Tutorial 2 - Estimating Heterogeneous Treatment Effects in Panel Data with Machine Learning Techniques

Victor Hugo C. Alexandrino da Silva

8/18/2021

This is a tutorial that extends *Tutorial 1 - Estimating Heterogeneous Treatment Effects in Randomized Data with Machine Learning Techniques* with panel data. The idea here is to provide comparisons between three different data process using the `grf` package for Causal Forest algorithms.

There are three different data generating processes:

- **1. Linear Interaction:** A linear interaction between a variable of interest and the treatment dummy.
- **2. Non-linear interaction:** A non linear function of a variable of interest X and a treatment dummy Y.
- **3. Log non-linear interaction:** A non-linear function of the variable X in log terms and a treatment variable W.

1. Linear Interaction

Let's first create our panel data setting. The model has a treatment variable W_{it} and a set of covariates $V1_{it}$ for each individual firm i and time t. Then, our panel model can be written as

$$y_{it} = \alpha_i + V1_{it} + W_{it} + V1_{it} \times W_{it}$$

Where α_i are non-observed individual fixed effects, correlated with the covariates set $V1$.

As a linear case, let's first run a fixed effect model, together with a random effect model and an OLS for panel data. Let's begin with our packages:

```
library(lfe, quietly = TRUE) # Linear Group Fixed Effects
library(lme4)                # Linear Mixed Models
require(snowfall)            # Cluster programming
library(MASS)
library(grf)                # Generalized Random Forest
library(tidyverse)          # Data manipulation
library(plm)
```

Now, we create our dataset. As the standard of machine learning models and since the causal forest algorithm assumes honest trees besides the causality assumptions, we split our dataset in train and test sample.

```r
set.seed(123)
rm(list = ls())

# Number of periods
t <- 10

# Number of variables
p <- 10

# Number of firms
n <- 400

# Generating p random variables
data1 = as.data.frame(matrix(rnorm(n*t*p), n*t, p))

firm = seq(1,n)

time = seq(1,t)

data <- expand.grid(firm = firm, time = time)

# Defining the treatment assignment
data$W <- rbinom(n*t, 1, 0.5)

# Defining train sample
data$train <- rbinom(n*t, 1, 0.5)

# Generate two correlated variables (V1 and V2)
covarMat = matrix( c(1, .95^2, .95^2, 1 ) , nrow=2 , ncol=2 )
data.2  = as.data.frame(mvrnorm(n=n*t , mu=rep(0,2), Sigma=covarMat ))

data <- bind_cols(data,data.2) %>%
    tbl_df()

summary(data$firm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   100.8   200.5   200.5   300.2   400.0
```

```r
data1$V1 <- NULL

data1$V2 <- NULL

#colnames(data1) <- c("var1","var2","var3","var4","var5","var6","var7","var8","var9","var10")

data <- bind_cols(data,data1)

#data <- subset(data, )

# Generate unit (fixed) effect by firm means of V2, which is correlated with V1
data <- data %>%
    group_by(firm) %>%
    mutate(unit.effect=mean(V2)) %>%
```

```r
    ungroup() %>%
    arrange(firm, time)

summary(data$firm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   100.8   200.5   200.5   300.2   400.0
```

```r
# Generate outcome variable y and fixed unit effects  index
y<-c()
unit<-c()
X<-c()
k <- 0
for(i in 1:n){
    for(j in 1:t){
        k<-k+1
        unit[k]<-i
        y[k]<-1+ data$V1[k] + data$W[k] + data$V1[k]*data$W[k] + data$unit.effect[k]+rnorm(1,mean=0,sd=
    }
}
data$unit=unit
data$y=y

# Split between train and test set
data.train <- data %>%
    filter(train==1)
data.test <- data %>%
    filter(train==0)

summary(data.train)
```

```
##       firm            time              W              train
##  Min.   :  1.0   Min.   : 1.000   Min.   :0.0000   Min.   :1
##  1st Qu.:101.0   1st Qu.: 3.000   1st Qu.:0.0000   1st Qu.:1
##  Median :200.0   Median : 6.000   Median :1.0000   Median :1
##  Mean   :200.3   Mean   : 5.503   Mean   :0.5046   Mean   :1
##  3rd Qu.:300.0   3rd Qu.: 8.000   3rd Qu.:1.0000   3rd Qu.:1
##  Max.   :400.0   Max.   :10.000   Max.   :1.0000   Max.   :1
##        V1                 V2                 V3                  V4
##  Min.   :-3.06127   Min.   :-3.34341   Min.   :-3.402024   Min.   :-3.244513
##  1st Qu.:-0.65971   1st Qu.:-0.63989   1st Qu.:-0.662341   1st Qu.:-0.641011
##  Median : 0.02782   Median : 0.01906   Median : 0.004654   Median :-0.000930
##  Mean   : 0.02294   Mean   : 0.03830   Mean   : 0.006782   Mean   : 0.006294
##  3rd Qu.: 0.71334   3rd Qu.: 0.74074   3rd Qu.: 0.701523   3rd Qu.: 0.655941
##  Max.   : 3.37882   Max.   : 3.55732   Max.   : 2.876658   Max.   : 3.313622
##        V5                 V6                  V7                 V8
##  Min.   :-3.12520   Min.   :-3.141838   Min.   :-3.32884   Min.   :-3.49729
##  1st Qu.:-0.72262   1st Qu.:-0.665314   1st Qu.:-0.67889   1st Qu.:-0.66944
##  Median :-0.03054   Median : 0.014644   Median :-0.02461   Median :-0.02189
##  Mean   :-0.01996   Mean   : 0.006389   Mean   :-0.02505   Mean   :-0.00540
##  3rd Qu.: 0.63730   3rd Qu.: 0.664496   3rd Qu.: 0.65844   3rd Qu.: 0.67252
##  Max.   : 3.70335   Max.   : 4.322815   Max.   : 3.41620   Max.   : 3.56045
```

```
##        V9                  V10              unit.effect            unit
##  Min.   :-2.954006   Min.   :-4.129135   Min.   :-0.89121   Min.   :  1.0
##  1st Qu.:-0.659852   1st Qu.:-0.674597   1st Qu.:-0.19681   1st Qu.:101.0
##  Median : 0.044575   Median : 0.003211   Median : 0.03965   Median :200.0
##  Mean   : 0.005281   Mean   : 0.004901   Mean   : 0.02286   Mean   :200.3
##  3rd Qu.: 0.658174   3rd Qu.: 0.692329   3rd Qu.: 0.26679   3rd Qu.:300.0
##  Max.   : 3.982778   Max.   : 3.053021   Max.   : 0.86816   Max.   :400.0
##        y
##  Min.   :-5.1861
##  1st Qu.: 0.1205
##  Median : 1.3756
##  Mean   : 1.5447
##  3rd Qu.: 2.8364
##  Max.   : 8.8032
```

```
summary(data.train$firm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   101.0   200.0   200.3   300.0   400.0
```

```
summary(data.test$firm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0    99.0   201.0   200.8   301.0   400.0
```

```
summary(data$firm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   100.8   200.5   200.5   300.2   400.0
```

```
summary(data$time)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     3.0     5.5     5.5     8.0    10.0
```

```
summary(data.train$time)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.000   6.000   5.503   8.000  10.000
```

```
summary(data.test$time)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.000   5.000   5.497   8.000  10.000
```

```
#setdiff(data.train,data.test)
```

