Data Science Methods in Causal Inference

Ryan T. Moore

American University

2024-08-23

Table of contents I

Data Science in Causal Inference

Approaches of Prediction and Causal Inference

Two Cultures, (Breiman 2001)

- *Data Models*: our "social science modeling"

Approaches of Prediction and Causal Inference

Two Cultures, (Breiman 2001)

- *Data Models*: our "social science modeling"
- *Algorithmic Models*: our "data science algorithms"

Methods for Prediction and Causal Inference

- Cross-validation
- Regression/Decision trees
- Random forests

James et al. (2021)

Cross-validation

Cross-validation

k-fold cross-validation to select method

Cross-validation

k-fold cross-validation to select method

- Randomly partition data into k groups

Cross-validation

k-fold cross-validation to select method

- Randomly partition data into k groups
- Apply method to $k - 1$ groups

Cross-validation

k-fold cross-validation to select method

- Randomly partition data into k groups
- Apply method to $k-1$ groups
- Use result to predict for left-out group

Cross-validation

k-fold cross-validation to select method

- Randomly partition data into k groups
- Apply method to $k-1$ groups
- Use result to predict for left-out group
- Calculate $\text{MSE}_i = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Cross-validation

k-fold cross-validation to select method

- Randomly partition data into k groups
- Apply method to $k-1$ groups
- Use result to predict for left-out group
- Calculate $\mathrm{MSE}_i = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- Calculate test error as average of the k MSE's:

Cross-validation

k-fold cross-validation to select method

- Randomly partition data into k groups
- Apply method to $k-1$ groups
- Use result to predict for left-out group
- Calculate $\text{MSE}_i = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- Calculate test error as average of the k MSE's:

Cross-validation

k-fold cross-validation to select method

- Randomly partition data into k groups
- Apply method to $k-1$ groups
- Use result to predict for left-out group
- Calculate $\text{MSE}_i = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- Calculate test error as average of the k MSE's:

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\text{MSE}_i$$

Cross-validation

k-fold cross-validation to select method

- Randomly partition data into k groups
- Apply method to $k-1$ groups
- Use result to predict for left-out group
- Calculate $\text{MSE}_i = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- Calculate test error as average of the k MSE's:

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\text{MSE}_i$$

- Select model that minimises $CV_{(k)}$

CV for Linear Model

```r
## Make data

mk_data <- function(n = 90, n_folds = 10){

  df <- tibble(
    x1 = rnorm(n),
    x2 = rnorm(n),
    x3 = rnorm(n),
    y = 0.1 * x1 + 0.2 * x2 + 0.5 * x3 + rnorm(n),
    cv_fold = sample(rep(1:n_folds, (n / n_folds)))
  )

}

df <- mk_data()
```

CV for Linear Model

```
head(df)
```

```
# A tibble: 6 x 5
       x1      x2      x3       y cv_fold
    <dbl>   <dbl>   <dbl>   <dbl>   <int>
1  1.34   -0.617   0.0877 -0.358       10
2 -0.333  -0.0812  0.817  -0.112        6
3 -0.267   0.262  -0.402   0.875        9
4 -1.15   -0.657   0.182   0.168        4
5 -0.857   0.320  -2.38   -0.288        2
6 -0.0667 -0.923   0.726   0.0846      10
```

CV for Linear Model

```
head(df)
```

```
# A tibble: 6 x 5
       x1      x2      x3       y cv_fold
    <dbl>   <dbl>   <dbl>   <dbl>   <int>
1  1.34   -0.617   0.0877 -0.358       10
2 -0.333  -0.0812  0.817  -0.112        6
3 -0.267   0.262  -0.402   0.875        9
4 -1.15   -0.657   0.182   0.168        4
5 -0.857   0.320  -2.38   -0.288        2
6 -0.0667 -0.923   0.726   0.0846      10
```

```
table(df$cv_fold)
```

```

 1  2  3  4  5  6  7  8  9 10
 9  9  9  9  9  9  9  9  9  9
```

CV for Linear Model

```
cv_lm <- function(data, fmla){

  n_folds <- max(data$cv_fold)
  store_mses <- vector("numeric", length = n_folds)

  for(idx in 1:n_folds){

    df_train <- data |> filter(cv_fold != idx)
    df_test <- data |> filter(cv_fold == idx)

    lm_out <- lm(fmla, data = df_train)

    predictions <- predict(lm_out, newdata = df_test)

    store_mses[idx] <- mean((df_test$y - predictions)^2)}

  test_error_cv_k <- mean(store_mses)
  return(test_error_cv_k)
```

CV for Linear Model

```
cv_lm(data = df, fmla = y ~ x1 + x2)
```

```
[1] 1.274226
```

CV for Linear Model

```r
cv_lm(data = df, fmla = y ~ x1 + x2)
```

```
[1] 1.274226
```

```r
df <- mk_data()
cv_lm(df, y ~ x1 + x2)
```

```
[1] 1.268971
```

CV for Linear Model

```
cv_lm(data = df, fmla = y ~ x1 + x2)
```

```
[1] 1.274226
```

```
df <- mk_data()
cv_lm(df, y ~ x1 + x2)
```

```
[1] 1.268971
```

```
df <- mk_data()
cv_lm(df, y ~ x1 + x2 + x3)
```

```
[1] 0.9664745
```

CV for Linear Model

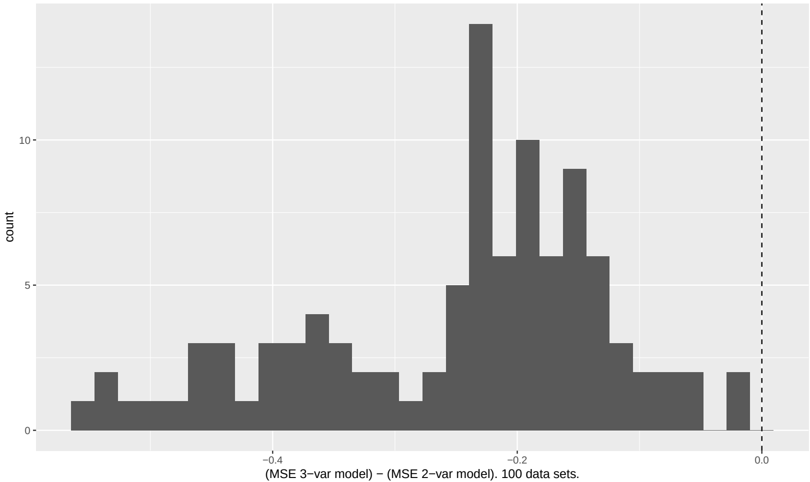


Figure 1: MSE always less (better) for 3-variable model.

Regression Trees

Regression Trees

- Partition predictor space into regions $R_1, R_2, \ldots, R_J$.

