Characterizing Effect Heterogeneity 2

Cyrus Samii

2019-06-20

Contents

1 Estimating Conditional Treatment Effects (CATES)

1.1 Setting

Some notation following Chernozhukov et al.:

- Potential outcomes, $Y(1), Y(0)$.
- Covariates X.
- Baseline conditional average (BCA): $b_0(X) = E[Y(0)|X]$.
- CATE: $s_0(X) = E[Y(1)|X] - E[Y(0)|X]$.
- CIA holds: $D \perp\!\!\!\perp (Y(1), Y(0))|X$.
- Subvector of stratifying covariates, $X_1 \subseteq X$, such that
- propensity score is $p(X) = P[D = 1|X_1]$, with $0 < p_0 \leq p(X) \leq p_1 < 1$.
- Observe $Y = DY(1) + (1 - D)Y(0)$, in which case,
- $Y = b_0(X) + Ds_0(X) + U$, with $E[U|X, D] = 0$, where
- $b_0(X) = E[Y|D = 0, X]$ and $s_0(X) = E[Y|D = 1, X] - E[Y|D = 0, X]$. CIA allows us to write b_0 and s_0 in terms of observables.
- Observe N iid draws of (Y, X, D) with law P. Draws are indexed by $i = 1, ..., N$.

A result to keep in mind (already noted in Athey and Imbens PNAS, I believe):
Define

$$H = H(D, X) = \frac{D - p(X)}{p(X)(1 - p(X))}.$$

Given the assumptions above, we have,

$$\begin{aligned}
E[YH|X] &= E\left[\frac{Y(D - p(X))}{p(X)(1 - p(X))}\middle| X\right] \\
&= \frac{1}{p(X)(1 - p(X))} E\left[D(D - p(X))Y(1) + (1 - D)(D - p(X))Y(0)|X\right] \\
&= E[Y(1)|X] - E[Y(0)|X] \\
&= s_0(X).
\end{aligned}$$

We can use YH as an "unbiased signal" about $s_0(X)$. It is, however, a noisy signal, and so in using this, Chernozhukov et al. make adjustments (e.g., for their second BLP method). Nonetheless, it does provide a target that one can use in trying to tune methods for estimating CATEs (see below the section on choosing an ML method).

1.2 Overview and Goals

We consider three types of high-dimensional methods for estimating heterogenous treatment effects: trees, random forests, and "elastic net" regularized regression. The focus initially will be on point predictions, rather than inference on such predictions. This is because the applications that we use work with summaries of the point predictions rather than the point predictions in and of themselves. See the section below on "features of CATES", referencing Chernozhukov et al. (2017, arxiv), for more on this point.

We are also interested in methods that work well when we entertain a high-dimensional covariate vector. I say "entertain" because it may be that, in fact, the covariates that predict heterogeneity are few, but this is something that we do not know *a priori,* and rather we have at our disposal many covariates that we want to consider as candidates for predicting effect heterogeneity. This leads us to machine learning approaches that use regularization to balance that ability to make very fine grained predictions with penalties for overfitting.

We consider the following algorithms:

- Trees
- Random Forests
- Elastic Net

1.3 Data preparation

Bringing in Data (note that we need to harmonize data wrt to section 1):

```r
mex_data <- as.data.frame(import("progresa_mat.dta"))
covs <- setdiff(names(mex_data),
                c('indiv_id',
                  'treatment',
                  'enrolled',
                  'treated_adj',
                  'y_adj',
                  'ml_single_parent',
                  'group'))
```

Checking the data:

```r
misCheck <- as.matrix(apply(mex_data, 2, function(x){sum(is.na(x))}))
colnames(misCheck) <- "Number missing"
kable(misCheck)
```

	Number missing
indiv_id	0
enrolled	0
ml_age	0
ml_n_child	0
ml_male	0
ml_hh_head_male	0
ml_single_parent	0
ml_al_father	0
ml_al_mother	0
ml_lw_father	0
ml_lw_mother	0
ml_literacy	0
ml_toilet	0
ml_water_piped	0
ml_lights	0
ml_tv	0
ml_car	0
ml_refrige	0
ml_n_total	0
ml_yrs_educ	0
ml_hh_head_edu	0
ml_father_educ	0
ml_mother_educ	0
ml_male_mi	0
ml_al_father_mi	0
ml_al_mother_mi	0
ml_literacy_mi	0
ml_toilet_mi	0
ml_water_piped_mi	0
ml_lights_mi	0
ml_tv_mi	0
ml_car_mi	0

	Number missing
ml_refrige_mi	0
ml_yrs_educ_mi	0
ml_father_educ_mi	0
ml_mother_educ_mi	0
treatment	0

2 Trees

2.1 Trees for potential outcomes

We start with using a regression tree to model potential outcomes. This helps us to understand how the trees work and the settings that we need to fix. We just use all available covariates. The `rpart()` function loads when we load the `causalTree` package. We will use 10-fold cross validation (CV) to estimated CV-error that we will minimize in selecting the complexity parameter to prune the tree.

We evaluate covariate importance using the built-in evaluation function. For a classification problem such that this, a given covariate's importance is based on the sum of standardized goodness of classification values for all splits that involves it.

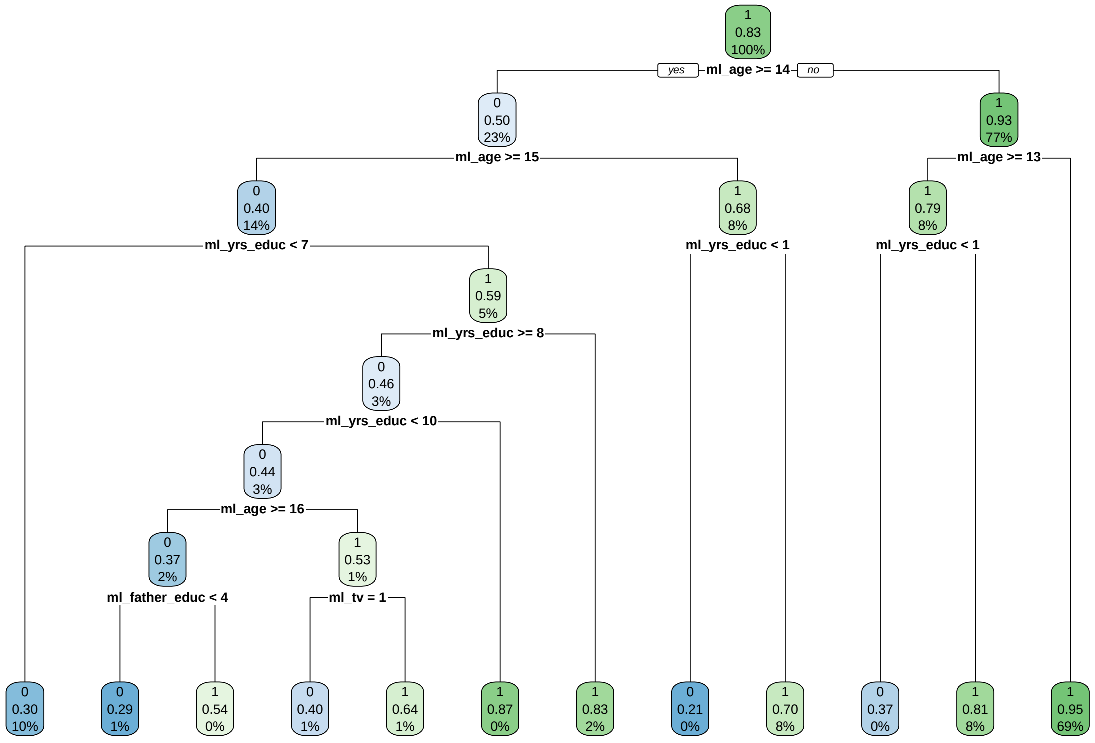
Here is the function:

```r
print(rpartOp)
```

