Machine learning for causal inference

Apoorva Lal and Zuhad Hai | Stanford

27 May 2021

Questions?

Uses of machine Learning for causal inference

- Effect heterogeneity (last week)
- Covariate adjustment and high-dimensional treatments (this week and next week)

Notation Refresher

- Data $\{D_i, Y_i, \mathbf{X}_i\}_i^N \in \{0,1\} \times \mathbb{R} \times \mathcal{X}$
- interested in estimating the treatment effect of D on Y
  - ATE : $\tau = \mathbb{E}\left[Y_i(1) - Y_i(0)\right]$

Selection on Observables assumptions

- Unconfoundedness: $Y_i(1), Y_i(0) \perp D_i | \mathbf{X}_i$
- Overlap: $0 < \mathbf{Pr}\left(D = 1 | \mathbf{X} = \mathbf{x}\right) < 1$

ATE Estimators using CEFs

- ML is excellent at fitting conditional expectation functions (CEFs)
- For causal inference, we use the following CEFs repeatedly
  - Conditional mean: $\mu_{(d)}(\mathbf{x}) := \mathbb{E}\left[Y_i(d)|\mathbf{X}_i = \mathbf{x}\right]$
  - Propensity score: $e(x) := \mathbb{E}\left[D_i = 1|\mathbf{X}_i - \mathbf{x}\right]$

Inverse Propensity Weighting

$$\tau = \mathbb{E}\left[\frac{D_i Y_i}{e(\mathbf{X}_i)} - \frac{(1-D_i)Y_i}{1-e(\mathbf{X}_i)}\right]$$

Plugin principle for estimator

Conditional Mean Estimation

$$\begin{aligned}\tau(\mathbf{x}) &= \mathbb{E}\left[Y_i(1) - Y_i(0)|\mathbf{X}_i = \mathbf{x}\right] \\ &= \mu_{(1)}(\mathbf{x}) - \mu_{(0)}(\mathbf{x})\end{aligned}$$

$$\hat{\tau} = \frac{1}{n}\sum_{i=1}^{N}(\widehat{\mu}_{(1)}(\mathbf{x}) - \widehat{\mu}_{(0)}(\mathbf{x}))$$

Combining the two approaches

- Augmented IPW / 'Double Robust' estimator (Robins, Rotnitzky, and Zhao 1994)

$$\hat{\tau}_{AIPW} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{\mu}_{(1)}(X_i) - \hat{\mu}_{(0)}(X_i) + D_i\frac{Y_i - \hat{\mu}_{(1)}(X_i)}{\hat{e}(X_i)} - (1-D_i)\frac{Y_i - \hat{\mu}_{(0)}(X_i)}{1-\hat{e}(X_i)}\right)$$

- Double Robustness: consistent if either $\widehat{\mu}_{(d)}(\mathbf{x})$ OR $\hat{e}(\mathbf{x})$ is consistent
- Honesty principle: use 'cross-fitting' (split the sample, fit $\widehat{\mu}$ and $\hat{e}$ on one half and predict on the other)

Orthogonal Scores / Double Machine Learning

- Chernozhukov, Hansen, Robins, et al

$$Y = \tau D + \mu(\mathbf{X}) + \varepsilon \quad , \mathbb{E}\left[\varepsilon | D, \mathbf{X}\right] = 0$$
$$D = e(\mathbf{X}_i) + \eta \quad , \mathbb{E}\left[\eta | \mathbf{X}\right] = 0$$

- Can use linear approximation for μ and e [Double selection - BCH 2014]
- Or generic machine learners and estimate residuals-on-residuals regression

$$y - \widehat{\mathbb{E}}[Y \mid \mathbf{x}] = \tau(D - \widehat{\mathbb{E}}[D \mid \mathbf{x}]) + \nu$$

Application to Gerber, Green, Larimer (2008)