Now, let's run our FE, RE and LPM model to see whether they are biased. Remember that the individual fixed effect α_i (variable `unit_effect`) is highly correlated with the covariate $V1$, built from $V1$ means. Moreover, we estimate in our train data.

```r
## Fixed Effects
fe_felm <- felm(y ~ V1*W | unit, data = data.train)

# With plm package
fe_plm <- plm(y ~ V1*W, data = data.train, index = c("unit"), model = "within")
```

```
## Warning in pdata.frame(data, index): column 'time' overwritten by time index
```

```r
## Random Effects
re_lmer <- lmer(y ~V1*W+(1 | unit), data = data.train)

# With plm package
re_plm <- plm(y ~V1 * W, data = data.train, index = c("unit"), model = "random")
```

```
## Warning in pdata.frame(data, index): column 'time' overwritten by time index
```

```r
## POLS

ols <- lm(y ~ V1*W, data=data.train)


## Summary

summary(fe_felm)
```

```
##
## Call:
##    felm(formula = y ~ V1 * W | unit, data = data.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3899 -0.5980  0.0118  0.6262  3.3078
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## V1    1.07764    0.03580   30.10   <2e-16 ***
## W     1.06925    0.05015   21.32   <2e-16 ***
## V1:W  0.92938    0.04942   18.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.023 on 1648 degrees of freedom
## Multiple R-squared(full model): 0.806   Adjusted R-squared: 0.7587
## Multiple R-squared(proj model): 0.7399   Adjusted R-squared: 0.6765
## F-statistic(full model):17.04 on 402 and 1648 DF, p-value: < 2.2e-16
## F-statistic(proj model):  1563 on 3 and 1648 DF, p-value: < 2.2e-16
```

```r
summary(fe_plm)
```

```
## Oneway (individual) effect Within Model
##
```

```
## Call:
## plm(formula = y ~ V1 * W, data = data.train, model = "within",
##     index = c("unit"))
##
## Unbalanced Panel: n = 400, T = 1-10, N = 2051
##
## Residuals:
##      Min.   1st Qu.    Median   3rd Qu.      Max.
## -3.389867 -0.597952  0.011781  0.626191  3.307844
##
## Coefficients:
##      Estimate Std. Error t-value  Pr(>|t|)
## V1   1.077641   0.035803  30.100 < 2.2e-16 ***
## W    1.069252   0.050152  21.320 < 2.2e-16 ***
## V1:W 0.929379   0.049420  18.805 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    6625.5
## Residual Sum of Squares: 1723.2
## R-Squared:      0.73992
## Adj. R-Squared: 0.67648
## F-statistic: 1562.83 on 3 and 1648 DF, p-value: < 2.22e-16
```

```
summary(re_lmer)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: y ~ V1 * W + (1 | unit)
##    Data: data.train
##
## REML criterion at convergence: 6076.5
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.2241 -0.6604  0.0201  0.6367  3.5353
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  unit     (Intercept) 0.08959  0.2993
##  Residual             1.04837  1.0239
## Number of obs: 2051, groups:  unit, 400
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  0.97364    0.03615   26.93
## V1           1.12736    0.03338   33.77
## W            1.08475    0.04656   23.30
## V1:W         0.93315    0.04618   20.21
##
## Correlation of Fixed Effects:
##      (Intr) V1     W
## V1   -0.040
## W    -0.652  0.031
## V1:W  0.028 -0.720 -0.024
```

```r
summary(re_plm)
```

```
## Oneway (individual) effect Random Effect Model
##    (Swamy-Arora's transformation)
##
## Call:
## plm(formula = y ~ V1 * W, data = data.train, model = "random",
##     index = c("unit"))
##
## Unbalanced Panel: n = 400, T = 1-10, N = 2051
##
## Effects:
##                   var std.dev share
## idiosyncratic 1.04561 1.02255 0.936
## individual    0.07122 0.26687 0.064
## theta:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.03241 0.13632 0.15746 0.14825 0.17712 0.22875
##
## Residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.3856 -0.6925  0.0251 -0.0015  0.6793  3.6981
##
## Coefficients:
##             Estimate Std. Error z-value  Pr(>|z|)
## (Intercept) 0.972333   0.035630  27.290 < 2.2e-16 ***
## V1          1.130411   0.033452  33.791 < 2.2e-16 ***
## W           1.085984   0.046643  23.283 < 2.2e-16 ***
## V1:W        0.933178   0.046288  20.160 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    8310.6
## Residual Sum of Squares: 2173
## R-Squared:      0.73853
## Adj. R-Squared: 0.73815
## Chisq: 5763.71 on 3 DF, p-value: < 2.22e-16
```

```r
summary(ols)
```

```
##
## Call:
## lm(formula = y ~ V1 * W, data = data.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6131 -0.7093  0.0145  0.6997  3.7441
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.96468    0.03349   28.80   <2e-16 ***
## V1           1.14763    0.03382   33.93   <2e-16 ***
```

```
## W            1.09363    0.04712   23.21   <2e-16 ***
## V1:W         0.93326    0.04687   19.91   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.067 on 2047 degrees of freedom
## Multiple R-squared:  0.7379, Adjusted R-squared:  0.7375
## F-statistic:  1921 on 3 and 2047 DF,  p-value: < 2.2e-16
```

We observe that our treatment variable is statically significant in all estimators, but there is a bias both in RE and OLS models due to non-observable effects.

Now, let's run a Causal Forest with the `grf` package including all other variables in the data set. However, first we need to deal with the panel structure. The solution is to include all other variables in the data set but not $V2$ (since it was used to generate the fixed effect α_i) together with all the firm dummies using the train sample:

```r
# Creating the dataset of firm dummies
firm_dummies <- as.data.frame(model.matrix(y ~ as.factor(firm), data.train))

#summary(firm_dummies)

# Separating between covariates, outcome and tratment, merging with the firm dummies

# Features (obs: we exclude V2 since it is correlated with fixed effects by our data structure)
X <- as.matrix(bind_cols(data.train[,c(5,7:16)],firm_dummies))

# Outcome
Y <- data.train$y

# Treatment
W <- data.train$W

# Running our causal forest. Recall that, in order to account firm fixed effects, we have created firm
tau.forest <- causal_forest(X,Y,W)

# Estimate ATE for the full sample
average_treatment_effect(tau.forest, target.sample = "all")
```

```
##   estimate    std.err
## 1.07853477 0.07751322
```

```r
# Estimate ATE for treated sample
average_treatment_effect(tau.forest, target.sample = "treated")
```

```
##   estimate    std.err
## 1.07380387 0.07767093
```

```r
# Best Linear Projection for CATE
cate_best <- best_linear_projection(tau.forest)

cate_best
```

```
##
## 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) 1.078531   0.078383   13.76 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

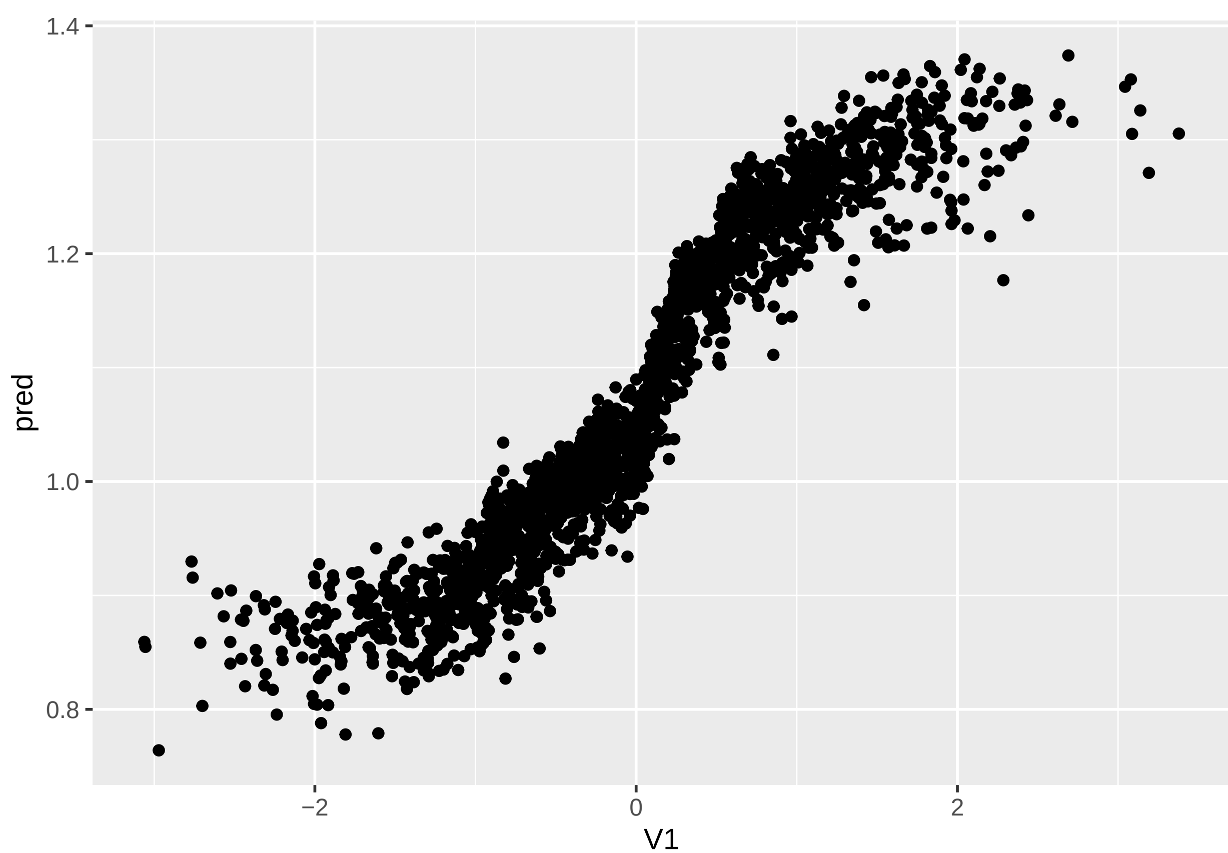
That is, our ATE estimated by the causal forest algorithm is quite similar with the ATE estimated by the linear model, especially when we account for firm fixed effects. Moreover, the best linear projection from the `grf` package also predicts similar values for the ATE.

We did the ATE prediction in the train sample. However, since the algorithm assumes honesty, we should do it at the test sample. For a full explanation, check Tutorial 1.

Therefore,