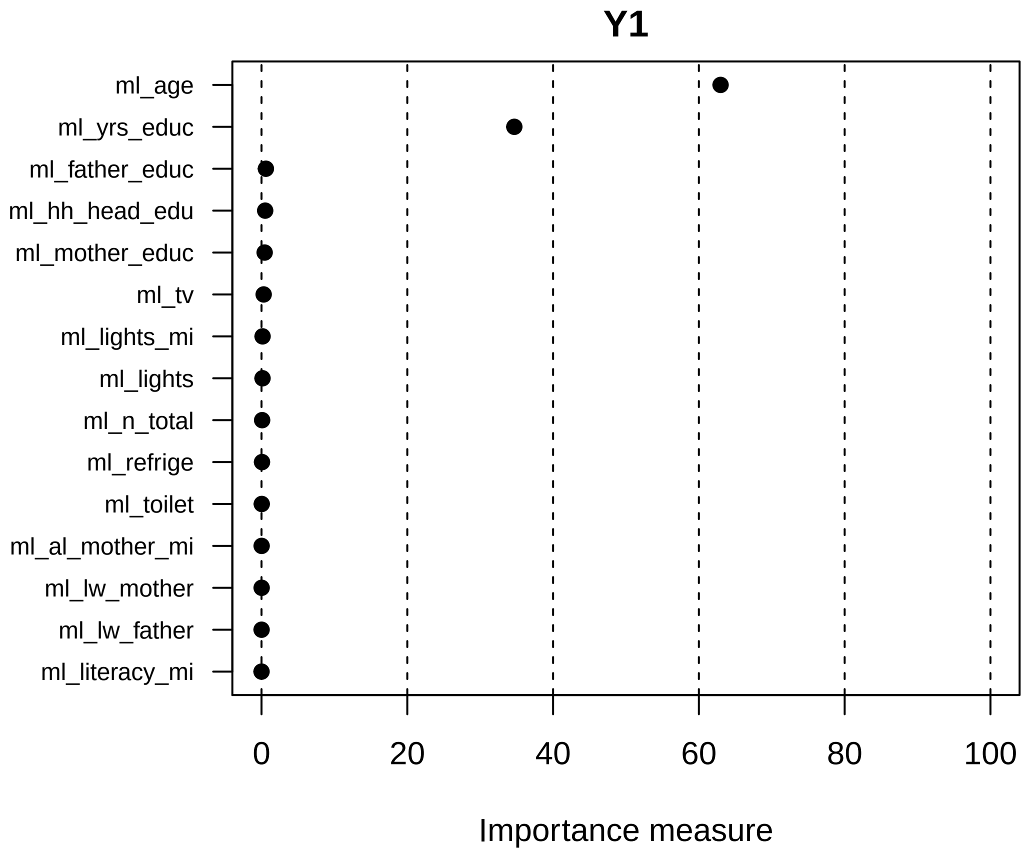
```
## function (formulaIn = NULL, methodIn = NULL, dataIn = NULL, xvalIn = NULL,
##     minbucketIn = NULL, cp_incIN = NULL, target = NULL, treatVar = NULL)
## {
##     if (target == "po") {
##         treeOut <- rpart(formulaIn, method = methodIn, data = dataIn,
##             control = rpart.control(xval = xvalIn, minbucket = minbucketIn,
##                 cp = cp_incIN), model = TRUE)
##     }
##     if (target == "fx") {
##         treeOut <- causalTree(formulaIn, data = dataIn, treatment = treatVar,
##             split.Rule = "CT", split.Honest = F, cv.option = "CT",
##             cv.Honest = F, split.alpha = 1, model = TRUE, control = rpart.control(xval = xvalIn,
##                 minbucket = minbucketIn, cp = cp_incIN))
##     }
##     opcp <- treeOut$cptable[, 1][which.min(treeOut$cptable[,
##         4])]
##     optreeOut <- prune(treeOut, opcp)
##     resList <- list(tree = optreeOut, varimp = 100 * (optreeOut$variable.importance/sum(optreeOut$va
##     return(resList)
## }
```

And now results:

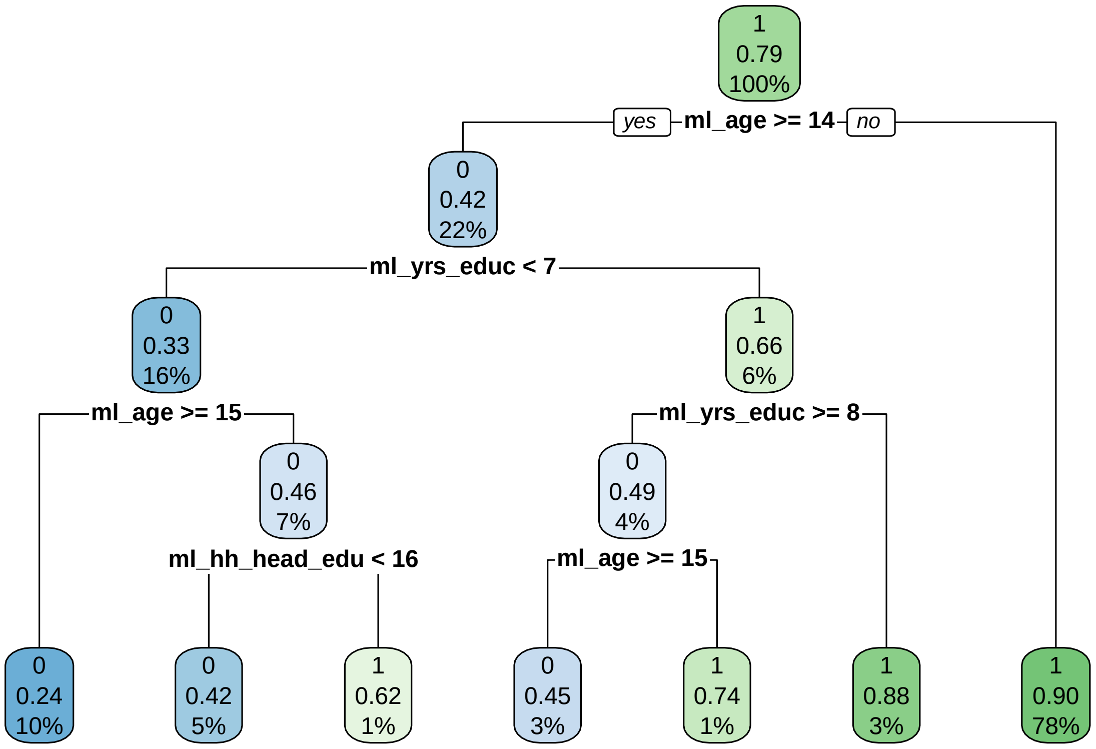
```r
mexFormula <- as.formula(paste("enrolled ~", paste(covs, collapse="+")))
mex_data1 <- subset(mex_data, treatment==1)
mex_data0 <- subset(mex_data, treatment==0)
tree_y1 <- rpartOp(formulaIn=mexFormula,
                   methodIn="class",
                   dataIn=mex_data1,
                   xvalIn=10,
                   minbucketIn=2,
                   cp_incIN=0,
                   target="po")
rpart.plot(tree_y1$tree)
```