Regression Trees

- Partition predictor space into regions $R_1, R_2, \dots, R_J$.
- If unit falls in region R_j, use average outcome in R_j as predicted value: $\hat{y}_{R_j}$

Regression Trees

- Partition predictor space into regions $R_1, R_2, \dots, R_J$.
- If unit falls in region R_j, use average outcome in R_j as predicted value: $\widehat{y}_{R_j}$
- (For *decision* on discrete outcome, count votes in R_j)

Regression Trees

- Partition predictor space into regions $R_1, R_2, \dots, R_J$.
- If unit falls in region R_j, use average outcome in R_j as predicted value: $\hat{y}_{R_j}$
- (For *decision* on discrete outcome, count votes in R_j)
- Goal: minimise residual sum of squares (RSS), just like LS regression:

Regression Trees

- Partition predictor space into regions $R_1, R_2, \ldots, R_J$.
- If unit falls in region R_j, use average outcome in R_j as predicted value: $\hat{y}_{R_j}$
- (For *decision* on discrete outcome, count votes in R_j)
- Goal: minimise residual sum of squares (RSS), just like LS regression:

Regression Trees

- Partition predictor space into regions $R_1, R_2, \ldots, R_J$.
- If unit falls in region R_j, use average outcome in R_j as predicted value: $\hat{y}_{R_j}$
- (For *decision* on discrete outcome, count votes in R_j)
- Goal: minimise residual sum of squares (RSS), just like LS regression:

$$\sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j} \right)$$

Regression Trees

How to define regions R_j?

Regression Trees

How to define regions R_j?

- Top-down, greedy recursive binary split

Regression Trees

How to define regions R_j?

- Top-down, greedy recursive binary split
- At each step, find predictor and cut-point that minimise

Regression Trees

How to define regions R_j?

- Top-down, greedy recursive binary split
- At each step, find predictor and cut-point that minimise

Regression Trees

How to define regions R_j?

- Top-down, greedy recursive binary split
- At each step, find predictor and cut-point that minimise

$$\sum_{i:x\in R_1(j,s)} \left(y_i - \hat{y}_{R_1(j,s)}\right)^2 + \sum_{i:x\in R_2(j,s)} \left(y_i - \hat{y}_{R_2(j,s)}\right)^2$$

Regression Trees

- Overfitting is a potential problem

- Overfitting is a potential problem
- Can we increase predictive quality by only using *part* of a tree?

- Overfitting is a potential problem
- Can we increase predictive quality by only using *part* of a tree?
- "Pruning"

Regression Trees

Pruning

- Build a large tree

Regression Trees

Pruning

- Build a large tree
- Select the subtree that gives least prediction error (via cross-validation)

Regression Trees

Pruning

- Build a large tree
- Select the subtree that gives least prediction error (via cross-validation)
- But, many possible subtrees, so penalise larger trees via α

Regression Trees

Pruning

- Build a large tree
- Select the subtree that gives least prediction error (via cross-validation)
- But, many possible subtrees, so penalise larger trees via α
- α: penalty parameter

Regression Trees

Pruning

- Build a large tree
- Select the subtree that gives least prediction error (via cross-validation)
- But, many possible subtrees, so penalise larger trees via α
- α: penalty parameter
- $|T|$: count of terminal nodes of T

Regression Trees

Pruning

- Build a large tree
- Select the subtree that gives least prediction error (via cross-validation)
- But, many possible subtrees, so penalise larger trees via α
- α: penalty parameter
- $|T|$: count of terminal nodes of T
- m: terminal node index

Regression Trees

Pruning

- Build a large tree
- Select the subtree that gives least prediction error (via cross-validation)
- But, many possible subtrees, so penalise larger trees via α
- α: penalty parameter
- $|T|$: count of terminal nodes of T
- m: terminal node index
- Find subtree that minimises

Regression Trees

Pruning

- Build a large tree
- Select the subtree that gives least prediction error (via cross-validation)
- But, many possible subtrees, so penalise larger trees via α
- α: penalty parameter
- $|T|$: count of terminal nodes of T
- m: terminal node index
- Find subtree that minimises

Regression Trees

Pruning

- Build a large tree
- Select the subtree that gives least prediction error (via cross-validation)
- But, many possible subtrees, so penalise larger trees via α
- α: penalty parameter
- $|T|$: count of terminal nodes of T
- m: terminal node index
- Find subtree that minimises

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2 + \alpha|T|$$

Regression Trees

Pruning

- Build a large tree
- Select the subtree that gives least prediction error (via cross-validation)
- But, many possible subtrees, so penalise larger trees via α
- α: penalty parameter
- $|T|$: count of terminal nodes of T
- m: terminal node index
- Find subtree that minimises

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2 + \alpha |T|$$

Sum squared pred. error (plus penalty that grows with tree size) across units in region, then regions.

Regression Trees

But, how to choose α? (Use cross-validation.)

1. Build big tree on training data (with some minimum terminal node size)

Regression Trees

But, how to choose α? (Use cross-validation.)

1. Build big tree on training data (with some minimum terminal node size)
2. For several values of α, find best subtree.

Regression Trees

But, how to choose α? (Use cross-validation.)

1. Build big tree on training data (with some minimum terminal node size)
2. For several values of α, find best subtree.
3. Find best value of α via CV. Create K folds. Then

Regression Trees

But, how to choose α? (Use cross-validation.)

1. Build big tree on training data (with some minimum terminal node size)
2. For several values of α, find best subtree.
3. Find best value of α via CV. Create K folds. Then
   3a. Do 1 and 2 on all but kth fold

Regression Trees

But, how to choose α? (Use cross-validation.)

1. Build big tree on training data (with some minimum terminal node size)
2. For several values of α, find best subtree.
3. Find best value of α via CV. Create K folds. Then
   3a. Do 1 and 2 on all but kth fold
   3b. Predict for kth fold, calculate MSE for several values of α

Regression Trees

But, how to choose α? (Use cross-validation.)

1. Build big tree on training data (with some minimum terminal node size)
2. For several values of α, find best subtree.
3. Find best value of α via CV. Create K folds. Then
   3a. Do 1 and 2 on all but kth fold
   3b. Predict for kth fold, calculate MSE for several values of α
   3c. Get avg MSE for each α

Regression Trees

But, how to choose α? (Use cross-validation.)

1. Build big tree on training data (with some minimum terminal node size)
2. For several values of α, find best subtree.
3. Find best value of α via CV. Create K folds. Then
   3a. Do 1 and 2 on all but kth fold
   3b. Predict for kth fold, calculate MSE for several values of α
   3c. Get avg MSE for each α
   3d. Pick α to minimise MSE

Regression Trees

But, how to choose α? (Use cross-validation.)