```
# Creating firm dummies for test sample
firm_dummies_test <- as.data.frame(model.matrix(y ~ as.factor(firm), data.test))

#setdiff(firm_dummies_test,firm_dummies)

#summary(firm_dummies_test)

# Creating each X, Y and W from test sample

# Features
X.test <- as.matrix(bind_cols(data.test[,c(5,7:16)],firm_dummies_test))

 # Outcome
Y.test <- data.test$y

# Treatment
W.test <- data.test$W

# Predicting CATE using the causal forest (trained by the train sample) and the test sample (for some r
tau.hat <- predict(tau.forest, estimate.variance = TRUE)

# Predicting variance
sigma.hat = sqrt(tau.hat$variance.estimates)

# Extracting CATE
data.train$pred <- tau.hat$predictions

# Histogram of CATE
hist(data.train$pred)
```

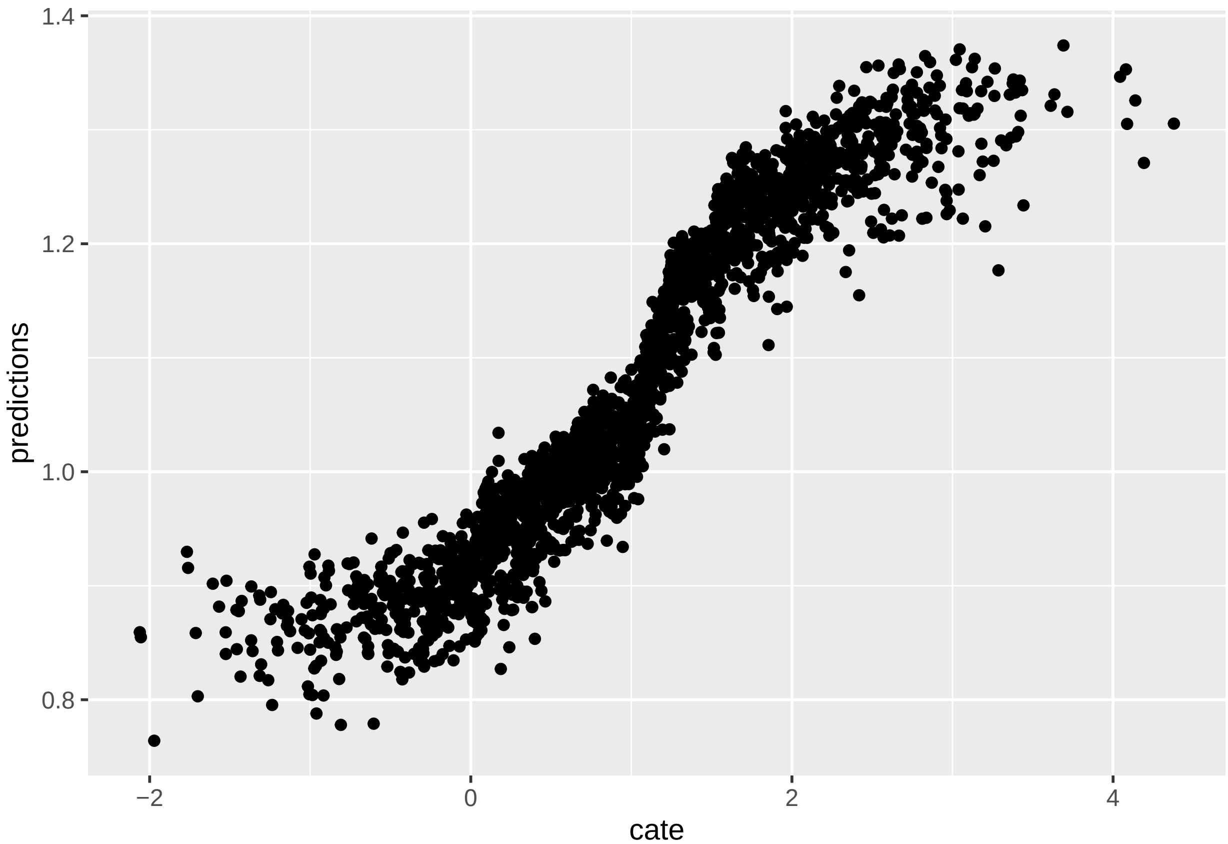

```
# Plotting CATE with respect to the covariate V1
ggplot(data.train, aes(x=V1, y=pred)) + geom_point()
```



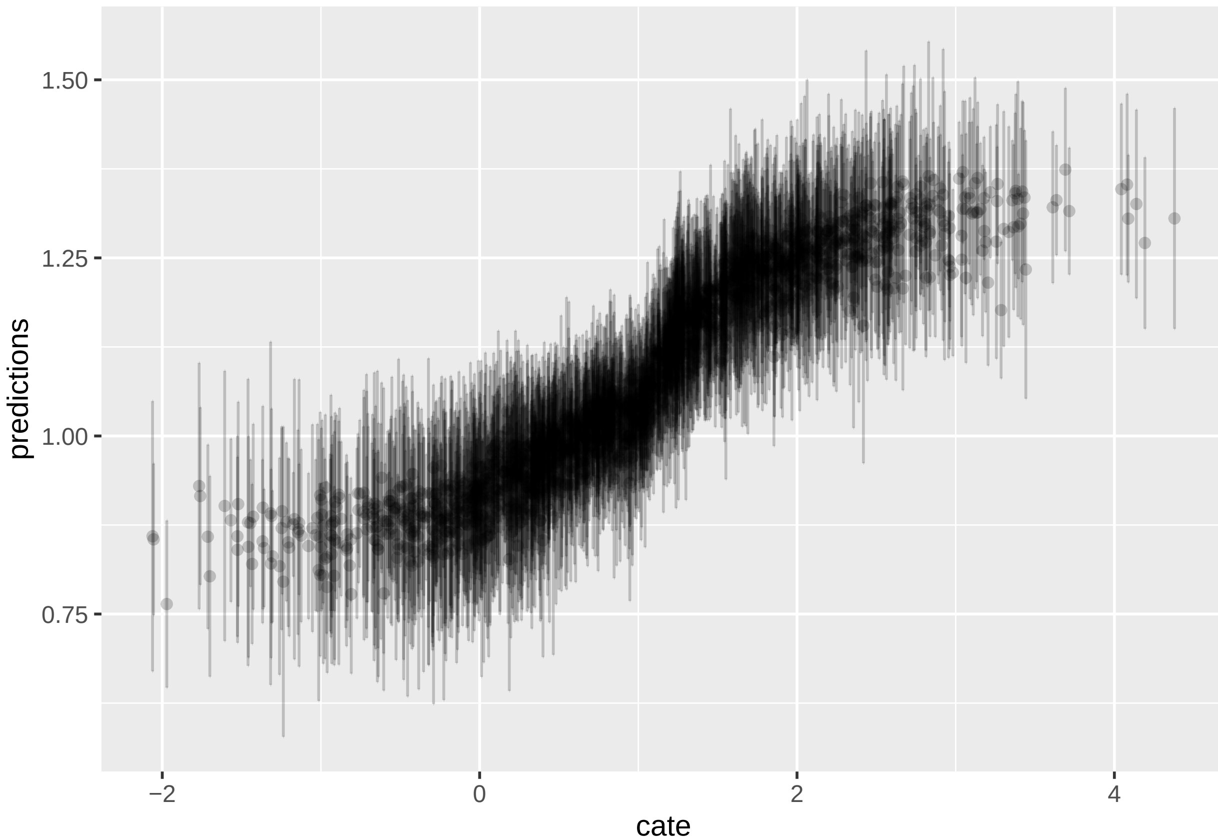
That is, this is the predicted value of CATE in our train data compared with the value of $V1$, our covariate that is correlated with the firm fixed effects (which was created from $V2$). It seems that there is a positive linear relationship between the feature $V1$ and the predicted CATE. This was true when we predicted our linear models.

Now, let's graph the predicted $CATE$ against the true one. We expect them to be similar:

```
# Create dataframe with predicted CATE info
graph.data <- as.data.frame(tau.hat)