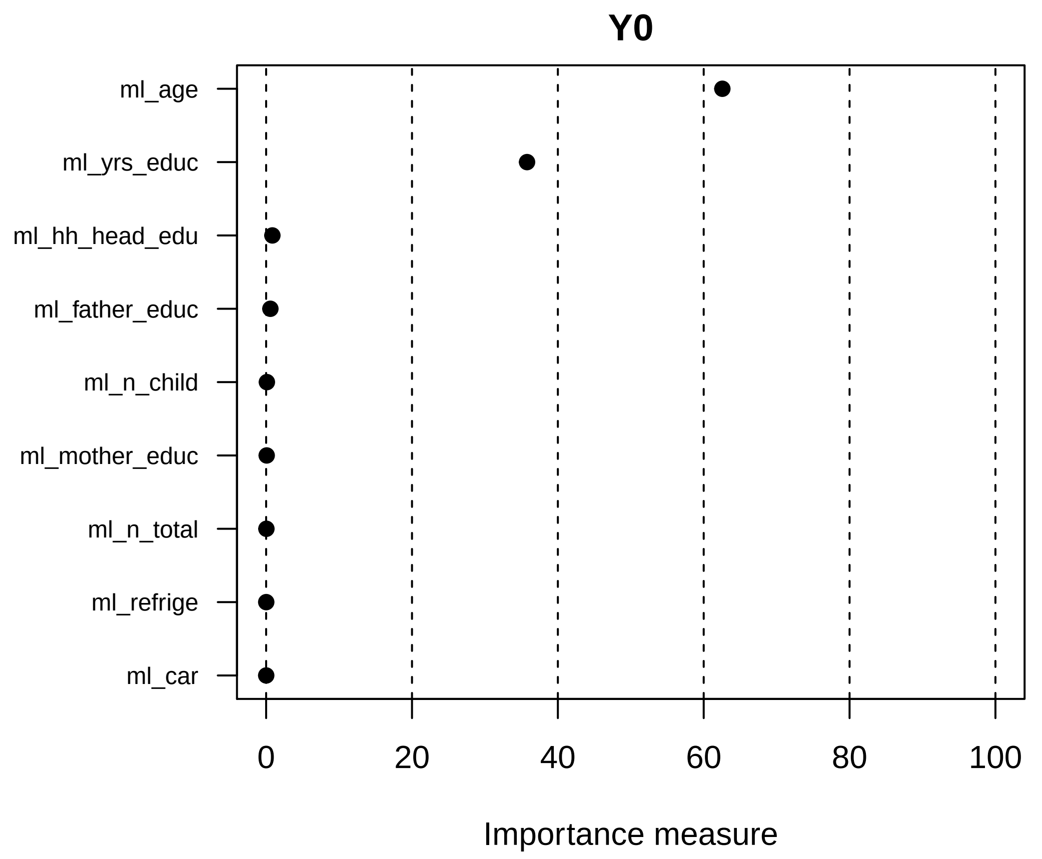
```r
par(mar=c(4,10,2,2))
rpart_impplot(treeIn=tree_y1, mainIn="Y1", labelScale=0.75)
```



```r
tree_y0 <- rpartOp(formulaIn=mexFormula,
                   methodIn="class",
                   dataIn=mex_data0,
                   xvalIn=10,
                   minbucketIn=2,
                   cp_incIN=0,
                   target="po")
rpart.plot(tree_y0$tree)
```



```r
par(mar=c(4,10,2,2))
rpart_impplot(treeIn=tree_y0, mainIn="Y0", labelScale=0.75)
```



Looking at the results for the treated and control outcomes, we see that in both cases, the optimal trees gain predictive traction almost entirely by splitting on age and years of education. The other variables contribute to predictive accuracy only to a neglible degree.

2.2 Interpreting the `rpart` fit

Here is a primer on what `rpart` is doing.

First, we can see a textual representation of the tree. Child nodes are always $(2x, 2x+1)$ of a parent x.

```
print(tree_y1$tree)
```

```
## n= 17083
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 17083 2866 1 (0.16776913 0.83223087)
##     2) ml_age>=13.5 3859 1926 0 (0.50090697 0.49909303)
##       4) ml_age>=14.5 2472  980 0 (0.60355987 0.39644013)
##         8) ml_yrs_educ< 6.5 1660  503 0 (0.69698795 0.30301205) *
##         9) ml_yrs_educ>=6.5 812  335 1 (0.41256158 0.58743842)
##          18) ml_yrs_educ>=7.5 525  240 0 (0.54285714 0.45714286)
##            36) ml_yrs_educ< 9.5 502  220 0 (0.56175299 0.43824701)
##              72) ml_age>=15.5 287  105 0 (0.63414634 0.36585366)
##               144) ml_father_educ< 3.5 204   60 0 (0.70588235 0.29411765) *
##               145) ml_father_educ>=3.5 83   38 1 (0.45783133 0.54216867) *
##              73) ml_age< 15.5 215  100 1 (0.46511628 0.53488372)
##               146) ml_tv>=0.5 95   38 0 (0.60000000 0.40000000) *
##                147) ml_tv< 0.5 120   43 1 (0.35833333 0.64166667) *
##             37) ml_yrs_educ>=9.5 23    3 1 (0.13043478 0.86956522) *
##           19) ml_yrs_educ< 7.5 287   50 1 (0.17421603 0.82578397) *
##        5) ml_age< 14.5 1387  441 1 (0.31795242 0.68204758)
##         10) ml_yrs_educ< 0.5 58   12 0 (0.79310345 0.20689655) *
##         11) ml_yrs_educ>=0.5 1329  395 1 (0.29721595 0.70278405) *
##      3) ml_age< 13.5 13224  933 1 (0.07055354 0.92944646)
##        6) ml_age>=12.5 1429  299 1 (0.20923723 0.79076277)
##         12) ml_yrs_educ< 0.5 59   22 0 (0.62711864 0.37288136) *
##         13) ml_yrs_educ>=0.5 1370  262 1 (0.19124088 0.80875912) *
##        7) ml_age< 12.5 11795  634 1 (0.05375159 0.94624841) *
```

The detailed output is given by `summary`.