1. Build big tree on training data (with some minimum terminal node size)
2. For several values of α, find best subtree.
3. Find best value of α via CV. Create K folds. Then
   3a. Do 1 and 2 on all but kth fold
   3b. Predict for kth fold, calculate MSE for several values of α
   3c. Get avg MSE for each α
   3d. Pick α to minimise MSE

4. Using that α, select best subtree from Step 2

Example: Regression Tree

Effect of office-holding on wealth
(Eggers and Hainmueller 2009):

```
library(qss)
library(rsample)
library(tree)

data("MPs")
mps <- MPs |> mutate(age = yod - yob,
                     is_labour = if_else(party == "labour",
                     is_london = if_else(region == "Greate
                     is_winner = if_else(margin > 0, 1, 0))
  select(ln.net, age, is_labour, is_london, is_winner) |>
  na.omit()
```

Example: Regression Tree

```
set.seed(765076184)

mp_split <- initial_split(mps, prop = 0.7)

mp_train <- training(mp_split)
mp_test <- testing(mp_split)
```

Example: Regression Tree

```
tree_mp <- tree(ln.net ~ ., data = mp_train)
plot(tree_mp)
text(tree_mp)
```

Figure 2: The regression tree (for training data)

Example: Regression Tree

Would pruning help?

Example: Regression Tree

Would pruning help?

```
cv_mps <- cv.tree(tree_mp, K = 10)

plot(cv_mps$size, cv_mps$dev, type = "b")
```

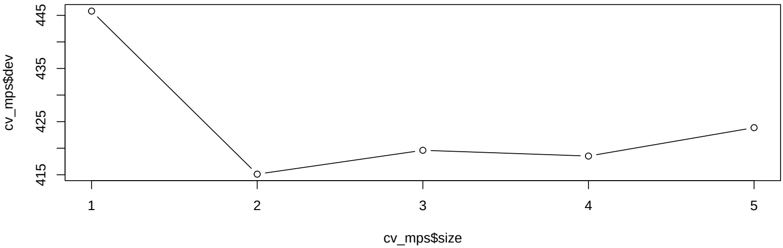


Figure 3: Subtree size 2 minimises SSR

Example: Regression Tree

Would pruning help?

```
cv_mps <- cv.tree(tree_mp, K = 10)

plot(cv_mps$size, cv_mps$dev, type = "b")
```

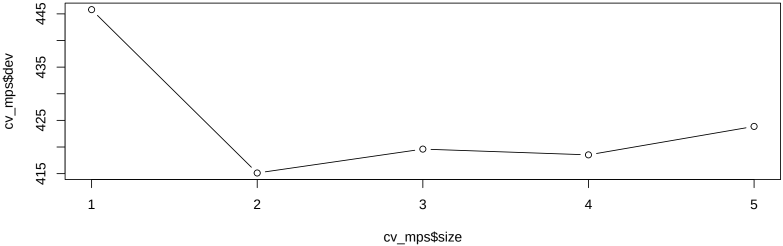


Figure 3: Subtree size 2 minimises SSR

Example: Regression Tree

```r
prune_mps <- prune.tree(tree_mp, best = 2)

plot(prune_mps)
text(prune_mps)
```

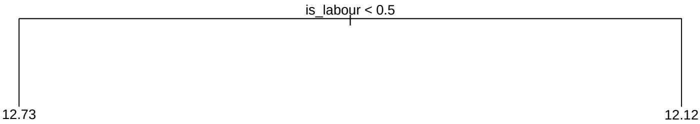


Figure 4: The pruned tree

Example: Regression Tree

Predict for test set:

- MSE for pruned: 1.922
- MSE for full: 1.945

Example: Regression Tree

Predict for test set:

- MSE for pruned: 1.922
- MSE for full: 1.945

So, pruning helped us avoid some overfitting.

Example: Regression Tree

Predict for test set:

- MSE for pruned: 1.922
- MSE for full: 1.945

So, pruning helped us avoid some overfitting.

(Typical pred error of $\sqrt{1.922} \approx 1.386$)

Example: Regression Tree

Predict for test set:

- MSE for pruned: 1.922
- MSE for full: 1.945

So, pruning helped us avoid some overfitting.

(Typical pred error of $\sqrt{1.922} \approx 1.386$)

(Bigger than IQR of 1.183, but range covers $[6.98, 16.3]$.)

Example: Regression Tree

Predict for test set:

- MSE for pruned: 1.922
- MSE for full: 1.945

So, pruning helped us avoid some overfitting.

(Typical pred error of $\sqrt{1.922} \approx 1.386$)

(Bigger than IQR of 1.183, but range covers $[6.98, 16.3]$.)

(Pretty good for 1 split!?)

Random Forests

Random Forests

Next: random forest algorithm

Random Forests

Next: random forest algorithm

Ensemble learning algorithms:

Random Forests

Next: random forest algorithm

Ensemble learning algorithms:

- Boosting: models build on prior models $\rightsquigarrow$ pick feature, predict, upweight mispredicted data, ... Do several times and combine.

Random Forests

Next: random forest algorithm

Ensemble learning algorithms:

- Boosting: models build on prior models $\rightsquigarrow$ pick feature, predict, upweight mispredicted data, ... Do several times and combine.
- Bagging: (random select units, model) $\rightarrow$ many times. No building.

Random Forests

Next: random forest algorithm

Ensemble learning algorithms:

- Boosting: models build on prior models $\rightsquigarrow$ pick feature, predict, upweight mispredicted data, ... Do several times and combine.
- Bagging: (random select units, model) $\rightarrow$ many times. No building.

Random Forests

Next: random forest algorithm

Ensemble learning algorithms:

- Boosting: models build on prior models $\rightsquigarrow$ pick feature, predict, upweight mispredicted data, ... Do several times and combine.
- Bagging: (random select units, model) $\rightarrow$ many times. No building.

Random Forests are bagging algorithms.

Random Forests

Next: random forest algorithm

Ensemble learning algorithms:

- Boosting: models build on prior models $\rightsquigarrow$ pick feature, predict, upweight mispredicted data, ... Do several times and combine.
- Bagging: (random select units, model) $\rightarrow$ many times. No building.

Random Forests are bagging algorithms.

Bagging: *b*ootstrap *agg*regation

Why bag?

Random Forests

Why bag?

- Trees are low bias, high variance
  (diff answers, depend on data split)

Why bag?

- Trees are low bias, high variance
  (diff answers, depend on data split)
- Bagging averages over data subsets, reducing variance

Random Forests

Why bag?

- Trees are low bias, high variance
  (diff answers, depend on data split)
- Bagging averages over data subsets, reducing variance
- (Linear regression: lower variance)

Random forests: decorrelated, bagged trees

Random Forests

Random forests: decorrelated, bagged trees

- Take bootstrapped training subsample

Random Forests

Random forests: decorrelated, bagged trees

- Take bootstrapped training subsample
- Build deep tree. At each split, *randomly sample* m of p predictors, build split from only those m.

