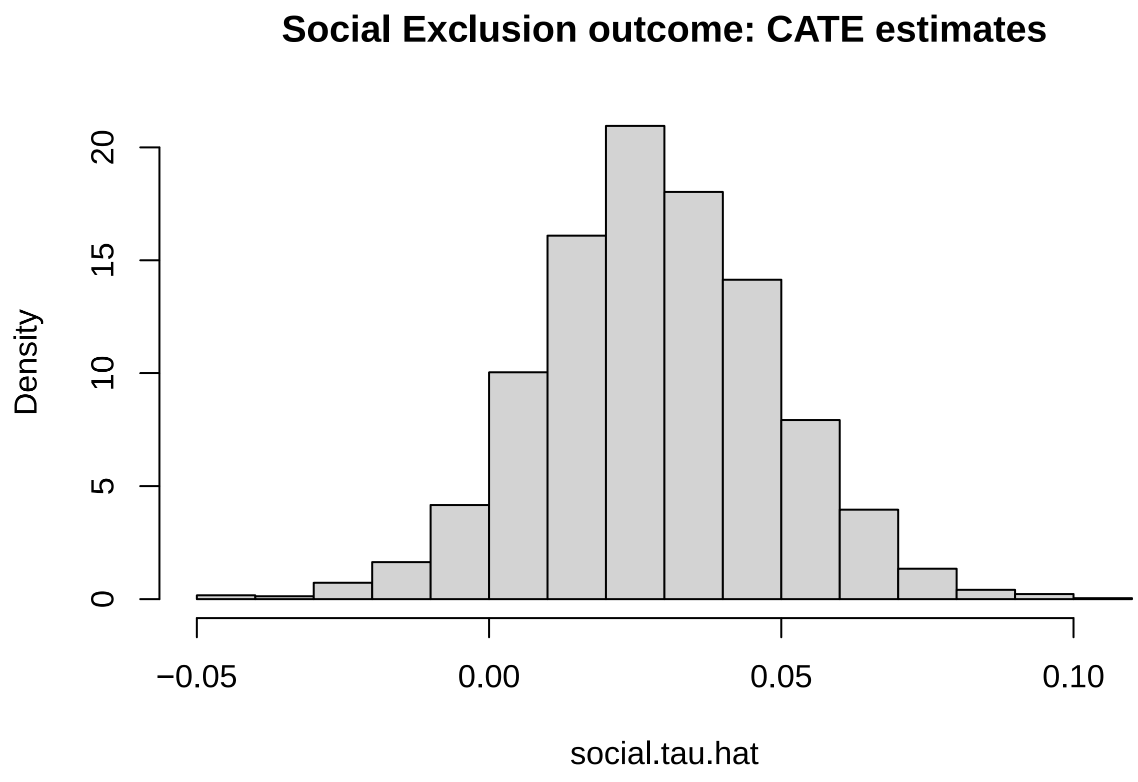
```
# Compare regions with above/below median CATEs (from Wager & Athey)
high_effect <- violence.tau.hat > median(violence.tau.hat)
ate.high <- average_treatment_effect( violence.cf , subset = high_effect )
ate.low <- average_treatment_effect( violence.cf, subset =! high_effect )
paste ("95% CI for difference in ATE:",
       round(ate.high[1] - ate.low[1] , 3) , "+/-",
       round(qnorm(0.975) * sqrt ( ate.high[2]^2 + ate.low[2]^2) , 3))
```

```
## [1] "95% CI for difference in ATE: -0.018 +/- 0.075"
```

```
# Partial dependence
partial_dependence_single(selected.covariate = 'refugee',
                          covariates = violence.covariates,
                          type = 'binary', X = violence_list$X,
                          causal.forest = violence.cf)
```



```
# Fit causal tree
social_list <- generate_X_Y_W_C_raw(outcome = social.outcome, covariates = social.covariates)
social.n <- dim(social_list$X)[1]
social.cf <- causal_forest(X = social_list$X, Y = social_list$Y,
                          W = social_list$W, clusters = social_list$C, W.hat = .5)

# CATE histogram
social.tau.hat <- predict(social.cf)$predictions
hist(social.tau.hat, main="Social Exclusion outcome: CATE estimates", freq=F)
```



```r
# Variable importance
social.var_imp <- c(variable_importance(social.cf))
names(social.var_imp) <- colnames(social_list$X)
social.sorted_var_imp <- sort(social.var_imp, decreasing = TRUE)

# Best linear projection
best_linear_projection(social.cf, social_list$X)
```

```
##
## 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)      -5.2825e-02  2.2170e-01 -0.2383   0.8117
## bhostsupportself -1.3092e-02  3.0755e-02 -0.4257   0.6703
## bstrata           2.1512e-03  6.0914e-03  0.3531   0.7240
## b_districtid     -1.1750e-02  2.0006e-02 -0.5873   0.5570
## ageinm            9.5540e-04  1.4044e-03  0.6803   0.4964
## male             -8.2781e-03  2.2777e-02 -0.3634   0.7163
## refugee          -6.0497e-02  5.9645e-02 -1.0143   0.3105
## astudent         -3.5833e-02  6.4585e-02 -0.5548   0.5790
## b_schoolsize      2.0332e-05  1.1336e-04  0.1794   0.8577
## braven_sd        -1.0469e-02  1.2170e-02 -0.8602   0.3897
## beyes_sd          7.3524e-03  1.2203e-02  0.6025   0.5469
## f_csize           3.1997e-04  4.8481e-03  0.0660   0.9474
```

