ECON 293/MGTECON 634:
Machine Learning and Causal Inference

Stefan Wager
Stanford University

Lecture 4: Heterogeneous Treatment Effects
in Observational Studies

Today's focus is on general principles for estimating **heterogeneous treatment effects** in **observational studies**. Such problems come up in many contexts:

- **Personalized medicine:** Which form of cancer therapy is most appropriate for this specific patient?
- **Targeted advertising:** How should a search engine adapt its advertising based on browsing history?
- **Resource allocation:** Which restaurants should health inspectors focus on?

All these problems can be attacked without **machine learning**.

But machine learning may help if we want to make algorithmic treatment recommendations from a CT scan of your brain.

Outline

The previous week's lectures focused on specific methods for treatment heterogeneity in **randomized trials**, with a focus on **causal trees** and **forests**.

This week, we'll discuss robust methods for treatment heterogeneity in **observational studies**. In particular, we'll

1. Review the key difficulties that arise in treatment effect estimation, **regularization bias** and **confounding bias**.
2. Discuss **confounding-robust** methods for estimating **constant treatment effects**.
3. Build on this discussion to develop confounding-robust **causal forests**.

Next week, I'll present a general approach to confounding-robust treatment effect estimation via generic machine learning methods.

Potential outcomes

For a set of **independent and identically distributed** units $i = 1, ..., n$, we observe

- A **feature vector** $X_i \in \mathcal{X}$,
- A **response** $Y_i \in \mathbb{R}$, and
- A **treatment assignment** $W_i \in \{0, 1\}$.

We posit **potential outcomes** $Y_i(0)$ and $Y_i(1)$ corresponding to treatment levels $W_i = 0, 1$ respectively (Neyman, 1923; Rubin, 1974), such that we observe $Y_i = Y_i(W_i)$ (SUTVA).

In our earlier discussions, we've focused on robust estimation of the **average treatment effect** $\tau = \mathbb{E}[Y_i(1) - Y_i(0)]$.

Treatment heterogeneity

Today, our goal is to move beyond a single average, and understand how treatment effects vary across people.

The **individual treatment effect** for the i-th unit is

$$\Delta_i = Y_i(1) - Y_i(0).$$

The ultimate analysis of treatment heterogeneity would seek to pinpoint Δ_i for every unit i on its own.

But this is **fundamentally impossible**.

Treatment heterogeneity

In practice, the best we can hope to do is to collect some covariates X_i, and see how the treatment effect varies with them.

Specifically, we seek to estimate the **conditional average treatment effect** (CATE)

$$\tau(x) = \mathbb{E}\left[Y_i(1) - Y_i(0) \,\middle|\, X_i = x\right],$$

i.e., the average treatment effect among all units who share the same covariate values.

Our goal is to estimate $\tau(X)$ accurately in terms of **mean-squared error**, $\mathbb{E}\left[(\hat{\tau}(X) - \tau(X))^2\right]$.

Unconfoundedness

In order to identify causal effects, we assume **unconfoundedness** (Rosenbaum and Rubin, 1983)

$$\left[\{Y_i(0),\ Y_i(1)\} \perp\!\!\!\perp W_i\right] \quad \Big| \quad X_i.$$

Informally, this means that the treatment is as good as **random** conditionally on observed features X_i.

Under unconfoundedness, methods based on **matching** are consistent (although not necessarily efficient) for the average treatment effect.

Similarly, the CATE is still a (localized) average, so it can be identified under unconfoundedness.

Why estimating treatment heterogeneity is hard

Identification of the CATE relies on essentially the same story as identification for the ATE. But there's a lot more to good **estimation** of the CATE.

Consider, e.g., a study on the effect of fibrinogen administration on mortality for victims of hemorrhagic shock. Features X_i is everything recorded by the first responders. Then:

- Treatment effects are probably **weak**, in the sense that

$$\text{Var}\left[\tau(X)\right] \ll \text{Var}\left[\mathbb{E}\left[Y(0) \,\middle|\, X\right]\right].$$

- There may be selection on **measured features** X_i.
- There may be selection on **unmeasured features** (not today).
- There may be **interference** across units (not today).

Regularization Bias and Confounding Bias

Machine learning for HTE

Under **unconfoundedness**,

$$\left[\{Y_i(0),\, Y_i(1)\} \perp\!\!\!\perp W_i\right] \;\; \Big| \;\; X_i,$$

we can write the CATE function as

$$\begin{aligned}
\tau(x) &= \mathbb{E}\left[Y_i(1) - Y_i(0) \,\middle|\, X_i = x\right] \\
&= \mathbb{E}\left[Y_i(1) \,\middle|\, X_i = x,\, W_i = 1\right] - \mathbb{E}\left[Y_i(0) \,\middle|\, X_i = x,\, W_i = 0\right] \\
&= \mathbb{E}\left[Y_i \,\middle|\, X_i = x,\, W_i = 1\right] - \mathbb{E}\left[Y_i \,\middle|\, X_i = x,\, W_i = 0\right] \\
&= \mu_{(1)}(x) - \mu_{(0)}(x).
\end{aligned}$$

This representation is the starting point for several machine learning based HTE estimation strategies.

Machine learning for HTE

The simplest way to use machine learning for treatment effect estimation is via **black box prediction**. For example:

The T-Learner fits separate models on the treated and controls.

1. Learn $\hat{\mu}_{(0)}(x)$ by predicting Y_i from X_i on the subset of observations with $W_i = 0$.
2. Learn $\hat{\mu}_{(1)}(x)$ by predicting Y_i from X_i on the subset of observations with $W_i = 1$.
3. Report $\hat{\tau}(x) = \hat{\mu}_{(1)}(x) - \hat{\mu}_{(0)}(x)$.

The S-Learner fits a single model to all the data.

1. Learn $\hat{\mu}(z)$ by predicting Y_i from $Z_i := (X_i, W_i)$ on all the data.
2. Report $\hat{\tau}(x) = \hat{\mu}((x, 1)) - \hat{\mu}((x, 0))$.

Machine learning for HTE

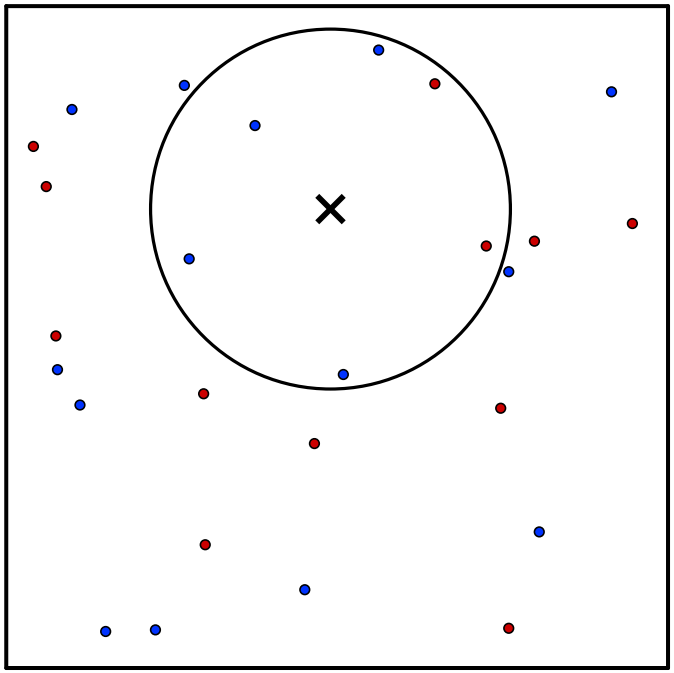
In order to use the S or T learner, one need to choose a **method for predicting** Y from X.

One could use the lasso, boosting, deep nets, etc. Today, we'll focus on random forests.

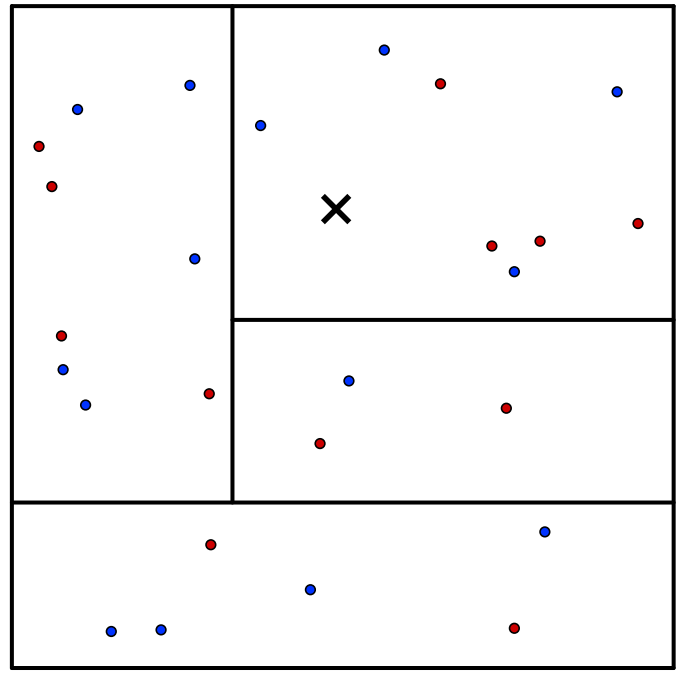
Over the next few slides, we'll briefly review random forests as a predictive method. Later today, we'll discuss how random forests can directly adapted to treatment heterogeneity (and not just used as a predictive black box).

Regression trees

k-NN neighborhood.



Tree-based neighborhood.