Random Forests

Random forests: decorrelated, bagged trees

- Take bootstrapped training subsample
- Build deep tree. At each split, *randomly sample* m of p predictors, build split from only those m.
- (Often choose $m \approx \sqrt{p}$)

Random forests: decorrelated, bagged trees

- Take bootstrapped training subsample
- Build deep tree. At each split, *randomly sample* m of p predictors, build split from only those m.
- (Often choose $m \approx \sqrt{p}$)
- So, different splits consider different predictors

Random Forests

Random forests: decorrelated, bagged trees

- Take bootstrapped training subsample
- Build deep tree. At each split, *randomly sample* m of p predictors, build split from only those m.
- (Often choose $m \approx \sqrt{p}$)
- So, different splits consider different predictors
- So, trees will look very different to each other

Example: Random Forests

```r
library(randomForest)

# Full bag:
bag_mps <- randomForest(ln.net ~ ., data = mp_train,
                        ntree = 500, mtry = 4,
                        importance = TRUE)

# Decorrelate:
rf_mps <- randomForest(ln.net ~ ., data = mp_train,
                       ntree = 500, mtry = 2,
                       importance = TRUE)
```

Example: Random Forests

Predict:

```
preds_bag <- predict(bag_mps, newdata = mp_test)

preds_rf <- predict(rf_mps, newdata = mp_test)
```

- MSE for RF: 1.995
- MSE for full bag: 2.536

Example: Random Forests

Predict:

```
preds_bag <- predict(bag_mps, newdata = mp_test)

preds_rf <- predict(rf_mps, newdata = mp_test)
```

- MSE for RF: 1.995
- MSE for full bag: 2.536

So, decorrelating helped us avoid some overfitting to each bootstrap subsample (and thus, reduced variance).

Example: Random Forests

Predict:

```
preds_bag <- predict(bag_mps, newdata = mp_test)

preds_rf <- predict(rf_mps, newdata = mp_test)
```

- MSE for RF: 1.995
- MSE for full bag: 2.536

So, decorrelating helped us avoid some overfitting to each bootstrap subsample (and thus, reduced variance).

(Typical pred error of $\sqrt{1.995} \approx 1.412$)

Example: Random Forests

Predict:

```
preds_bag <- predict(bag_mps, newdata = mp_test)

preds_rf <- predict(rf_mps, newdata = mp_test)
```

- MSE for RF: 1.995
- MSE for full bag: 2.536

So, decorrelating helped us avoid some overfitting to each bootstrap subsample (and thus, reduced variance).

(Typical pred error of $\sqrt{1.995} \approx 1.412$)

(Bigger than IQR of 1.183, but range covers $[6.98, 16.3]$.)

Heterogeneous Treatment Effects

Homogeneous and Heterogeneous Effects

- Most causal inference starts at *average treatment effects*

Homogeneous and Heterogeneous Effects

- Most causal inference starts at *average treatment effects*
- Average may be interesting on its own, ...

Homogeneous and Heterogeneous Effects

- Most causal inference starts at *average treatment effects*
- Average may be interesting on its own, ...

Homogeneous and Heterogeneous Effects

- Most causal inference starts at *average treatment effects*
- Average may be interesting on its own, ...but often masks assumption of *homogeneous* effects

Homogeneous and Heterogeneous Effects

- Most causal inference starts at *average treatment effects*
- Average may be interesting on its own, ...but often masks assumption of *homogeneous* effects
- Notationally, often assume $\tau_i = \tau \quad \forall i$

Homogeneous and Heterogeneous Effects

- Most causal inference starts at *average treatment effects*
- Average may be interesting on its own, ...but often masks assumption of *homogeneous* effects
- Notationally, often assume $\tau_i = \tau \quad \forall i$
- But, *heterogeneous* effects often of central interest

Homogeneous and Heterogeneous Effects

- Most causal inference starts at *average treatment effects*
- Average may be interesting on its own, ...but often masks assumption of *homogeneous* effects
- Notationally, often assume $\tau_i = \tau \quad \forall i$
- But, *heterogeneous* effects often of central interest
- Different effects for different groups

Homogeneous and Heterogeneous Effects

- Most causal inference starts at *average treatment effects*
- Average may be interesting on its own, ...but often masks assumption of *homogeneous* effects
- Notationally, often assume $\tau_i = \tau \quad \forall i$
- But, *heterogeneous* effects often of central interest
- Different effects for different groups
    - Subgroup variability (research)

Homogeneous and Heterogeneous Effects

- Most causal inference starts at *average treatment effects*
- Average may be interesting on its own, ...but often masks assumption of *homogeneous* effects
- Notationally, often assume $\tau_i = \tau \quad \forall i$
- But, *heterogeneous* effects often of central interest
- Different effects for different groups
  - Subgroup variability (research)
  - Targeting resources (campaigns, marketing)

Homogeneous and Heterogeneous Effects

- Most causal inference starts at *average treatment effects*
- Average may be interesting on its own, ...but often masks assumption of *homogeneous* effects
- Notationally, often assume $\tau_i = \tau \quad \forall i$
- But, *heterogeneous* effects often of central interest
- Different effects for different groups
  - Subgroup variability (research)
  - Targeting resources (campaigns, marketing)
  - Constituency effects (public policy)

Homogeneous and Heterogeneous Effects

- Most causal inference starts at *average treatment effects*
- Average may be interesting on its own, ...but often masks assumption of *homogeneous* effects
- Notationally, often assume $\tau_i = \tau \quad \forall i$
- But, *heterogeneous* effects often of central interest
- Different effects for different groups
  - Subgroup variability (research)
  - Targeting resources (campaigns, marketing)
  - Constituency effects (public policy)
- Notationally, $\exists i : \tau_i \neq \tau$

Homogeneous and Heterogeneous Effects: Estimation

Homogeneous effects:

$$\text{Outcome} = \beta_0 + \beta_1 \text{Treatment} + \epsilon$$

Homogeneous and Heterogeneous Effects: Estimation

Homogeneous effects:

$$\text{Outcome} = \beta_0 + \beta_1 \text{Treatment} + \epsilon$$

```r
lm_out <- lm(ln.net ~ is_winner, data = mps)
lm_out
```

```

Call:
lm(formula = ln.net ~ is_winner, data = mps)

Coefficients:
(Intercept)    is_winner
    12.2464       0.5176
```

Homogeneous and Heterogeneous Effects: Estimation

Homogeneous effects:

```r
t.test(ln.net ~ is_winner, data = mps)
```

```

	Welch Two Sample t-test

data:  ln.net by is_winner
t = -3.9552, df = 287.65, p-value = 9.636e-05
alternative hypothesis: true difference in means betwee
95 percent confidence interval:
 -0.7751044 -0.2599998
sample estimates:
mean in group 0 mean in group 1
       12.24641        12.76396
```

Homogeneous and Heterogeneous Effects: Estimation

Homogeneous effects:

$$\text{Outcome} = \beta_0 + \beta_1 \text{Treatment} + \sum \beta_j X_j + \epsilon$$