# Create variance observations for CATE for each observation and confidence interval bounders
graph.data <- graph.data %>%
    mutate(cate = 1 + data.train$V1,0) %>%
    mutate(lower = predictions - sigma.hat, upper = predictions + sigma.hat)

# Plot CATE prediction x CATE true
ggplot(graph.data , aes(x=cate, y=predictions)) + geom_point()
```




```
# And finally, plotting error bar from CI:
ggplot() + geom_errorbar(graph.data , mapping=aes(x=cate,  ymin=lower,ymax=upper), alpha = 0.2) + geom_p
```



In general, our both CATE and ATE estimates are similar in the linear models and the causal forest algorithm. Let's look at the power of the causal forest algorithm when comparing with the non-linear estimation.

2. Non-linear interaction

Assume now that our ATE/CATE assuming that we now have a non-linear model:

$$y_{it} = 1 + \alpha_i + \max(V1_{it}, 0) * W_{it} + \varepsilon_{it}$$

Let's re-create our dataset with the non-linear relationship between V_1 and W:

```r
set.seed(321)
rm(list = ls())

# Number of periods
t <- 10

# Number of variables
p <- 10

# Number of firms
n = 400

# Generate the random variables
data1 = as.data.frame(matrix(rnorm(n*t*p), n*t, p))
```

```r
firm = seq(1,n)

time = seq(1,t)

data <- expand.grid(firm = firm, time = time)

# Treatment variable
data$W <- rbinom(n*t,1,0.5)

# Train dummy
data$train <- rbinom(n*t,1,0.5)

# Generate two correlated variables V1 and V2
covarMat = matrix( c(1, .95^2, .95^2, 1 ) , nrow=2 , ncol=2 )
data.2  = as.data.frame(mvrnorm(n=n*t , mu=rep(0,2), Sigma=covarMat ))

data <- bind_cols(data,data.2) %>%
    tbl_df()

data1$V1 = NULL

data1$V2 = NULL

data <- bind_cols(data,data1)

# Generate fixed effects by firm means of V2, correlated with V1:
data <- data %>%
    group_by(firm) %>%
    mutate(unit.effect=mean(V2)) %>%
    ungroup() %>%
    arrange(firm, time)

# Create non-linear outcome and fixed effects index
y<-c()
unit<-c()
X<-c()
k <- 0
for(i in 1:n){
    for(j in 1:t){
        k<-k+1
        unit[k]<-i
        y[k]<-1+ pmax(data$V1[k],0)*data$W[k]  + data$unit.effect[i]+rnorm(1,mean=0,sd=1)
    }
}
data$y=y
data$unit <- unit

# Split between train and test sample
data.train <- data %>%
    filter(train==1)

data.test <- data %>%
    filter(train==0)
```

Now, let's run our linear models:

```r
# Fixed effects
fe <- felm(y ~V1*W | unit, data = data.train)

# Random effects
re <- lmer(y ~ V1*W + (1 | unit), data = data.train)

# POLS
ols <- lm(y ~ V1 * W, data = data.train)

# Summary for all linear models
summary(fe)
```

```
##
## Call:
##    felm(formula = y ~ V1 * W | unit, data = data.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2806 -0.6299 -0.0094  0.6136  3.5679
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## V1    0.02574    0.03719   0.692    0.489
## W     0.35896    0.05081   7.064 2.43e-12 ***
## V1:W  0.49366    0.05262   9.383  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.021 on 1565 degrees of freedom
## Multiple R-squared(full model): 0.3475   Adjusted R-squared: 0.1799
## Multiple R-squared(proj model): 0.1355   Adjusted R-squared: -0.08653
## F-statistic(full model):2.073 on 402 and 1565 DF, p-value: < 2.2e-16
## F-statistic(proj model): 81.78 on 3 and 1565 DF, p-value: < 2.2e-16
```

```r
summary(re)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: y ~ V1 * W + (1 | unit)
##    Data: data.train
##
## REML criterion at convergence: 5779.5
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.7636 -0.6828 -0.0111  0.6607  3.6284
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  unit     (Intercept) 0.06835  0.2614
##  Residual             1.03513  1.0174
## Number of obs: 1968, groups:  unit, 400
```

```
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  0.97539    0.03583  27.221
## V1           0.02719    0.03381   0.804
## W            0.38373    0.04692   8.179
## V1:W         0.49548    0.04809  10.303
##
## Correlation of Fixed Effects:
##      (Intr) V1     W
## V1    0.001
## W    -0.667 -0.004
## V1:W -0.002 -0.702 -0.009
```

```
summary(ols)
```

```
##
## Call:
## lm(formula = y ~ V1 * W, data = data.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0428 -0.7427 -0.0052  0.7216  3.7375
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.96735    0.03376  28.658  < 2e-16 ***
## V1           0.02789    0.03400   0.820    0.412
## W            0.39154    0.04738   8.263 2.58e-16 ***
## V1:W         0.49494    0.04848  10.209  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.051 on 1964 degrees of freedom
## Multiple R-squared:  0.1333, Adjusted R-squared:  0.132
## F-statistic: 100.7 on 3 and 1964 DF,  p-value: < 2.2e-16
```

Note that in our non-biased estimation, the covariate $V1$ is no longer statistically significant. But we have heterogeneity from the interaction and the treatment W seems to be relevant, but with small goodness of fit variables.

One solution for these problems in non-linear relations between treatment, covariate and outcome is the causal forest algorithm. Let's predict first the CATE against $V1$, as we did in the linear case.

```
# Setting firm dummies for the train sample
firm_dummies <- as.data.frame(model.matrix(y~as.factor(firm),data.train))

# Splitting our train sample
X <- as.matrix(bind_cols(data.train[,c(5,7:16)],firm_dummies))
Y <- data.train$y
W <- data.train$W

# Train the causal forest
```

```r
tau.forest2 <- causal_forest(X,Y,W)

# Estimating ATE from the full train sample
average_treatment_effect(tau.forest2, target.sample = "all")
```

```
##   estimate    std.err
## 0.31463897 0.04169315
```

```r
# Estimating ATE from the treated train sample
average_treatment_effect(tau.forest2, target.sample = "treated")
```

```
##   estimate    std.err
## 0.33031705 0.04233338
```

```r
# Estimating CATE with Best Linear Projection
cate_best_nl <- best_linear_projection(tau.forest2)

cate_best_nl
```

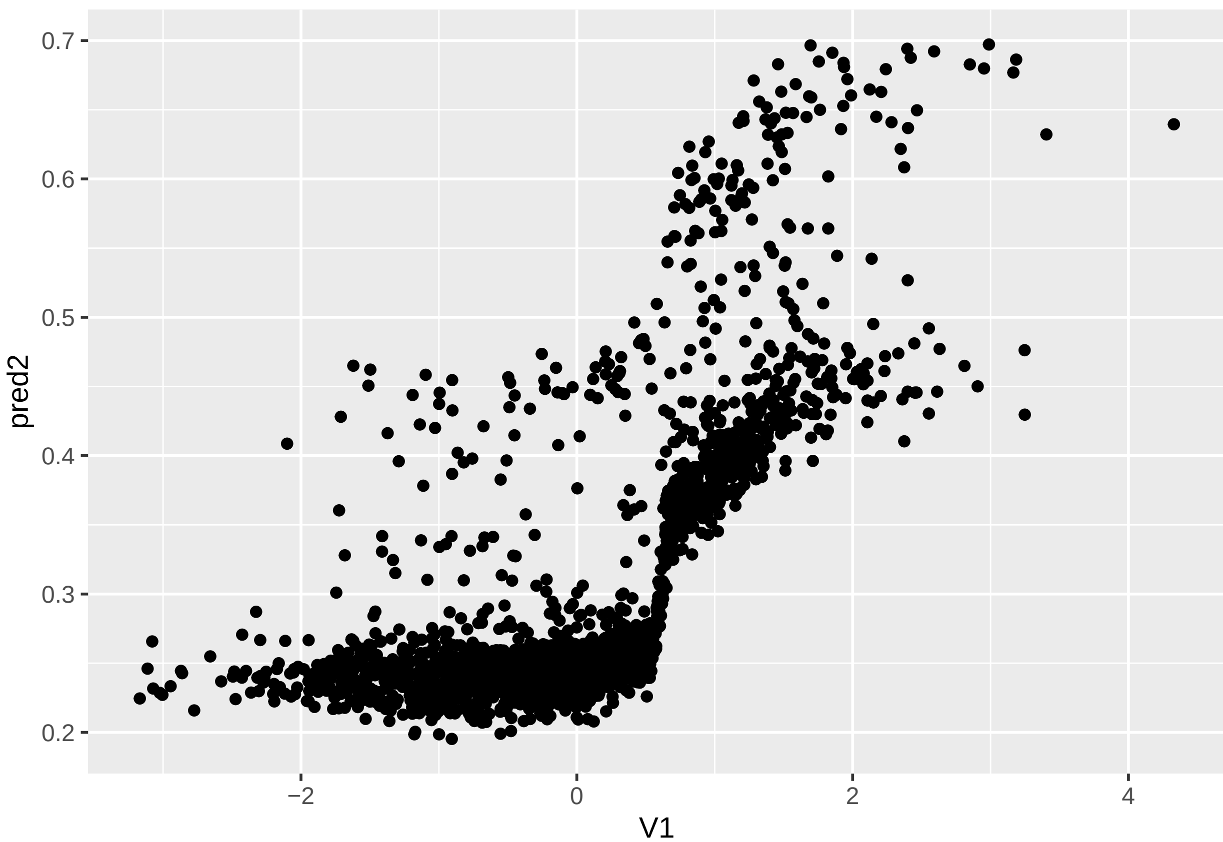
```
##
## Best linear projection of the conditional average treatment effect.
## Confidence intervals are cluster- and heteroskedasticity-robust (HC3):
##
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 0.314563   0.042201  7.4539 1.351e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now, let's create our test sample in order to predict the CATE:

```r
# Create firm dummies in the test sample
firm_dummies_test <- as.data.frame(model.matrix(y~as.factor(firm),data.test))

X.test <- as.matrix(bind_cols(data.test[,c(5,7:16)],firm_dummies_test))
Y.test <- data.test$y
W.test <- data.test$W

# Predicting CATE with the test sample
tau.hat2 = predict(tau.forest2, X.test, estimate.variance = TRUE)

# Defining standard errors from CATE
sigma.hat2 <- sqrt(tau.hat2$variance.estimates)

# Creating the column "pred2" for the CATE predictions in the test sample
data.test$pred2 <- tau.hat2$predictions
```