- At the top are the results of the CV assessment of error. We use this to find the CV-optimal tree.

- Then we have the scaled (to sum to 100) variable importance scores for variables with scaled scores of at least one.
- Then we have information on the candidate splits at each node going down the tree. Here are details for the binary classification case for a node, which we will refer to as node A.
  - We have number of observations in node A, a complexity parameter for this split, an indicator for what class is being predicted, and then expected loss is just $1 - p_1(A)$, where $p_1(A) = \Pr[Y = 1|A]$, and then the probability that a random unit would arrive at this node.
  - Candidate splits are ordered on the basis of their contributions to increasing predictive accuracy. The resulting tree just uses the first of these. We have "primary" and "surrogate" splits listed. The "surrogates" are what one uses if the "primary" variable exhibits missing data. If there is no missingness, one just uses the "primary".
  - For each of the candidate splits, we have an "improve" measure. These are what we add up to determine variable importance. As per the `rpart` vignette, the "improve" measure equals N times the change in the impurity. The default impurity function for binary classification uses Gini ($f(p) = p(1-p)$) and thus defines impurity at node A as

$$I(A) = p_0(A)(1 - p_0(A)) + p_1(A)(1 - p_1(A)) = 2p_1(A)(1 - p_1(A)).$$

    Let $C_k(A)$ be the optimal split at node A for variable k. Then, the change in impurity after splitting at node A on the basis of $C_k(A)$ is given by

$$\Delta_I(C_k(A), A) = I(A) - \Pr[A_L(C_k(A))] * I(A_L(C_{d,k})) - \Pr[A_R(C_{d,k})] * I(A_R(C_k(A))).$$

    Finally, the "improve" score is given by $N(A)\Delta_I(C_k(A), A)$. We can show the calculation for the first and second nodes:

```
# improve at node 1:
N_A <- 10361
I_A <- 2*0.79432487*(1-0.79432487)
p_AL <- 2324/N_A
I_AL <- 2*0.42211704*(1-0.42211704)
p_AR <- 8037/N_A
I_AR <- 2*0.90195347*(1-0.90195347)
N_A*(I_A - p_AL*I_AL - p_AR*I_AR)
```

```
## [1] 830.1277
```

```r
# improve at node 2:
N_A <- 2324
I_A <- 2*0.42211704*(1-0.42211704)
p_AL <- 1673/N_A
I_AL <- 2*0.32994620*(1-0.32994620)
p_AR <- 651/N_A
I_AR <- 2*0.65898618*(1-0.65898618)
N_A*(I_A - p_AL*I_AL - p_AR*I_AR)
```

```
## [1] 101.4771
```

We will see that this matches what we have below. Then, the unscaled variable importance score for variable k is the sum of these improve scores for any split in which the variable is used. That is, if the optimal split at node A is $C(A)$, then

$$VI_{k,unsc} = \sum_A I[C(A) = C_k(A)]\Delta_I(C_k(A), A).$$

The scaled variable importance scores rescale everything so that they sum to 100:

