Characterizing Effect Heterogeneity 3

Cyrus Samii

2019-06-21

Contents

1 Random Forests

We can proceed in a similar manner as above, looking at models for potential outcomes and then for treatment effects per se. What we want to do is to define variable importance measures for the random forests.

We work with the "generalized random forest" algorithm of Athey et al.

We will see below that the forests do not prune trees as much as the regression tree method, presumably the guard against overfitting comes from the aggregation over trees, which are each fit on the basis of perturbed or sampled data.

1.1 Some background on forests

Hastie et al. *Elements...* present forests in conventional terms, as ensembles of trees.

Athey et al. (AoS) take a different view, presenting them in terms of adaptive kernel estimators.

1.2 Forests for potential outcomes

```
numTreeVec <- c(2,20, 200, 2000)
for(i in 1:length(numTreeVec) ){
numTrees <- numTreeVec[i]
  mu0_up <- regression_forest(X=mex_data0[,covs],
                         Y=mex_data0[,"enrolled"],
                         num.trees = numTrees,
                         honesty=FALSE,
                         tune.parameters = TRUE,
                         num.fit.trees = numTrees,
                         sample.fraction=.5,
                         min.node.size=2,
                         seed=111)
  if(i == 1){mu0      <- list(mu0_up)}
  if(i >  1){mu0[[i]] <- mu0_up}

  mu1_up <- regression_forest(X=mex_data1[,covs],
                         Y=mex_data1[,"enrolled"],
                         num.trees = numTrees,
                         honesty=FALSE,
                         tune.parameters = TRUE,
                         num.fit.trees = numTrees,
                         sample.fraction=.5,
                         min.node.size=2,
                         seed=111)
  if(i == 1){mu1      <- list(mu1_up)}
  if(i >  1){mu1[[i]] <- mu1_up}
}
```

1.2.1 Variable importance

The variable importance measure that they encode is one that is based on the number of times, across the trees in the forest, that a variable is used to split for different node depths up to some pre-specified depth. For each variable, the frequency at each node depth is multipled by a depth weight, these weighted frequencies are added up, and then the total is standardized with respect to all of the other variables' sums to give a variable importance measure that sums to 1 across all of the variables. Here we can see it in action:

```r
# Using the built in function:
cbind(covs,round(variable_importance(mu0[[1]], max.depth=10), 2))
```

