Experimental Design & A/B Testing

MKTG 6279

Week 1

This course

- Basics of causal inference
- Customer segmentation
- Feature selection
- Recommendation engines

This week

- Experiments
- A/B testing
- Causal forests
- Power tests

Experiments

What is an experiment?

- Take sample from a population of subjects
- Randomly assign subjects to either treatment or control
- Measure average outcome for each subjects
- Compare outcomes between treatment and control groups

This procedure allows you to estimate the causal effect of the **treatment** on some **outcome**

Why experiments?

- Causation vs. correlation: experiments help establish cause-and-effect relationships
- Reduce risk: test changes on a small scale before rolling them out to everyone
- Optimize ROI: identify what works best and allocate resources effectively

Example: testing different ad creatives to see which generates the most clicks

Randomized Control Trials (RCTs): The Gold Standard

Randomization: participants are randomly assigned to one of the treatment groups or a control group

- Control Group: receives the existing treatment or no treatment
- Treatment Group: receives the new treatment or intervention
- Minimizes bias (incorrect estimate) and allows for *causal* inference

Potential Outcomes in Causal Inference

Each subject has two potential outcomes in the experiment

- $Y_i(1)$: outcome from being treated
- $Y_i(0)$: outcome from not being treated

The individual treatment effect is the difference between these two outcomes:

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

Unfortunately, we can only observe one potential outcome for each subject, so we can never know what τ_i is for each subject

Average Treatment Effects

Since we can't observe these individual effects, we start with the **Average Treatment Effect** (ATE)

$$\tau = \mathbb{E}_i[Y_i(1) - Y_i(0)]$$

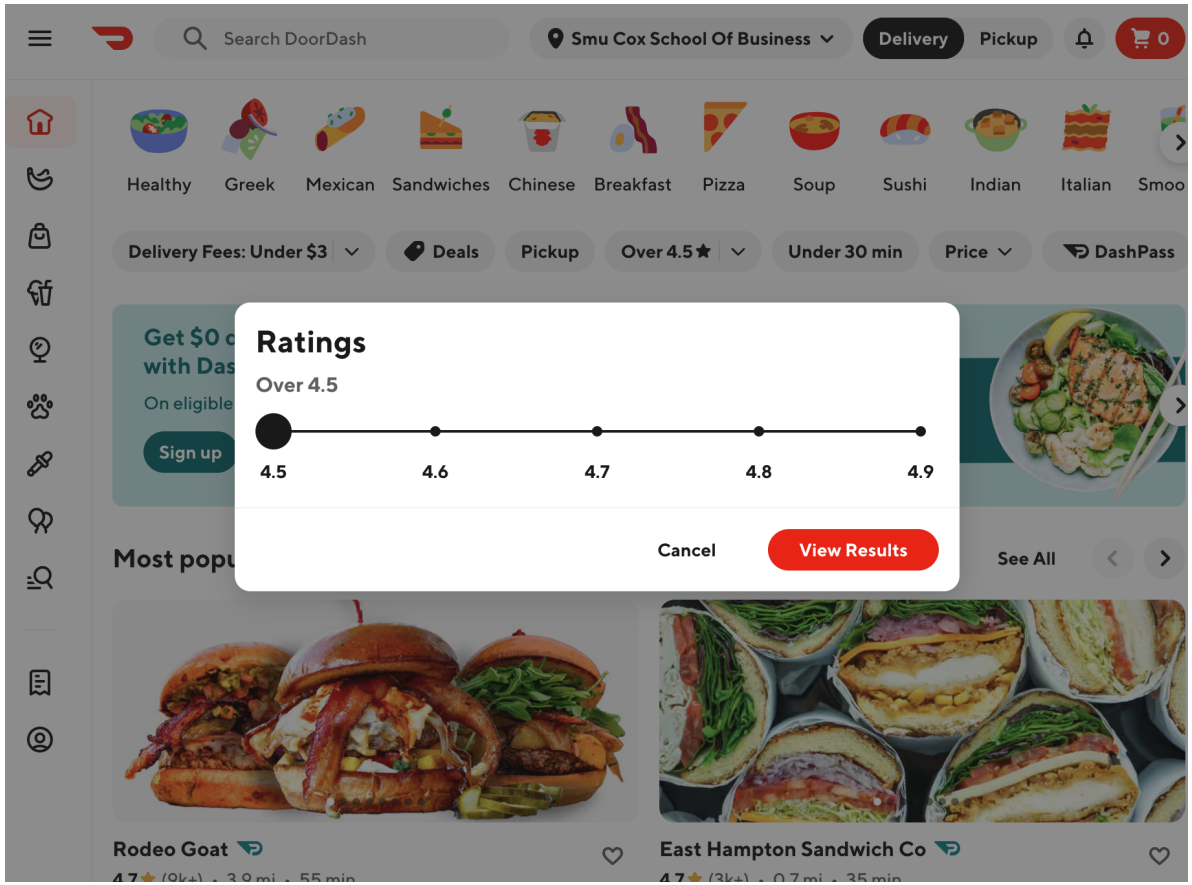
which we estimate as the difference-in-means

$$\widehat{\tau} = \frac{1}{n_1}\sum Y_i(1) - \frac{1}{n_0}\sum Y_i(0)$$

where n_1 and n_0 are the number of treated and control subjects.

Simple Example: DoorDash Ratings Scale

DoorDash changes the ratings minimum from 4.5 to 4



Why might DoorDash test this?

A lower minimum range expands the range of options

But, this increases search costs and can lead to abandoned carts since "worse" restaurants are included in the results list

Use a RCT to see what happens. . .

Setup

Control: current design (4.5+)

Treatment: new, lower range (4+)

Outcome: time on site (in seconds)

Other potential outcomes: purchase, amount spent, rate of cart abandonment, etc.

- The outcome selected depends on who cares about these results and what the goal of the analysis is

Doordash Data

```
load('data/doordash.rdata')
str(doordash)
```

```
## 'data.frame':    1000 obs. of  7 variables:
##  $ ip          : num  10001 10002 10003 10004 10005 ...
##  $ age         : num  59 63 59 39 27 22 45 21 43 40 ...
##  $ mthsActive: num  19.26 11.51 4.26 9.53 19.54 ...
##  $ treatment : num  0 0 1 1 0 1 1 1 1 0 ...
##  $ mobile    : num  1 1 0 1 0 0 1 1 1 1 ...
##  $ seconds   : num  116.3 146.6 159.8 120.9 40.3 ...
##  $ purchase  : int  0 0 1 0 0 0 1 0 0 0 ...
```

ATE

```
ate = doordash %>%
  group_by(treatment) %>%
  summarise(seconds_bar = mean(seconds))
ate
```

```
## # A tibble: 2 x 2
##   treatment seconds_bar
##       <dbl>       <dbl>
## 1         0        99.3
## 2         1       105.
```

So the ATE is about 6.1 seconds.

Another way...

```
with(doordash,
     mean(seconds[treatment == 1]) -
     mean(seconds[treatment == 0]))
```

```
## [1] 6.051932
```

Is this significantly different from zero?

Use a t-test

Testing the difference in means (i.e., average seconds on the site) between the **treated** and **control** groups

```
t.test(seconds ~ treatment,doordash)
```

```
##
##  Welch Two Sample t-test
##
## data:  seconds by treatment
## t = -2.6885, df = 982.28, p-value = 0.0073
## alternative hypothesis: true difference in means betwee
## 95 percent confidence interval:
##  -10.469417  -1.634448
## sample estimates:
## mean in group 0 mean in group 1
##        99.25854       105.31048
```

Or use regression...

Slightly different 95% CI on treatment, but this is splitting hairs...

```
r1 = lm(seconds ~ treatment,doordash)
tidy(r1) %>% select(1,2,5) %>%
  mutate(across(-1, ~round(., 3)))
```

```
## # A tibble: 2 x 3
##   term        estimate p.value
##   <chr>          <dbl>   <dbl>
## 1 (Intercept)    99.3    0
## 2 treatment       6.05   0.007
```

```
confint(r1)
```

```
##                   2.5 %    97.5 %
## (Intercept) 96.203769 102.31332
## treatment    1.642745  10.46112
```

Why use experiments?