Decision trees are **adaptive nearest neighbor** predictors. Given pairs $(X_i,\ Y_i)$, a tree estimates $\mu(x) = \mathbb{E}\left[Y_i \,\middle|\, X_i = x\right]$ as

$$\hat{\mu}_{TREE}(x) = \text{avg}\left\{Y_i : X_i \in \text{leaf}(x)\right\}.$$

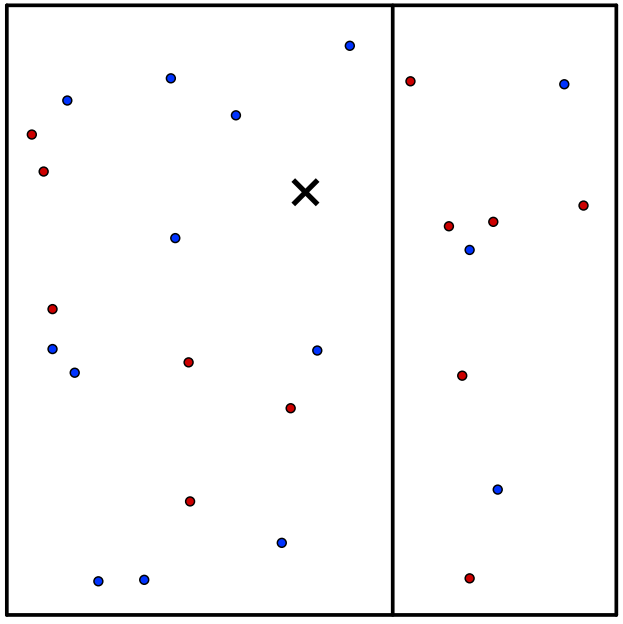
Random forests are like decision trees, except they smooth over all the sharp decision points.

Regression trees

Trees recursively apply a **greedy splitting criterion**.

In the **regression case**, the CART (Breiman et al., 1984) is standard.

- Compute $\hat{y}$ by averaging data in left/right leaf.
- Split minimizes $\sum_i (y_i - \hat{y}(X_i))^2$.
- Equivalently, pick a split to maximize the **weighted difference** $n_L n_R (\hat{y}_L - \hat{y}_R)^2$.



Trees and random forests (Breiman, 2001)

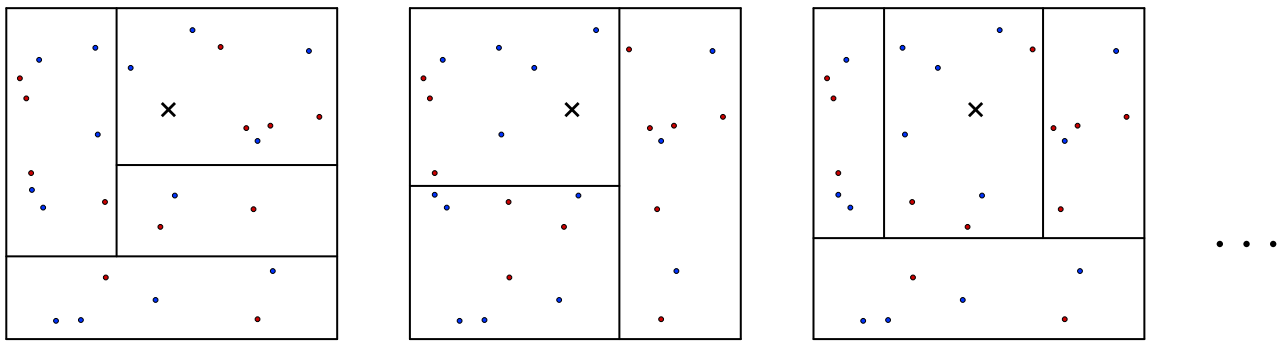
Suppose we have a training set $\{(X_i,\ Y_i)\}_{i=1}^n$, a test point x, and a tree predictor

$$\hat{\mu}\left(x\right) = T\left(x;\ \{(X_i,\ Y_i)\}_{i=1}^n\right).$$

Random forest idea: build and average many different trees T^*:

$$\hat{\mu}\left(x\right) = \frac{1}{B}\sum_{b=1}^{B} T_b^*\left(x;\ \{(X_i,\ Y_i)\}_{i=1}^n\right).$$

This is a simple idea, but improves performance considerably.



Trees and random forests (Breiman, 2001)

Suppose we have a training set $\{(X_i,\ Y_i)\}_{i=1}^n$, a test point x, and a tree predictor

$$\hat{\mu}\left(x\right) = T\left(x;\ \{(X_i,\ Y_i)\}_{i=1}^n\right).$$

Random forest idea: build and average many different trees T^*:

$$\hat{\mu}\left(x\right) = \frac{1}{B}\sum_{b=1}^{B} T_b^*\left(x;\ \{(X_i,\ Y_i)\}_{i=1}^n\right).$$

This is a simple idea, but improves performance considerably.

We turn T into T^* by:

- Bagging / subsampling the training set (Breiman, 1996); this helps smooth over discontinuities (Bühlmann and Yu, 2002).
- Selecting the splitting variable at each step from m out of p randomly drawn features (Amit and Geman, 1997).

Back to treatment heterogeneity...

The simplest way to use machine learning for treatment effect estimation is via **black box prediction**. For example:

The T-Learner fits separate models on the treated and controls.

1. Learn $\hat{\mu}_{(0)}(x)$ by predicting Y_i from X_i on the subset of observations with $W_i = 0$.
2. Learn $\hat{\mu}_{(1)}(x)$ by predicting Y_i from X_i on the subset of observations with $W_i = 1$.
3. Report $\hat{\tau}(x) = \hat{\mu}_{(1)}(x) - \hat{\mu}_{(0)}(x)$.

The S-Learner fits a single model to all the data.

1. Learn $\hat{\mu}(z)$ by predicting Y_i from $Z_i := (X_i, W_i)$ on all the data.
2. Report $\hat{\tau}(x) = \hat{\mu}((x, 1)) - \hat{\mu}((x, 0))$.

How robust are these methods to regularization bias?

Implementing the T-learner

```
tf0 = regression_forest(X[W==0,], Y[W==0])
tf1 = regression_forest(X[W==1,], Y[W==1])

tf.preds.0 = predict(tf0, X)$predictions
tf.preds.1 = predict(tf1, X)$predictions
tf.preds.0[W==0] = predict(tf0)$predictions  #OOB
tf.preds.1[W==1] = predict(tf1)$predictions  #OOB
preds.tf = tf.preds.1 - tf.preds.0
```

Implement the T-learner via a **random forest**:

1. Learn $\hat{\mu}_{(0)}(x)$ by predicting Y_i from X_i on the subset of observations with $W_i = 0$.
2. Learn $\hat{\mu}_{(1)}(x)$ by predicting Y_i from X_i on the subset of observations with $W_i = 1$.
3. Report $\hat{\tau}(x) = \hat{\mu}_{(1)}(x) - \hat{\mu}_{(0)}(x)$, with **out-of-bag** predictions when applicable.

Implementing the S-learner

```
sf = regression_forest(cbind(X, W), Y)

pred.sf.0 = predict(sf, cbind(X, 0))$predictions
pred.sf.1 = predict(sf, cbind(X, 1))$predictions

preds.sf.oob = predict(sf)$predictions
pred.sf.0[W==0] = preds.sf.oob[W==0]
pred.sf.1[W==1] = preds.sf.oob[W==1]

preds.sf = pred.sf.1 - pred.sf.0
```

Implement the S-learner via a **random forest**:

1. Learn $\hat{\mu}(z)$ by predicting Y_i from $Z_i := (X_i, W_i)$ on all the data, with **out-of-bag** predictions when applicable.
2. Report $\hat{\tau}(x) = \hat{\mu}((x, 1)) - \hat{\mu}((x, 0))$ on the test set.

Simulation Example: RCT

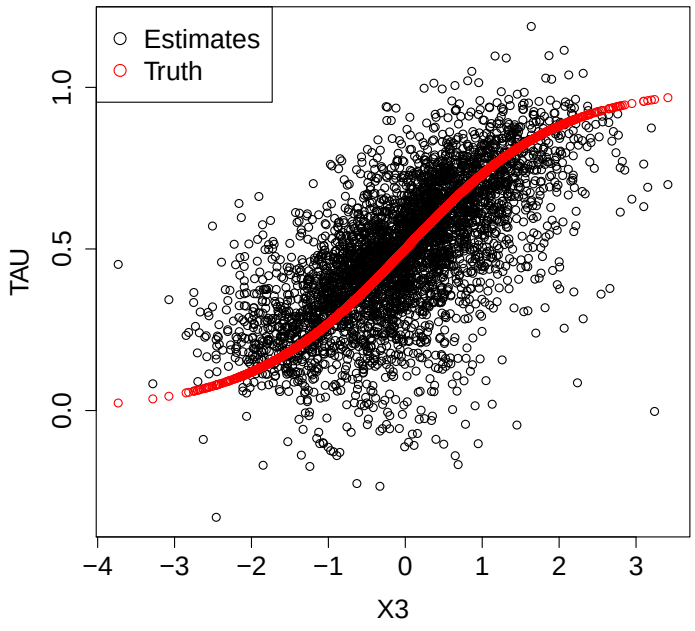
```
n = 4000; p = 10; treat.prob = 0.3
X = matrix(rnorm(n * p), n, p)
W = rbinom(n, 1, treat.prob)
TAU = 1/(1 + exp(-X[,3]))
Y = pmax(X[,1] + X[,2], 0) + W * TAU + rnorm(n)
```

Note in particular:

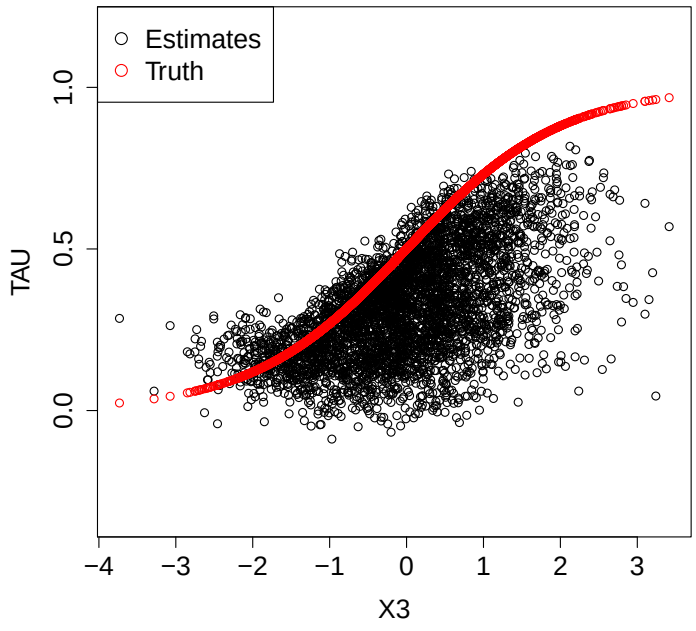
- This is a **randomized trial** with treatment fraction 0.3 (because treatment propensities don't depend on X).
- The treatment effect function is **simpler** than the main effect (which has interactions).

Simulation Example: RCT

T-forest



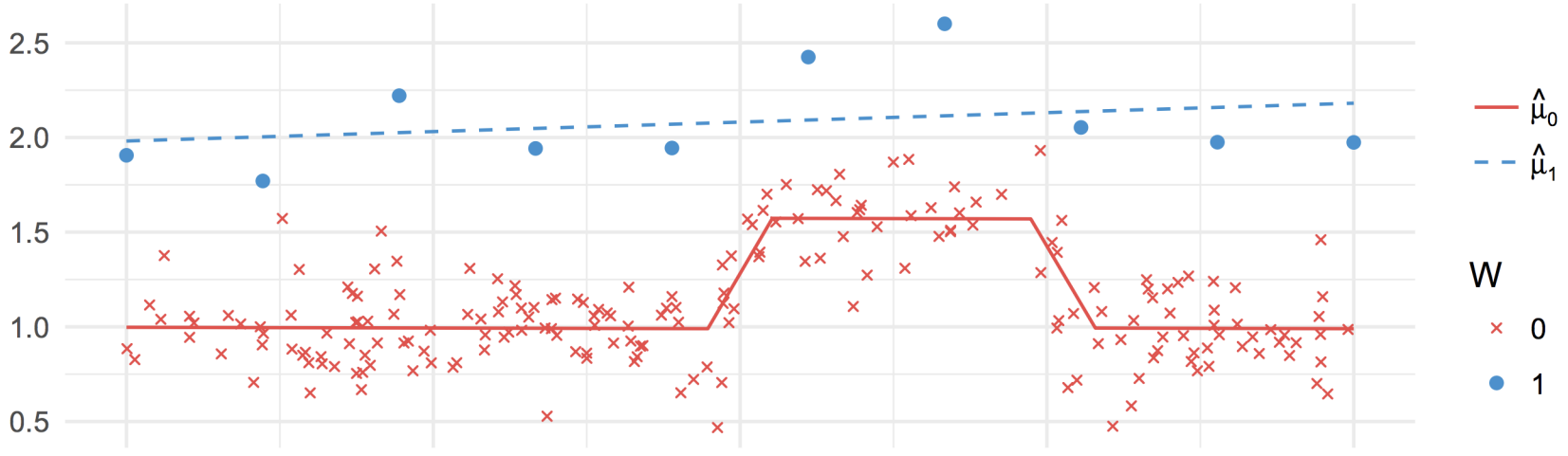
S-forest



The T- and S-learners have a hard time even approximating the treatment effect function.

- The T- and S-learners are only designed to make accurate **predictions**, not to estimate **treatment effects**.

Regularization Bias and the T-learner



[Figure: Künzel & al., 2019]

Machine learning for HTE

As a first improvement, Künzel et al. (2019) recently proposed a two-step method that **explicitly fits** the CATE function in an effort to **avoid regularization bias**.

The X-Learner imputes unobserved outcomes, and uses them to learn the HTE.

1. Learn $\hat{\mu}_{(0)}(x)$ by predicting Y_i from X_i on the subset of observations with $W_i = 0$.
2. Define $\Delta_i(1) = Y_i - \hat{\mu}_{(0)}(X_i)$, and learn $\hat{\tau}_{(1)}(x)$ by predicting $\Delta_i(1)$ from X_i on those observations with $W_i = 1$.
3. Learn $\hat{\tau}_{(0)}(x)$ by swapping the roles of treated/controls.
4. Learn $\hat{e}(x)$ by predicting W_i from X_i.
5. Report $\hat{\tau}(x) = \hat{e}(x)\hat{\tau}_{(0)}(x) + (1 - \hat{e}(x))\hat{\tau}_{(1)}(x)$.

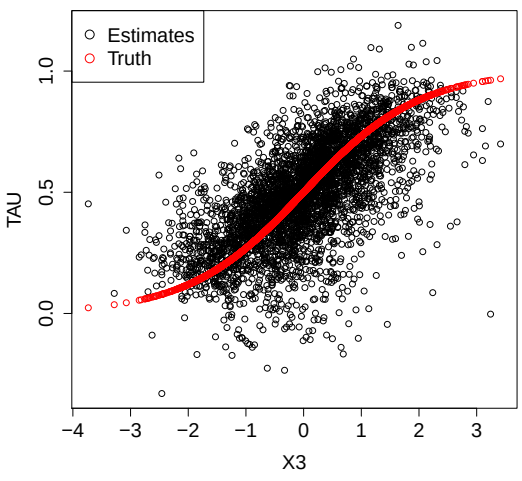
NB: The quantities Δ_i are used as noisy proxies for the individual treatment effect $Y_i(1) - Y_i(0)$.

Implementing the X-learner

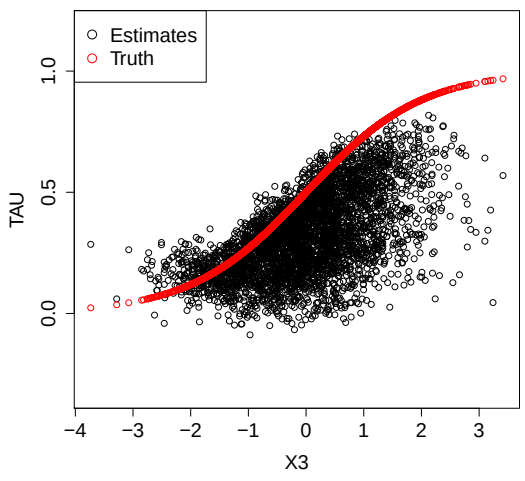
```
tf0 = regression_forest(X[W==0,], Y[W==0])
yhat0 = predict(tf0, X[W==1,])$predictions
xf1 = regression_forest(X[W==1,], Y[W==1]-yhat0)
xf.preds.1 = predict(xf1, X)$predictions
xf.preds.1[W==1] = predict(xf1)$predictions

tf1 = regression_forest(X[W==1,], Y[W==1])
yhat1 = predict(tf1, X[W==0,])$predictions
xf0 = regression_forest(X[W==0,], yhat1-Y[W==0])
xf.preds.0 = predict(xf0, X)$predictions
xf.preds.0[W==0] = predict(xf0)$predictions

propf = regression_forest(X, W, tune.parameters = TRUE)
ehat = predict(propf)$predictions
preds.xf = (1 - ehat) * xf.preds.1 + ehat * xf.preds.0
```