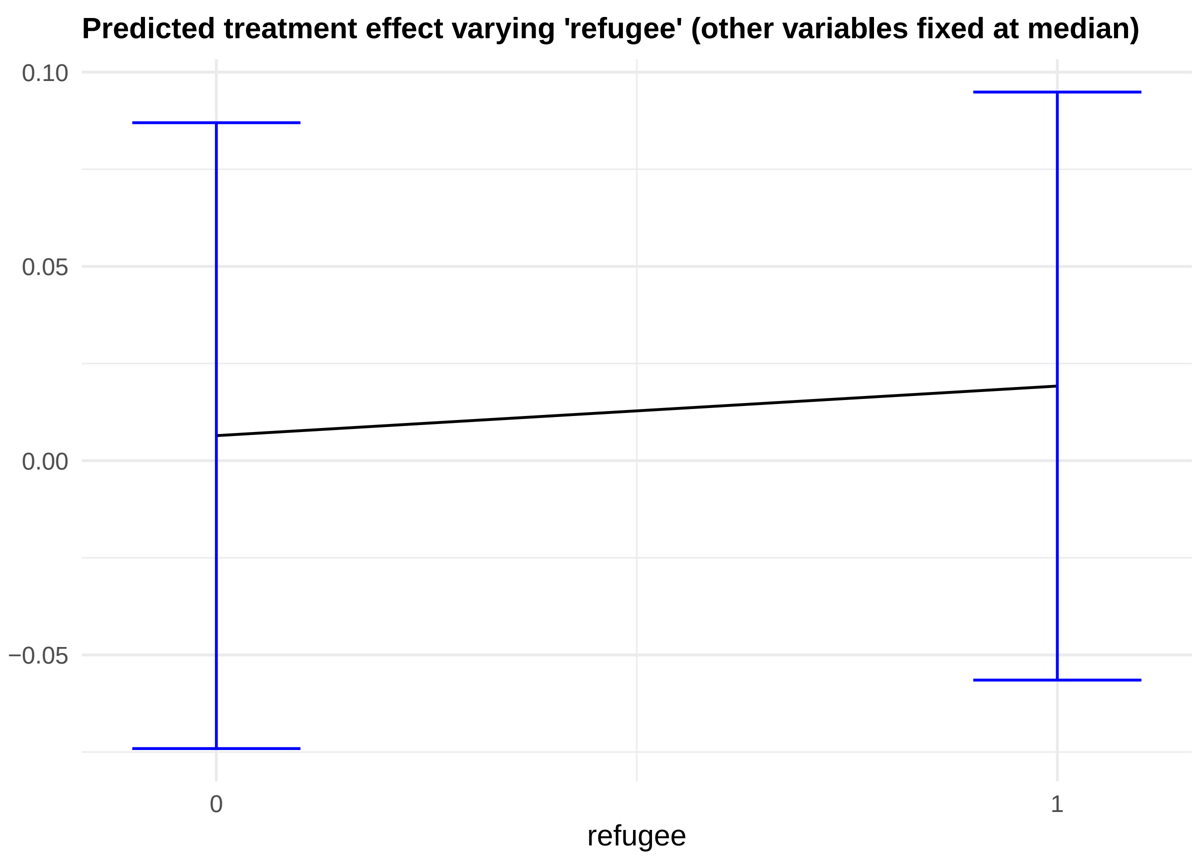
```r
# Calibration
test_calibration(social.cf)
```

```
##
## Best linear fit using forest predictions (on held-out data)
## as well as the mean forest prediction as regressors, along
## with one-sided heteroskedasticity-robust (HC3) SEs:
##
##                                Estimate Std. Error t value  Pr(>t)
## mean.forest.prediction          1.03595    0.54191  1.9117 0.02799 *
## differential.forest.prediction -2.27367    1.03884 -2.1887 0.98567
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

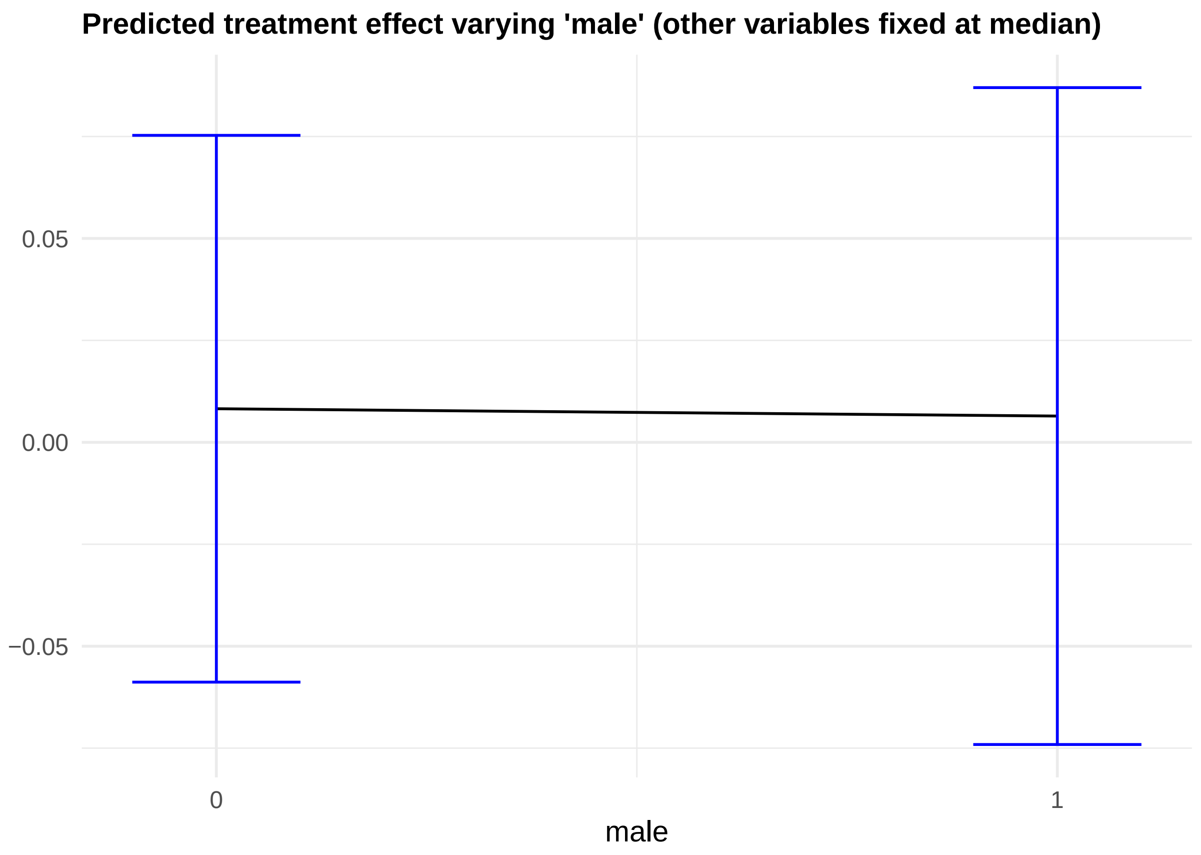
```r
# Compare regions with above/below median CATEs (from Wager & Athey)
high_effect <- social.tau.hat > median(social.tau.hat)
ate.high <- average_treatment_effect( social.cf , subset = high_effect )
ate.low <- average_treatment_effect( social.cf, subset =! high_effect )
paste ("95% CI for difference in ATE:",
       round(ate.high[1] - ate.low[1] , 3) , "+/-",
       round(qnorm(0.975) * sqrt ( ate.high[2]^2 + ate.low[2]^2) , 3))
```