The next step is to plot our CATE. Let's first see the relationship between the covariate $V1$ with the predicted CATE:

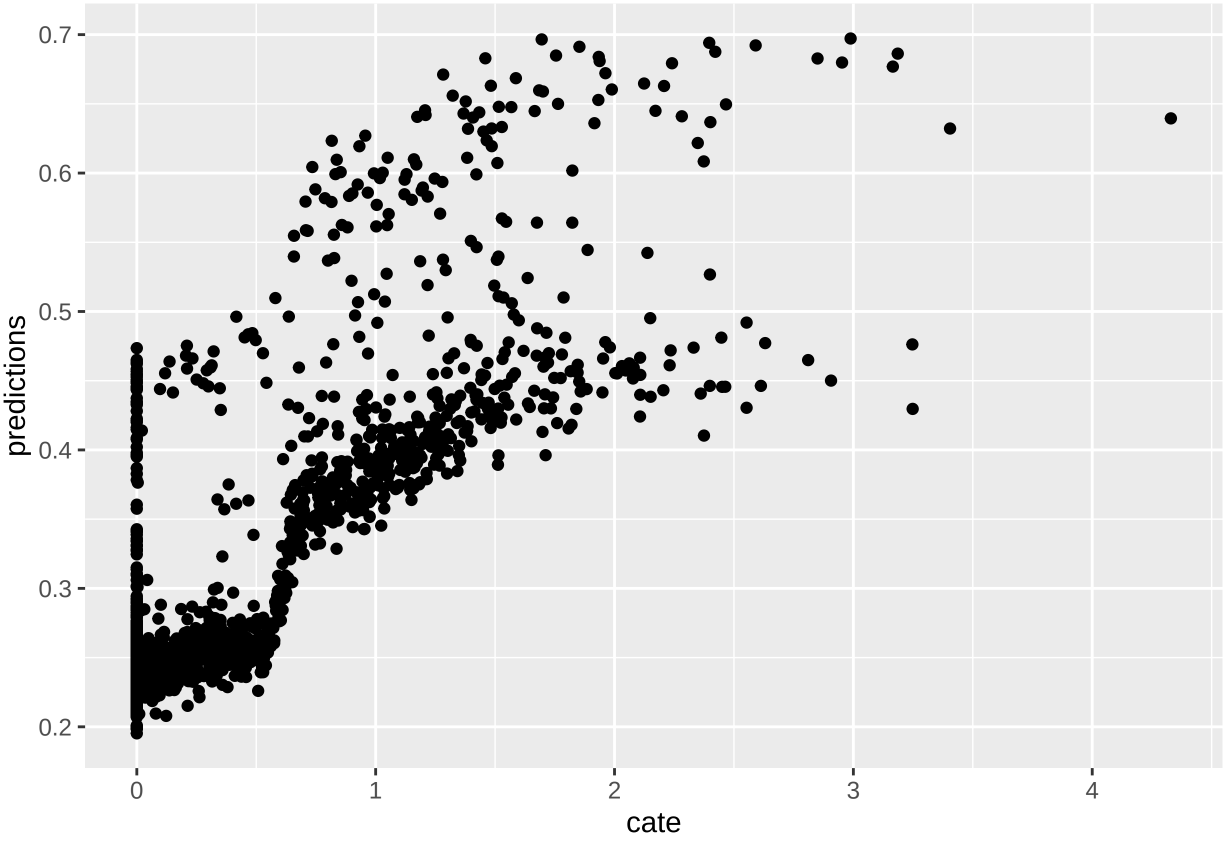
```r
# Plot CATE vs V1
ggplot(data.test,aes(x = V1, y = pred2)) +
    geom_point()
```



Note that in fact there is a non-linear relationship between $V1$ and the predicted CATE in our covariates.

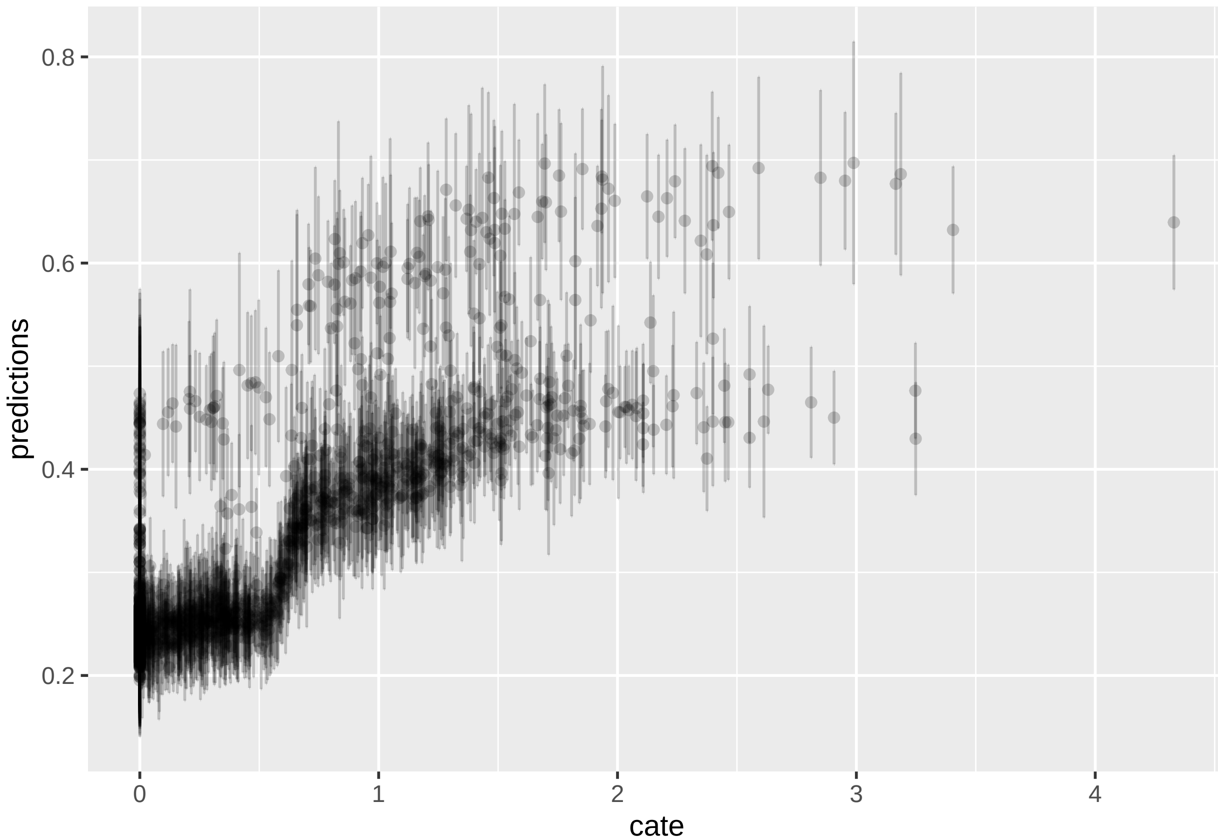
Now, let's plot the predicted CATE with the real CATE:

```r
# Storing out CATE info in a dataframe
graph.data2 <- as.data.frame(tau.hat2)

# Create cate and CI bounders
graph.data2 <- graph.data2 %>%
    mutate(cate=pmax(data.test$V1,0)) %>%
    mutate(lower=predictions - sigma.hat2, upper=predictions + sigma.hat2)

# Plot the relationship between the predicted CATE and the real CATE
ggplot(graph.data2, aes(x=cate, y=predictions)) + geom_point()
```



And finally, our error plot:

```
# Plot error plot with standard errors
ggplot() + geom_errorbar(graph.data2, mapping=aes(x=cate,  ymin=lower,ymax=upper), alpha = 0.2) + geom_p
```



It seems that now we have a better estimation of the ATE/CATE when there are non-linearities between our treatment and covariates. Moreover, the prediction performs really well, with close estimations of the coefficients.