Simulation Example: RCT

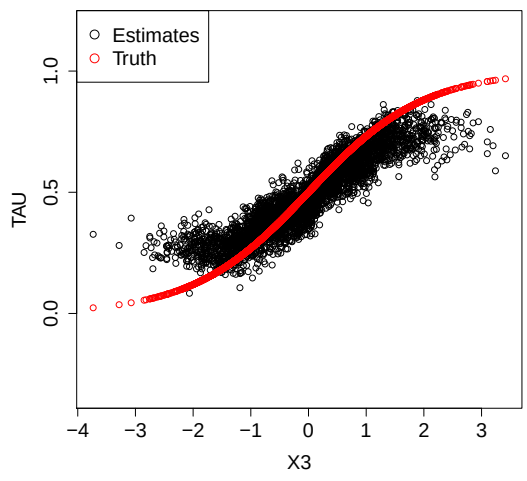
T-forest



S-forest

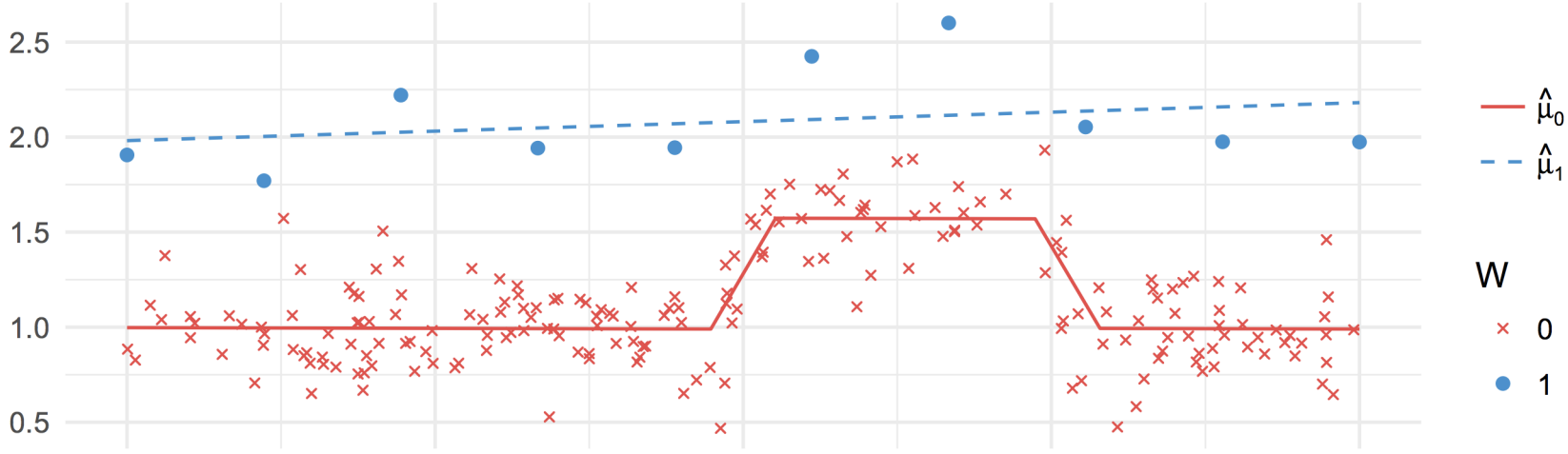


X-forest



In **randomized trials**, the *X*-construction can get at treatment effects directly.

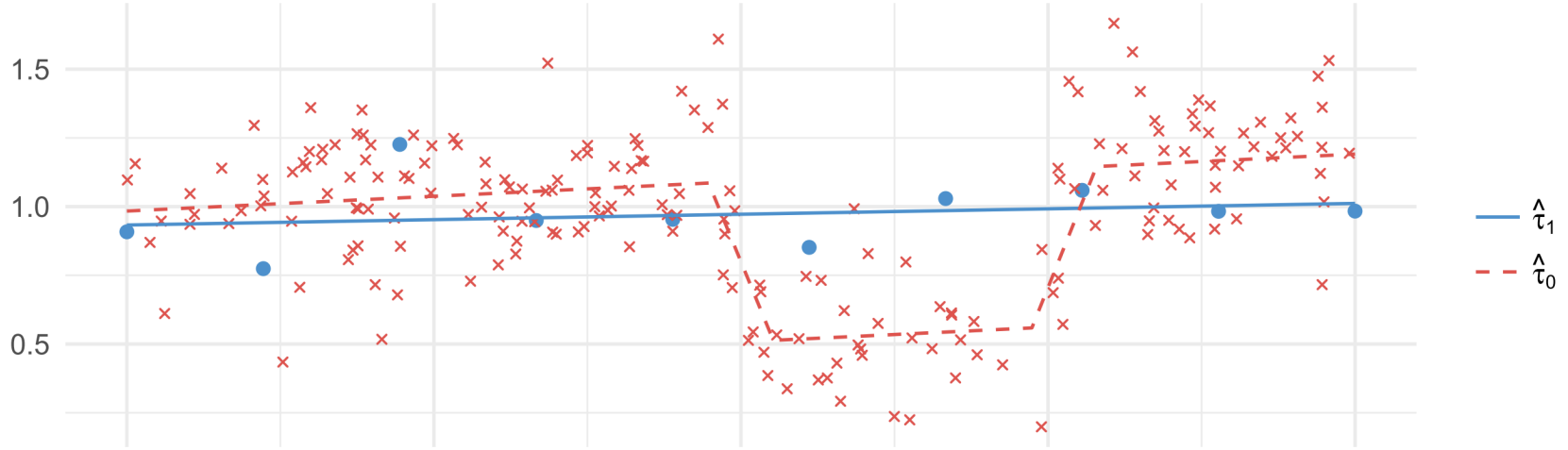
Intuition for the X-learner



Just like the T-learner, the X-learner starts by fitting **separate regressions** for $\hat{\mu}_{(0)}(x)$ and $\hat{\mu}_{(1)}(x)$.

However, due to **regularization bias**, the difference between these two functions may be a poor estimate of $\tau(x)$.

Intuition for the X-learner



In a second step, the X-learner forms differences between the **observed outcomes** and the predicted **counterfactual outcomes** with the other treatment, e.g., $\Delta_i(1) = Y_i - \hat{\mu}_{(0)}(X_i)$.

In this case, regressing $\Delta_i(1)$ on X_i for the treated samples gives a **good estimate** $\hat{\tau}_1(x)$ of the CATE.

The regression $\hat{\tau}_0(x)$ on $\Delta_i(0) = \hat{\mu}_{(1)}(X_i) - Y_i$ for the controls is less good, but is **down-weighted** by the X-learner.

Simulation Example: Not an RCT

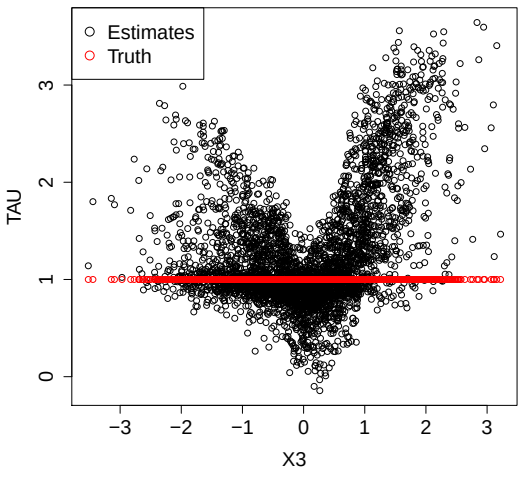
```
n = 4000; p = 10
X = matrix(rnorm(n * p), n, p)
W = rbinom(n, 1, 1 / (1 + exp(-X[,3])))
TAU = 1
Y = 2 * pmax(X[,1] + X[,2] + X[,3], 0) +
      W * TAU + rnorm(n)
```

Note in particular:

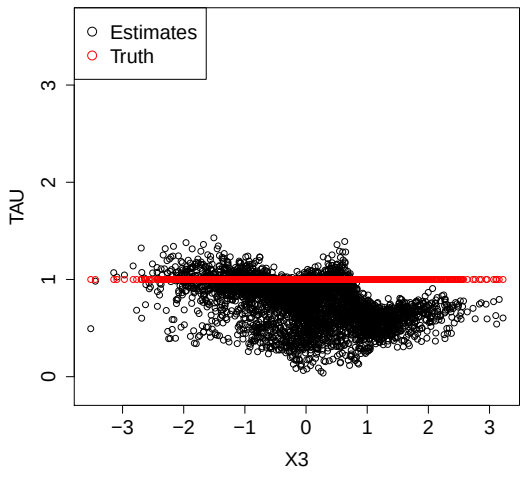
- This is **not** a randomized trial (because treatment propensities depend on X).
- The propensity function is **correlated** with the main effect.
- The treatment effect is **constant**.

Simulation Example: Not an RCT

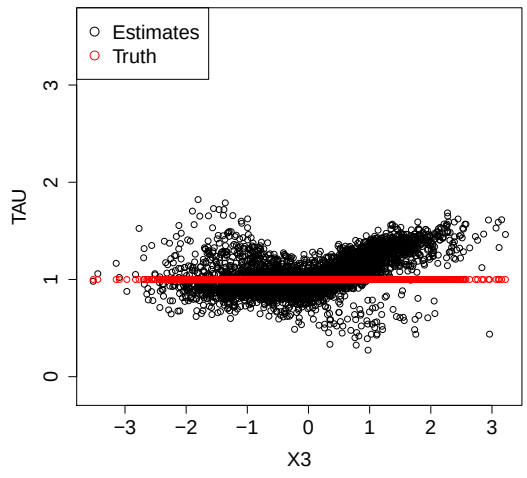
T-forest



S-forest



X-forest



All three methods, including the X-learner, **hallucinate treatment effects** here.

- This is because the baseline effect is correlated with the **propensity score**, and the $S/T/X$-learners don't robustly adjust for this.

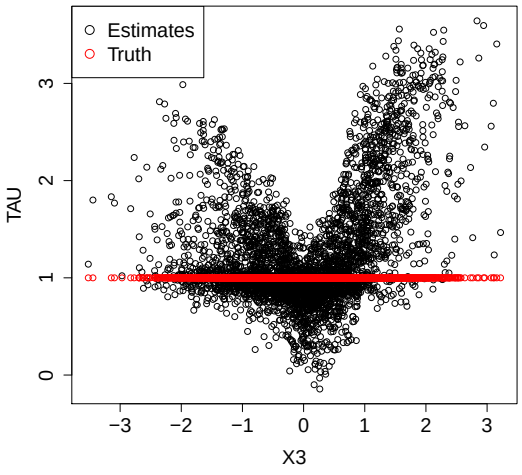
By directly fitting the individual pseudo-effects of the form $\Delta_i(1) = Y_i - \hat{\mu}_{(0)}(X_i)$, etc., the X-learner can avoid some of the **regularization bias** that plagues the direct methods.

- In **randomized trials**, this is enough to make the X-learner perform very well.
- However, in observational studies, we also need to be robust to **confounding**.