```
## [1] "95% CI for difference in ATE: -0.08 +/- 0.067"
```

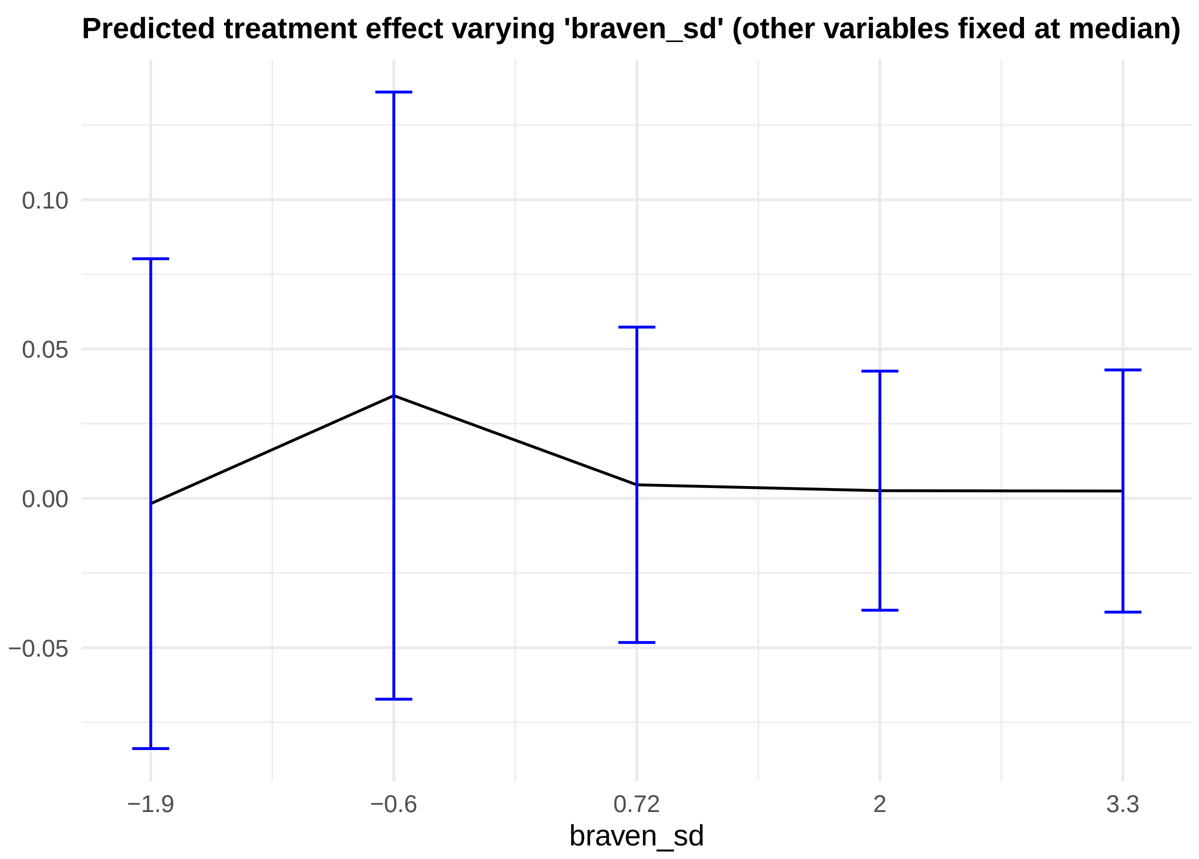
```r
# Partial dependence
partial_dependence_single(selected.covariate = 'refugee',
                          covariates = colnames(social_list$X)[2:length(colnames(social_list$X))],
                          type = 'binary', X = social_list$X,
                          causal.forest = social.cf)
```



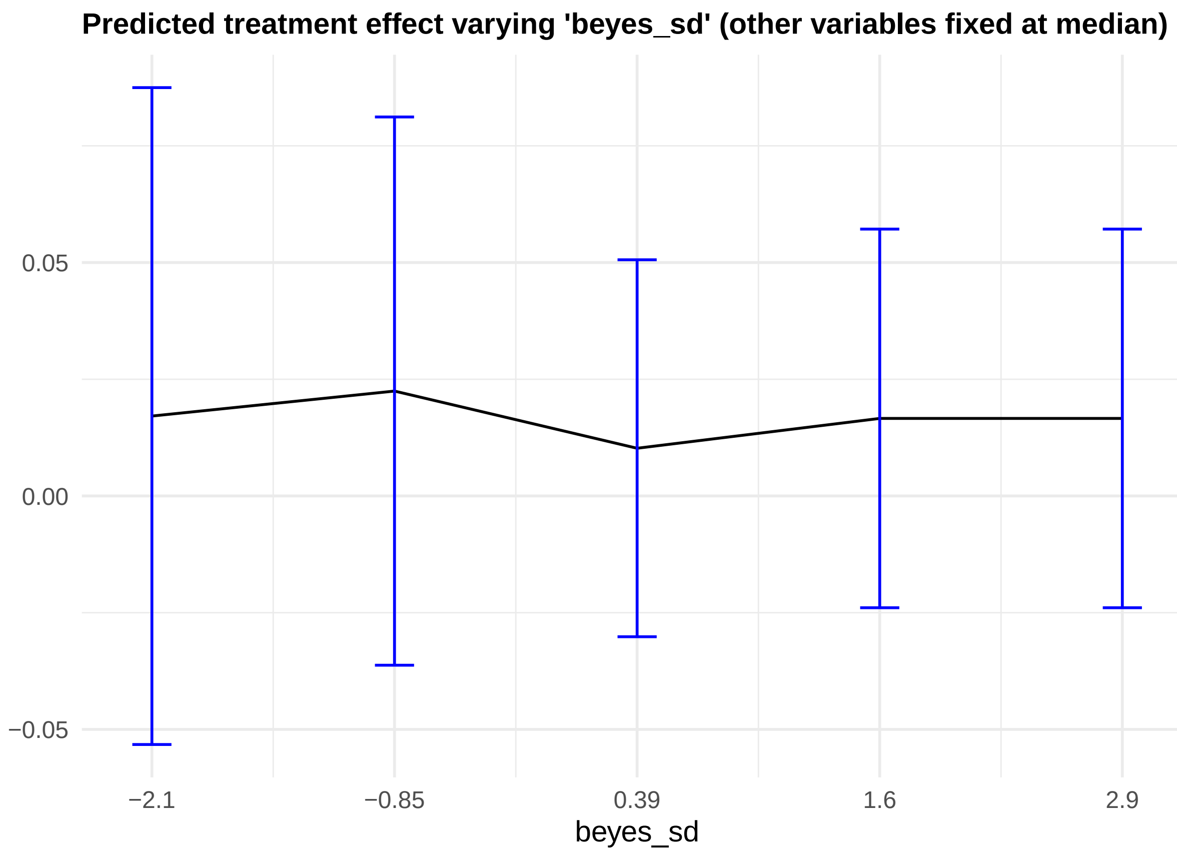
```
partial_dependence_single(selected.covariate = 'male',
                          covariates = social.covariates,
                          type = 'binary', X = social_list$X,
                          causal.forest = social.cf)
```



```r
partial_dependence_single(selected.covariate = 'braven_sd',
                          covariates = social.covariates,
                          type = 'non-binary', X = social_list$X,
                          causal.forest = social.cf, grid_size = 5)
```



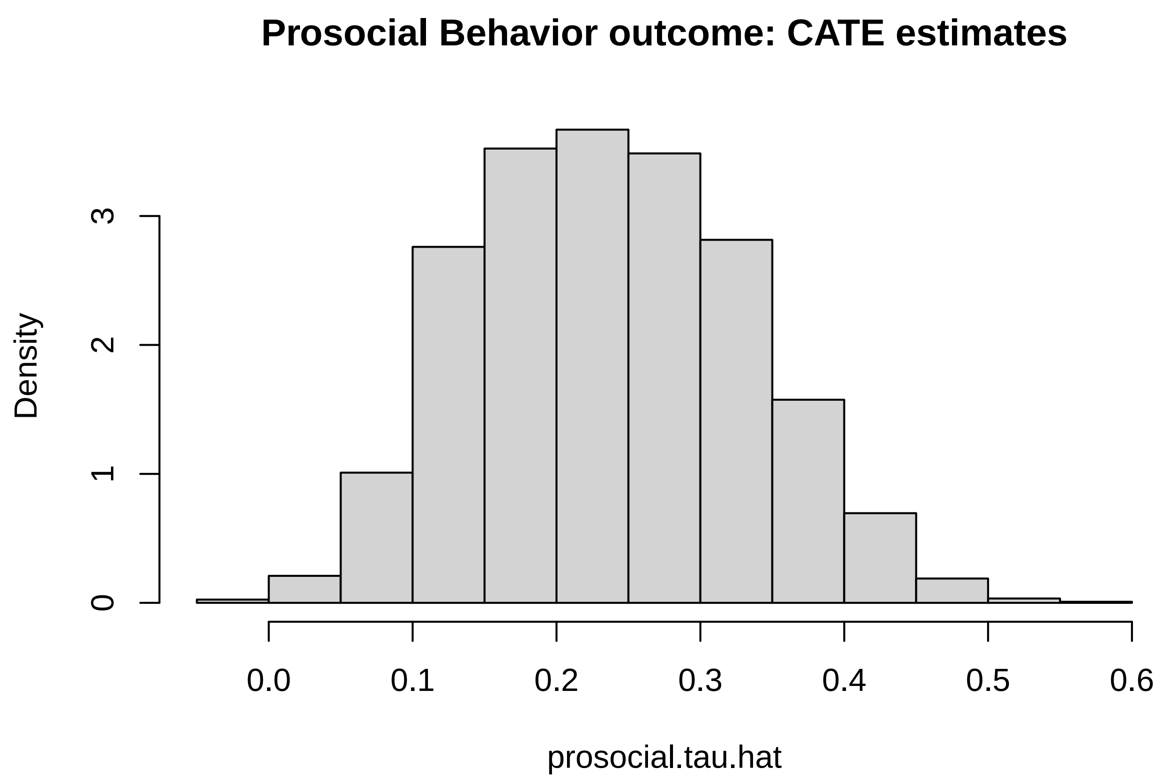
```
partial_dependence_single(selected.covariate = 'beyes_sd',
                          covariates = social.covariates,
                          type = 'non-binary', X = social_list$X,
                          causal.forest = social.cf, grid_size = 5)
```