Homogeneous and Heterogeneous Effects: Estimation

Homogeneous effects:

$$\text{Outcome} = \beta_0 + \beta_1 \text{Treatment} + \sum \beta_j X_j + \epsilon$$

```r
lm_out <- lm(ln.net ~ is_winner + is_labour +
               is_london + age, data = mps)
lm_out
```

```

Call:
lm(formula = ln.net ~ is_winner + is_labour + is_london + a
    data = mps)

Coefficients:
(Intercept)    is_winner    is_labour    is_london
  12.078838     0.398818    -0.477549     0.161134      0.00
```

Homogeneous and Heterogeneous Effects: Estimation

Homogeneous effects:

```
lm_lin(ln.net ~ is_winner, covariates = ~ is_labour + is_lo
```

```
                          Estimate  Std. Error     t valu
(Intercept)            1.226687e+01 0.078894901 155.483661
is_winner              3.459885e-01 0.131207672   2.636953
is_labour_c           -1.613663e-01 0.152608515  -1.057387
is_london_c            2.427360e-01 0.250214401   0.970111
age_c                  4.740367e-03 0.007031323   0.674178
is_winner:is_labour_c -9.104022e-01 0.264395760  -3.443331
is_winner:is_london_c -8.847770e-02 0.426241818  -0.207576
is_winner:age_c       -4.778657e-05 0.012753800  -0.003746
                         CI Lower    CI Upper  DF
(Intercept)           12.111785723 12.42195044 416
is_winner              0.088075873  0.60390123 416
is_labour_c           -0.461346226  0.13861367 416
is_london_c           -0.249106208  0.73457813 416
```

CATEs: Conditional ATEs

- *Conditional average treatment effect* (CATE): avg treatment effect for subset of population

CATEs: Conditional ATEs

- *Conditional average treatment effect* (CATE): avg treatment effect for subset of population
- Sometimes "CACE"

CATEs: Conditional ATEs

- *Conditional average treatment effect* (CATE): avg treatment effect for subset of population
- Sometimes "CACE"
- Inference: not "evidence against TE $= 0$?", but "evidence against $\text{CATE}_1 = \text{CATE}_2$?"

Homogeneous and Heterogeneous Effects: Estimation

Heterogeneous effects:

$$\text{Outcome} = \beta_0 + \beta_1 \text{Treatment} + \beta_2 \text{Group} + \beta_3 \text{Treatment} \cdot \text{Group} + \epsilon$$

Homogeneous and Heterogeneous Effects: Estimation

Heterogeneous effects:

$$\text{Outcome} = \beta_0 + \beta_1 \text{Treatment} + \beta_2 \text{Group} + \beta_3 \text{Treatment} \cdot \text{Group} + \epsilon$$

- β_1 gives TE for `Group == 0`

Homogeneous and Heterogeneous Effects: Estimation

Heterogeneous effects:

$$\text{Outcome} = \beta_0 + \beta_1 \text{Treatment} + \beta_2 \text{Group} + \beta_3 \text{Treatment} \cdot \text{Group} + \epsilon$$

- β_1 gives TE for `Group == 0`
- $\beta_1 + \beta_3$ gives TE for `Group == 1`

Homogeneous and Heterogeneous Effects: Estimation

Heterogeneous effects:

```r
lm_out <- lm(ln.net ~ is_winner * is_labour +
               is_london + age, data = mps)
coef(lm_out) |> round(3)
```

```
      (Intercept)           is_winner          is_labou
           11.959               0.780            -0.162
              age is_winner:is_labour
            0.005              -0.914
```

Causal Forests

Causal Forests

- Our regression trees had terminal nodes (“leaves”) that were sufficiently homogeneous for prediction.

Causal Forests

- Our regression trees had terminal nodes ("leaves") that were sufficiently homogeneous for prediction.
- Use $\hat{y}_{R_j}$ as pred value for obs in R_j

Causal Forests

- Our regression trees had terminal nodes ("leaves") that were sufficiently homogeneous for prediction.
- Use $\hat{y}_{R_j}$ as pred value for obs in R_j
- (Tory, winner, London $\rightarrow$ 13.84)

Causal Forests

- Our regression trees had terminal nodes ("leaves") that were sufficiently homogeneous for prediction.
- Use $\hat{y}_{R_j}$ as pred value for obs in R_j
- (Tory, winner, London $\rightarrow$ 13.84)
- $$\hat{y}_{R_j} = \frac{1}{|R_j|} \sum_{i \in R_j} Y_i$$

Causal Forests

- Similarly, consider each leaf R_j small, homogeneous enough that potential outcomes independent

Causal Forests

- Similarly, consider each leaf R_j small, homogeneous enough that potential outcomes independent
- Treateds in R_j provide good estimates of what controls in R_j would have done under treatment

Causal Forests

- Similarly, consider each leaf R_j small, homogeneous enough that potential outcomes independent
- Treateds in R_j provide good estimates of what controls in R_j would have done under treatment
- Controls in R_j provide good estimates of what treateds in R_j would have done under control

Causal Forests

- Similarly, consider each leaf R_j small, homogeneous enough that potential outcomes independent
- Treateds in R_j provide good estimates of what controls in R_j would have done under treatment
- Controls in R_j provide good estimates of what treateds in R_j would have done under control
- Assignment w/in leaf R_j is as-good-as-random

Causal Forests

- Similarly, consider each leaf R_j small, homogeneous enough that potential outcomes independent
- Treateds in R_j provide good estimates of what controls in R_j would have done under treatment
- Controls in R_j provide good estimates of what treateds in R_j would have done under control
- Assignment w/in leaf R_j is as-good-as-random
- I.e., each leaf contains an experiment

Causal Forests

- Similarly, consider each leaf R_j small, homogeneous enough that potential outcomes independent
- Treateds in R_j provide good estimates of what controls in R_j would have done under treatment
- Controls in R_j provide good estimates of what treateds in R_j would have done under control
- Assignment w/in leaf R_j is as-good-as-random
- I.e., each leaf contains an experiment

Causal Forests

- Similarly, consider each leaf R_j small, homogeneous enough that potential outcomes independent
- Treateds in R_j provide good estimates of what controls in R_j would have done under treatment
- Controls in R_j provide good estimates of what treateds in R_j would have done under control
- Assignment w/in leaf R_j is as-good-as-random
- I.e., each leaf contains an experiment

$$Y(0), Y(1) \perp\!\!\!\perp T | \mathbf{X}$$

Causal Forests

- Let $\{T, R_j\} = \{i : T_i = 1, i \in R_j\} \quad$ (Tr obs in R_j)

Causal Forests

- Let $\{T, R_j\} = \{i : T_i = 1, i \in R_j\}$ (Tr obs in R_j)
- Let $\{C, R_j\} = \{i : T_i = 0, i \in R_j\}$ (Co obs in R_j)

Causal Forests

- Let $\{T, R_j\} = \{i : T_i = 1, i \in R_j\}$ (Tr obs in R_j)
- Let $\{C, R_j\} = \{i : T_i = 0, i \in R_j\}$ (Co obs in R_j)
- Natural estimation of