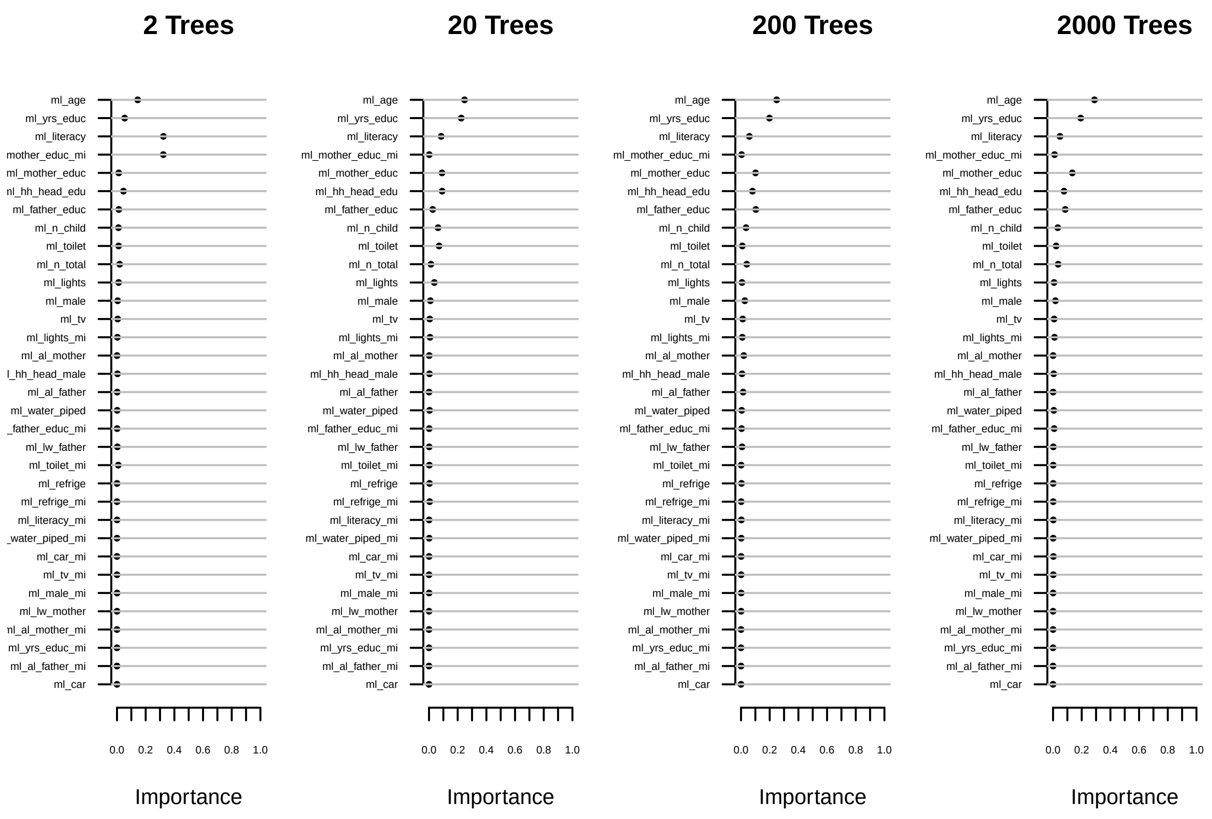
```
##       covs
##  [1,] "ml_age"             "0.14"
##  [2,] "ml_n_child"         "0.01"
##  [3,] "ml_male"            "0"
##  [4,] "ml_hh_head_male"    "0"
##  [5,] "ml_al_father"       "0"
##  [6,] "ml_al_mother"       "0"
##  [7,] "ml_lw_father"       "0"
##  [8,] "ml_lw_mother"       "0"
##  [9,] "ml_literacy"        "0.32"
## [10,] "ml_toilet"          "0.01"
## [11,] "ml_water_piped"     "0"
## [12,] "ml_lights"          "0.01"
## [13,] "ml_tv"              "0"
## [14,] "ml_car"             "0"
## [15,] "ml_refrige"         "0"
## [16,] "ml_n_total"         "0.02"
## [17,] "ml_yrs_educ"        "0.05"
## [18,] "ml_hh_head_edu"     "0.05"
## [19,] "ml_father_educ"     "0.01"
## [20,] "ml_mother_educ"     "0.01"
## [21,] "ml_male_mi"         "0"
## [22,] "ml_al_father_mi"    "0"
## [23,] "ml_al_mother_mi"    "0"
## [24,] "ml_literacy_mi"     "0"
## [25,] "ml_toilet_mi"       "0.01"
## [26,] "ml_water_piped_mi"  "0"
## [27,] "ml_lights_mi"       "0"
## [28,] "ml_tv_mi"           "0"
## [29,] "ml_car_mi"          "0"
## [30,] "ml_refrige_mi"      "0"
## [31,] "ml_yrs_educ_mi"     "0"
## [32,] "ml_father_educ_mi"  "0"
## [33,] "ml_mother_educ_mi"  "0.32"
```

```r
# Code that replicates:
split.freq <- split_frequencies(mu0[[1]], max.depth=10)
split.freq <- split.freq / pmax(1L, rowSums(split.freq))
weight <- seq_len(nrow(split.freq)) ^ -2
var.importance <- t(split.freq) %*% weight / sum(weight)
cbind(covs, round(var.importance, 2))
```