```r
# Regress AIPW scores on group membership
social.aipw <- get_AIPW_scores(social_list, social.cf)

social.aipw.ols <- lm(formula = 'social.aipw ~ refugee',
          data = transform(social_list$X, aipw.scores = social.aipw))
social.ols.res <- coeftest(social.aipw.ols, vcov = vcovHC(social.aipw.ols, "HC2"))

social.aipw.ols <- lm(formula = 'social.aipw ~ I(braven_sd < -0.6)',
                      data = transform(social_list$X, aipw.scores = social.aipw))
social.ols.res <- coeftest(social.aipw.ols, vcov = vcovHC(social.aipw.ols, "HC2"))

social.aipw.ols <- lm(formula = 'social.aipw ~ male',
                      data = transform(social_list$X, aipw.scores = social.aipw))
social.ols.res <- coeftest(social.aipw.ols, vcov = vcovHC(social.aipw.ols, "HC2"))

social.aipw.ols <- lm(formula = 'social.aipw ~ beyes_sd',
                      data = transform(social_list$X, aipw.scores = social.aipw))
social.ols.res <- coeftest(social.aipw.ols, vcov = vcovHC(social.aipw.ols, "HC2"))
```

```r
prosocial_list <- generate_X_Y_W_C_raw(outcome = 'fs_decision_out', covariates = prosocial.covariates)
prosocial.n <- dim(prosocial_list$X)[1]
prosocial.cf <- causal_forest(X = prosocial_list$X, Y = prosocial_list$Y,
                           W = prosocial_list$W, clusters = prosocial_list$C, W.hat = .5)

# CATE histogram
prosocial.tau.hat <- predict(prosocial.cf)$predictions
```

```r
hist(prosocial.tau.hat, main="Prosocial Behavior outcome: CATE estimates", freq=F)
```



```r
# Variable importance
prosocial.var_imp <- c(variable_importance(prosocial.cf))
names(prosocial.var_imp) <- prosocial.covariates # TODO What IS NA? The clusters?
prosocial.sorted_var_imp <- sort(prosocial.var_imp, decreasing = TRUE)

# Data-driven subgroups
# TODO pending: How to deal with use of clusters argument for folds vs. for clusters?

# Best linear projection
best_linear_projection(prosocial.cf, prosocial_list$X)
```

```
##
## 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)   0.25506524  0.70405662  0.3623   0.7172
## bstrata       0.00984625  0.01948704  0.5053   0.6134
## b_districtid -0.03175718  0.04548329 -0.6982   0.4851
## ageinm       -0.00254889  0.00516891 -0.4931   0.6220
## male          0.00054830  0.06254567  0.0088   0.9930
## refugee       0.11565937  0.15181799  0.7618   0.4462
## astudent      0.00168500  0.18428468  0.0091   0.9927
## b_schoolsize  0.00023155  0.00029187  0.7933   0.4276
## braven_sd    -0.04873611  0.04507745 -1.0812   0.2797
## beyes_sd     -0.01567330  0.04278361 -0.3663   0.7141
## f_csize       0.00152803  0.01283569  0.1190   0.9052
```

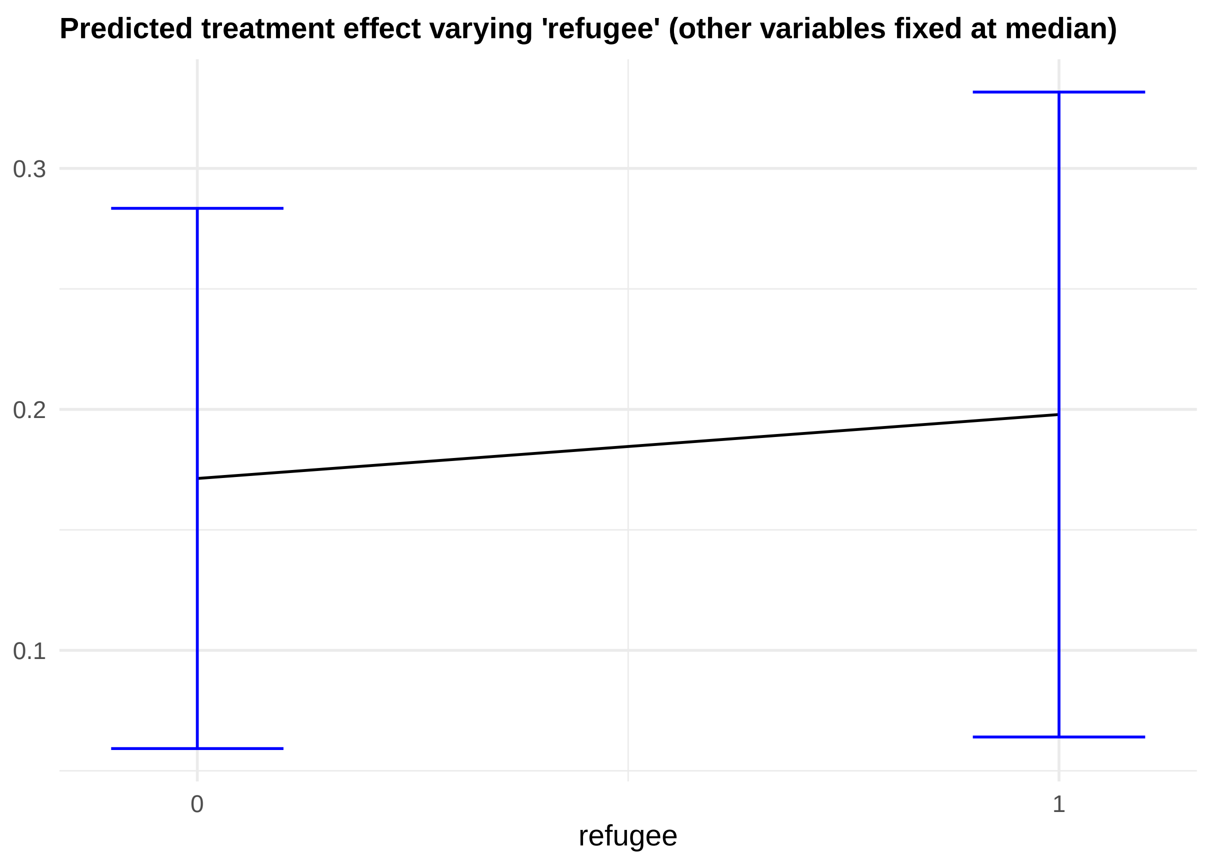
```
# Calibration
test_calibration(prosocial.cf)
```

```
##
## Best linear fit using forest predictions (on held-out data)
## as well as the mean forest prediction as regressors, along
## with one-sided heteroskedasticity-robust (HC3) SEs:
##
##                                Estimate Std. Error t value    Pr(>t)
## mean.forest.prediction          0.99221    0.28740  3.4524 0.0002802 ***
## differential.forest.prediction -1.12079    0.65767 -1.7042 0.9557930
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