The difference-in-means $\widehat{\tau}$ is an **unbiased** estimate of the average treatment effect (ATE)

- If we only gave mobile users the treatment and desktop users the control, our estimate may be biased because it captures both the treatment effect *and* the effect of the device type

In other words, randomization eliminates the possibility of **confounders**

SUTVA

An assumption is that the outcomes for one subject do not affect the outcomes for other subjects

This is called the **Stable Unit Treatment Value Assumption** (SUTVA)

- Usually a problem when subjects might share their experiences with others (e.g., if other subjects become aware that a test is occurring)

Aside: random sampling

The average treatment effect averages over the **population in the experiment**.

The ideal experiment randomly samples from the target population.

- Trivial in engineering experiments (e.g., Amazon, DoorDash, etc.)
- Do-able in marketing experiments
- Unethical in clinical trials

So, there are two types of randomization in the ideal experiment: **random sampling** to select subjects and **random assignment** to treatments

In later classes we talk about about how to deal with violations of these.

Wait, what if the treatment effect depends on...

- City vs rural individuals?
- Mobile vs desktop users?
- DashPass member status?
- Age?
- Tenure with DoorDash?
- Anything else...

Enter HTE and CATE

When treatment effects vary by subgroup (even at the individual level) we have **heterogeneous treatment effects** (HTEs)

We allow for HTEs by estimating **conditional average treatment effects** (CATEs):

$$\mathbb{E}[\tau_i|X] = \mathbb{E}[Y_i(1) - Y_i(0)|X_i]$$

"The estimated treatment effect *conditional on* some value X_i"

DoorDash CATE

Does the treatment effect *depend on* whether this was a mobile (vs desktop) user?

```
r2=lm(seconds ~ treatment*mobile,doordash)
tidy(r2) %>% select(1,2,5) %>%
  mutate(across(-1, ~round(., 3)))
```

```
## # A tibble: 4 x 3
##   term             estimate p.value
##   <chr>               <dbl>   <dbl>
## 1 (Intercept)          94.4   0
## 2 treatment             3.73  0.226
## 3 mobile               10.1   0.001
## 4 treatment:mobile      4.72  0.287
```

Confidence intervals

```
confint(r2)
```

```
##                       2.5 %    97.5 %
## (Intercept)       90.246986 98.583938
## treatment         -2.304148  9.756347
## mobile             4.090144 16.137987
## treatment:mobile -3.975074 13.411416
```

DoorDash CATE

But make sure to control for potentially confounding variables for more precise estimates...

```
r3=lm(seconds ~ treatment*mobile + age + mthsActive,
      doordash)
coef(r3)
```

```
##      (Intercept)        treatment           mobile
##        45.215676         7.897196        11.860501
##       mthsActive treatment:mobile
##        -2.996878         6.104693
```

More precise CIs

```
confint(r3)
```

```
##                       2.5 %    97.5 %
## (Intercept)       44.961696 45.469656
## treatment          7.719270  8.075123
## mobile            11.682812 12.038191
## age                1.991776  2.001127
## mthsActive        -3.005754 -2.988002
## treatment:mobile   5.848315  6.361070
```

Use Causal Forests for more flexible mappings...

Rather than use linear regression to define subgroups, use random forests to accommodate HTEs

Instead of predicting outcomes, predict treatment effects

- *Random* forests split data to minimize prediction errors
- *Causal* forests split data to maximize the differences in estimated treatment effects between subgroups

"Honest" tree construction

In a causal forest, data is usually split into separate subsets:

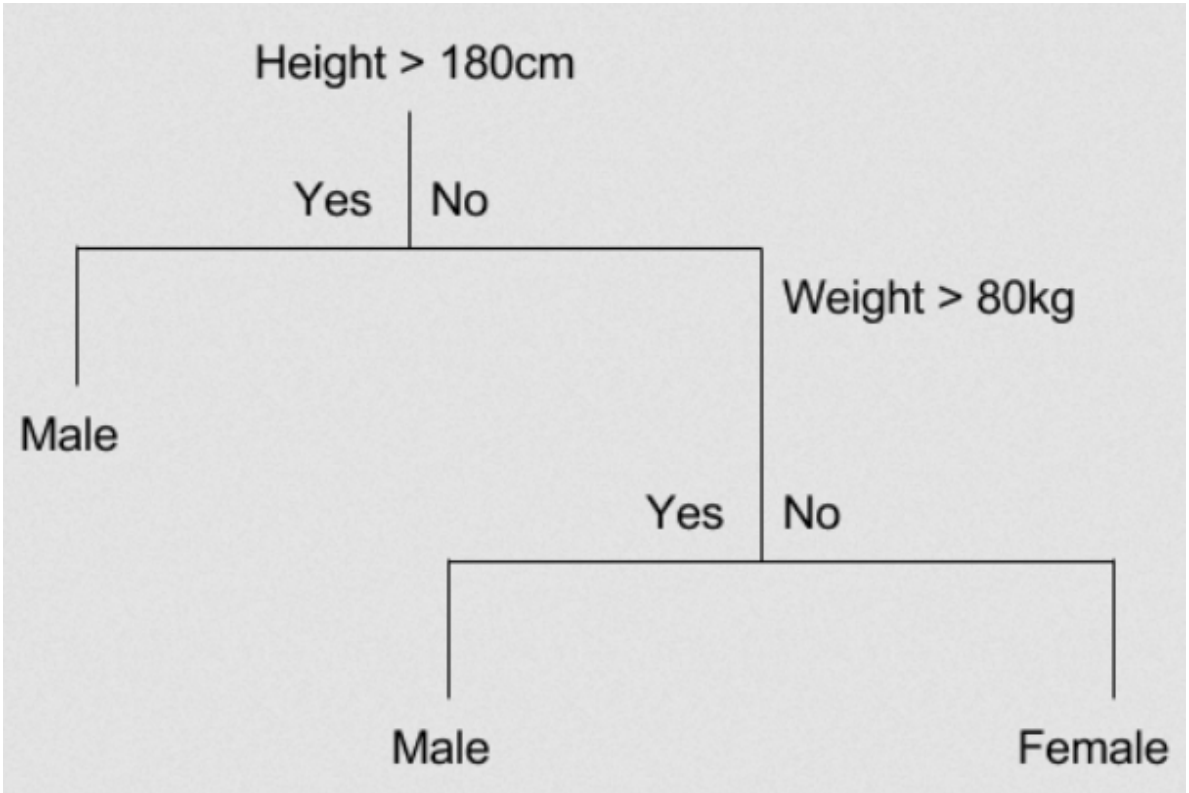
1) One for building the tree structure
2) Another for estimating treatment effects within the resulting nodes

Also known as "split-sample" or "double-sample" estimation

Driven by CART

A **classification and regression tree (CART)** is the building block for causal forests

These partition the data into subsets based on feature values



Classification tree for categorical outcomes

Example

- Predicting churn (yes/no)

Splitting criteria

- Gini Impurity: probability of incorrectly classifying a randomly chosen element
- Entropy: disorder or randomness in a set of data
- Minimize impurity in the child nodes
- Intuition: a group of {A,A,A} has less impurity in the final node versus {A,B,C}, so create branches and nodes accordingly

Prediction

- Most common value within a node

Regression tree for continuous outcomes

Example

- Predicting house prices

Splitting criteria

- Minimize the sum of squared residuals (differences between predicted and actual values) within the child nodes
- In other words, create groups of similar values

Prediction

- Average value of outcomes within a node

Simple regression tree in DoorDash

We can fit a random forest using the `grf` package

`grf` stands for "Generalized Random Forests"

```
library(grf)
X = doordash %>% select(mobile,age,mthsActive)
Y = doordash$seconds

rf   = regression_forest(X, Y, num.trees = 1)
tree = get_tree(rf, 1)
#plot(tree)
```

Random forests

A **random forest** is a collection of trees estimated from different sub-samples of the data