3. Log non-linear interaction

Finally, we can see that the causal forest algorithm also performs really well when the non-linearity comes from the log relation. This is useful, for example, when we account for GDP growth in macro panels, or level-log models that we know that there is no linear relationship between the feature and treatment.

Assume now that our model is such that

$$y_{it} = 1 + \alpha_i + \log(V1_{it}^2) * W_{it} + \varepsilon_{it}$$

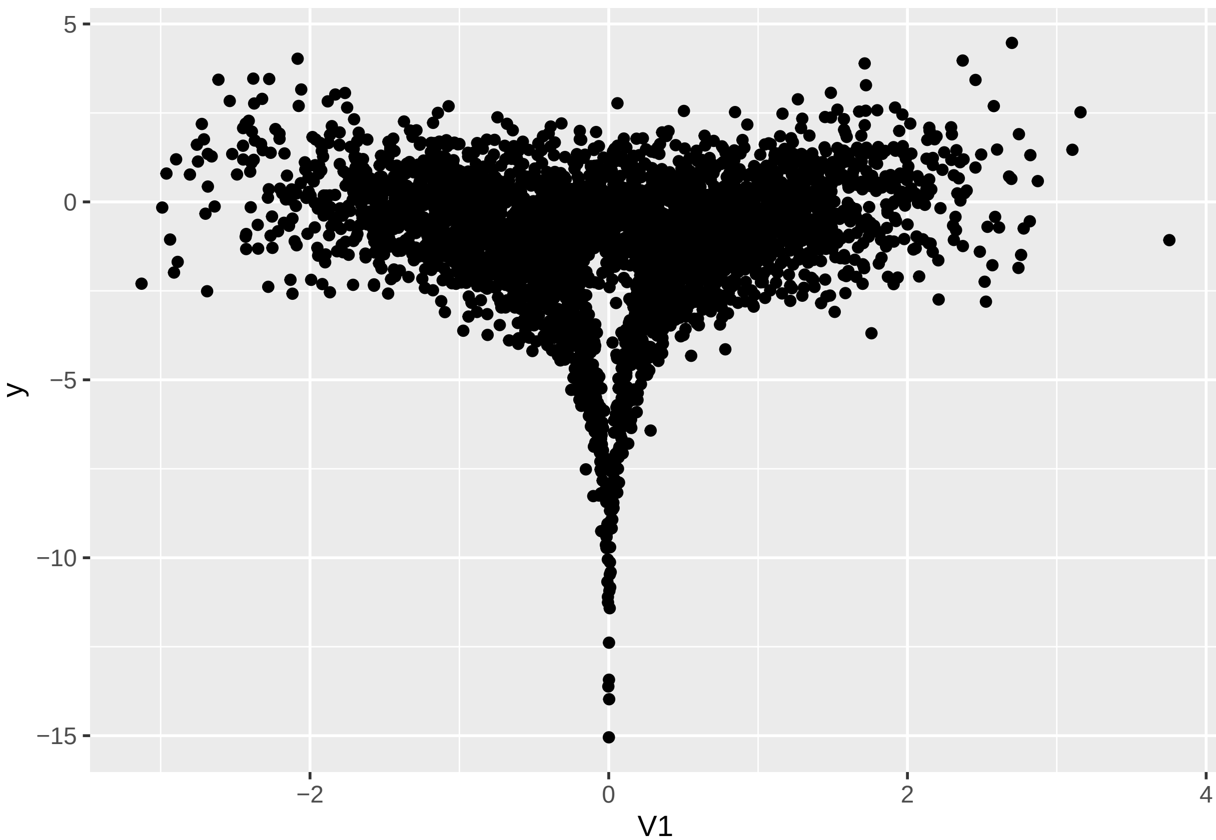
But we observe only $V1_{it}$, not $\log(V1_{it}^2)$.

Let's again create our panel with $V1$ being correlated with $V2$ and the fixed effect `unit`:

```
set.seed(666)

rm(list=ls())

# Number of periods
t <- 10

# Number of covariates
p <- 10
```

```r
# Number of firms
n = 400

# Generate random covariates
data1 = as.data.frame(matrix(rnorm(n*t*p), n*t, p))

firm = seq(1,n)

time = seq(1,t)

data <- expand.grid(firm=firm,time=time)

# Create treatment variable
data$W <- rbinom(n*t,1,0.5)

# Create train sample indicator
data$train <- rbinom(n*t,1,0.5)

# Generate V1 and V2, two correlated covariates
covarMat = matrix( c(1, .95^2, .95^2, 1 ) , nrow=2 , ncol=2 )
data.2  = as.data.frame(mvrnorm(n=n*t , mu=rep(0,2), Sigma=covarMat ))

data <- bind_cols(data,data.2) %>%
    tbl_df()

data1$V1 = NULL
data1$V2 = NULL

data <- bind_cols(data,data1)

# Generate treatment effects by firm means of V2, correlated with V1
data <- data %>%
    group_by(firm) %>%
    mutate(unit.effect=mean(V2)) %>%
    ungroup() %>%
    arrange(firm, time)

# Create non-linear outcome
y<-c()
unit<-c()
X<-c()
k <- 0

for(i in 1:n){
    for(j in 1:t){
        k<-k+1
        unit[k]<-i
        y[k]<-1+ log(data$V1[k]^2)*data$W[k] + pmin(data$V3, 0) + data$unit.effect[i]+rnorm(1,mean=0,sd
    }
}
data$y=y
data$unit <- unit
```

```r
# Split between test and train data
data.train <- data %>%
    filter(train==1)
data.test <- data %>%
    filter(train==0)

ggplot(data, aes(x = V1, y = y)) +
    geom_point()
```



Thus, we have indeed a non-linear relationship between $V1$ and the outcome y. Let's run the FE, RE and the OLS regressions using the train sample:

```r
# FE
fe <- felm(y ~ V1*W | unit, data=data.train)

# RE
re <- lmer(y ~ V1*W + (1 | unit), data=data.train)

# POLS
ols <- lm(y ~ V1*W, data = data.train)

# Summary

summary(fe)
```

```
##
```

```
## Call:
##    felm(formula = y ~ V1 * W | unit, data = data.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4938  -0.8785   0.0891   0.9992   5.2491
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## V1   -0.04144    0.06812  -0.608    0.543
## W    -1.23652    0.09257 -13.358   <2e-16 ***
## V1:W  0.01293    0.09518   0.136    0.892
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.85 on 1612 degrees of freedom
## Multiple R-squared(full model): 0.2947   Adjusted R-squared: 0.1193
## Multiple R-squared(proj model): 0.09981   Adjusted R-squared: -0.1241
## F-statistic(full model): 1.68 on 401 and 1612 DF, p-value: 2.282e-12
## F-statistic(proj model): 59.58 on 3 and 1612 DF, p-value: < 2.2e-16
```

```
summary(re)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: y ~ V1 * W + (1 | unit)
##    Data: data.train
##
## REML criterion at convergence: 8240.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -6.2590 -0.4652  0.0663  0.5790  2.6978
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  unit     (Intercept) 0.05461  0.2337
##  Residual             3.43184  1.8525
## Number of obs: 2014, groups:  unit, 399
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept) -0.45725    0.05967  -7.664
## V1          -0.02699    0.06208  -0.435
## W           -1.26979    0.08324 -15.254
## V1:W        -0.03438    0.08552  -0.402
##
## Correlation of Fixed Effects:
##      (Intr) V1     W
## V1    0.015
## W    -0.692 -0.010
## V1:W -0.010 -0.727  0.028
```

```
summary(ols)
```

```
##
## Call:
## lm(formula = y ~ V1 * W, data = data.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.7006  -0.8590   0.1257   1.0919   5.0430
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.45475    0.05864  -7.754  1.4e-14 ***
## V1          -0.02610    0.06214  -0.420    0.675
## W           -1.27236    0.08326 -15.282  < 2e-16 ***
## V1:W        -0.03617    0.08552  -0.423    0.672
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.867 on 2010 degrees of freedom
## Multiple R-squared:  0.1044, Adjusted R-squared:  0.103
## F-statistic: 78.08 on 3 and 2010 DF,  p-value: < 2.2e-16
```

Note that now we only have statistically significance on the treatment, but none in the heterogeneity, even knowing that our true model has a non-linear relationship between $V1$ and W.