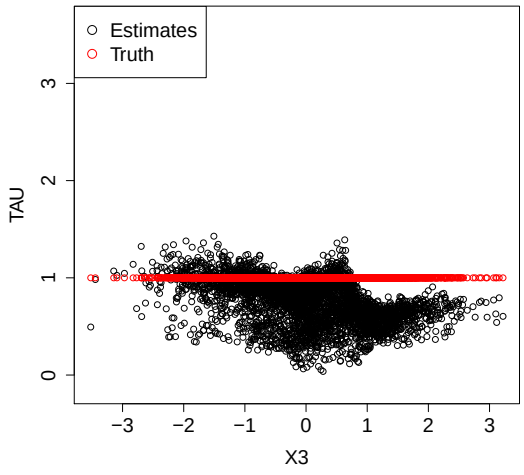
Our focus for the rest of this lecture: How can we design **robust CATE estimators** in a setting where we need to **control for confounders**?

Simulation Example: Not an RCT

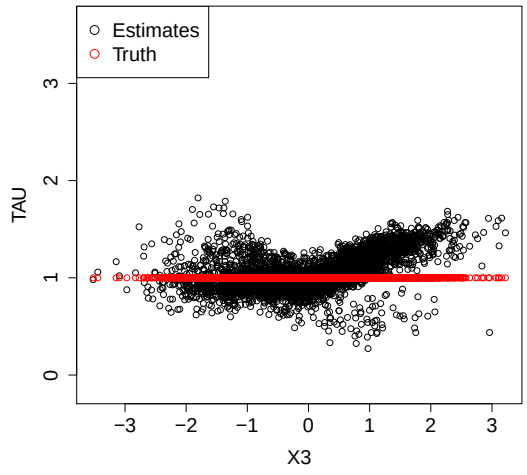
T-forest



S-forest



X-forest



None of the T-, S-, or X-learners use **propensity scores** to guide treatment effect estimation.

- Makes methods vulnerable to confounding outside of RCTs.
- The X-learner does use the propensity score, but only in a minor role (for aggregation).

How can we **leverage good propensity score estimates** for accurate heterogeneous treatment effect estimation?

By directly fitting the individual pseudo-effects of the form $\Delta_i(1) = Y_i - \hat{\mu}_{(0)}(X_i)$, etc., the X-learner can avoid some of the **regularization bias** that plagues the direct methods.

- In **randomized trials**, this is enough to make the X-learner perform very well.
- However, in observational studies, we also need to be robust to **confounding**.

Guiding question for the rest of this lecture: How can we use the **propensity score** for better heterogeneous treatment effect estimation?

Confounding-Robust Estimation of Constant Treatment Effects

Constant treatment effects

Suppose we assume a **constant treatment effect** τ, i.e.

$$Y_i(1) - Y_i(0) = \tau$$

for all units $i = 1, \ldots, n$. We want to estimate τ.

In terms of our earlier discussion, this implies

$$\tau = \tau(x) = \mathbb{E}\left[Y_i(1) - Y_i(0) \,\middle|\, X_i = x\right] \text{ for all } x \in \mathcal{X}.$$

This extra restriction means that this is not the same problem as estimating an **average treatment effect**, i.e., ATE $= \mathbb{E}\left[\tau(X)\right]$ for a potentially heterogeneous function $\tau(\cdot)$.

Constant treatment effects

Given **unconfoundedness**, i.e.,

$$\left[\left\{Y_i^{(0)},\, Y_i^{(1)}\right\} \perp\!\!\!\perp W_i\right] \;\; | \;\; X_i,$$

constant treatment effects imply a **partially linear** model

$$\mathbb{E}\left[Y \,\middle|\, X = x,\, W = w\right] = \mu_{(0)}(x) + w\tau.$$

Our goal now is to estimate τ.

NB: In this setting, the **difference in means** is still biased:

$$\begin{aligned}\mathbb{E}\left[Y_i \,\middle|\, W_i = 1\right] - \mathbb{E}\left[Y_i \,\middle|\, W_i = 0\right] \\ = \tau + \mathbb{E}\left[\mu_{(0)}(X_i) \,\middle|\, W_i = 1\right] - \mathbb{E}\left[\mu_{(0)}(X_i) \,\middle|\, W_i = 0\right].\end{aligned}$$

Robinson's transformation

Constant treatment effects imply a **partially linear** model

$$\mathbb{E}\left[Y \,\middle|\, X = x,\, W = w\right] = \mu_{(0)}(x) + w\tau.$$

This is a **semiparametric** problem, in that we have a low-dimensional parameter of interest τ, but have a non-parametric nuisance component $\mu_{(0)}(x)$.

Robinson (1988) proposed a simple approach to **estimation** in this model, based on the following observation. Defining

$$\begin{aligned} e(x) &= \mathbb{E}\left[W_i \,\middle|\, X_i = x\right], \quad \text{and} \\ m(x) &= \mathbb{E}\left[Y_i \,\middle|\, X_i = x\right] = \mu_{(0)}(x) + \tau e(x), \end{aligned}$$

we can **re-write** the partially linear model as

$$Y_i - m(X_i) = \tau(W_i - e(X_i)) + \varepsilon_i, \quad \mathbb{E}\left[\varepsilon_i \,\middle|\, X_i,\, W_i\right] = 0.$$

The Partially Linear Model

Robinson (1988) starts by re-writing the model in "centered" form:

$$
\begin{aligned}
e(x) &= \mathbb{E}\left[W_i \,\middle|\, X_i = x\right], \\
m(x) &= \mathbb{E}\left[Y_i \,\middle|\, X_i = x\right] = f(x) + \tau e(x), \\
Y_i - m(X_i) &= \tau(W_i - e(X_i)) + \varepsilon_i.
\end{aligned}
$$

This suggests the following **3-step approach** to estimation:

1. Get an estimate $\hat{e}(\cdot)$ by predicting W_i from X_i.
2. Get an estimate $\hat{m}(\cdot)$ by predicting Y_i from X_i.
3. Estimate τ by **residual-on-residual regression**:

$$
\hat{\tau} = \sum_{i=1}^{n} \left(Y_i - \hat{m}(X_i)\right)\left(W_i - \hat{e}(X_i)\right) \Big/ \sum_{i=1}^{n} \left(W_i - \hat{e}(X_i)\right)^2.
$$

The Partially Linear Model

This suggests the following **3-step approach** to estimation:

1. Get an estimate $\hat{e}(\cdot)$ by predicting W_i from X_i.
2. Get an estimate $\hat{m}(\cdot)$ by predicting Y_i from X_i.
3. Estimate τ by **residual-on-residual regression**:

$$\hat{\tau} = \sum_{i=1}^{n} \left(Y_i - \hat{m}(X_i)\right) \left(W_i - \hat{e}(X_i)\right) \Big/ \sum_{i=1}^{n} \left(W_i - \hat{e}(X_i)\right)^2.$$

Theorem. Under modest regularity conditions, and assuming that

$$\mathbb{E}\left[\left(m(X_i) - \hat{m}(X_i)\right)^2\right]^{\frac{1}{2}} \ll \frac{1}{n^{1/4}}, \quad \mathbb{E}\left[\left(e(X_i) - \hat{e}(X_i)\right)^2\right]^{\frac{1}{2}} \ll \frac{1}{n^{1/4}},$$

we get a **central limit theorem** $\sqrt{n}\left(\hat{\tau} - \tau\right) \Rightarrow \mathcal{N}\left(0,\, V\right)$.

The Partially Linear Model

This suggests the following **3-step approach** to estimation:

1. Get an estimate $\hat{e}(\cdot)$ by predicting W_i from X_i.
2. Get an estimate $\hat{m}(\cdot)$ by predicting Y_i from X_i.
3. Estimate τ by **residual-on-residual regression**:

$$\hat{\tau} = \sum_{i=1}^{n} \left(Y_i - \hat{m}(X_i)\right) \left(W_i - \hat{e}(X_i)\right) \Big/ \sum_{i=1}^{n} \left(W_i - \hat{e}(X_i)\right)^2.$$

The partially linear model offers a **practical approach** to $\sqrt{n}$-consistent estimation and **confidence intervals** for τ.

- All we need is an ability to estimate $m(\cdot)$ and $e(\cdot)$ at **reasonable rates**.
- Generic **machine learning** tools can be used for this task. **Cross-fitting** can be used to address any regularity concerns.

The Partially Linear Model

This suggests the following **3-step approach** to estimation:

1. Get an estimate $\hat{e}(\cdot)$ by predicting W_i from X_i.
2. Get an estimate $\hat{m}(\cdot)$ by predicting Y_i from X_i.
3. Estimate τ by **residual-on-residual regression**:

$$\hat{\tau} = \sum_{i=1}^{n} \left(Y_i - \hat{m}(X_i)\right) \left(W_i - \hat{e}(X_i)\right) \Big/ \sum_{i=1}^{n} \left(W_i - \hat{e}(X_i)\right)^2.$$

Linear regression is actually a **special case** of this. The following is an equivalent characterization of OLS:

1. Fit a linear model $W_i \sim X_i\,\gamma_e$.
2. Fit a linear model $Y_i \sim X_i\,\gamma_m$.
3. $\hat{\tau} = \sum_{i=1}^{n} \left(Y_i - X_i\hat{\gamma}_m\right) \left(W_i - X_i\hat{\gamma}_e\right) \Big/ \sum_{i=1}^{n} \left(W_i - X_i\hat{\gamma}_e\right)^2.$

We've now seen two "robust" estimators in this class:

- The **AIPW estimator** for the avg. treatment effect (week 2).
- The **residual-on-residual regression estimator** for a constant treatment effect (now).

Both estimators have the property that if **"nuisance components"** (i.e., non-focal parts of the problem) are estimated reasonably accurately, then we get $1/\sqrt{n}$-rate **central limit theorem** for our **target estimand**.