$$VI_{k,sc} = 100 \times \frac{VI_{k,unsc}}{\sum_k VI_{k,unsc}}.$$

Okay, now we can look at the output for the control potential outcomes tree:

```r
summary(tree_y0$tree)
```

```
## Call:
## rpart(formula = formulaIn, data = dataIn, method = methodIn,
##     model = TRUE, control = rpart.control(xval = xvalIn, minbucket = minbucketIn,
##         cp = cp_incIN))
##   n= 10361
##
##            CP nsplit rel error    xerror       xstd
## 1 0.169873299      0 1.0000000 1.0000000 0.01930667
## 2 0.097137494      1 0.8301267 0.8301267 0.01797326
## 3 0.007977475      2 0.7329892 0.7329892 0.01709121
## 4 0.007742844      4 0.7170343 0.7358048 0.01711817
## 5 0.004223369      6 0.7015486 0.7114031 0.01688162
##
## Variable importance
##         ml_age    ml_yrs_educ ml_hh_head_edu ml_father_educ
##             63             36              1              1
##
## Node number 1: 10361 observations,    complexity param=0.1698733
##   predicted class=1  expected loss=0.2056751  P(node) =1
##     class counts:  2131  8230
##    probabilities: 0.206 0.794
##   left son=2 (2324 obs) right son=3 (8037 obs)
##   Primary splits:
##       ml_age         < 13.5 to the right, improve=830.12760, (0 missing)
##       ml_yrs_educ    < 4.5  to the right, improve=442.53780, (0 missing)
##       ml_mother_educ < 5.5  to the left,  improve= 59.74968, (0 missing)
##       ml_father_educ < 3.5  to the left,  improve= 42.11586, (0 missing)
##       ml_hh_head_edu < 12.5 to the left,  improve= 36.91415, (0 missing)
##   Surrogate splits:
##       ml_yrs_educ < 5.5  to the right, agree=0.873, adj=0.435, (0 split)
##
## Node number 2: 2324 observations,    complexity param=0.09713749
##   predicted class=0  expected loss=0.422117  P(node) =0.2243027
##     class counts:  1343   981
##    probabilities: 0.578 0.422
##   left son=4 (1673 obs) right son=5 (651 obs)
##   Primary splits:
##       ml_yrs_educ    < 6.5  to the left,  improve=101.47710, (0 missing)
##       ml_age         < 14.5 to the right, improve= 42.56597, (0 missing)
##       ml_hh_head_edu < 15.5 to the left,  improve= 22.39856, (0 missing)
##       ml_mother_educ < 5.5  to the left,  improve= 18.66620, (0 missing)
##       ml_father_educ < 3.5  to the left,  improve= 15.13115, (0 missing)
##   Surrogate splits:
##       ml_hh_head_edu < 25.5 to the left,  agree=0.727, adj=0.026, (0 split)
##       ml_father_educ < 7.5  to the left,  agree=0.722, adj=0.009, (0 split)
##       ml_mother_educ < 6.5  to the left,  agree=0.722, adj=0.006, (0 split)
##       ml_car         < 0.5  to the left,  agree=0.720, adj=0.002, (0 split)
##
## Node number 3: 8037 observations
##   predicted class=1  expected loss=0.09804653  P(node) =0.7756973
##     class counts:   788  7249
##    probabilities: 0.098 0.902
##
## Node number 4: 1673 observations,    complexity param=0.007977475
##   predicted class=0  expected loss=0.3299462  P(node) =0.1614709
##     class counts:  1121   552
##    probabilities: 0.670 0.330
##   left son=8 (987 obs) right son=9 (686 obs)
##   Primary splits:
##       ml_age         < 14.5 to the right, improve=39.723920, (0 missing)
##       ml_hh_head_edu < 15.5 to the left,  improve=16.298820, (0 missing)
##       ml_father_educ < 3.5  to the left,  improve=15.190950, (0 missing)
##       ml_mother_educ < 4.5  to the left,  improve=10.541080, (0 missing)
##       ml_literacy    < 0.5  to the left,  improve= 8.683832, (0 missing)
##   Surrogate splits:
##       ml_n_child     < 1.5  to the right, agree=0.595, adj=0.012, (0 split)
##       ml_hh_head_edu < 26   to the left,  agree=0.594, adj=0.010, (0 split)
##       ml_father_educ < 8.5  to the left,  agree=0.593, adj=0.007, (0 split)
##
## Node number 5: 651 observations,    complexity param=0.007742844
##   predicted class=1  expected loss=0.3410138  P(node) =0.06283177
##     class counts:   222   429
##    probabilities: 0.341 0.659
##   left son=10 (369 obs) right son=11 (282 obs)
##   Primary splits:
##       ml_yrs_educ    < 7.5  to the right, improve=46.811680, (0 missing)
##       ml_age         < 15.5 to the right, improve=22.969920, (0 missing)
##       ml_water_piped < 0.5  to the right, improve= 3.196313, (0 missing)
##       ml_lw_father   < 0.5  to the left,  improve= 1.942517, (0 missing)
##       ml_literacy_mi < 0.5  to the left,  improve= 1.900351, (0 missing)
##   Surrogate splits:
##       ml_age         < 14.5 to the right, agree=0.682, adj=0.266, (0 split)
##       ml_n_child     < 6.5  to the left,  agree=0.576, adj=0.021, (0 split)
##       ml_n_total     < 10.5 to the left,  agree=0.571, adj=0.011, (0 split)
##       ml_mother_educ < 8.5  to the left,  agree=0.570, adj=0.007, (0 split)
##       ml_refrige     < 0.5  to the left,  agree=0.568, adj=0.004, (0 split)
##
## Node number 8: 987 observations
##   predicted class=0  expected loss=0.2391084  P(node) =0.09526108
##     class counts:   751   236
##    probabilities: 0.761 0.239
##
## Node number 9: 686 observations,    complexity param=0.007977475
##   predicted class=0  expected loss=0.4606414  P(node) =0.06620983
##     class counts:   370   316
##    probabilities: 0.539 0.461
##   left son=18 (546 obs) right son=19 (140 obs)
##   Primary splits:
##       ml_hh_head_edu < 15.5 to the left,  improve=9.094782, (0 missing)
##       ml_father_educ < 3.5  to the left,  improve=9.062705, (0 missing)
##       ml_yrs_educ    < 5.5  to the right, improve=6.623714, (0 missing)
##       ml_mother_educ < 4.5  to the left,  improve=4.994739, (0 missing)
##       ml_n_child     < 5.5  to the right, improve=3.608422, (0 missing)
##   Surrogate splits:
##       ml_father_educ < 3.5  to the left,  agree=0.953, adj=0.771, (0 split)
##       ml_mother_educ < 4.5  to the left,  agree=0.799, adj=0.014, (0 split)
##
## Node number 10: 369 observations,    complexity param=0.007742844
##   predicted class=0  expected loss=0.4932249  P(node) =0.03561432
##     class counts:   187   182
##    probabilities: 0.507 0.493
##   left son=20 (311 obs) right son=21 (58 obs)
##   Primary splits:
##       ml_age            < 14.5 to the right, improve=8.475549, (0 missing)
##       ml_literacy_mi    < 0.5  to the left,  improve=6.355477, (0 missing)
##       ml_water_piped_mi < 0.5  to the left,  improve=6.355477, (0 missing)
##       ml_car_mi         < 0.5  to the left,  improve=6.355477, (0 missing)
##       ml_refrige_mi     < 0.5  to the left,  improve=6.355477, (0 missing)
##
## Node number 11: 282 observations
##   predicted class=1  expected loss=0.1241135  P(node) =0.02721745
##     class counts:    35   247
##    probabilities: 0.124 0.876
##
## Node number 18: 546 observations
##   predicted class=0  expected loss=0.4194139  P(node) =0.05269762
##     class counts:   317   229
##    probabilities: 0.581 0.419
##
## Node number 19: 140 observations
##   predicted class=1  expected loss=0.3785714  P(node) =0.01351221
##     class counts:    53    87
##    probabilities: 0.379 0.621
##
## Node number 20: 311 observations
##   predicted class=0  expected loss=0.4469453  P(node) =0.03001641
##      class counts:   172   139
##     probabilities: 0.553 0.447
##
## Node number 21: 58 observations
##    predicted class=1  expected loss=0.2586207  P(node) =0.005597915
##      class counts:    15    43
##     probabilities: 0.259 0.741
```

2.3 Potential outcome heterogeneity and effect heterogeneity

What are the implications of a covariate's importance for predicting potential outcomes when it comes to predicting effect heterogeneity? We have

$$E[Y(1) - Y(0)|X] = E[Y(1)|X] - E[Y(0)|X]$$

by the linearity of expectations. That being the case, if $E[Y(1)|X] = E[Y(1)]$ and $E[Y(0)|X] = E[Y(0)]$, then it must be that $E[Y(1) - Y(0)|X] = E[Y(1) - Y(0)]$. As such, being a variable that is predictive of either of the potential outcomes is a *necessary* condition for a covariate for it to be predictive of effects. That said, variables that are predictive of potential outcomes may not be predictive of effects. E.g., we could have that