Let's see if the causal forest algorithm work well in this case:

```
# Firm dummies
firm_dummies <- as.data.frame(model.matrix(y~as.factor(firm),data.train))

# Creating our train sample for X, Y and W

# Covariates
X <- as.matrix(bind_cols(data.train[,c(5,7:16)],firm_dummies))

str(X)
```

```
##  num [1:2014, 1:410] 0.445 -0.145 -0.88 1.866 0.805 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:2014] "1" "2" "3" "4" ...
##   ..$ : chr [1:410] "V1" "V3" "V4" "V5" ...
```

```
# Outcome
Y <- data.train$y

# Treatment
W <- data.train$W

# Train our causal forest with train test
tau.forest3 <- causal_forest(X,Y,W)

# Estimate ATE with all train sample
average_treatment_effect(tau.forest3, target.sample = "all")
```

```
##    estimate     std.err
## -0.87532728  0.06408444
```

```
# Estimate ATE with trated train sample
average_treatment_effect(tau.forest3, target.sample = "treated")
```

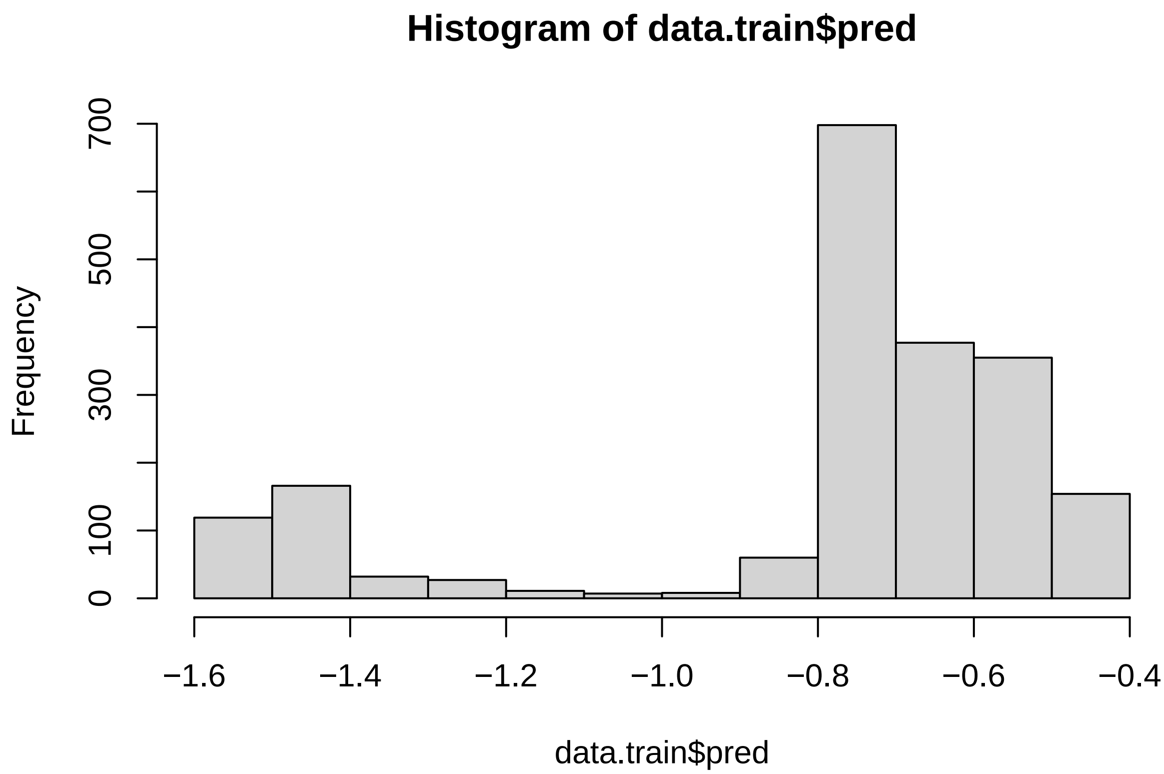
```
##    estimate     std.err
## -1.08992711  0.06829477
```

```
# Predicting CATE with Best Linear Prediction
cate_best_nl2 <- best_linear_projection(tau.forest3)

cate_best_nl2
```

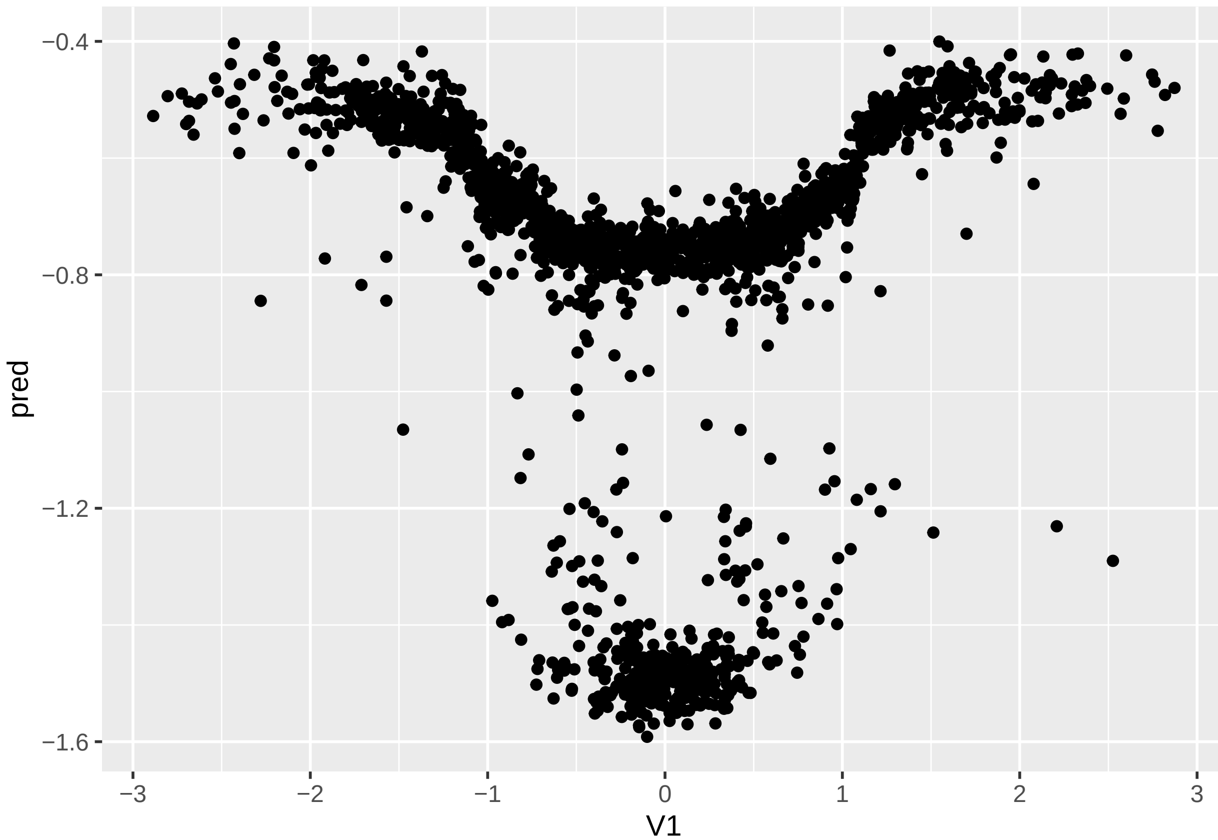
```
##
## Best linear projection of the conditional average treatment effect.
## Confidence intervals are cluster- and heteroskedasticity-robust (HC3):
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -0.872477   0.066125 -13.194 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now let's predict our CATE with the test sample:

```
# Create firm dummies for test sample
firm_dummies_test <- as.data.frame(model.matrix(y ~as.factor(firm), data.test))

# Create test set for X, Y and W, excluding V2
# Features
X.test <- as.matrix(bind_cols(data.test[,c(5,7:16)],firm_dummies_test))