- Athey et al Tutorial

Data Prep

```
# If you need to, you can download the data from GitHub
if(!file.exists(here::here("data", "socialneighbor.csv"))){
  datafile <- "https://raw.githubusercontent.com/gsbDBI/ExperimentData/master/Social/ProcessedData/socialpre

  # get data file and load into memory
  data_raw <- read.csv(datafile)

  # create a data folder if not already available
  dir.create(here::here("data"), showWarnings = FALSE)
  # Save the data
  write.csv(data_raw, here::here("data", "socialneighbor.csv"))
}else{
  # Read in the data
  data_raw <- read.csv(here::here("data", "socialneighbor.csv"))
}

# These are the covariates we'll use
cts_variables_names <- c("yob", "hh_size", "totalpopulation_estimate",
                         "percent_male", "median_age",
                         "percent_62yearsandover",
                         "percent_white", "percent_black",
                         "percent_asian", "median_income",
                         "employ_20to64", "highschool", "bach_orhigher",
                         "percent_hispanicorlatino")
```

Experimental ATE using Linear Regression

```
(tauhat_linear <- lm(Y~W, data=df))
```

```
##
## Call:
## lm(formula = Y ~ W, data = df)
##
## Coefficients:
## (Intercept)            W
##      0.2925       0.0855
```

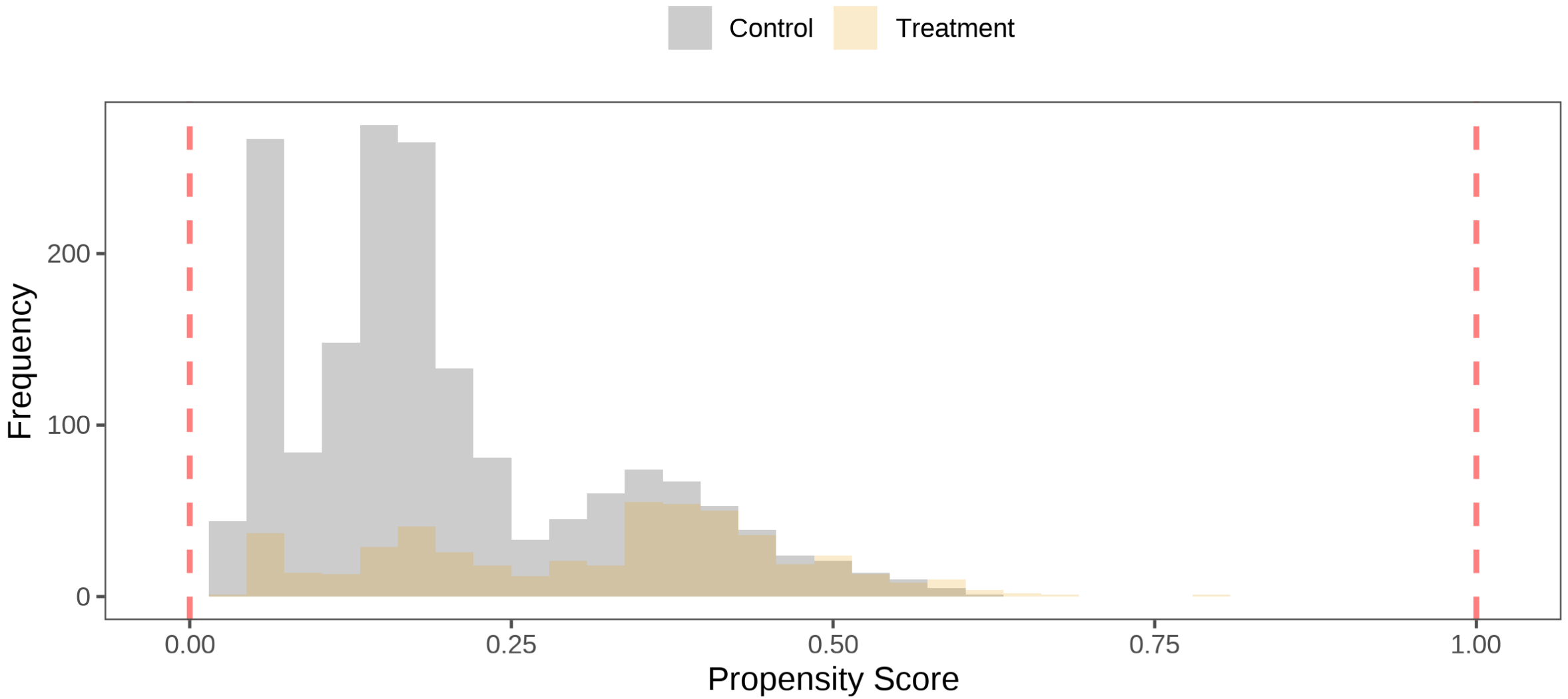
Adding Confounding

```
likely_voter = (((df$g2002 == 1) & (df$sex == 0)) | (df$p2004 == 1))
prob = 0.25 # proportion of voters to drop

#drop proportion of likely voters from the treated group
drop_from_treat <- base::sample(which(likely_voter & df$W == 1), round(prob * sum(likely_voter)))
#drop proportion of non-likely voters from the control group
drop_from_control <- base::sample(which(!likely_voter & df$W == 0), round(prob * sum(!likely_voter)))

# Modified dataset
df_mod <- df[-c(drop_from_treat, drop_from_control),]

# The difference in means is now biased!
(tauhat_conf_linear <- lm(Y~W, data=df_mod))
```

```
##
## Call:
## lm(formula = Y ~ W, data = df_mod)
##
## Coefficients:
## (Intercept)            W
##      0.3018       0.0473
```