The predictions for each unit are averaging the predictions across trees

This is an example of **bagging** which is an **ensemble technique**

Simply increase `num.trees` from the prior slide to turn the single "tree" into a "forest"

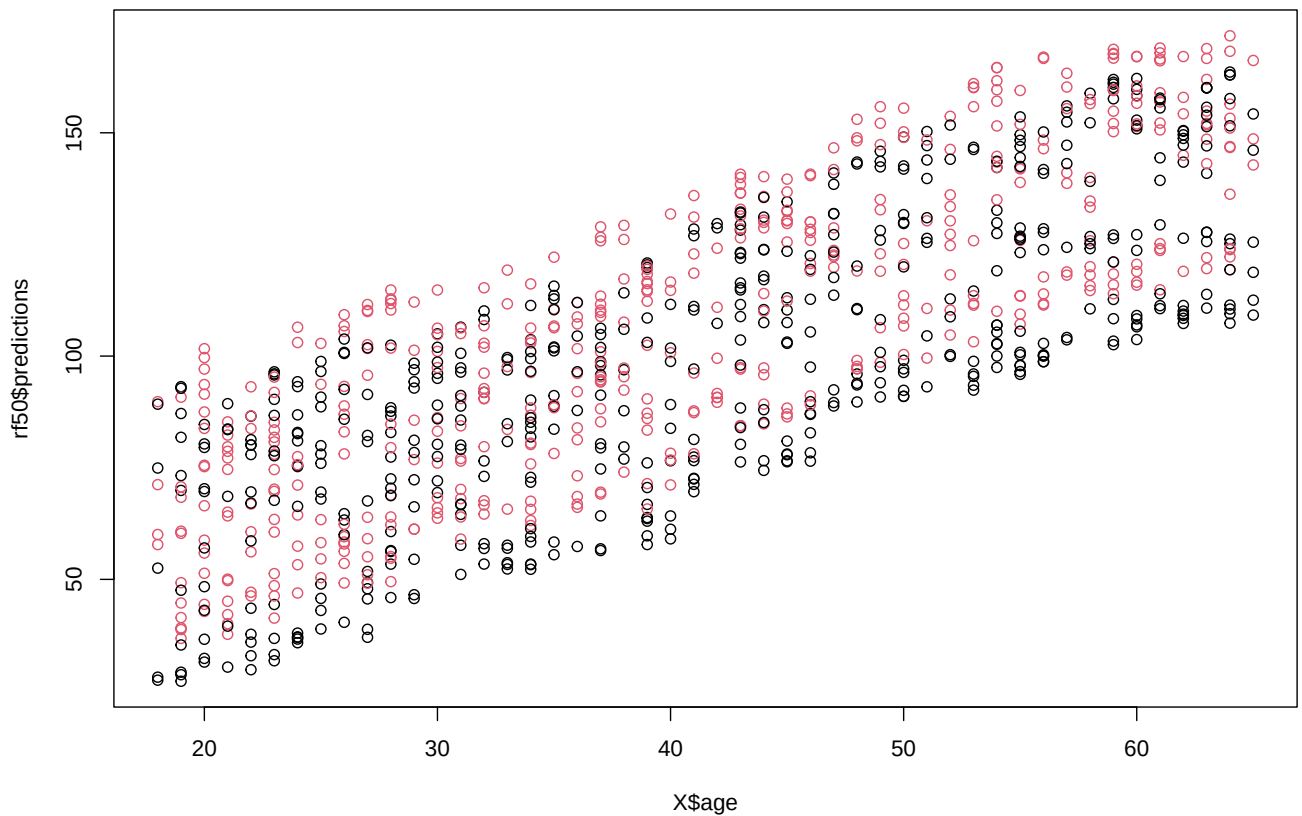
```
rf50 = regression_forest(X, Y, num.trees = 50)
#plot(get_tree(rf50, 1))
#plot(get_tree(rf50, 50))
```

Random forest predictions

Random forests aggregate across trees to get the prediction

Unlike linear regression, predictions don't *necessarily* follow a linear pattern

```
plot(X$age, rf50$predictions, col=factor(X$mobile))
```



Now for *causal* forests

A **causal forest** is a random forest built to predict each unit's *treatment effect* $\tau_i = Y_i(1) - Y_i(0)$ as a function of potential variables X_i

It allows us to sort through the potential variables quickly and identify non-linear relationships

Causal forest for DoorDash

To fit a causal forest, we call `grf::causal_forest`

The inputs are the variables `X`, the outcomes `Y`, the treatment `W`, and the probability of treatment in the experiment `W.hat`

```
W     = doordash$treatment
W.hat = .5 #randomly assigned, equally sized groups
cf = causal_forest(X, Y, W, W.hat=0.5, seed=1)
```

Causal forest CATEs

The goal of the causal forest is to estimate the conditional average treatment effects (CATEs):

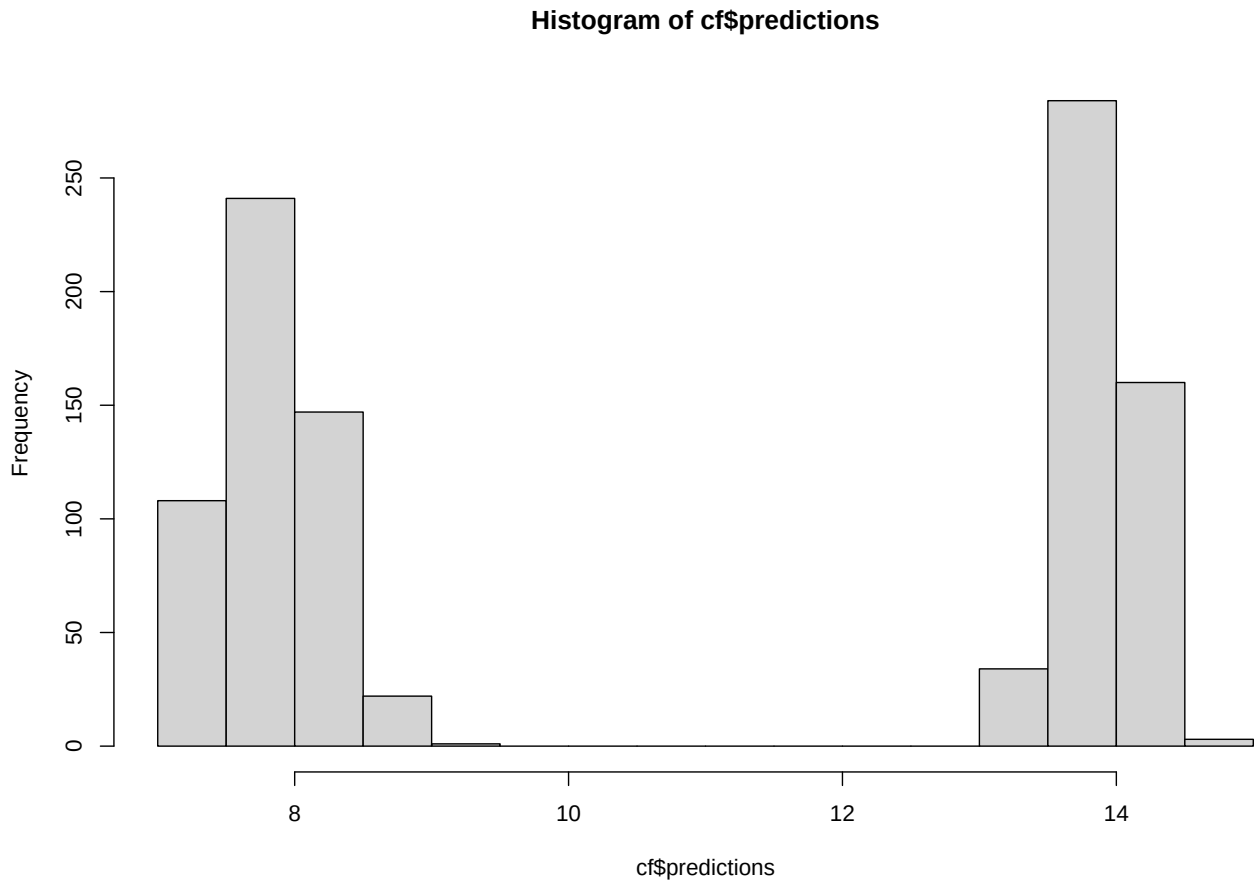
$$\mathbb{E}[\tau_i|X_i] = \mathbb{E}[Y_i(1) - Y_i(0)|X_i]$$

```
doordash$CATE = cf$predictions
str(doordash)
```

```
## 'data.frame':    1000 obs. of  8 variables:
##  $ ip          : num  10001 10002 10003 10004 10005 ...
##  $ age         : num  59 63 59 39 27 22 45 21 43 40 ...
##  $ mthsActive: num  19.26 11.51 4.26 9.53 19.54 ...
##  $ treatment : num  0 0 1 1 0 1 1 1 1 0 ...
##  $ mobile    : num  1 1 0 1 0 0 1 1 1 1 ...
##  $ seconds   : num  116.3 146.6 159.8 120.9 40.3 ...
##  $ purchase  : int  0 0 1 0 0 0 1 0 0 0 ...
##  $ CATE      : num [1:1000, 1] 13.97 13.77 7.43 13.76 8.
```