 # Outcome
Y.test <- data.test$y

# Treatment
W.test <- data.test$W

# Predicting CATE (Again, form some reason we are having problems with columns. But to have full honesty
tau.hat3 <- predict(tau.forest3, estimate.variance = TRUE)

# Predicting variance
sigma.hat3 = sqrt(tau.hat3$variance.estimates)

# Extracting CATE
data.train$pred <- tau.hat3$predictions

# Histogram of CATE
hist(data.train$pred)
```

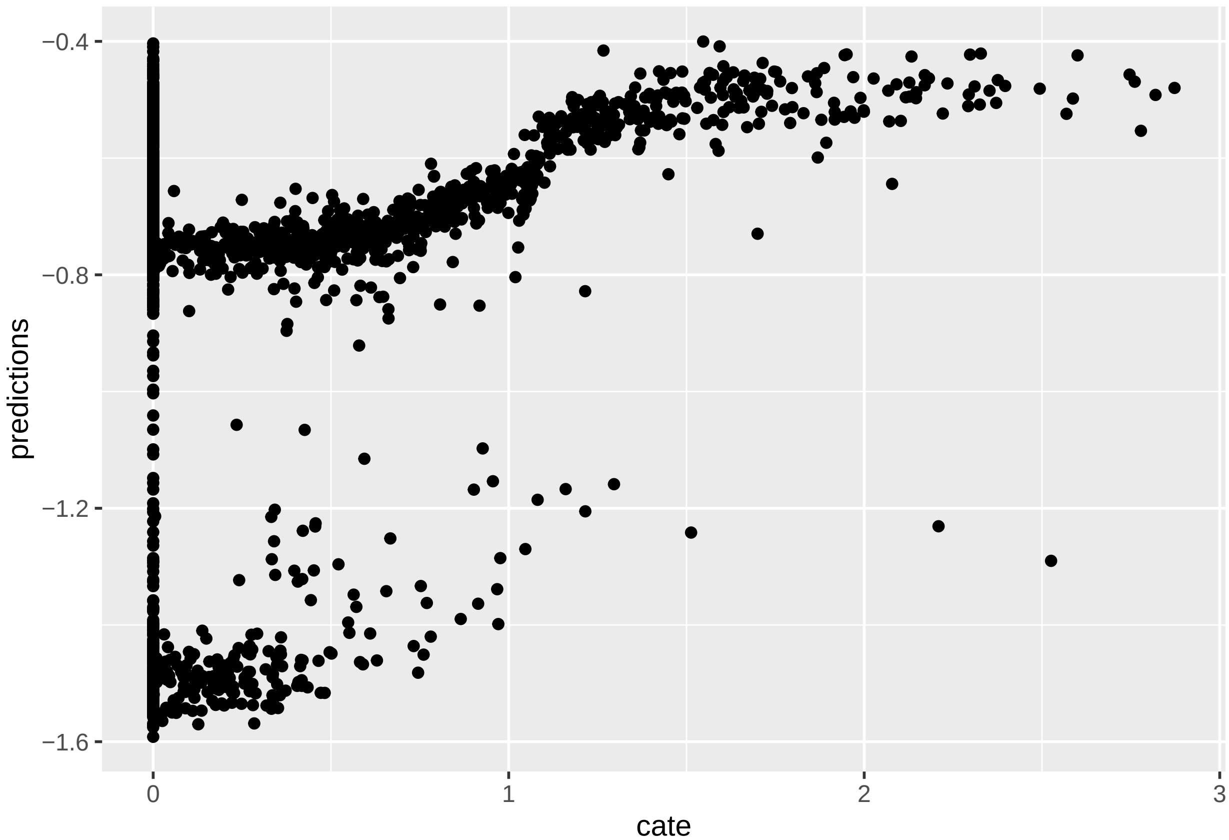



```r
# Plotting CATE vs V1. It should be non-linear
ggplot(data.train, aes(x=V1, y = pred)) +
    geom_point()
```



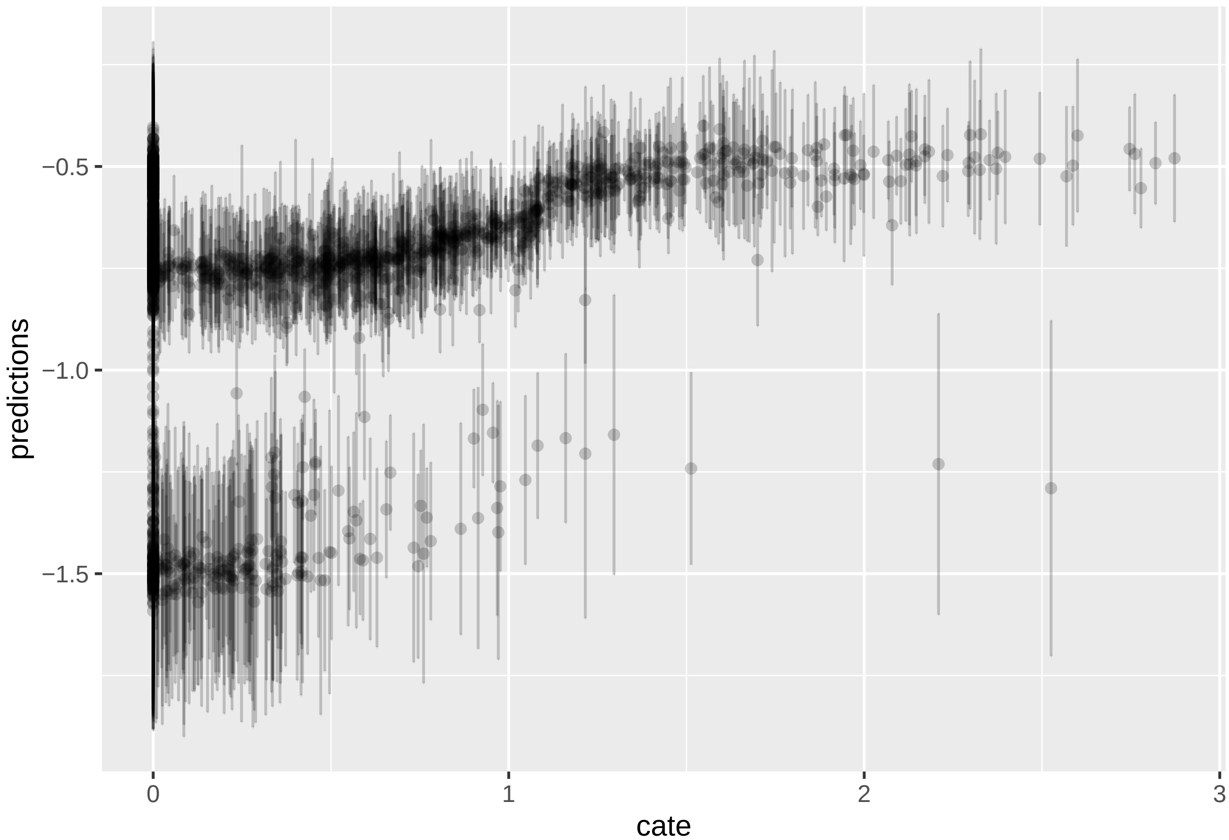
Thus, it seems that our CATE prediction from the causal forest has indeed a non-linear relationship with $V1$. Let's compare the real CATE with the prediction:

```
# Storing out CATE info in a dataframe
graph.data3 <- as.data.frame(tau.hat3)

# Create cate and CI bounders
graph.data3 <- graph.data3 %>%
    mutate(cate=pmax(data.train$V1,0)) %>%
    mutate(lower=predictions - sigma.hat3, upper=predictions + sigma.hat3)

# Plot the relationship between the predicted CATE and the real CATE
ggplot(graph.data3, aes(x=cate, y=predictions)) + geom_point()
```



Which seems that the CATE's causal forest prediction matches with the real one. Finally, with we look at the error plot:

```
# Plot error plot with standard errors
ggplot() + geom_errorbar(graph.data3, mapping=aes(x=cate,  ymin=lower,ymax=upper), alpha = 0.2) + geom_p
```



That is, our prediction in fact has a non-linear relationship with the CATE, which corresponds with our real model.

Is interesting to see that the CATE estimated by the causal forest algorithm, even not performing good for simpler models (See Tutorial 1), brings us good results when there is a clear non-linear relationship between the covariate and the treatment. The next step is to do with observational data, rather than randomized.

References

Tutorial 1 - Estimating Heterogeneous Treatment Effects in Randomized Data with Machine Learning Techniques (available at https://sites.google.com/view/victor-hugo-alexandrino/)

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

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

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

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

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

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

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