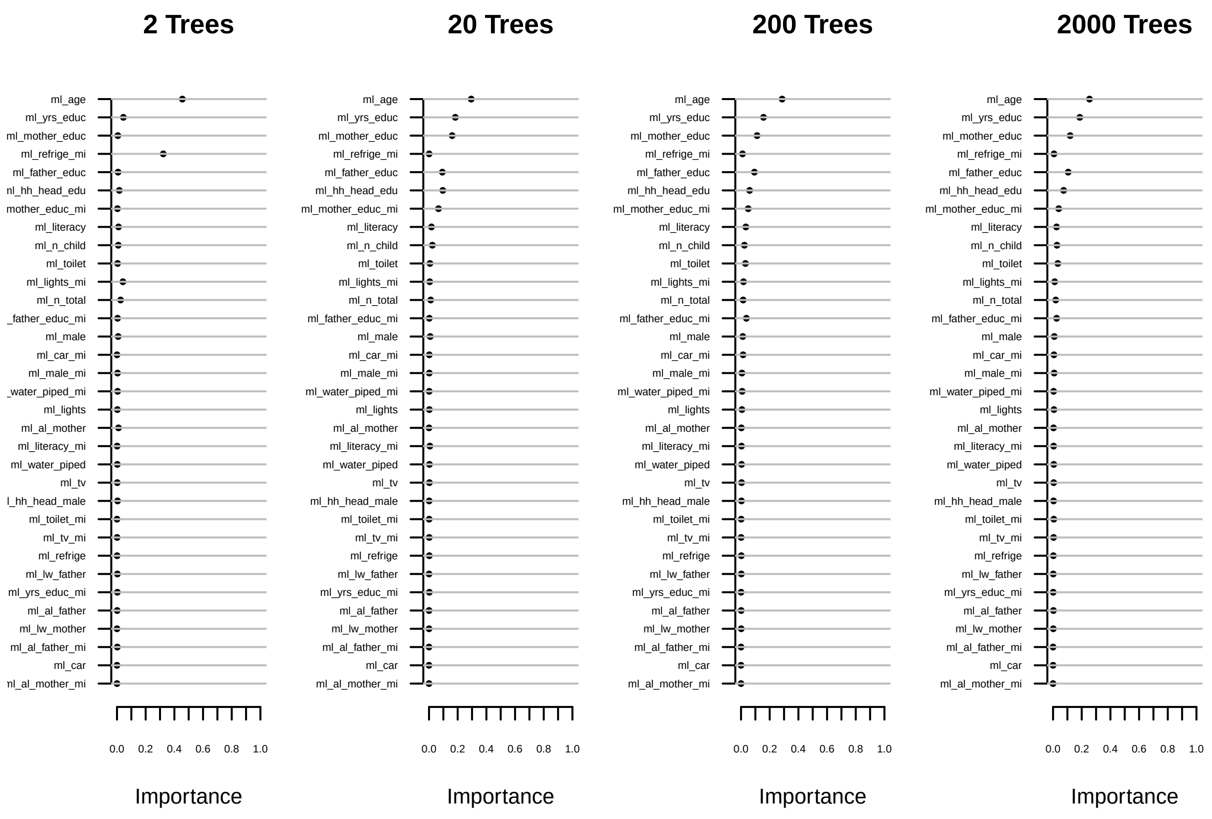
```
##       covs
##  [1,] "ml_age"             "0.14"
##  [2,] "ml_n_child"         "0.01"
##  [3,] "ml_male"            "0"
##  [4,] "ml_hh_head_male"    "0"
##  [5,] "ml_al_father"       "0"
##  [6,] "ml_al_mother"       "0"
##  [7,] "ml_lw_father"       "0"
##  [8,] "ml_lw_mother"       "0"
##  [9,] "ml_literacy"       "0.32"
## [10,] "ml_toilet"         "0.01"
## [11,] "ml_water_piped"    "0"
## [12,] "ml_lights"         "0.01"
## [13,] "ml_tv"             "0"
## [14,] "ml_car"            "0"
## [15,] "ml_refrige"        "0"
## [16,] "ml_n_total"        "0.02"
## [17,] "ml_yrs_educ"       "0.05"
## [18,] "ml_hh_head_edu"    "0.05"
## [19,] "ml_father_educ"    "0.01"
## [20,] "ml_mother_educ"    "0.01"
## [21,] "ml_male_mi"        "0"
## [22,] "ml_al_father_mi"   "0"
## [23,] "ml_al_mother_mi"   "0"
## [24,] "ml_literacy_mi"    "0"
## [25,] "ml_toilet_mi"      "0.01"
## [26,] "ml_water_piped_mi" "0"
## [27,] "ml_lights_mi"      "0"
## [28,] "ml_tv_mi"          "0"
## [29,] "ml_car_mi"         "0"
## [30,] "ml_refrige_mi"     "0"
## [31,] "ml_yrs_educ_mi"    "0"
## [32,] "ml_father_educ_mi" "0"
## [33,] "ml_mother_educ_mi" "0.32"
```

Now we can look at variable importance over the different sized forests:

```
varImpMat0.or <- grfimp_orderedMat(forest_list = mu0,
                  tree_sizeVec = numTreeVec,
                  covsIn = covs)
varImpMat1.or <- grfimp_orderedMat(forest_list = mu1,
                  tree_sizeVec = numTreeVec,
                  covsIn = covs)
grfimp_multiPlot(varImpMat0.or)
```



```r
grfimp_multiPlot(varImpMat1.or)
```



We note that this variable importance measure is basically a counting exercise, and is motivated by the fact that the trees that constitute the forests develop splits in a manner that maximizes discrimination at each step. That said, the scaling of the measure is not guaranteed to be proportional to the variance explained by each variable. Something like what we had for trees may be difficult to implement, however there are permutation/bootstrap based alternatives that one could use—cf. Su et al. (2009).