Heterogeneity in predicted CATES

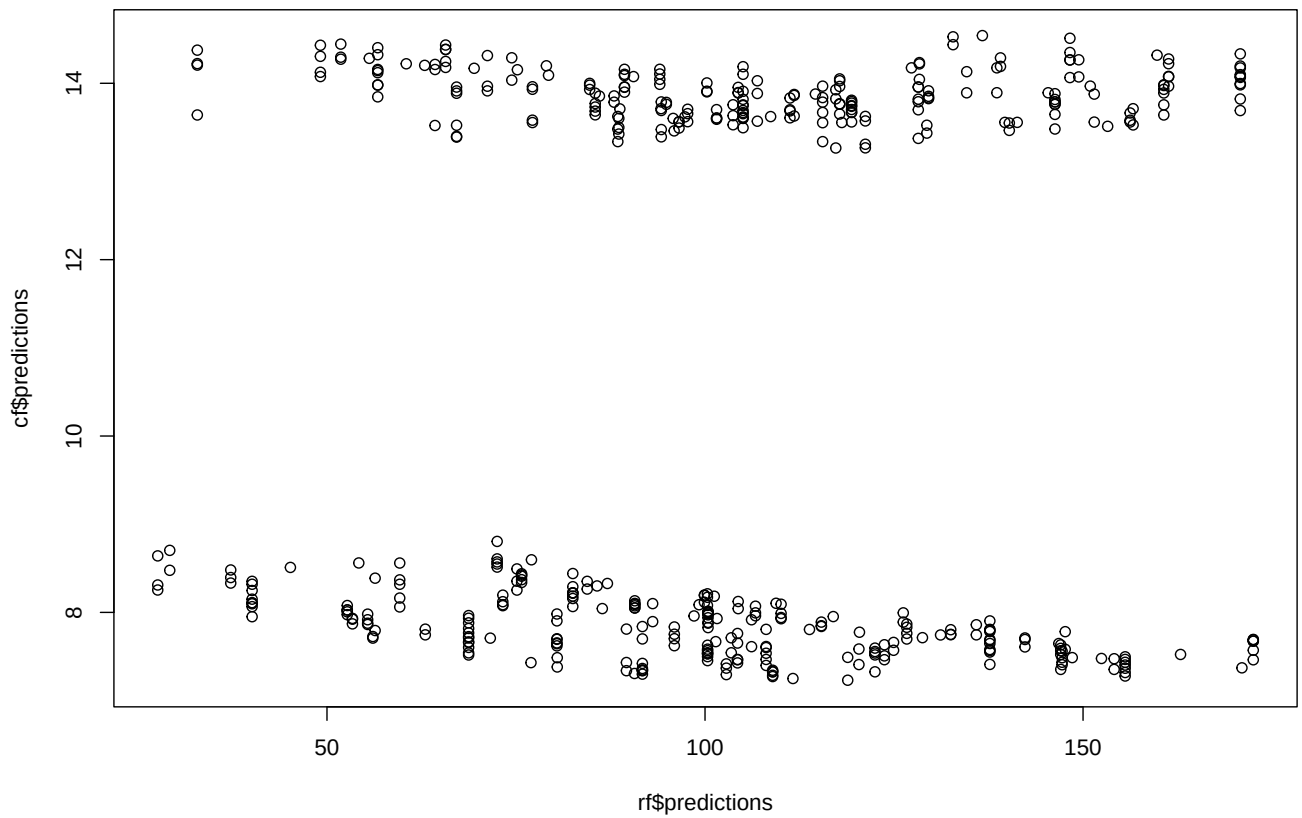
```
hist(cf$predictions)
```



Causal forest versus random forest

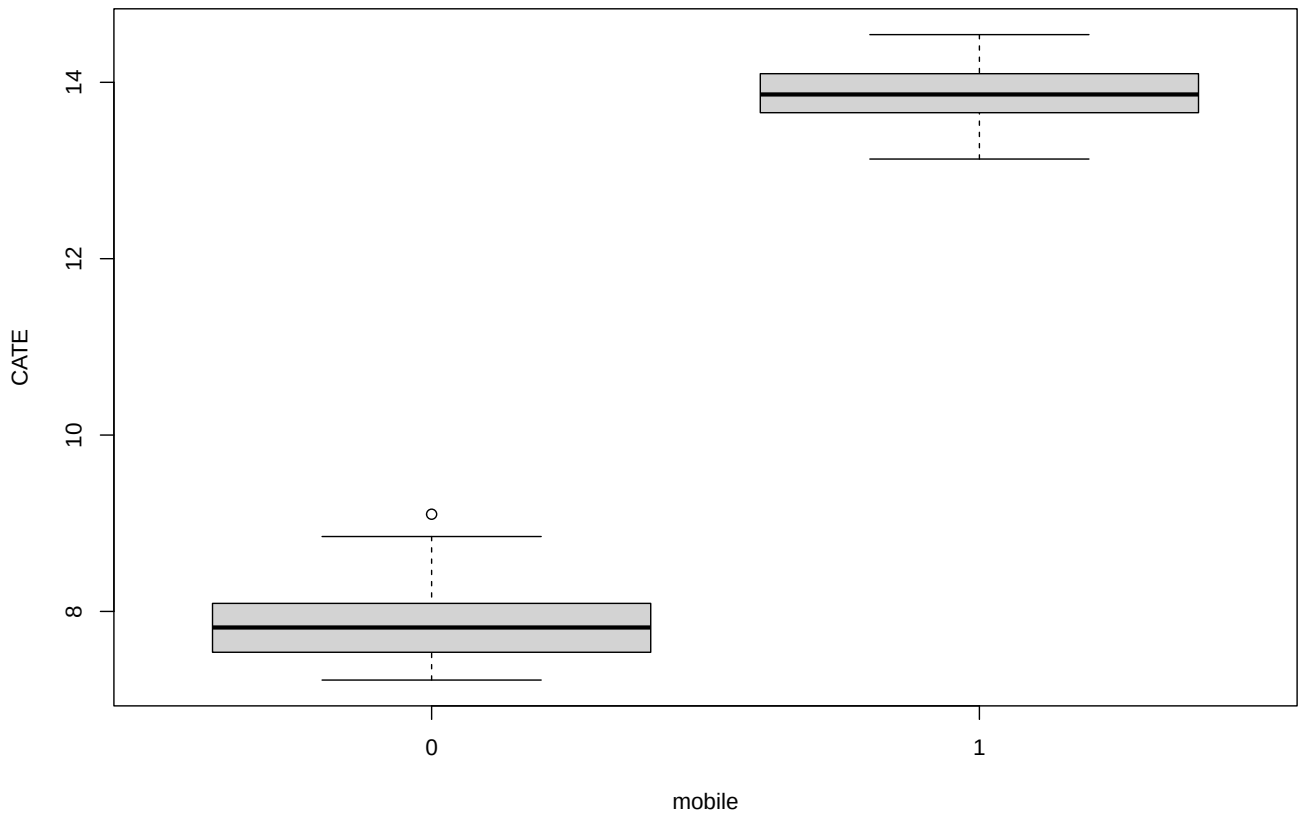
The random forest predicts *seconds* but the causal forest predicts *CATE*s

```
plot(rf$predictions, cf$predictions)
```