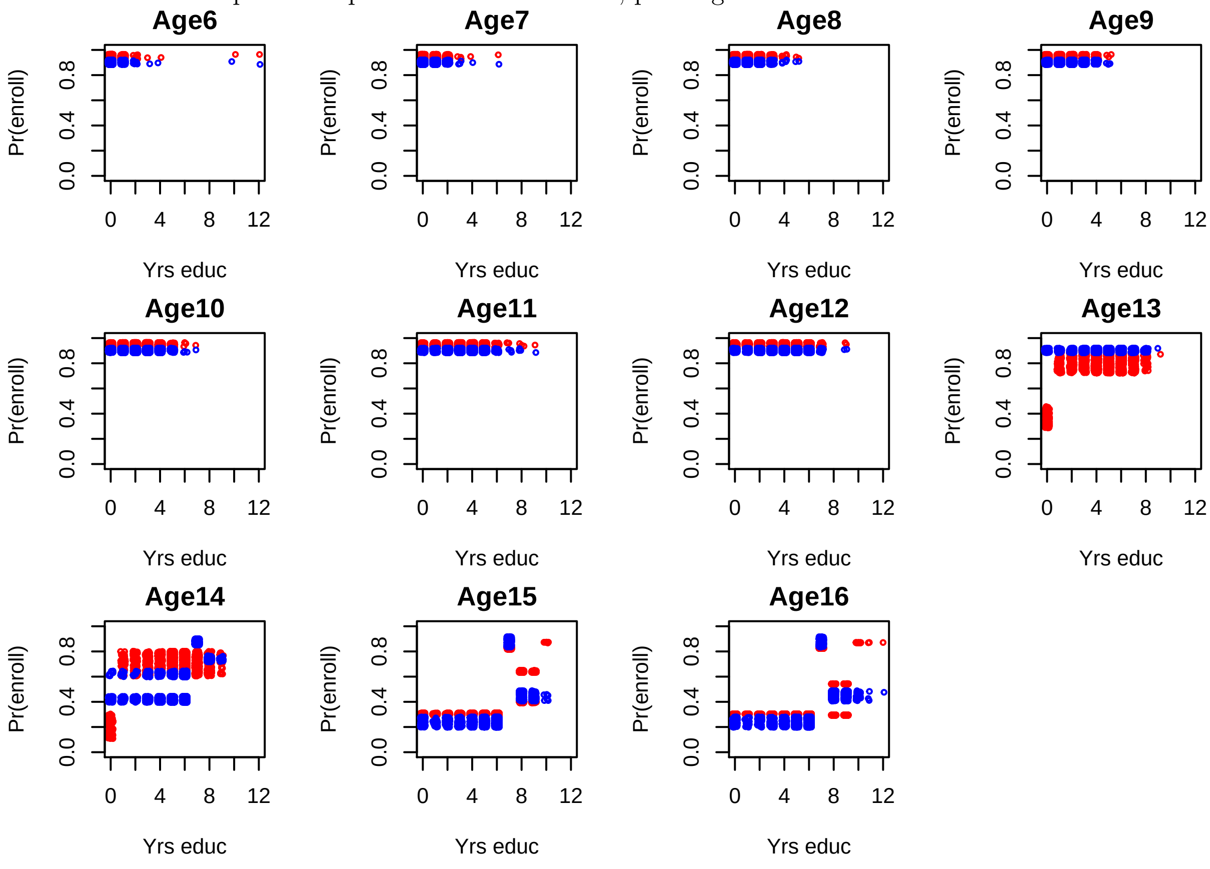
$$E[Y(d)|X] = \alpha + \beta d + \gamma X$$

for $d = 0, 1$, in which case

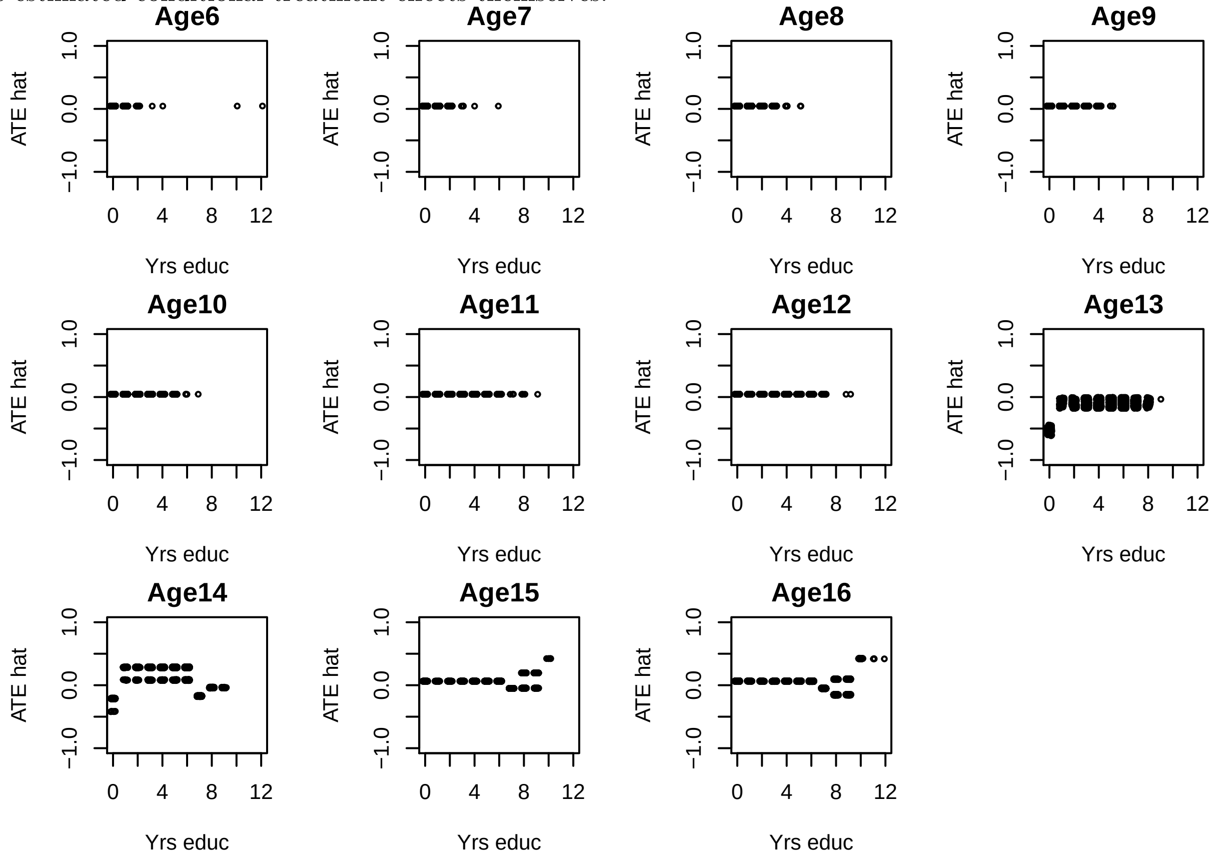
$$E[Y(1) - Y(0)|X] = (\alpha + \beta + \gamma X) - (\alpha + \gamma X) = \beta,$$

which does not depend on X. That said, a starting point for selecting variables to consider as contributors to effect heterogeneity would those that have some predictive relationship with potential outcomes. (Note that this is similar to what is done in Imai and Strauss 2011).

Let us view the predicted potential outcome trees, plotting over the two covariates that seem the matter:



From these plots we can see that for lower age groups, the predicted means are very close, and so effects are basically zero. Then, we have variation for higher age groups. We can see all of this more clearly by plotting the estimated conditional treatment effects themselves:



While this approach is useful for getting a handle on effect heterogeneity, it is not optimized with respect to treatment effects per se. We will turn to such methods next.

2.4 Trees for effect heterogeneity

We now turn to trees for effect heterogeneity. There are different ways to go about this. The first is to use Horvitz-Thompson scaling. The next to construct a tree optimized with respect to effect heterogeneity.