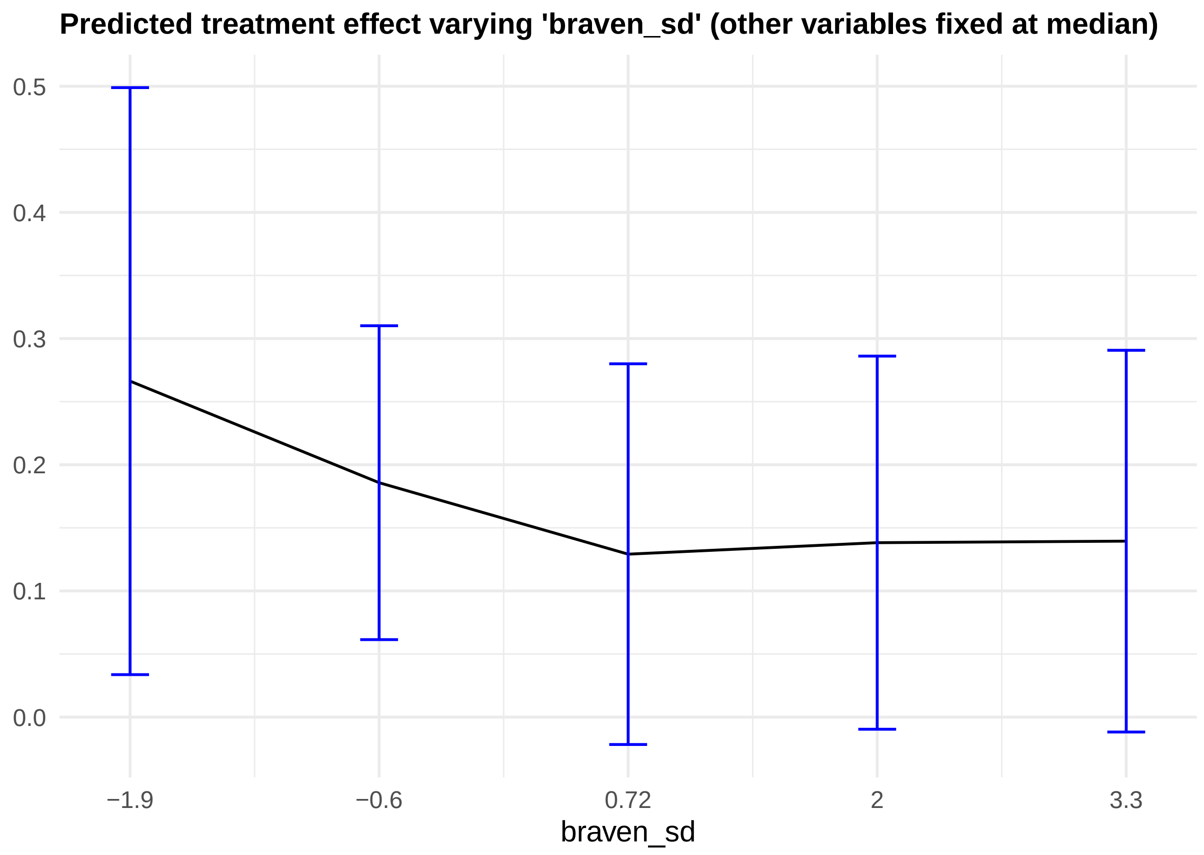
```
# Compare regions with above/below median CATEs (from Wager & Athey)
high_effect <- prosocial.tau.hat > median(prosocial.tau.hat)
ate.high <- average_treatment_effect( prosocial.cf , subset = high_effect )
ate.low <- average_treatment_effect( prosocial.cf, subset =! high_effect )
paste ("95% CI for difference in ATE:",
       round(ate.high[1] - ate.low[1]  , 3)  , "+/-",
       round(qnorm(0.975) * sqrt ( ate.high[2]^2 + ate.low[2]^2) , 3))
```

```
## [1] "95% CI for difference in ATE: -0.125 +/- 0.253"
```