Causal Forests

- Let $\{T, R_j\} = \{i : T_i = 1, i \in R_j\}$ (Tr obs in R_j)
- Let $\{C, R_j\} = \{i : T_i = 0, i \in R_j\}$ (Co obs in R_j)
- Natural estimation of

Causal Forests

- Let $\{T, R_j\} = \{i : T_i = 1, i \in R_j\}$ (Tr obs in R_j)
- Let $\{C, R_j\} = \{i : T_i = 0, i \in R_j\}$ (Co obs in R_j)
- Natural estimation of

$$\hat{\hat{\tau}}_{R_j} = \frac{1}{|\{T, R_j\}|} \sum_{\{T,R_j\}} Y_i - \frac{1}{|\{C, R_j\}|} \sum_{\{C,R_j\}} Y_i$$

Causal Forests

- Let $\{T, R_j\} = \{i : T_i = 1, i \in R_j\}$ (Tr obs in R_j)
- Let $\{C, R_j\} = \{i : T_i = 0, i \in R_j\}$ (Co obs in R_j)
- Natural estimation of

$$\hat{\hat{\tau}}_{R_j} = \frac{1}{|\{T, R_j\}|} \sum_{\{T, R_j\}} Y_i - \frac{1}{|\{C, R_j\}|} \sum_{\{C, R_j\}} Y_i$$

So, we can use RF methods to estimate conditional (heterogeneous) treatment effects, CATEs.

Causal Forests

- Let $\{T, R_j\} = \{i : T_i = 1, i \in R_j\}$ (Tr obs in R_j)
- Let $\{C, R_j\} = \{i : T_i = 0, i \in R_j\}$ (Co obs in R_j)
- Natural estimation of

$$\hat{\hat{\tau}}_{R_j} = \frac{1}{|\{T, R_j\}|} \sum_{\{T, R_j\}} Y_i - \frac{1}{|\{C, R_j\}|} \sum_{\{C, R_j\}} Y_i$$

So, we can use RF methods to estimate conditional (heterogeneous) treatment effects, CATEs.

☺

Causal Forests: Honesty

- But need one more thing for asymptotics to work out ...

Causal Forests: Honesty

- But need one more thing for asymptotics to work out ...
- Each tree must be *honest*

Causal Forests: Honesty

- But need one more thing for asymptotics to work out ...
- Each tree must be *honest*
- Each unit i *either*

Causal Forests: Honesty

- But need one more thing for asymptotics to work out ...
- Each tree must be *honest*
- Each unit i *either*
  - used to determine tree splits, *or*

Causal Forests: Honesty

- But need one more thing for asymptotics to work out ...
- Each tree must be *honest*
- Each unit i *either*
  - used to determine tree splits, *or*
  - used to estimate $\hat{\hat{\tau}}_{R_j}$

Causal Forests: Honesty

- But need one more thing for asymptotics to work out ...
- Each tree must be *honest*
- Each unit i *either*
  - used to determine tree splits, *or*
  - used to estimate $\hat{\hat{\tau}}_{R_j}$
- But **not both**!

Causal Forests: Honesty

- But need one more thing for asymptotics to work out ...
- Each tree must be *honest*
- Each unit i *either*
  - used to determine tree splits, *or*
  - used to estimate $\hat{\hat{\tau}}_{R_j}$
- But **not both**!
- One way: "Double-sample" causal trees

Causal Forests: Honesty

- But need one more thing for asymptotics to work out ...
- Each tree must be *honest*
- Each unit i *either*
  - used to determine tree splits, *or*
  - used to estimate $\hat{\hat{\tau}}_{R_j}$
- But **not both**!
- One way: "Double-sample" causal trees
  - Split training data into $\mathcal{I}$ and $\mathcal{J}$

Causal Forests: Honesty

- But need one more thing for asymptotics to work out ...
- Each tree must be *honest*
- Each unit i *either*
  - used to determine tree splits, *or*
  - used to estimate $\hat{\hat{\tau}}_{R_j}$
- But **not both**!
- One way: "Double-sample" causal trees
  - Split training data into $\mathcal{I}$ and $\mathcal{J}$
  - Splits chosen to maximise variance on $\hat{\hat{\tau}}$ for $i \in \mathcal{J}$

Causal Forests: Honesty

- But need one more thing for asymptotics to work out ...
- Each tree must be *honest*
- Each unit i *either*
  - used to determine tree splits, *or*
  - used to estimate $\hat{\hat{\tau}}_{R_j}$
- But **not both**!
- One way: "Double-sample" causal trees
  - Split training data into $\mathcal{I}$ and $\mathcal{J}$
  - Splits chosen to maximise variance on $\hat{\hat{\tau}}$ for $i \in \mathcal{J}$
  - Splitting cannot use y_i from $\mathcal{I}$

Causal Forests: Honesty

- But need one more thing for asymptotics to work out ...
- Each tree must be *honest*
- Each unit i *either*
  - used to determine tree splits, *or*
  - used to estimate $\hat{\hat{\tau}}_{R_j}$
- But **not both**!
- One way: "Double-sample" causal trees
  - Split training data into $\mathcal{I}$ and $\mathcal{J}$
  - Splits chosen to maximise variance on $\hat{\hat{\tau}}$ for $i \in \mathcal{J}$
  - Splitting cannot use y_i from $\mathcal{I}$
  - Prediction, estimation of $\hat{\hat{\tau}}$ uses only $\mathcal{I}$

Causal Forests: Honesty

- But need one more thing for asymptotics to work out ...
- Each tree must be *honest*
- Each unit i *either*
  - used to determine tree splits, *or*
  - used to estimate $\hat{\hat{\tau}}_{R_j}$
- But **not both**!
- One way: "Double-sample" causal trees
  - Split training data into $\mathcal{I}$ and $\mathcal{J}$
  - Splits chosen to maximise variance on $\hat{\hat{\tau}}$ for $i \in \mathcal{J}$
  - Splitting cannot use y_i from $\mathcal{I}$
  - Prediction, estimation of $\hat{\hat{\tau}}$ uses only $\mathcal{I}$
- Build a random forest (decorrelated deep trees picking from m predictors) of causal trees

Example: Causal Forests

```r
library(grf)

X <- mps |> select(age, is_labour, is_london)

W <- mps |> select(is_winner) |>
  unlist() |> as.numeric()

Y <- mps |> select(ln.net) |> unlist()

cf_out <- causal_forest(X, Y, W)
```

Example: Causal Forests

```
cf_out
```

```
GRF forest object of type causal_forest
Number of trees: 2000
Number of training samples: 424
Variable importance:
    1     2     3
0.537 0.393 0.070
```

Example: Causal Forests

```
cf_out
```

```
GRF forest object of type causal_forest
Number of trees: 2000
Number of training samples: 424
Variable importance:
    1     2     3
0.537 0.393 0.070
```

(“How frequently was i the split feature?”)