2.4.1 Horvitz-Thompson scaling

Recall the HT scaling result above. We can construct our unbiased signal of $s_0(X)$ as

$$\hat{S}(X_i) = Y_i \frac{(D_i - p(X_i))}{p(X_i)(1 - p(X_i))}.$$

We assume complete random assignment, in which case we have, $p(X_i) = p$. For the Progresa experiment:

```
pUp <- mean(mex_data$treatment)
pUp
```

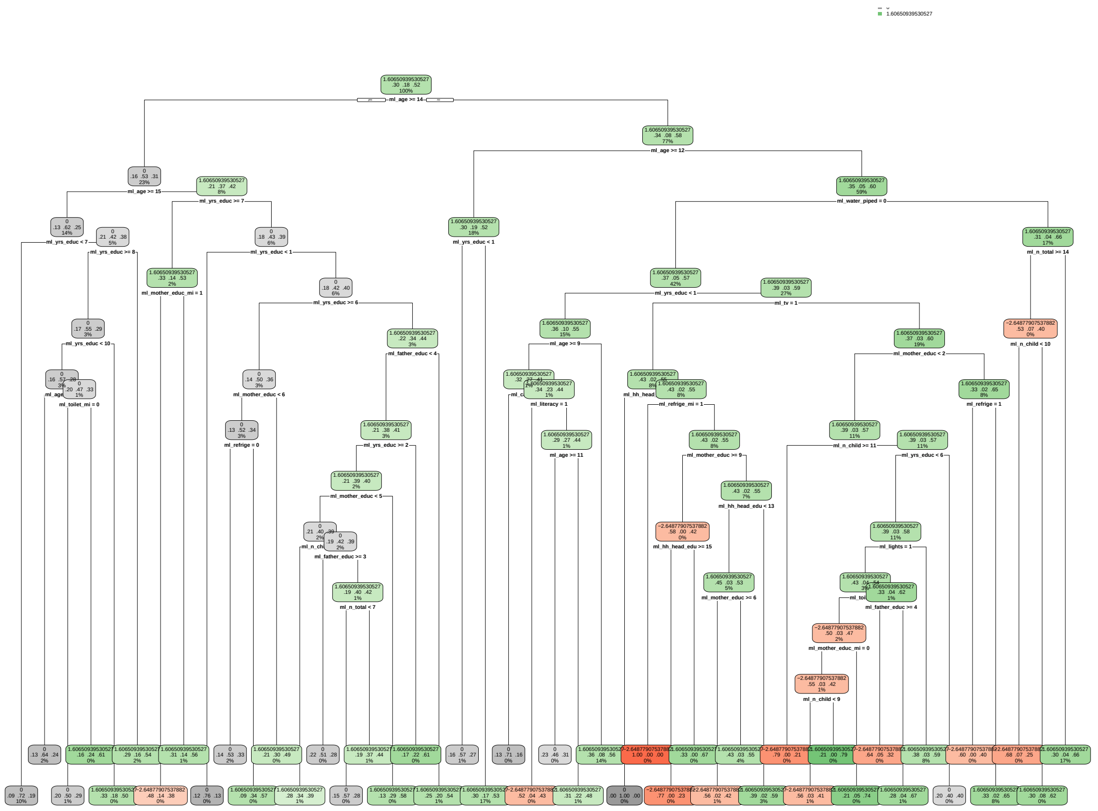
```
## [1] 0.6224676
```

```
mex_data$Senrolled <- with(mex_data, enrolled*((treatment-pUp)/(pUp*(1-pUp))))
```

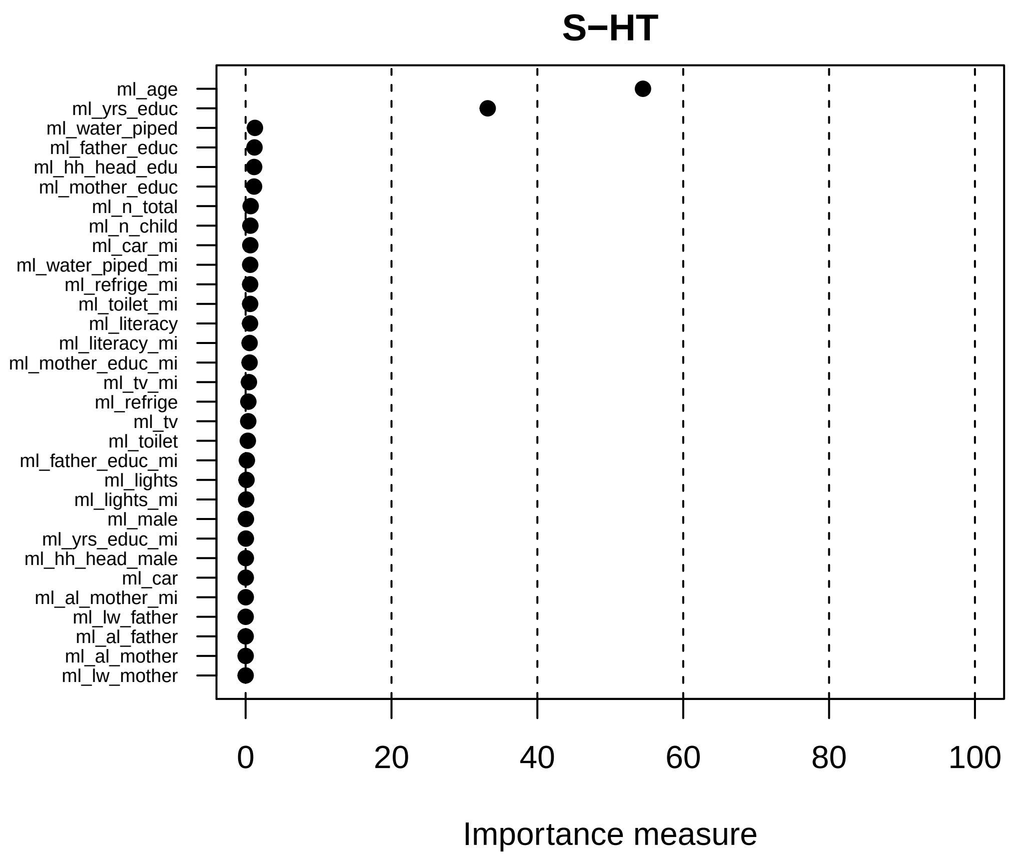
Then we fit the tree to this:

```
mexFormula_S <- as.formula(paste("Senrolled ~", paste(covs, collapse="+")))
tree_S <- rpartOp(formulaIn=mexFormula_S,
                  methodIn="class",
                  dataIn=mex_data,
                  xvalIn=10,
                  minbucketIn=2,
                  cp_incIN=0,
                  target="po")
rpart.plot(tree_S$tree)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```
par(mar=c(4,10,2,2))
rpart_impplot(treeIn=tree_S, mainIn="S-HT", labelScale=0.6)
```



Pretty much the same story as what we had seen before.