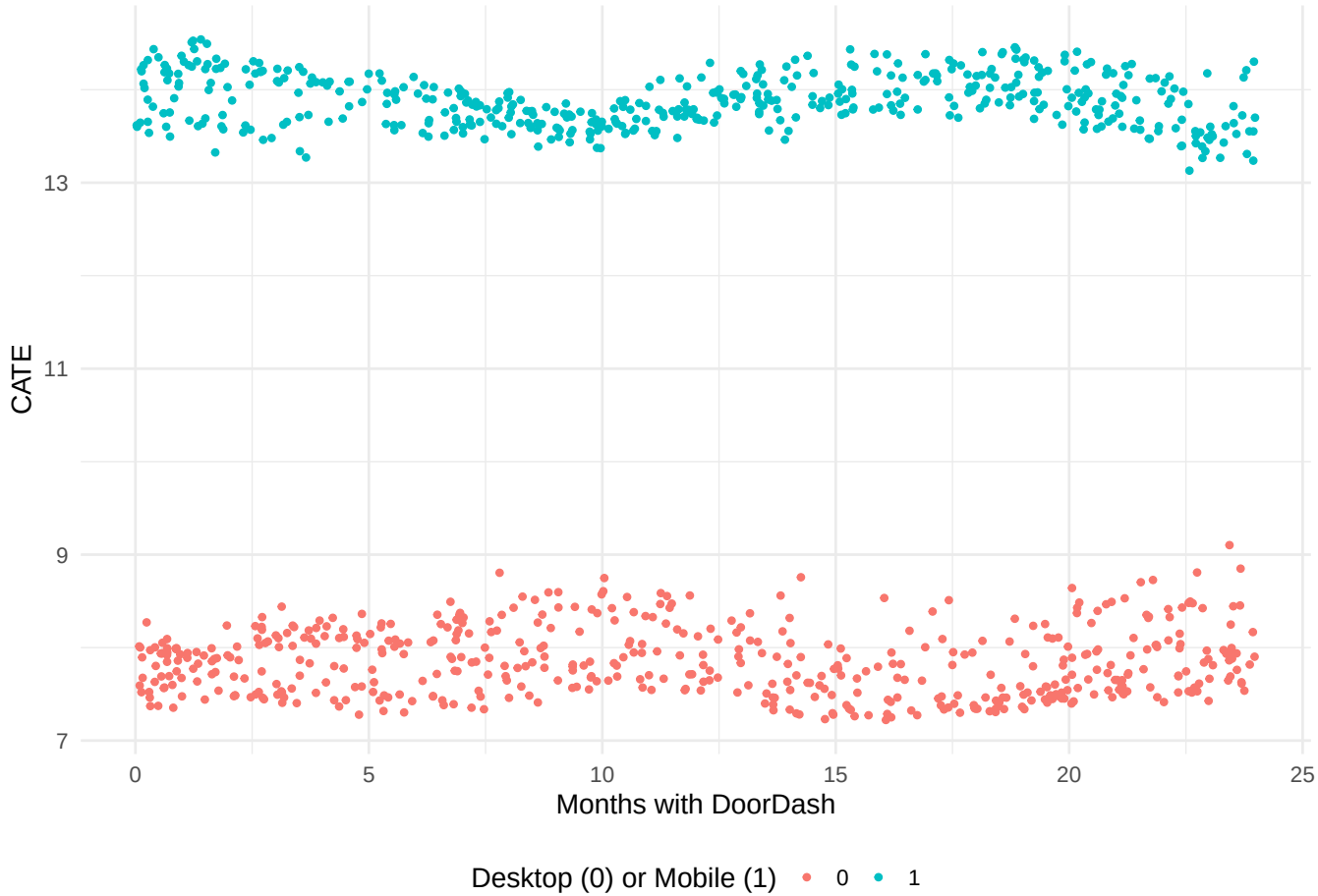
Causal forest CATES by desktop/mobile

```
boxplot(CATE~mobile,doordash)
```




```
#or stripchart(CATE~mobile,doordash,vertical=TRUE)
```

Causal forest CATEs versus age



Optimal treatement given X?

```
library(policytree)
#opportunity cost of not treating = -tau
rewards = cbind(control=-get_scores(cf),
                treatment=get_scores(cf))
tree = policy_tree(X, rewards, min.node.size = 1)
plot(tree, leaf.labels=c("Control", "Treatment"))
```

```
#see R for this plot
```

Which predictors are the most important?

It can be hard to figure out which modifiers are "important"

One way to do that is to use `grf:best_linear_projection` which finds the linear model that fits best to the random forest.

```
best_linear_projection(cf, X)
```

```
##
## Best linear projection of the conditional average treatm
## Confidence intervals are cluster- and heteroskedasticity
##
##                Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  8.529637   1.111182  7.6762 3.901e-14 ***
## mobile       5.754905   0.584360  9.8482 < 2.2e-16 ***
## age         -0.027341   0.024240 -1.1279    0.2596
## mthsActive   0.033474   0.047492  0.7048    0.4811
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
```

Causal forest average treatment effect

`grf::average_treatment_effect` combines the CATES to obtain an average treatment effect

It is similar to our difference-in-means estimate

```
average_treatment_effect(cf)
```

```
##  estimate   std.err
## 10.576744  0.298686
```

Sometimes `std.err` of the causal forest ATE will be smaller than the difference-in-means ATE.

Accounting for variability due to differences in X_i can make the ATE estimate more precise.

Power Calculations

Hypothesis testing review

Alpha α

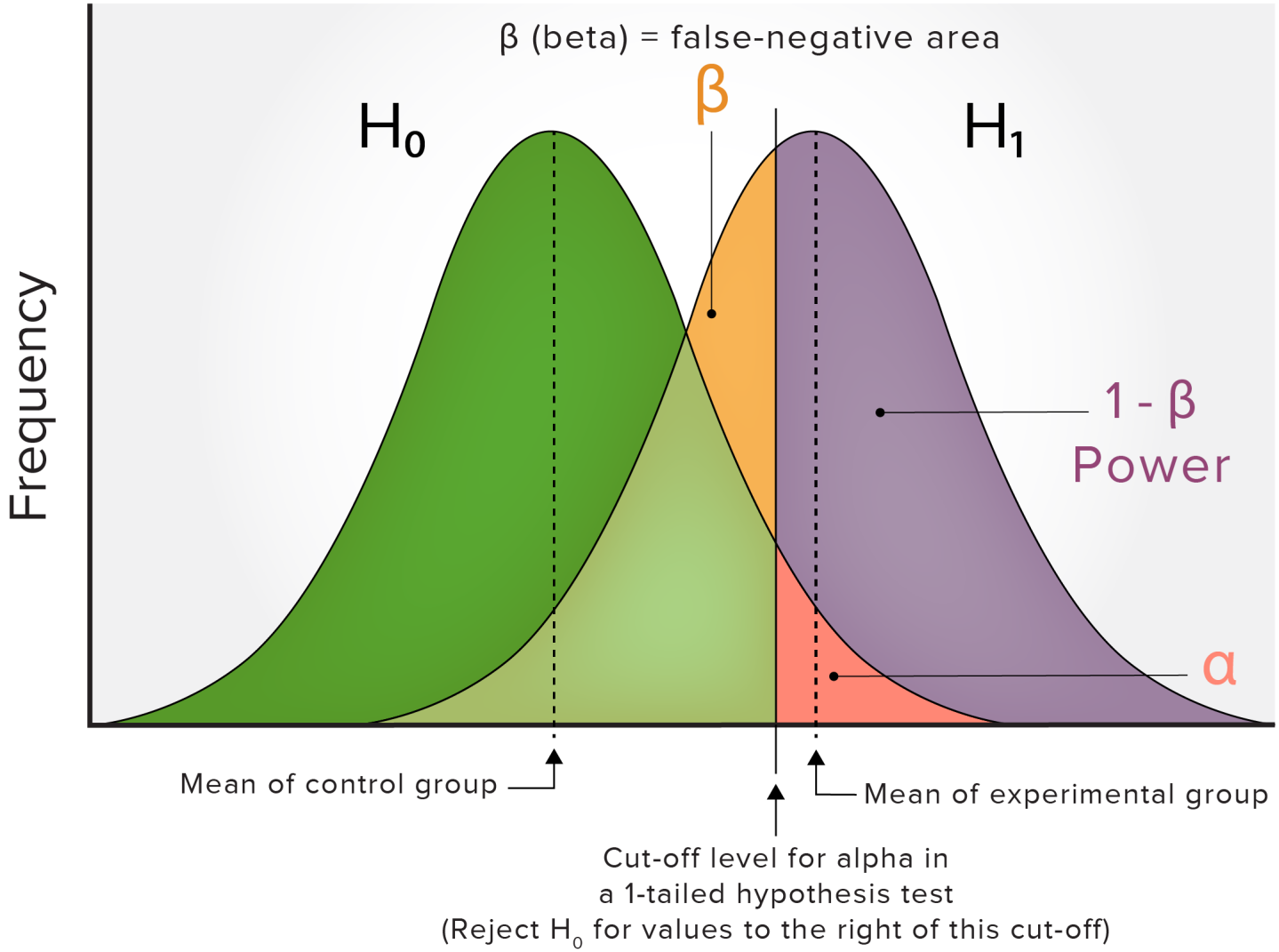
- Significance level (usually .05)
- Probability of making a Type I error
- A Type I error is a "false positive"
- False positive: an incorrect "yes", or we think there *is* an effect when there *is not*
- Specifically: *rejecting* the null when the null is actually *true*