Example: Causal Forests

```
cf_pred_est_var <- predict(cf_out, X,
                           estimate.variance = TRUE)
```

Example: Causal Forests

```
cf_pred_est_var <- predict(cf_out, X,
                           estimate.variance = TRUE)
```

```
cf_preds <- cf_pred_est_var$predictions

df_cf <- tibble(X,
                cf_te = cf_preds,
                cf_se = sqrt(cf_pred_est_var$variance.e
                te_1se_lower = cf_te - cf_se,
                te_1se_upper = cf_te + cf_se)
```

Example: Causal Forests

Avg pred treatment effect in honest sample:

```
mean(cf_preds)
```

```
[1] 0.3805919
```

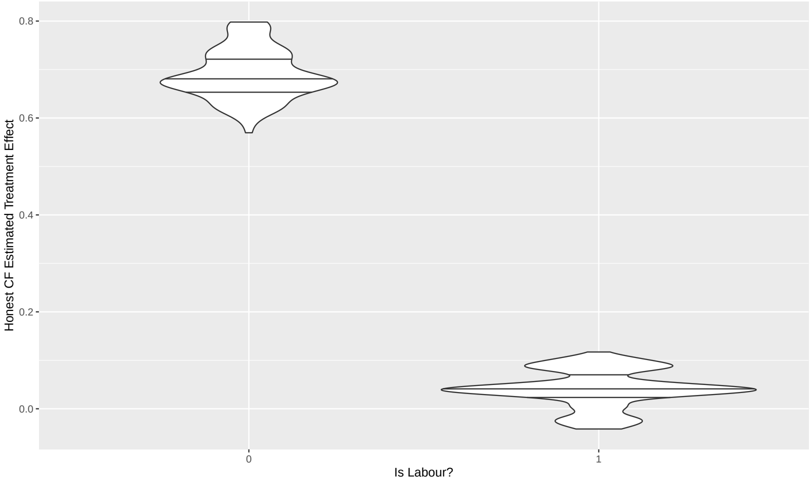
Example: Causal Forests

A doubly-robust ATE from honest sample:

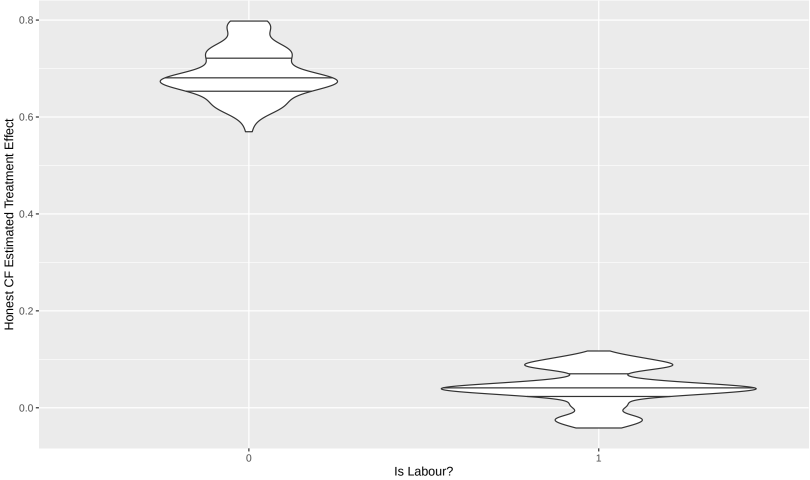
```
average_treatment_effect(cf_out)
```

```
 estimate   std.err
0.3673061 0.1376409
```

Example: Causal Forests Results, Party

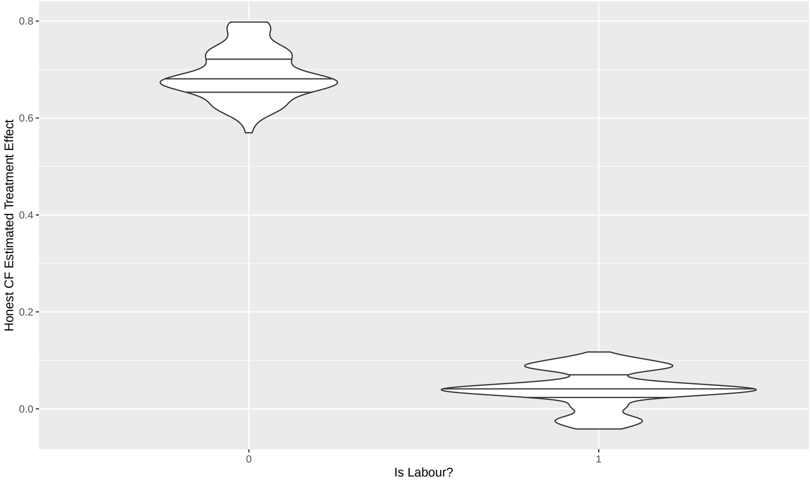


Example: Causal Forests Results, Party



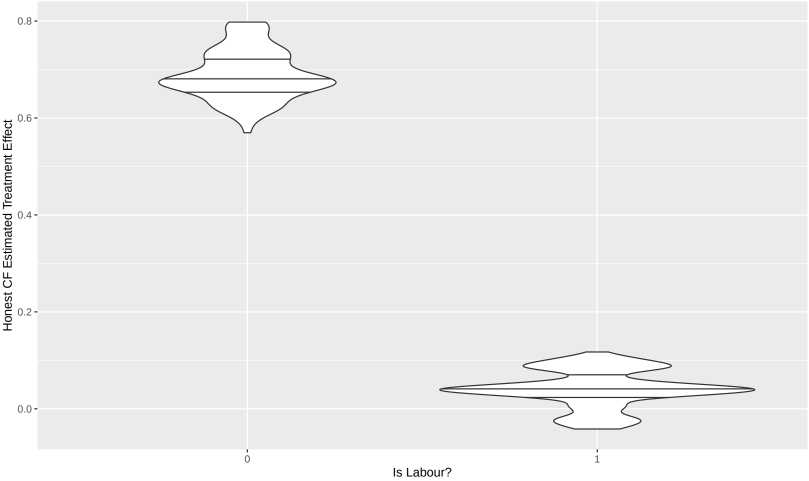
- Mean CF TE, Tory: 0.69

Example: Causal Forests Results, Party



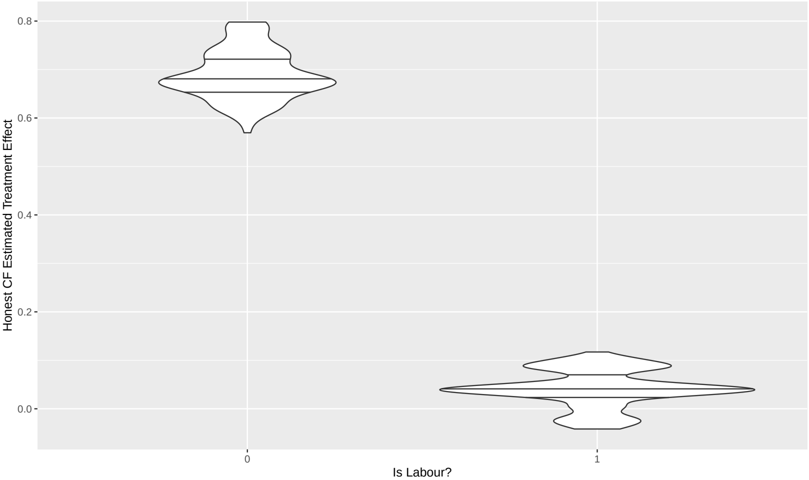
- Mean CF TE, Tory: $0.69 \rightsquigarrow £242,000$

Example: Causal Forests Results, Party



- Mean CF TE, Tory: 0.69 $\rightsquigarrow$ £242, 000
- Mean CF TE, Labour: 0.041

Example: Causal Forests Results, Party



- Mean CF TE, Tory: 0.69 $\rightsquigarrow$ £242,000
- Mean CF TE, Labour: 0.041 $\rightsquigarrow$ £10,000

(mix of medians/means here ...)