These are two special cases of a **general recipe** for robust, semiparamtric causal inference. For more, see:

Chernozhukov, V., Escanciano, J. C., Ichimura, H., Newey, W. K., & Robins, J. M. **Locally robust semiparametric estimation.** *arXiv*, 2016.

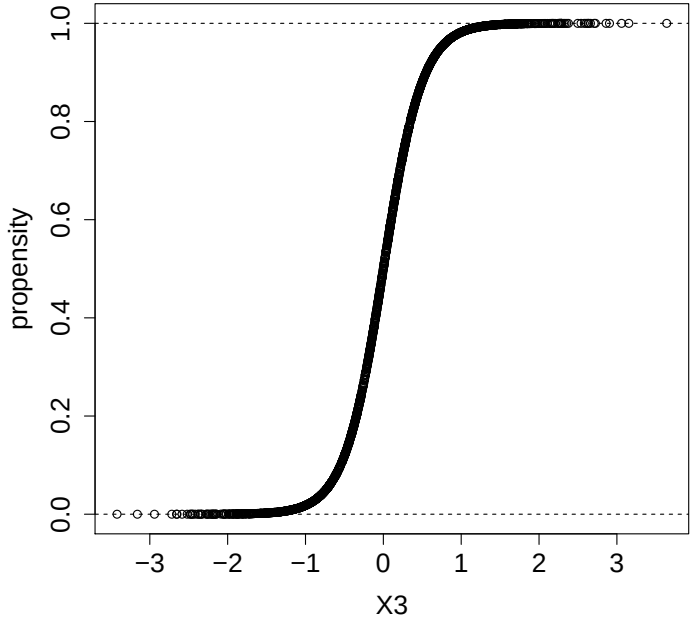
What's the difference?

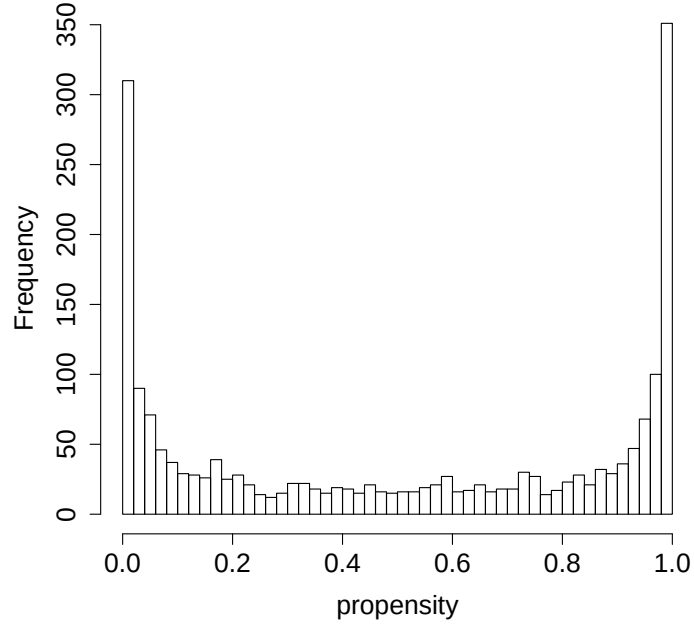
Estimating a **constant treatment effect** $\tau(X) = \tau$ is not the same problem as estimating an **average treatment effect**, i.e., $\text{ATE} = \mathbb{E}[\tau(X)]$ for a potentially heterogeneous function $\tau(\cdot)$.

- To estimate an **average effect** we need reasonably accurate estimates of $\tau(x)$ everywhere.
- To estimate a **constant effect** we can opportunistically focus on areas with the most signal.

Constant vs average treatment effects

```
n = 2000; p = 6; TAU = 0.3
X = matrix(rnorm(n * p), n, p)
pscore = 1 / (1 + exp(-4 * X[,3]))
W = rbinom(n, 1, pscore)
Y = log(1 + exp((X[,1] + X[,2]) / 3)) +
  TAU * W + rnorm(n)
```





Constant vs average treatment effects

```
rf.y = regression_forest(X, Y, tune.parameters = TRUE)
m.hat = predict(rf.y)$predictions
tY = Y - m.hat

lr.w = glm(W ~ X, family = binomial)
e.hat = predict(lr.w, type = "response")
tW = W - e.hat

ols.fit = lm(tY ~ tW)
tau.hat = coef(ols.fit)["tW"]
tau.se = sqrt(vcovHC(ols.fit)["tW", "tW"])
paste("95% CI:", round(tau.hat, 3),
      "+/-",  round(1.96 * tau.se, 3))
```

If we **know** that the treatment effect is constant, we can accurately estimate it, and get 95% CI for τ of $\mathbf{0.322 \pm 0.145}$.

Constant vs average treatment effects

```
rf.y = regression_forest(X, Y, tune.parameters = TRUE)
m.hat = predict(rf.y)$predictions
lr.w = glm(W ~ X, family = binomial)
e.hat = predict(lr.w, type = "response")

cf = causal_forest(X, Y, W, Y.hat = m.hat, W.hat = e.hat,
                   tune.parameters = TRUE)
ate.hat = average_treatment_effect(cf,
                                   target.sample = "all")
paste("95% CI:", round(ate.hat["estimate"], 3),
      "+/-",  round(1.96 * ate.hat["std.err"], 3))
```

If we **don't know** that the treatment effect is constant, it's harder to estimate it, and we get 95% CI for τ of $\mathbf{0.56 \pm 0.346}$.

- The `average_treatment_effect` function does **augmented inverse-propensity weighted** estimation (Lecture 2).

Constant vs average treatment effects

Here, we have 2 different choices:

- Assume a **constant effect** τ, in which case accurate estimation of τ is possible.
- Estimate an **average effect** $\mathbb{E}[\tau(X)]$ in a way that's robust to heterogeneity, at the cost of precision.

Constant vs average treatment effects

Here, we have 2 different choices:

- Assume a **constant effect** τ, in which case accurate estimation of τ is possible.
- Estimate an **average effect** $\mathbb{E}[\tau(X)]$ in a way that's robust to heterogeneity, at the cost of precision.

What about **Robinson's method** to get "$\hat{\tau}$", but **without assuming a constant effect** $\tau = \tau(x)$? In this case,

$$\sqrt{n}\left(\hat{\tau} - \tau_e\right) \Rightarrow \mathcal{N}\left(0,\, V\right), \quad \tau_e = \frac{\mathbb{E}\left[e(X)(1 - e(X))\tau(X)\right]}{\mathbb{E}\left[e(X)(1 - e(X))\right]}.$$

In other words, there are **two ways** to justify Robinson's method:

- **Assume** a constant effect.
- **Relax** the target of inference to τ_e.

In `grf`, the `average_treatment_effect` does Robinson's method if we set `target.sample = "overlap"`.

Confounding-Robust Forests for Treatment Heterogeneity

Causal forests

In the previous segment, we discussed how to optimally estimate a **constant treatment effect**.

Idea #1: Estimating $\tau(x)$ amounts to finding sub-regions where $\tau(x)$ is stable, and then estimating the CATE over them.

Idea #2: We can find regions that express the heterogeneity in $\tau(x)$ via trees & forests.

Neighborhood averaging

To get started, consit the **nearest neighbors** estimator for $\tau(x)$:

$$\hat{\tau}(x) = \frac{\sum_{\{i \in \mathcal{S}(x): W_i = 1\}} Y_i}{|\{i \in \mathcal{S}(x) : W_i = 1\}|} - \frac{\sum_{\{i \in \mathcal{S}(x): W_i = 0\}} Y_i}{|\{i \in \mathcal{S}(x) : W_i = 0\}|},$$

where $\mathcal{S}(x) = \{i : |X_i - x| \leq \delta_n\}$.

Problem #1: For this to be valid, we essentially need to assume that W_i is approximately randomized over $\mathcal{S}(x)$ (i.e., we can't have variation in the propensity score).

Problem #2: For this to work, we need δ_n to be small enough for treatment effects to be roughly constant over $\mathcal{S}(x)$, but also large enough to contain a non-negligible amount of data. This is difficult in even moderately high dimensions.

Neighborhood averaging

What if someone gives us a **neighborhood** $\mathcal{S}(x)$ and tells us that $\tau(\cdot)$ is constant over $\mathcal{S}(x)$, but that other quantities (e.g., the propensity score) may vary.

The resulting problem amounts to estimating a constant treatment effect over $\mathcal{S}(x)$, under unconfoundedness!

Why not use **Robinson's method** for it?

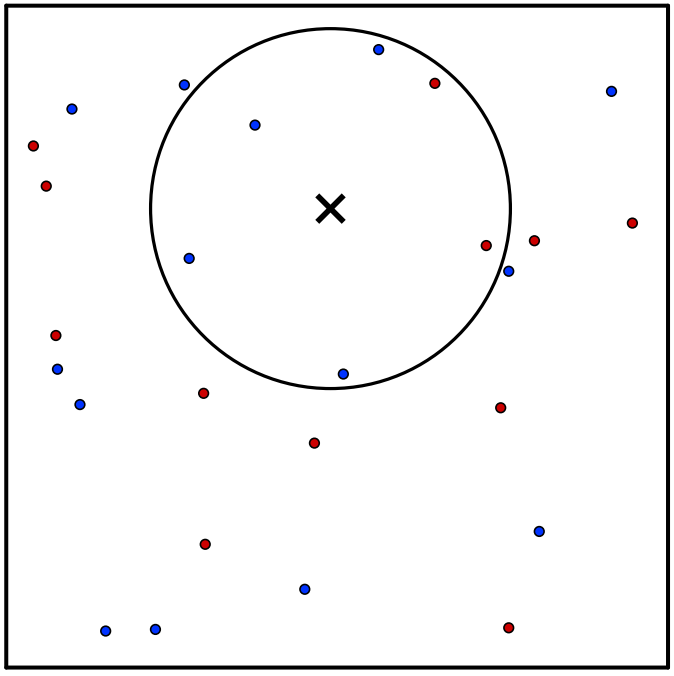
$$\hat{\tau}(x) \leftarrow \text{OLS}\left(\widetilde{Y}_i \sim \widetilde{W}_i,\ \texttt{subset:}\ X_i \in \mathcal{S}(x)\right),$$

where $\widetilde{W}_i = W_i - \hat{e}^{(-i)}(X_i)$, etc., rely on residualization.