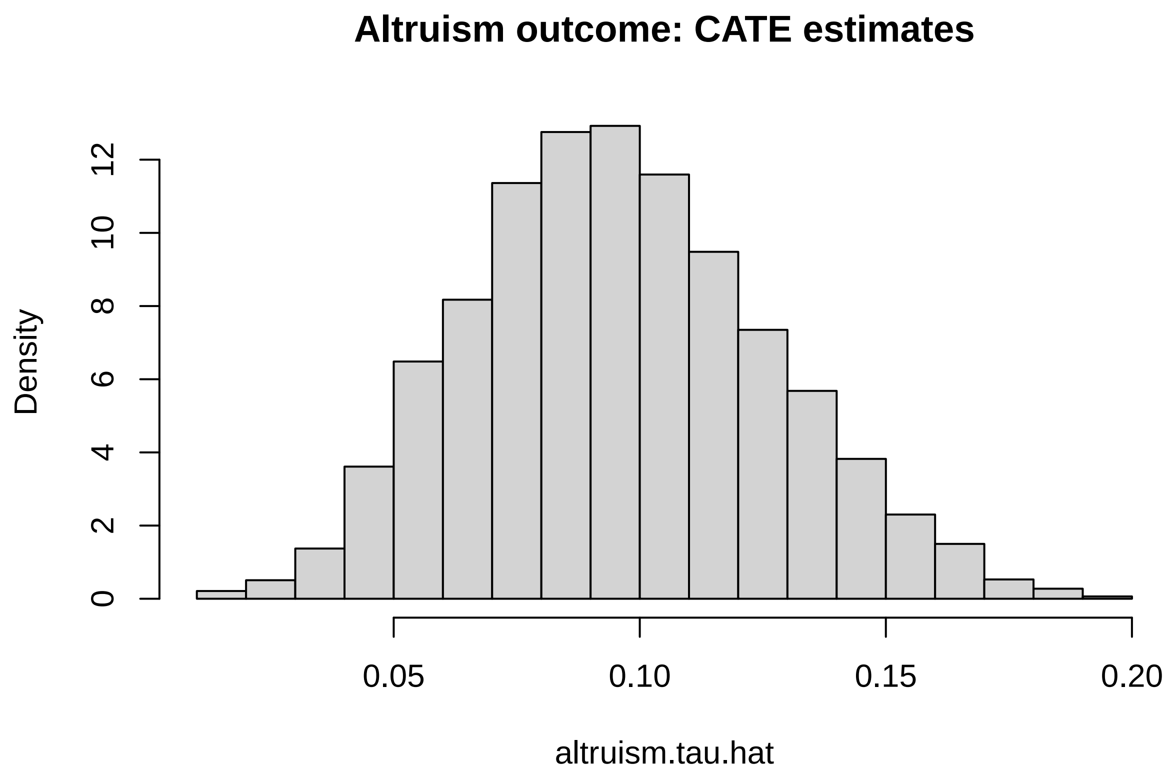
```
# Partial dependence
partial_dependence_single(selected.covariate = 'refugee',
                          covariates = prosocial.covariates,
                          type = 'binary', X = prosocial_list$X,
                          causal.forest = prosocial.cf)
```



```r
partial_dependence_single(selected.covariate = 'braven_sd',
                          covariates = prosocial.covariates,
                          type = 'non-binary', X = prosocial_list$X,
                          causal.forest = prosocial.cf, grid_size = 5)
```



```r
# Fit causal tree
altruism_list <- generate_X_Y_W_C_raw(outcome = 'fdonate', covariates = altruism.covariates)
altruism.n <- dim(altruism_list$X)[1]
altruism.cf <- causal_forest(X = altruism_list$X, Y = altruism_list$Y,
                             W = altruism_list$W, clusters = altruism_list$C, W.hat = .5)

# CATE histogram
altruism.tau.hat <- predict(altruism.cf)$predictions
hist(altruism.tau.hat, main="Altruism outcome: CATE estimates", freq=F)
```



```r
# Variable importance
altruism.var_imp <- c(variable_importance(altruism.cf))
names(altruism.var_imp) <- altruism.covariates # TODO What IS NA? The clusters?
altruism.sorted_var_imp <- sort(altruism.var_imp, decreasing = TRUE)

# Data-driven subgroups
# TODO pending: How to deal with use of clusters argument for folds vs. for clusters?

# Best linear projection
best_linear_projection(altruism.cf, altruism_list$X)
```

```
##
## 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)    0.20091578  0.27813065  0.7224  0.47010
## bdonation_sd -0.00160187  0.01451342 -0.1104  0.91212
## a2             0.01899112  0.02645699  0.7178  0.47291
## bstrata        0.00072805  0.00634683  0.1147  0.90868
## b_districtid  0.01060121  0.01750831  0.6055  0.54488
## ageinm        -0.00182537  0.00179766 -1.0154  0.30996
## male          -0.02225215  0.02603331 -0.8548  0.39273
## refugee        0.05016080  0.05275716  0.9508  0.34176
## astudent      -0.11995136  0.06335417 -1.8933  0.05837 .
## b_schoolsize  0.00005704  0.00010958  0.5205  0.60272
## braven_sd     -0.01022339  0.01740027 -0.5875  0.55687
## beyes_sd       0.00752103  0.01381089  0.5446  0.58607
## f_csize       -0.00089831  0.00535964 -0.1676  0.86690
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

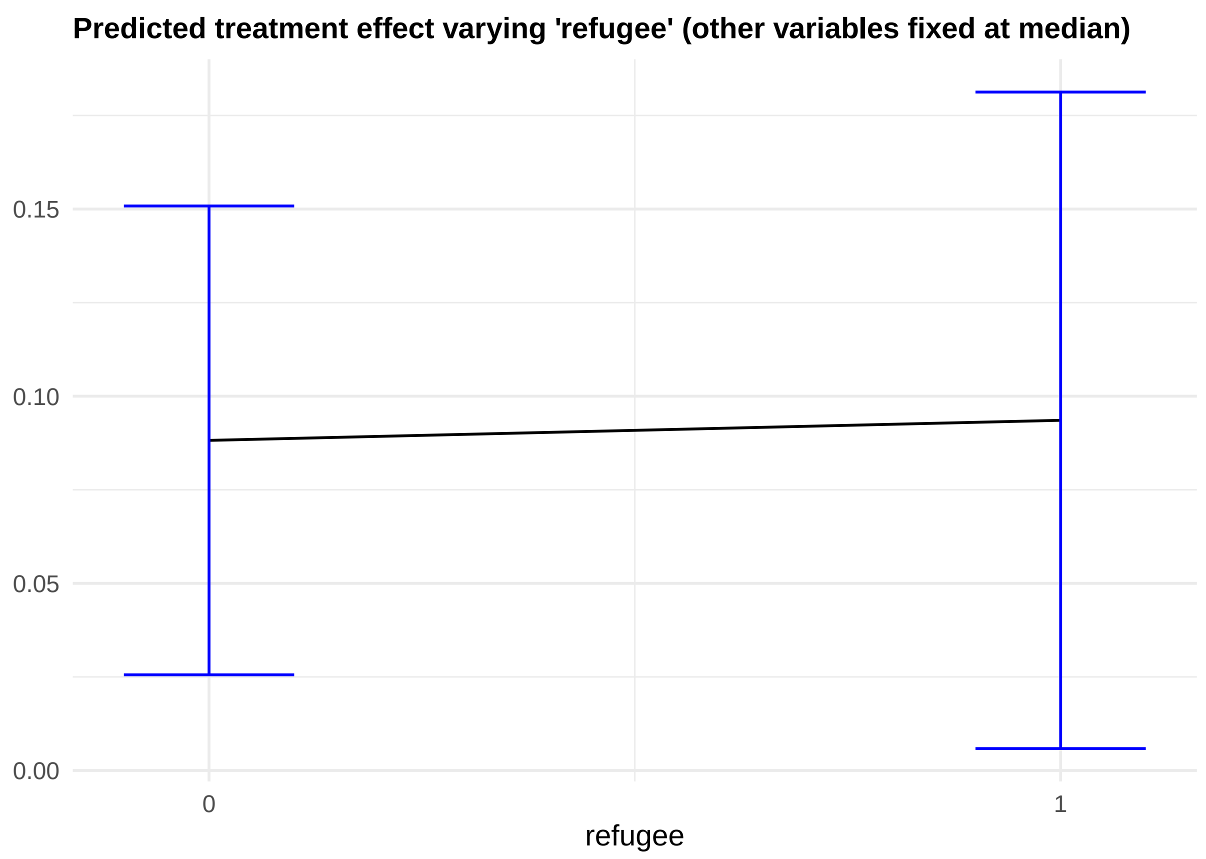
```
# Calibration
test_calibration(altruism.cf)
```

```
##
## Best linear fit using forest predictions (on held-out data)
## as well as the mean forest prediction as regressors, along
## with one-sided heteroskedasticity-robust (HC3) SEs:
##
##                                Estimate Std. Error t value    Pr(>t)
## mean.forest.prediction          0.96102    0.29192  3.2921 0.0005009 ***
## differential.forest.prediction -1.15669    0.74286 -1.5571 0.9402409
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

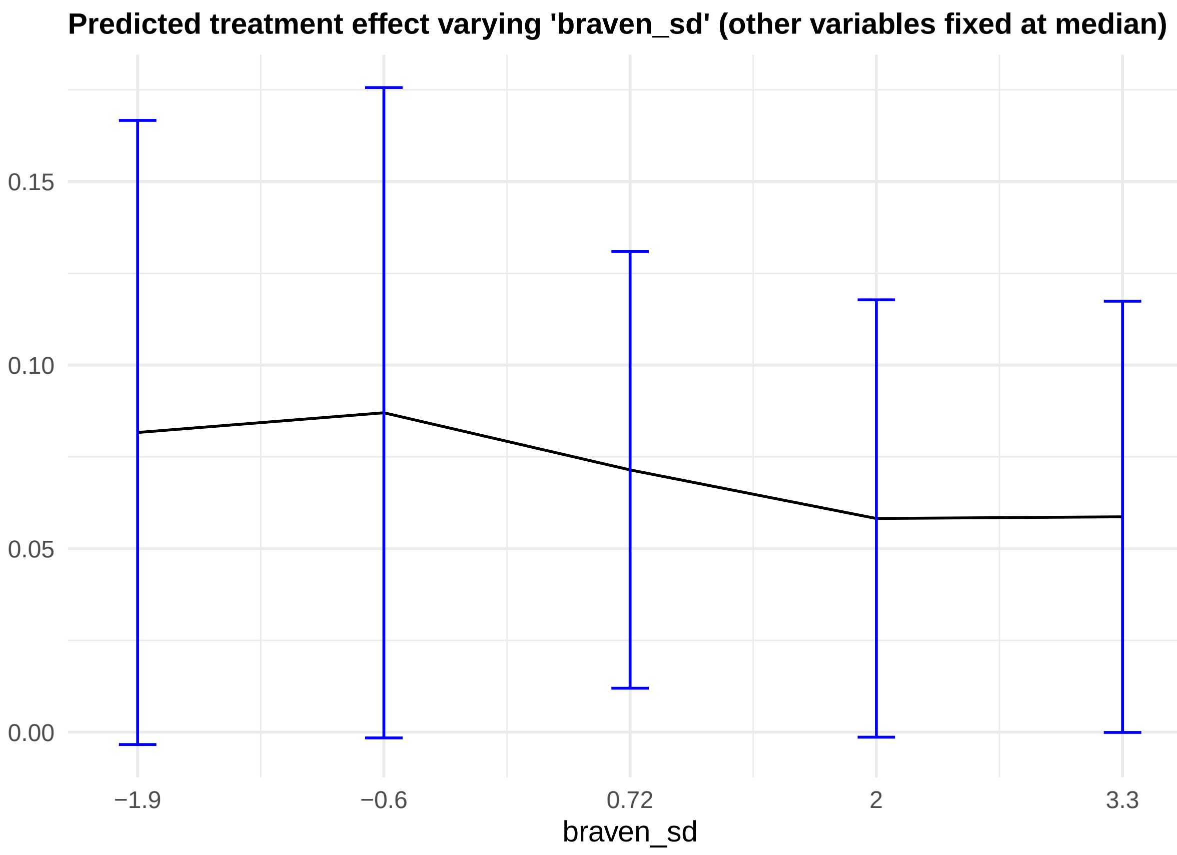
```
# Compare regions with above/below median CATEs (from Wager & Athey)
high_effect <- altruism.tau.hat > median(altruism.tau.hat)
ate.high <- average_treatment_effect( altruism.cf , subset = high_effect )
ate.low <- average_treatment_effect( altruism.cf, subset =! high_effect )
paste ("95% CI for difference in ATE:",
       round(ate.high[1] - ate.low[1] , 3) , "+/-",
       round(qnorm(0.975) * sqrt ( ate.high[2]^2 + ate.low[2]^2) , 3))
```

```
## [1] "95% CI for difference in ATE: -0.059 +/- 0.098"
```