Beta β

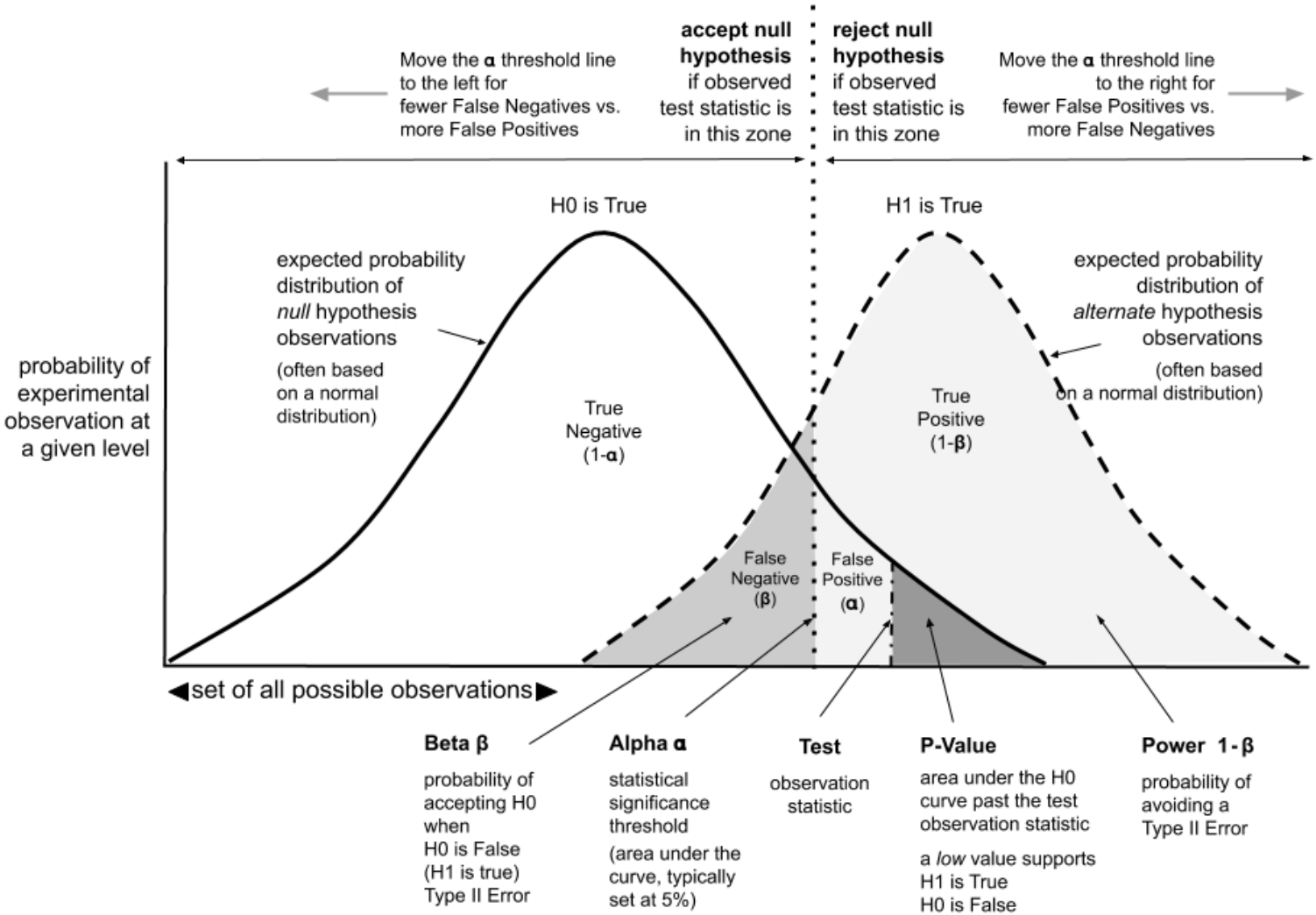
- Probability of a Type II error
- A Type II error is a "false negative"
- False negative: an incorrect "no", or we think there *is no* effect when there *is*
- Specifically: *failing* to reject the null when the null is actually *false*

Note: $\alpha \neq (1 - \beta)$, but they are inversely related: increasing α decreases β

One picture. . .


β (beta) = false-negative area
β
H0
H1
Frequency
1 - β
Power
α
Mean of control group
Mean of experimental group
Cut-off level for alpha in
a 1-tailed hypothesis test
(Reject H0 for values to the right of this cut-off)

Another option...



The Importance of Power

Power is $1 - \beta$: the probability of finding a effect when a true effect exists

In other words, reject the null hypothesis when it is in fact false

Typically set to .8

How to increase power:

- Increase the sample size
- Reduce variance (noise in the data) by controlling for confounding variables and make units as similar as possible
- Increase alpha (say .05 to .10), but this increases the probability of a Type I error (false positive)

Power Calculations in R

See `library(pwr)`

```r
pwr.t.test(d = 0.2,          #difference in means
    sig.level = 0.05,     #Type I error (alpha)
    power = 0.8,          #Power (1 - pr(Type II))
    type = "two.sample", #one/two/paired
    alternative = "two.sided")
```

```
##
##      Two-sample t test power calculation
##
##               n = 393.4057
##               d = 0.2
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

Interpreting power calculations

The output shows the required sample size per group to achieve the desired power

A larger effect size requires a smaller sample size, and vice-versa

Next week we look at how to reduce variance to increase the power

Smaller Effect → Larger N

```r
pwr.t.test(d = 0.01,      #difference in means
    sig.level = 0.05,     #Type I error (alpha)
    power = 0.8,          #Power (1 - pr(Type II))
    type = "two.sample", #one/two/paired
    alternative = "two.sided")
```

```
##
##      Two-sample t test power calculation
##
##               n = 156978.2
##               d = 0.01
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

Case Study: Amazon Website Optimization

Goal

Increase conversion rate

- Percentage of visitors who make a purchase

Hypothesis: a redesigned product page with larger images and customer testimonials will lead to higher conversions

Experimental Design

RCT: randomly assign website visitors to either the existing product page (control) or the redesigned page (treatment)

Metric: conversion rate (% of customers who end up purchasing)

Test duration: 2 weeks

Data

```
load('data/amazon.rdata')
head(amazon)
```

```
##   conversion treatment mobile prime monthlySpend
## 1          0         0      0     0          4.41
## 2          0         0      1     1         37.68
## 3          0         0      0     0          3.75
## 4          0         1      0     1         65.35
## 5          0         0      1     1         61.42
## 6          0         0      0     0          5.83
```

ATE for proportions

```
ate_p = amazon %>%
  group_by(treatment) %>%
  summarise(conversion_bar = mean(conversion))
ate_p
```

```
## # A tibble: 2 x 2
##   treatment conversion_bar
##       <dbl>          <dbl>
## 1         0         0.0984
## 2         1         0.126
```

So the ATE is about 2.8%.