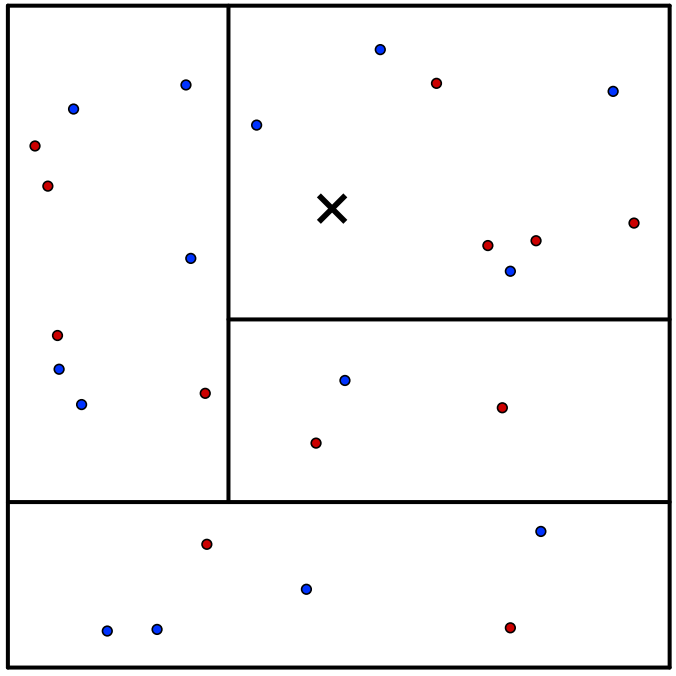
Our goal is to use a localized Robinson's method as our guide to estimation. Now we just need to modify **forests** to do so.

Trees and forests

k-NN neighborhood.



Tree-based neighborhood.



Trees and random forests (Breiman, 2001)

Suppose we have a training set $\{(X_i,\ Y_i)\}_{i=1}^n$, a test point x, and a tree predictor

$$\hat{\mu}\left(x\right) = T\left(x;\ \{(X_i,\ Y_i)\}_{i=1}^n\right).$$

Random forest idea: build and average many different trees T^*:

$$\hat{\mu}\left(x\right) = \frac{1}{B}\sum_{b=1}^{B} T_b^*\left(x;\ \{(X_i,\ Y_i)\}_{i=1}^n\right).$$

Challenges:

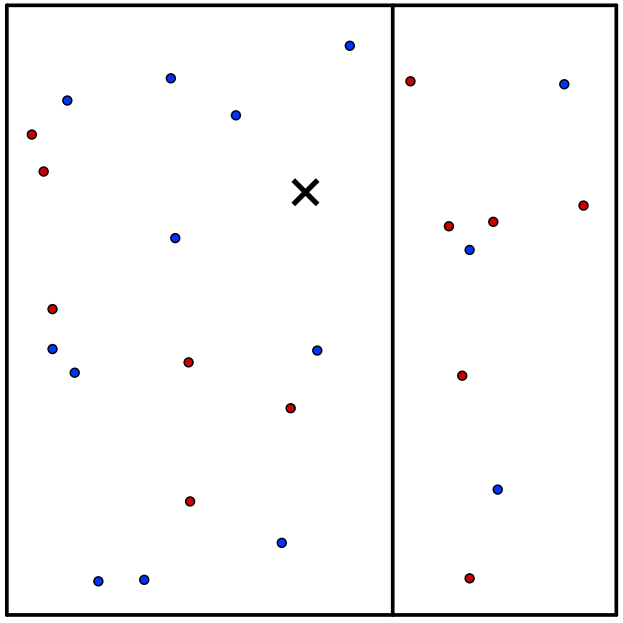
- How should we adapt splitting?
- How should we adapt prediction?

Regression tree splitting: Review

Trees recursively apply a **greedy splitting criterion**.

In the **regression case**, the CART (Breiman et al., 1984) is standard.

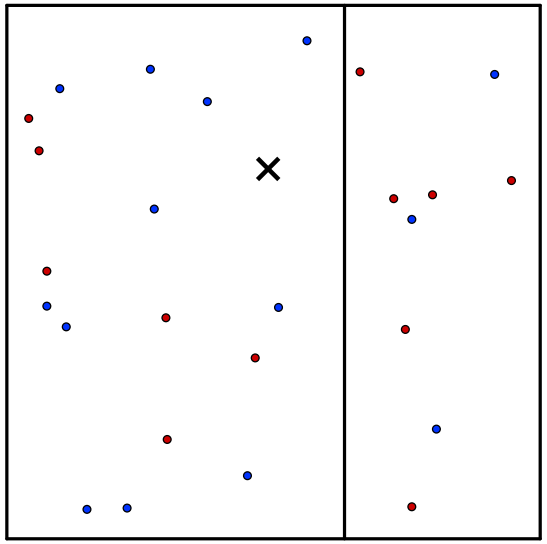
- Compute $\hat{y}$ by averaging data in left/right leaf.
- Split minimizes $\sum_i (y_i - \hat{y}(X_i))^2$.
- Equivalently, pick a split to maximize the **weighted difference** $n_L n_R (\hat{y}_L - \hat{y}_R)^2$.



Recursive partitioning for causal effects

How can we design a **splitting rule** that targets treatment heterogeneity?

As before, we seek to proceed **greedily**, and seek to maximize the amount of signal expressed in each split.



For each candidate "left-right" split (L, R), we do the following:

- Compute $\hat{\tau}_L$ and $\hat{\tau}_R$ **assuming homogeneous leaf-effects**:

$$\hat{\tau}_L \leftarrow \texttt{lm}\left(\left(Y_i - \hat{m}^{(-i)}(X_i)\right) \sim \left(W_i - \hat{e}^{(-i)}(X_i)\right) : X_i \in L\right).$$

- Split to maximize the **weighted difference** $n_L n_R (\hat{\tau}_L - \hat{\tau}_R)^2$.
- In the **regression case**, this is equivalent to CART.

Last week, discussed similar idea, but focused on RCT.

Aggregating causal estimates

For regression, natural to write a forest as an **average of trees**:

$$\hat{\mu}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b^*(x; \{(X_i, Y_i)\}_{i=1}^n).$$

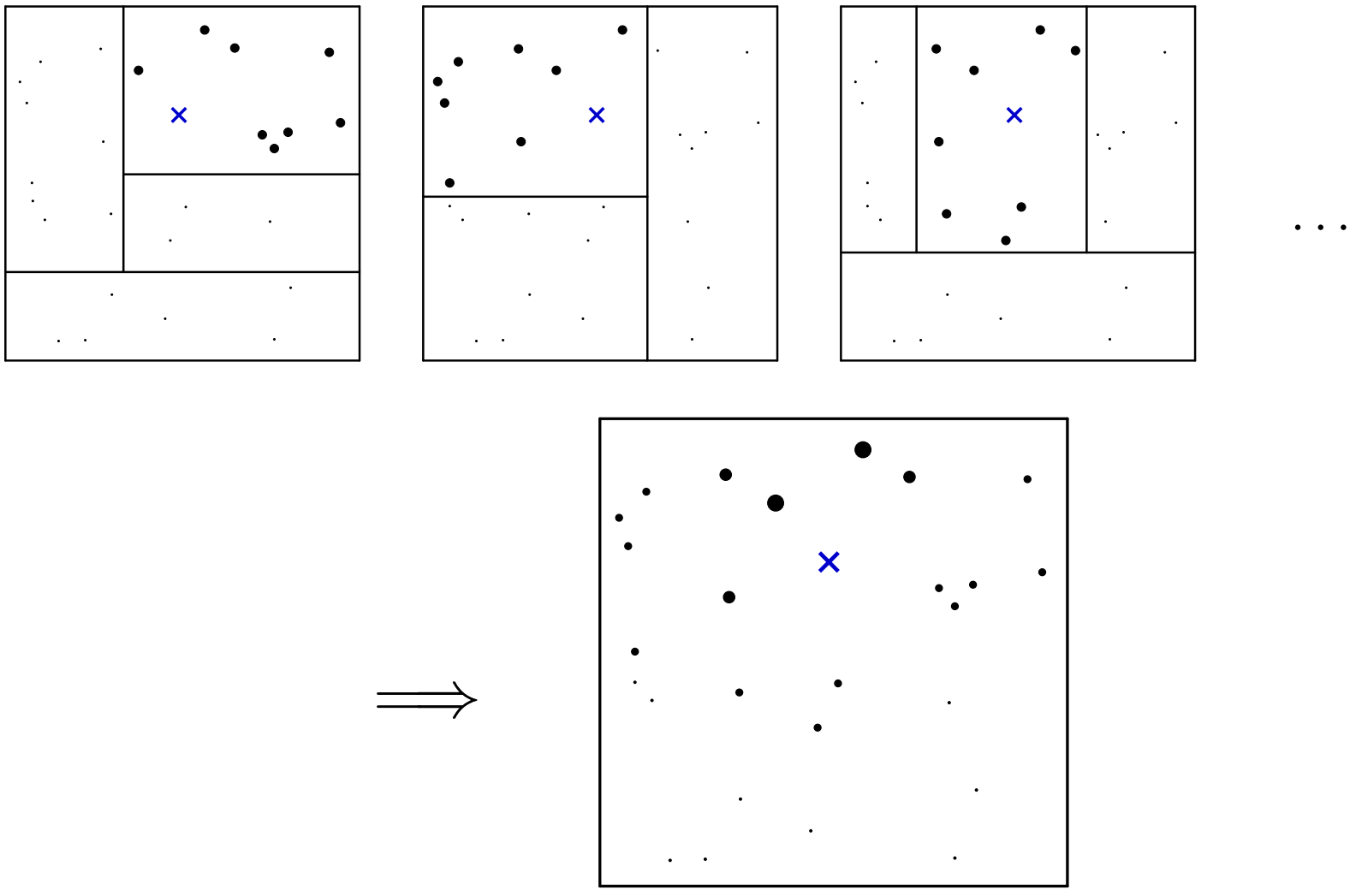
However, in causal forests, some leaves may be **highly variable**, and so averaging is undesirable.