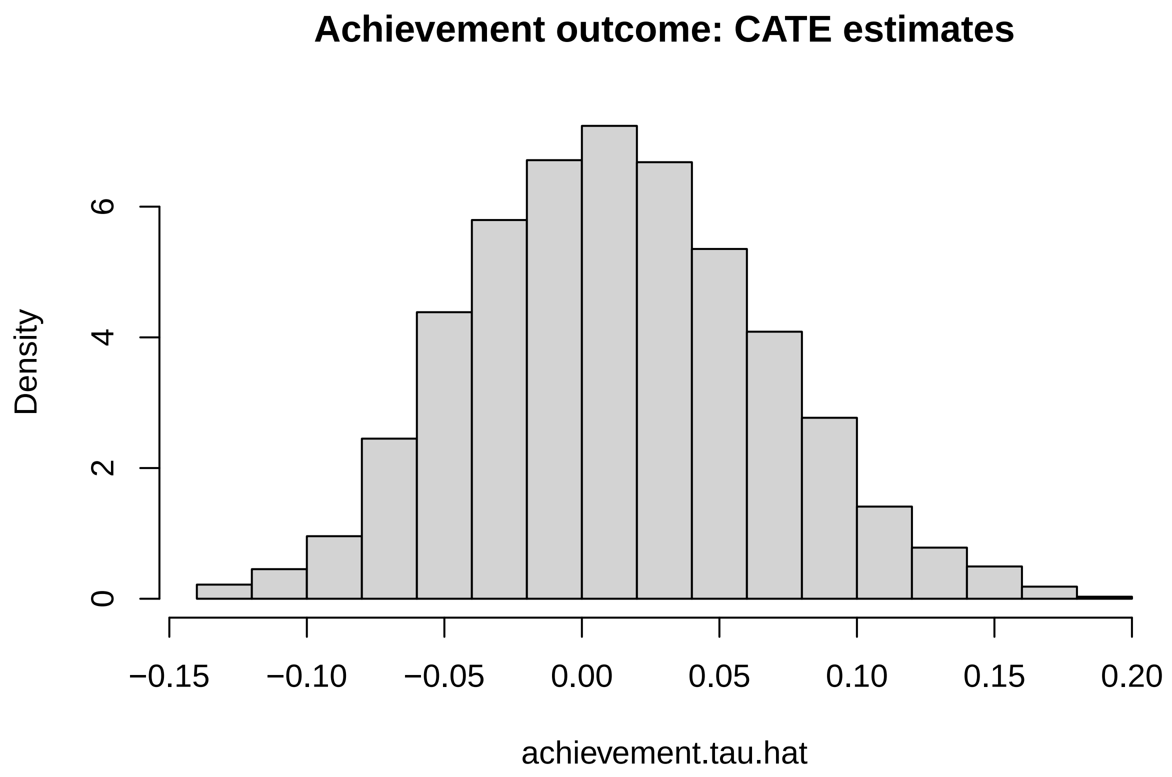
```
# Partial dependence
partial_dependence_single(selected.covariate = 'refugee',
                          covariates = altruism.covariates,
                          type = 'binary', X = altruism_list$X,
                          causal.forest = altruism.cf)
```



```
partial_dependence_single(selected.covariate = 'braven_sd',
                          covariates = altruism.covariates,
                          type = 'non-binary', X = altruism_list$X,
                          causal.forest = altruism.cf, grid_size = 5)
```



```r
# Fit causal tree
achievement_list <- generate_X_Y_W_C_raw(outcome = 'fturk_sd', covariates = achievement.covariates)
achievement.n <- dim(achievement_list$X)[1]
achievement.cf <- causal_forest(X = achievement_list$X, Y = achievement_list$Y,
                              W = achievement_list$W, clusters = achievement_list$C, W.hat = .5)

# CATE histogram
achievement.tau.hat <- predict(achievement.cf)$predictions
hist(achievement.tau.hat, main="Achievement outcome: CATE estimates", freq=F)
```



```r
# Variable importance
achievement.var_imp <- c(variable_importance(achievement.cf))
names(achievement.var_imp) <- achievement.covariates # TODO What IS NA? The clusters?
achievement.sorted_var_imp <- sort(achievement.var_imp, decreasing = TRUE)

# Data-driven subgroups
# TODO pending

# Best linear projection
best_linear_projection(achievement.cf, achievement_list$X)
```

```
##
## 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)   0.30263189  0.49626221  0.6098   0.5420
## bturk_sd      0.03857284  0.03304693  1.1672   0.2432
## bstrata      -0.00370950  0.01702414 -0.2179   0.8275
## b_districtid -0.03294763  0.04321881 -0.7623   0.4459
## ageinm       -0.00201537  0.00396986 -0.5077   0.6117
## male         -0.02896814  0.04815415 -0.6016   0.5475
## refugee       0.00039259  0.13380377  0.0029   0.9977
## astudent      0.08502162  0.11657845  0.7293   0.4658
## b_schoolsize  0.00030936  0.00022080  1.4011   0.1612
## braven_sd    -0.01132873  0.02968488 -0.3816   0.7028
## beyes_sd      0.00085939  0.03078287  0.0279   0.9777
## f_csize       0.00227010  0.01087499  0.2087   0.8347
```