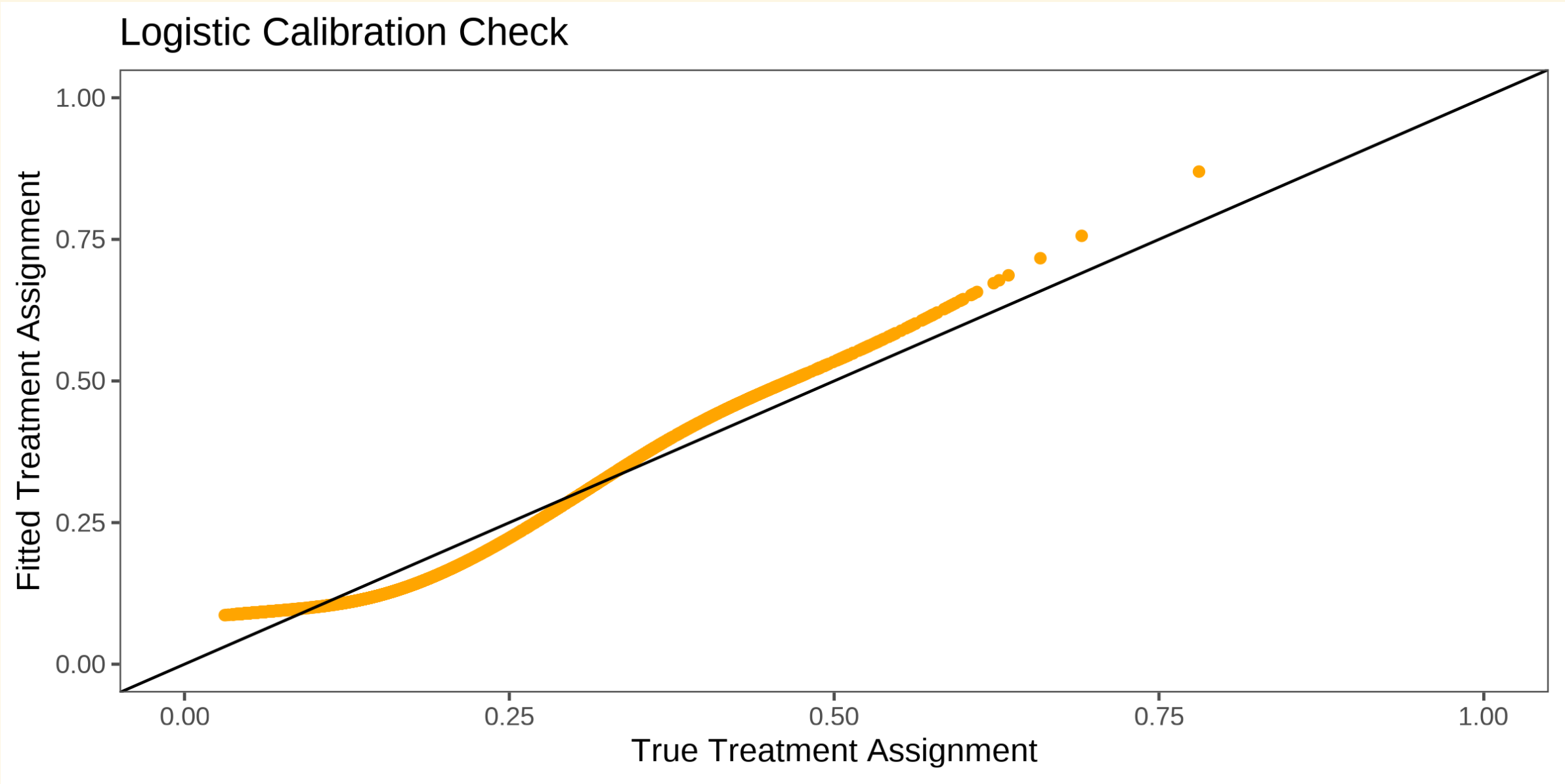
Estimating the Pscore

```
covariates_names <- names(df_mod[,!names(df_mod) %in% c("Y", "W")])
#Create a matrix for covariates other than treatment(W) and outcome(Y)
Xmod = df_mod[,!names(df_mod) %in% c("Y", "W")]
#Create a vectors for the outcome variable (Y) and the treatment (W)
Ymod = df_mod$Y
Wmod = df_mod$W
# Computing the propensity score by logistic regression of W on all the covariates except the outcome (Y).

# Declare formula for logistic regression
p_logistic.frmla <- as.formula(paste("W ~ ", paste(covariates_names, collapse= "+")))
p_logistic.fit <- glm(p_logistic.frmla , data = df_mod, family = "binomial")
#These are predicted probabilities of getting the treatment (W) as a function of the covariates (X)
#Add the predicted probabilities to our original data frame (df_mod)
df_mod$p_logistic<- predict(p_logistic.fit, type = "response")

#The following plot is the histogram of the propensity scores.
p <- ggplot(df_mod, aes(x = p_logistic, fill = as.factor(W))) +
  geom_histogram(alpha = 0.2, position = "identity", bins = 35) +
  geom_vline(aes(xintercept=1),color="red", linetype="dashed", size=1, alpha = 0.5) +
  geom_vline(aes(xintercept=0), color="red", linetype="dashed", size=1, alpha = 0.5) +
  labs(title="", x="Propensity Score", y = "Frequency") +
  ggthemes::theme_few() +
  scale_fill_colorblind(name="",labels=c("Control", "Treatment")) +
  theme(legend.position = "top")
```