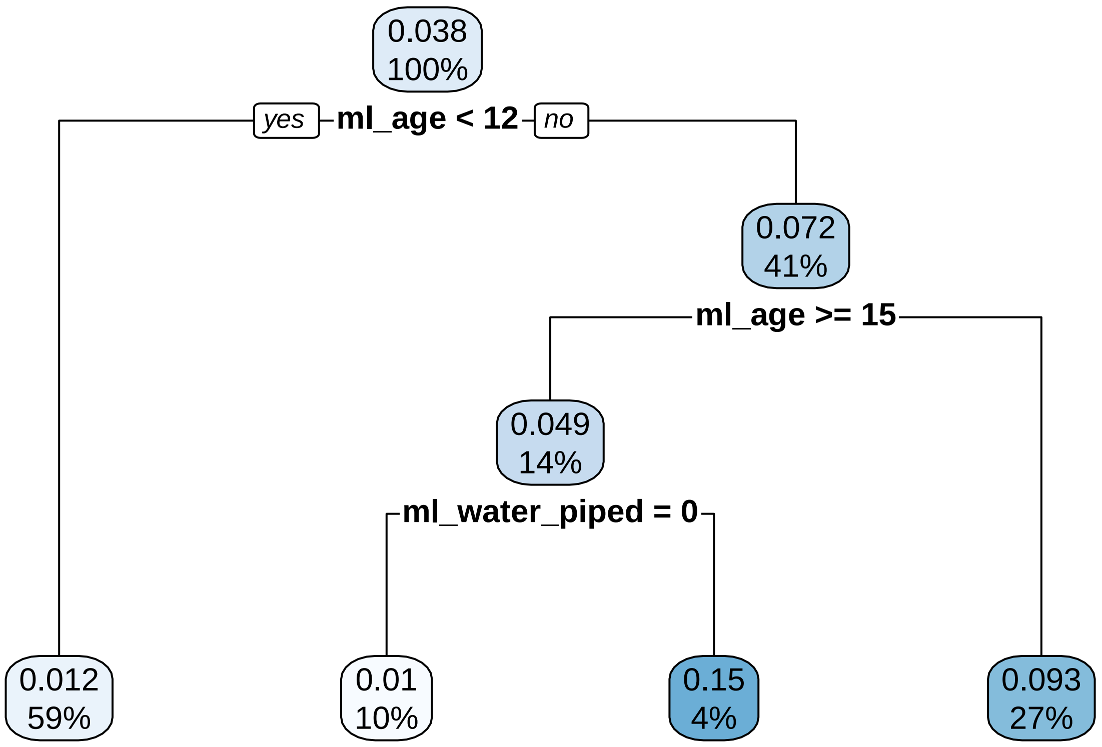
2.4.2 Causal Tree

We now turn to the "causal tree," as per Athey and Imbens. We use their recommended causal tree (CT) splitting rule. We do not use their honest splitting or CV rules because we are not interested in in-sample inference.

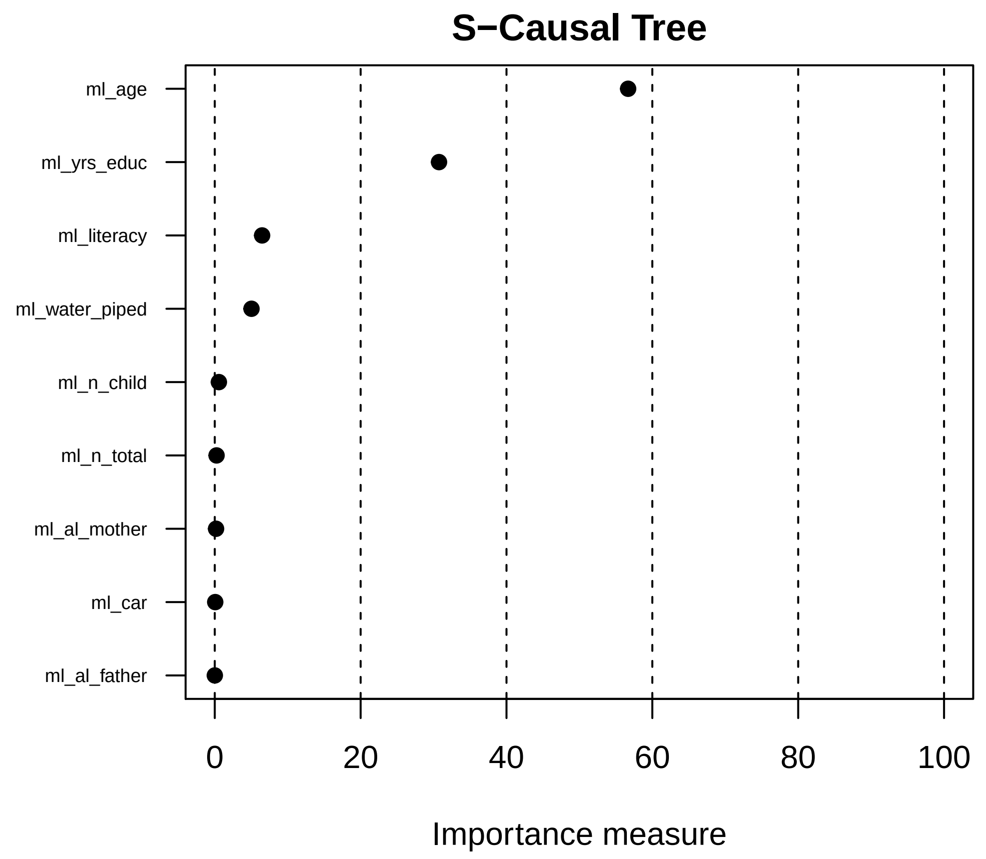
```
tree_fx_tree <- causalTree(mexFormula,
                           data=mex_data,
                           treatment=mex_data$treatment,
                           split.Rule="CT",
                           split.Honest=F,
                           cv.option="CT",
                           cv.Honest=F,
                           split.alpha=1,
                           control=rpart.control(xval=10,
                                                 minbucket=2,
                                                 cp=0.00005))
```

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

```
opcp_fx <- tree_fx_tree$cptable[,1][which.min(tree_fx_tree$cptable[,4])]
optreeOut_fx <- prune(tree_fx_tree, opcp_fx)
rpart.plot(optreeOut_fx)
```



```
tree_fx <- list(tree = optreeOut_fx,
                varimp = 100*(optreeOut_fx$variable.importance/sum(optreeOut_fx$variable.importance)))
par(mar=c(4,10,2,2))
rpart_impplot(treeIn=tree_fx, mainIn="S-Causal Tree", labelScale=0.6)
```



Similar story as above.

Note that the CV-optimal tree and the variable importance ranking don't quite match in that the latter puts some weight on literacy. This is because the variable importance measure looks at all trees where it is included in a split. But presumably what it offers is redundant relative to something else that is included in the optimal tree.