Example: Causal Forests Results, Party

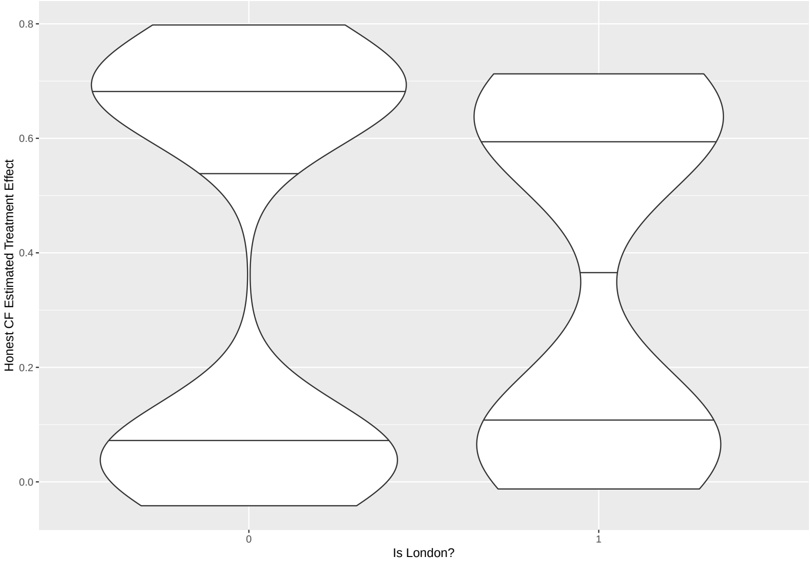
```r
average_treatment_effect(
  cf_out,
  subset = X$is_labour == 0)
```

```
 estimate   std.err
0.8530553 0.1983866
```

```r
average_treatment_effect(
  cf_out,
  subset = X$is_labour == 1)
```

```
  estimate    std.err
-0.1665371  0.1828122
```

Example: Causal Forests Results, London



Example: Causal Forests Results, London

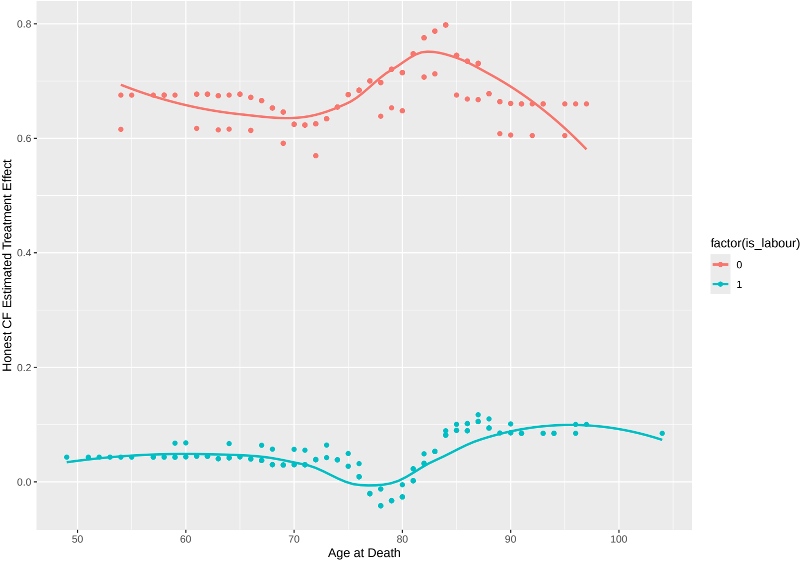
```
average_treatment_effect(
  cf_out,
  subset = X[, "is_london"] == 1)
```

```
 estimate   std.err
0.2454707 0.3964525
```

```
average_treatment_effect(
  cf_out,
  subset = X[, "is_london"] == 0)
```

```
 estimate   std.err
0.3847111 0.1469377
```

Example: Causal Forests Results, Age



Some Next Ideas ...

- Feature Selection
- Regularization/Shrinkage
  (LASSO, ridge, elastic net)
- Double LASSO for treatment effects
  (models for treatment and outcome)

Conclusions

The Big Picture

	Treatment Assignment	
Unit Selection	Randomized	Not Randomized
Randomized	Randomized Experiment (gold standard)	Survey Sampling (allows population inference)
Not Randomized	Controlled Experiment (allows causal inference)	Observational Study (large potential for bias)

Final Thought on Importance of Comparison Groups (Tufte 1974)

Final Thought on Importance of Comparison Groups (Tufte 1974)

One day when I was a junior medical student, a very important Boston surgeon visited the school and delivered a great treatise on a large number of patients who had undergone successful operations for vascular reconstruction. At the end of the lecture, a young student at the back of the room timidly asked, "Do you have any controls?" Well, the great surgeon drew himself up to his full height, hit the desk, and said, "Do you mean did I not operate on half of the patients?" The hall grew very quiet then. The voice at the back of the room very hesitantly replied, "Yes, that's what I had in mind." Then the visitor's fist really came down as he thundered, "Of course not. That would have doomed half of them to their death." God, it was quiet then, and one could scarcely hear the small voice ask, "Which half?"[4]

Thank you.

Thank you.

Your engagement, your ideas, your questions, your participation, your good nature, your stamina (800 minutes!), and your hard work have been a joy to share.

It has been a great honor to teach you this week.

Thank you.

Your engagement, your ideas, your questions, your participation, your good nature, your stamina (800 minutes!), and your hard work have been a joy to share.

It has been a great honor to teach you this week.

Stay in touch.

rtm@american.edu
www.ryantmoore.org

References I

Breiman, Leo. 2001. "Statistical Modeling: The Two Cultures." *Statistical Science* 16 (3): 199–215. http://www.jstor.org/stable/2676681.

Eggers, Andrew C., and Jens Hainmueller. 2009. "MPs for Sale? Returns to Office in Postwar British Politics." *American Political Science Review* 103 (4): 513–33.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2021. *An Introduction to Statistical Learning with Applications in R.* 2nd ed. New York, NY: Springer.