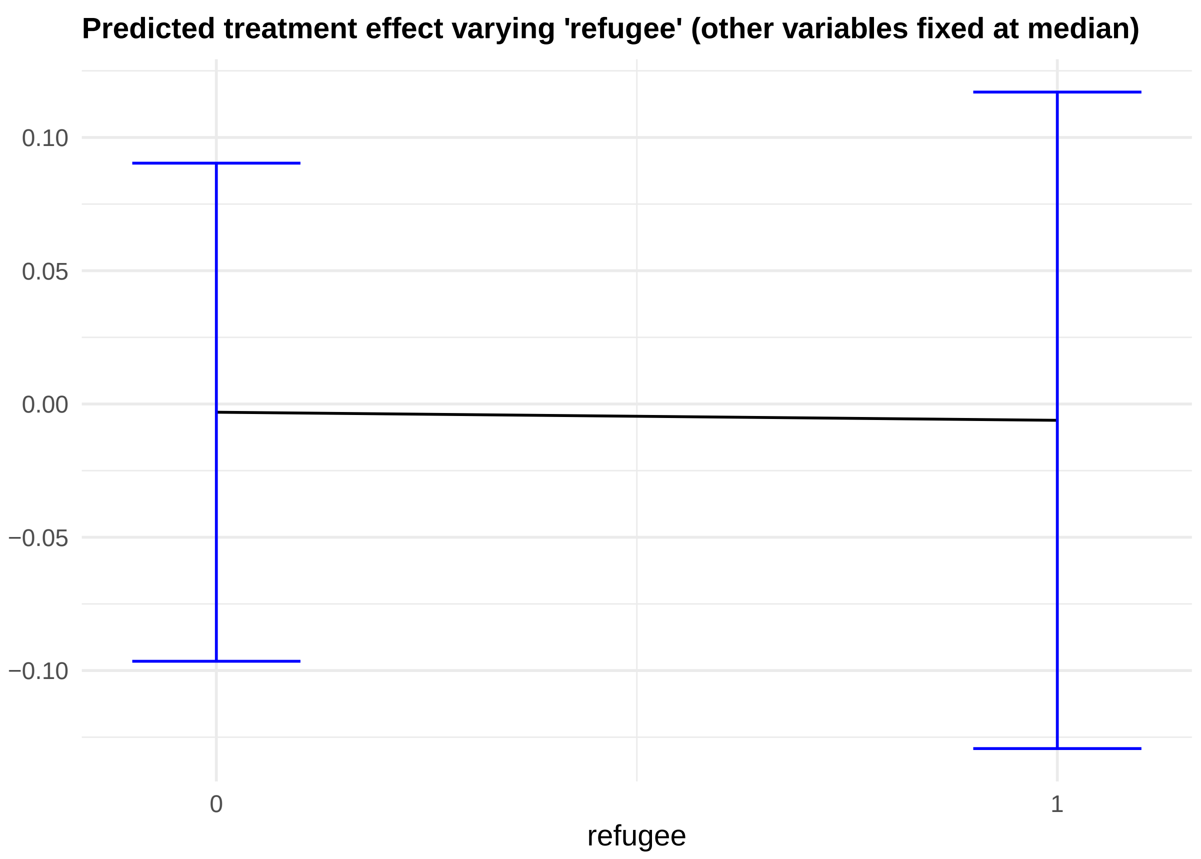
```
# Calibration
test_calibration(achievement.cf)
```

```
##
## Best linear fit using forest predictions (on held-out data)
## as well as the mean forest prediction as regressors, along
## with one-sided heteroskedasticity-robust (HC3) SEs:
##
##                                Estimate Std. Error t value Pr(>t)
## mean.forest.prediction          1.55781    4.34490  0.3585 0.3600
## differential.forest.prediction -1.43385    0.83401 -1.7192 0.9572
```

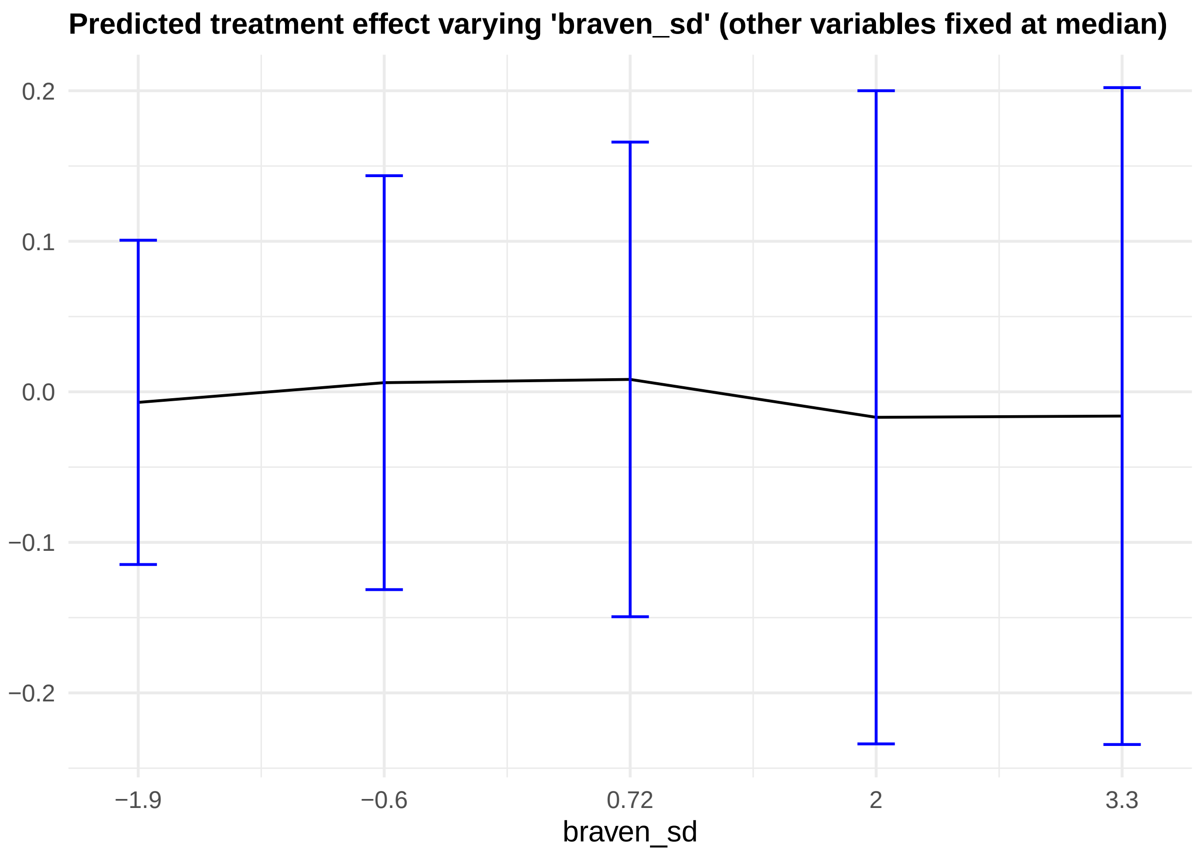
```
# Compare regions with above/below median CATEs (from Wager & Athey)
high_effect <- achievement.tau.hat > median(achievement.tau.hat)
ate.high <- average_treatment_effect( achievement.cf , subset = high_effect )
ate.low <- average_treatment_effect( achievement.cf, subset =! high_effect )
paste ("95% CI for difference in ATE:",
       round(ate.high[1] - ate.low[1] , 3) , "+/-",
       round(qnorm(0.975) * sqrt ( ate.high[2]^2 + ate.low[2]^2) , 3))
```

```
## [1] "95% CI for difference in ATE: -0.073 +/- 0.203"
```

```
# Partial dependence
partial_dependence_single(selected.covariate = 'refugee',
                          covariates = achievement.covariates,
                          type = 'binary', X = achievement_list$X,
                          causal.forest = achievement.cf)
```



```
partial_dependence_single(selected.covariate = 'braven_sd',
                          covariates = achievement.covariates,
                          type = 'non-binary', X = achievement_list$X,
                          causal.forest = achievement.cf, grid_size = 5)
```


Predicted treatment effect varying 'braven_sd' (other variables fixed at median)
0.2
0.1
0.0
−0.1
−0.2
−1.9
−0.6
0.72
2
3.3
braven_sd