1.2.2 Iterated estimation and screening covariates

In an applied paper, Athey and Wager (2019) also propose iteration to first see in a generalized estimation problem the forest picks out as important, and then in a second iteration constraining the estimation to work only with covariates that pass some importance threshold after the first pass. We can do this here for the large forest (2000 trees).

```
covs.short0 <- colnames(varImpMat0.or)[varImpMat0.or[4,]>.01]
covs.short1 <- colnames(varImpMat1.or)[varImpMat1.or[4,]>.01]

mu0_short <- regression_forest(X=mex_data0[,covs.short0],
                         Y=mex_data0[,"enrolled"],
                         num.trees = 2000,
                         honesty=FALSE,
                         tune.parameters = TRUE,
                         num.fit.trees = 10,
                         sample.fraction=.5,
                         min.node.size=2,
                         seed=111)
mu1_short <- regression_forest(X=mex_data1[,covs.short1],
                         Y=mex_data1[,"enrolled"],
                         num.trees = 2000,
                         honesty=FALSE,
                         tune.parameters = TRUE,
                         num.fit.trees = 10,
                         sample.fraction=.5,
                         min.node.size=2,
                         seed=111)
cbind(covs.short0, variable_importance(mu0_short, max.depth=10))
```

```
##       covs.short0
##  [1,] "ml_male"           "0.0317208191373628"
##  [2,] "ml_n_total"        "0.0476515892723341"
##  [3,] "ml_toilet"         "0.0338818965571456"
##  [4,] "ml_n_child"        "0.0421327323894612"
##  [5,] "ml_father_educ"    "0.0893377815617847"
##  [6,] "ml_hh_head_edu"    "0.0858583128081488"
##  [7,] "ml_mother_educ"    "0.131091871458474"
##  [8,] "ml_mother_educ_mi" "0.0200114687345336"
##  [9,] "ml_literacy"       "0.0599184556346349"
## [10,] "ml_yrs_educ"       "0.197652168391464"
## [11,] "ml_age"            "0.260742904054656"
```

```
cbind(covs.short1, variable_importance(mu1_short, max.depth=10))
```

```
##       covs.short1
##  [1,] "ml_father_educ_mi" "0.0355528251541182"
##  [2,] "ml_n_total"        "0.029920787910997"
##  [3,] "ml_lights_mi"      "0.0243657106348473"
##  [4,] "ml_toilet"         "0.0444915516146963"
##  [5,] "ml_n_child"        "0.0422566851478603"
##  [6,] "ml_literacy"       "0.0444079307341879"
##  [7,] "ml_mother_educ_mi" "0.0497839547805342"
##  [8,] "ml_hh_head_edu"    "0.0876862309818392"
##  [9,] "ml_father_educ"    "0.0952326569258715"
## [10,] "ml_mother_educ"    "0.121208437228804"
## [11,] "ml_yrs_educ"       "0.180553733173711"
## [12,] "ml_age"            "0.244539495712532"
```

1.2.3 Summary conclusions

Not clear that doing the forests with small number of trees is really adding anything. Better to just stick with the large number (e.g., default 2000) and then contrast to the one-tree case. The larger is motivated by the idea of the adaptive kernel estimation, which surely is better with many trees than a few.

Below we will stick with that.

1.3 Forest for effect heterogeneity

Athey et al. (AoS) propose methods to estimate causal effects that first perform what Chernozhukov et al. refer to as "Neyman orthogonalization" — that is, partialing out variation in treatment and outcome due to covariates, and then running the forest on these residualized treatment and outcome values. This contributes to efficiency. This is implemented automatically in the `causal_forest` package. We will also use the pre-screening algorithm that was discussed above.

```r
# Start by just running on the defaults:
tauX_raw <- causal_forest(X = mex_data[,covs],
                      Y = mex_data[,"enrolled"],
                      W = mex_data[,"treatment"],
                      seed = 111)

# Now prune:
covs_short_tauX <- covs[variable_importance(tauX_raw) > .01]

# Now estimate again:
tauX <- causal_forest(X = mex_data[,covs_short_tauX],
                      Y = mex_data[,"enrolled"],
                      W = mex_data[,"treatment"],
                      seed = 111)
cbind(covs_short_tauX, variable_importance(tauX))
```

```
##       covs_short_tauX
##  [1,] "ml_age"            "0.464610261873223"
##  [2,] "ml_n_child"        "0.0366727390623697"
##  [3,] "ml_literacy"       "0.0163970357470381"
##  [4,] "ml_n_total"        "0.0501448037571906"
##  [5,] "ml_yrs_educ"       "0.297310047891212"
##  [6,] "ml_hh_head_edu"    "0.0349925004400063"
##  [7,] "ml_father_educ"    "0.03116558687577"
##  [8,] "ml_mother_educ"    "0.0389052087482516"
##  [9,] "ml_literacy_mi"    "0.0148154252313507"
## [10,] "ml_mother_educ_mi" "0.0149863903735885"
```

Similar story as what we've seen all along.