Another way...

```
with(amazon,
     mean(conversion[treatment == 1]) -
     mean(conversion[treatment == 0]))
```

```
## [1] 0.0277879
```

Is it significant?

Confidence Intervals for Proportions

```
n = nrow(amazon)
binom.confint(sum(subset(amazon,treatment == 1)$conversion)
              n = sum(amazon$treatment),method='prop.test')
```

```
##       method   x    n      mean    lower     upper
## 1 prop.test 314 2489 0.1261551 0.113493 0.1399871
```

```
binom.confint(sum(subset(amazon,treatment == 0)$conversion)
              n = sum(amazon$treatment == 0),method='prop.t
```

```
##       method   x    n       mean      lower     upper
## 1 prop.test 247 2511 0.09836718 0.08713672 0.1108456
```

Use a prop-test

Testing the difference in proportions (i.e., purchase rate) between the treated and control groups

```
propData = amazon %>% group_by(treatment) %>%
  summarise(conversions = sum(conversion),n = n())
prop.test(propData$conversions,propData$n)
```

```
##
##  2-sample test for equality of proportions with continu
##
## data:  propData$conversions out of propData$n
## X-squared = 9.4126, df = 1, p-value = 0.002155
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.045675782 -0.009900014
## sample estimates:
##     prop 1     prop 2
## 0.09836718 0.12615508
```

Controlling for confounders

```
cate_p = glm(conversion ~ treatment*prime +
  monthlySpend + mobile,amazon,family='binomial')
#coef(cate_p)
tidy(cate_p) %>% select(1,2,5) %>%
  mutate(across(-1, ~round(., 3)))
```

```
## # A tibble: 6 x 3
##   term              estimate p.value
##   <chr>                <dbl>   <dbl>
## 1 (Intercept)         -2.55    0
## 2 treatment            0.715   0
## 3 prime                0.075   0.579
## 4 monthlySpend         0.007   0
## 5 mobile               0.07    0.443
## 6 treatment:prime     -1.02    0
```

Confidence intervals

But these are on log-odds effects. . .

```
confint(cate_p)
```

```
## Waiting for profiling to be done...
```

```
##                              2.5 %       97.5 %
## (Intercept)          -2.776116719 -2.335524123
## treatment             0.476479550  0.956953153
## prime                -0.190293374  0.340833209
## monthlySpend          0.005177537  0.008603113
## mobile               -0.108817882  0.249322626
## treatment:prime      -1.394691840 -0.654120440
```

Predict probabilities

Remember, treatment effect depends on X since this is non-linear

Setup various conditions and compare predicted conversion probabilities

```
cate_p_df = expand.grid(
  treatment = 0:1,
  mobile = 0:1,
  monthlySpend = round(mean(amazon$monthlySpend),2),
  prime = 0:1)
lo_hat = predict(cate_p, newdata = cate_p_df,
                 se.fit = TRUE)
lower_lo = lo_hat$fit - 1.96 * lo_hat$se.fit
upper_lo = lo_hat$fit + 1.96 * lo_hat$se.fit
```

Compare results

Looks like Prime users have a null/negative treatment effect

```
cbind(cate_p_df,lb = round(plogis(lower_lo),3),
                ub = round(plogis(upper_lo),3))
```

```
##   treatment mobile monthlySpend prime    lb    ub
## 1         0      0        32.62     0 0.073 0.107
## 2         1      0        32.62     0 0.143 0.192
## 3         0      1        32.62     0 0.078 0.114
## 4         1      1        32.62     0 0.153 0.202
## 5         0      0        32.62     1 0.079 0.115
## 6         1      0        32.62     1 0.058 0.089
## 7         0      1        32.62     1 0.084 0.121
## 8         1      1        32.62     1 0.062 0.094
```

*Bootstrap SEs on the probabilities

Another option is to bootstrap the predicted probabilities

Probably more useful in more complex models, but might be useful to see here

Steps:

1) Sample with replacement from the data
2) Estimate the model and save predicted probabilities
3) Repeat many times

*In R

Using a simplified model, but the idea generalizes

```
nBootstraps = 100
n = nrow(amazon)
bootstrapOut = data.frame(control = rep(NA,nBootstraps),
                          treatment = rep(NA,nBootstraps))

for(b in 1:nBootstraps){
  df_b = amazon[sample(1:n,n,replace=TRUE),]
  temp = glm(conversion ~ treatment,family='binomial',data
  bootstrapOut[b,] = predict(temp,data.frame(treatment = 0
}
```

*Now take quantiles

```
#head(bootstrapOut)
apply(bootstrapOut,2,quantile)
```

```
##          control treatment
## 0%   0.08320189 0.1092808
## 25%  0.09384953 0.1233001
## 50%  0.09828241 0.1272361
## 75%  0.10194856 0.1309408
## 100% 0.11353712 0.1426269
```

*Or get quantiles this way. . .

```
bootstrapOut %>%
  reframe(across(c(control,treatment),
                 ~quantile(.x,c(.025,.975))))
```

```
##       control treatment
## 1 0.08767708 0.1112726
## 2 0.10940749 0.1394809
```

*Another option for bootstrapping...

Slight modification

```
singleBootstrap = function() {
  df_b = amazon[sample(1:n,n,replace=TRUE),]
  temp = glm(conversion ~ treatment,family='binomial',data
  predict(temp,data.frame(treatment = 0:1),type='response')
}

#control | treated
bootOut = t(replicate(10,singleBootstrap()))
apply(bootOut,2,quantile)
```

```
##                1         2
## 0%   0.08660194 0.1133936
## 25%  0.08874525 0.1238639
## 50%  0.09371650 0.1272913
## 75%  0.09646334 0.1307292
## 100% 0.10760741 0.1384430
```

Causal forest

```
W = amazon$treatment
Y = amazon$conversion
X = amazon %>% select(-c(treatment,conversion))
cf_amzn = causal_forest(X, Y, W, W.hat = 0.5, seed=1)
amazon$CATE = cf_amzn$predictions
```