A helpful alternative perspective is to view forests as weighting:

$$\hat{\mu}(x) = \frac{1}{B} \sum_{b=1}^{B} \sum_{i=1}^{n} Y_i \frac{1(Y_i \in L_b(x))}{|L_b(x)|} = \sum_{i=1}^{n} Y_i \underbrace{\frac{1}{B} \sum_{b=1}^{B} \frac{1(Y_i \in L_b(x))}{|L_b(x)|}}_{\alpha_i(x)}.$$

In other words, we understand random forests as a **data-adaptive "kernel"** with weights $\alpha_i(x)$.

The random forest kernel



Forests induce a kernel via **averaging tree-based neighborhoods**.

Aggregating causal estimates

Regression forests can also be understood as weighted estimators with a **forest kernel**,

$$\hat{\mu}(x) = \frac{1}{B}\sum_{b=1}^{B}\sum_{i=1}^{n} Y_i \frac{1\left(Y_i \in L_b(x)\right)}{|L_b(x)|} = \sum_{i=1}^{n} Y_i \underbrace{\frac{1}{B}\sum_{b=1}^{B} \frac{1\left(Y_i \in L_b(x)\right)}{|L_b(x)|}}_{\alpha_i(x)}.$$

This kernel-based approach naturally **extends** to the causal case. For a given test point x, we propose estimating $\tau(x)$ as follows:

$$\hat{\tau}(x) \leftarrow \texttt{lm}\Bigg(\left(Y_i - \hat{m}^{(-i)}(X_i)\right) \sim \left(W_i - \hat{e}^{(-i)}(X_i)\right), \\ \texttt{weights} = \alpha_i(x)\Bigg).$$

This is what's done in the (confounding robust) implementation of **causal forests in GRF**. The arguments `W.hat` and `Y.hat` in the API correspond to $\hat{e}^{(-i)}(X_i)$ and $\hat{m}^{(-i)}(X_i)$ in our notation.

Simulation Example: Not an RCT

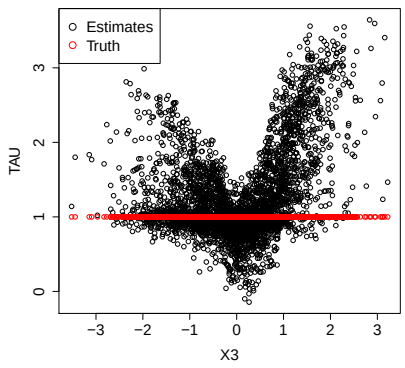
```
n = 4000; p = 10
X = matrix(rnorm(n * p), n, p)
W = rbinom(n, 1, 1 / (1 + exp(-X[,3])))
TAU = 1
Y = 2 * pmax(X[,1] + X[,2] + X[,3], 0) +
      W * TAU + rnorm(n)
```

Note in particular:

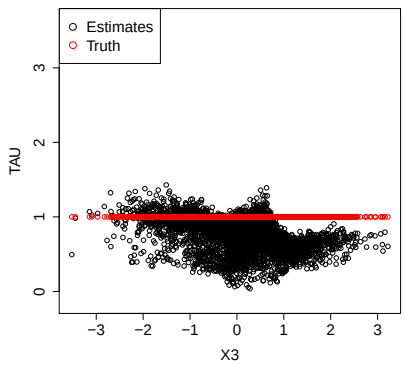
- This is **not** a randomized trial (because treatment propensities depend on X).
- The propensity function is **correlated** with the main effect.

Simulation example revisted: Not an RCT

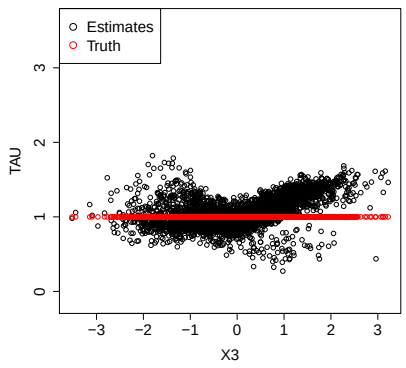
T-forest



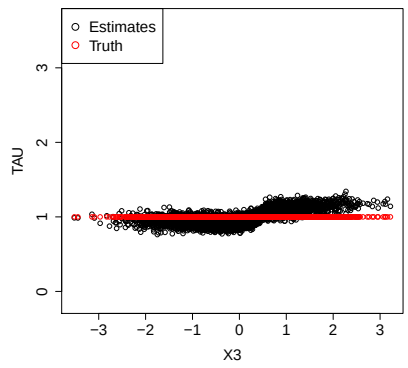
S-forest



X-forest



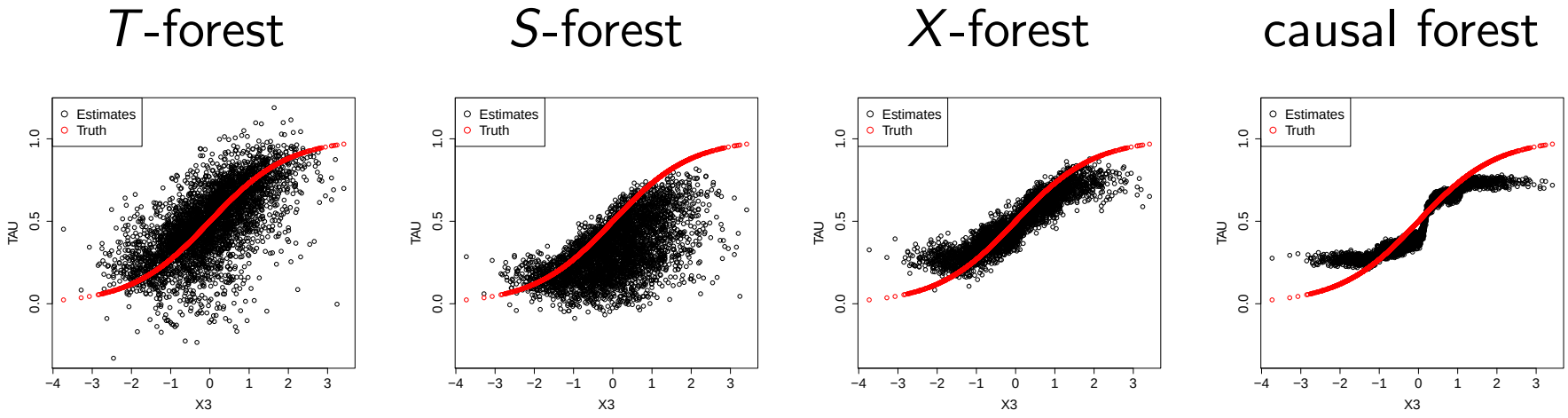
causal forest



The ability of a causal forest to rely on a propensity score fit helps accuracy outside of RCTs.

```
cf = causal_forest(X, Y, W, tune.parameters = TRUE)
preds.cf = predict(cf, X.test)$predictions
```

Simulation example revisted: RCT



In an RCT, both the X-forest and causal forest qualitatively fit the signal. Here, causal forest regularizes more aggressively, which helps slightly in RMSE $\sqrt{\mathbb{E}\left[(\hat{\tau}(X) - \tau(X))^2\right]}$.

	T-forest	S-forest	X-forest	causal forest
RMSE	0.173	0.246	0.087	0.074

Next week, we'll continue for here, and discuss:

- Confounding-robust loss functions for machine learning.
- Methods for evaluating CATE estimates.
- Applications.

References

S/T/X learners & regularization bias

- Künzel, Sören R., Jasjeet S. Sekhon, Peter J. Bickel, and Bin Yu. **Metalearners for estimating heterogeneous treatment effects using machine learning.** *Proceedings of the National Academy of Sciences*, 116(10), 2019.
- Hahn, P. Richard, Jared S. Murray, and Carlos M. Carvalho. **Bayesian regression tree models for causal inference: Regularization, confounding, and heterogeneous effects.** *Bayesian Analysis*, 15(3), 2020.

Causal forests in observational studies

- Athey, Susan and Stefan Wager. **Estimating treatment effects with causal forests: An application.** *Observational Studies*, 5, 2019.
- Athey, Susan, Julie Tibshirani, and Stefan Wager. **Generalized random forests.** *Annals of Statistics*, 47(2), 2019.