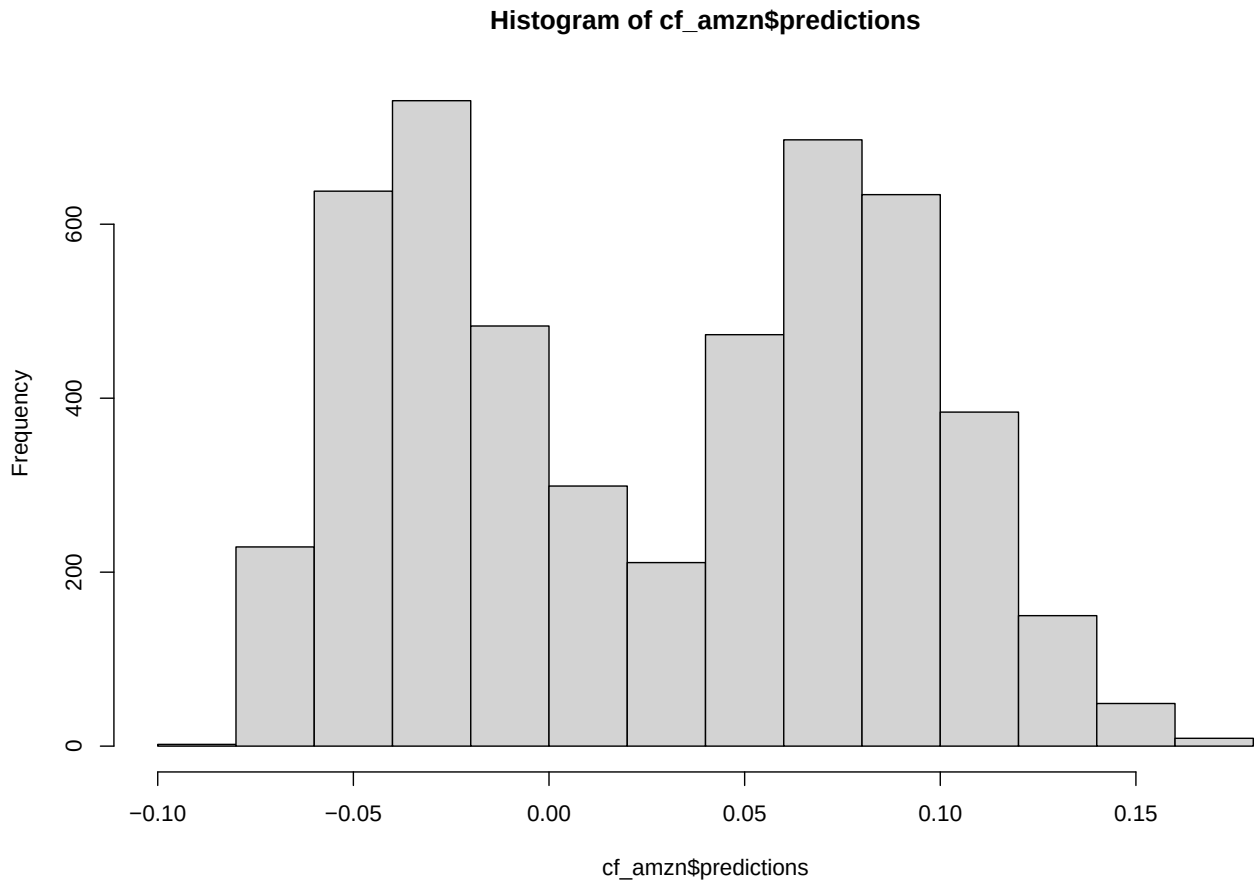
Causal forest ATE

```
average_treatment_effect(cf_amzn)
```

```
##    estimate     std.err
## 0.027855967 0.008837823
```

Heterogeneity in predicted CATES

```
hist(cf_amzn$predictions)
```



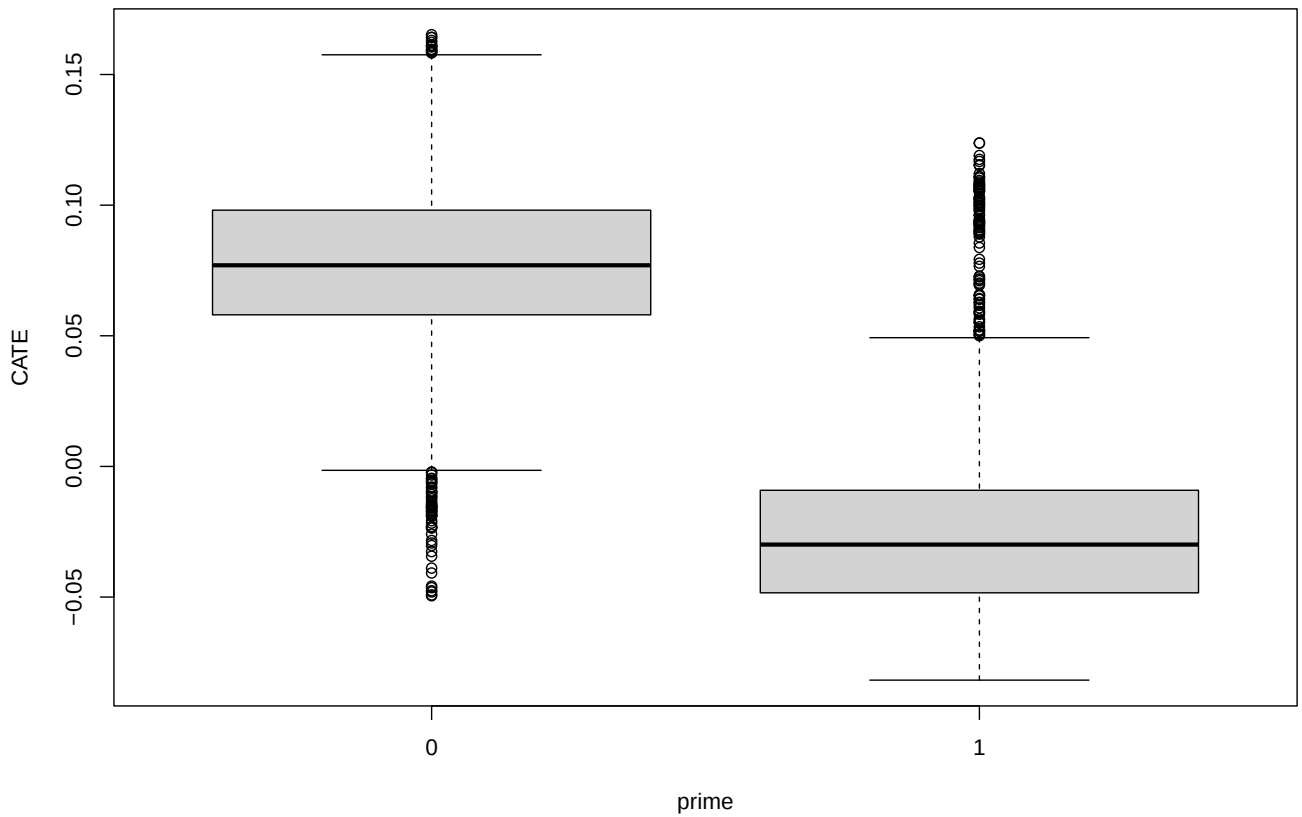
Which predictors are the most important?

```
best_linear_projection(cf_amzn, X)
```

```
##
## Best linear projection of the conditional average treatm
## Confidence intervals are cluster- and heteroskedasticity
##
##                  Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)   7.8249e-02  1.7333e-02  4.5143 6.497e-06 *
## mobile        7.1840e-03  1.7590e-02  0.4084    0.6830
## prime        -1.0507e-01  1.7611e-02 -5.9663 2.594e-09 *
## monthlySpend -6.9163e-05  2.9865e-04 -0.2316    0.8169
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
```

Causal forest CATES versus mobile users

```
boxplot(CATE~prime,amazon)
```



Causal forest CATEs versus Prime

Which treatement given X?

```
rewards = cbind(control=-get_scores(cf_amzn),
                treatment=get_scores(cf_amzn))
tree = policy_tree(X, rewards, min.node.size = 1)
plot(tree, leaf.labels=c("Control", "Treatment"))
```

Interpretation

- On average, the redesigned page increased conversions by about 2.8%
- But, the treatment effect was significantly lower for Prime users (why? Maybe they were so used to the old layout)
- We also see heterogeneity in monthly spend levels: the Prime differential is greater for those with lower monthly spend

What to do?

What are the limitations of releasing this?

Is the negative Prime effect a concern?

Is the difference worth it?

- Think about proportion of Prime users to everyone else, if it is small then it doesn't matter if there is a negative effect in this group

Any other ideas?

Bonus Tips

On Writing I of II

- **For your analysis to have impact, it needs to be understood**
- Write like you speak: plainly with small words, simple sentences, and short paragraphs
- Read your writing out loud. Good writing sounds natural.
- Be specific: "increases substantially" $\rightarrow$ "increases by 5%"
- Focus on actionable insights

Writing Tips II

Use active voice rather than passive voice:

{subject} + {verb} + {object}

Active

- Our campaign generated 15% more leads
- We collected survey responses from 5,000 subjects
- We recommend implementing an A/B test on the home page

Passive

- 15% more leads were generated from our campaign
- Survey responses were collected from 5,000 subjects
- It is recommended that A/B testing is implemented on the home page

Next Week

- Variance reduction with CUPED
- PSM
- AIPW
- DoubleML