Control
Treatment
Frequency
200
100
0
0.00
0.25
0.50
0.75
1.00
Propensity Score

Calibration Plot for pscore

```
log_calibration <- smooth.spline(df_mod$p_logistic, Wmod, df = 4)
log_calibration <- as.data.frame(cbind(log_calibration$x, log_calibration$y))

log_spline <- ggplot(log_calibration,aes(x=V1, y= V2)) +
  geom_point(color = "orange")+
  labs(title = "Logistic Calibration Check",
       x= "True Treatment Assignment",
       y="Fitted Treatment Assignment") +
  geom_abline(intercept = 0) + xlim(c(0,1))+ ylim(c(0,1))+
  ggthemes::theme_few()
```


Logistic Calibration Check
Fitted Treatment Assignment
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
True Treatment Assignment

Conditional Mean Estimation

```
library(estimatr)
lm_interact = lm_robust(Y ~ (. - p_logistic) * W , data = df_mod)
lm_interact %>% tidy %>% filter(term == "W")
```

```
##   term estimate std.error statistic p.value conf.low conf.high   df outcome
## 1    W  0.08971   0.07527     1.192  0.2335  -0.0579    0.2373 2180       Y
```

IPW

```
signif.level <- qnorm(0.025, lower.tail = FALSE)
ipw <- function(dataset, p) {
  W <- dataset$W
  Y <- dataset$Y
  G <- ((W - p) * Y) / (p * (1 - p))
  tau.hat <- mean(G)
  se.hat <- sqrt(var(G) / (length(G) - 1))
  c(ATE=tau.hat, lower_ci = tau.hat - signif.level * se.hat,
      upper_ci = tau.hat + signif.level * se.hat)
}

(tauhat_logistic_ipw <- ipw(df_mod, df_mod$p_logistic))
```

```
##       ATE lower_ci upper_ci
##  0.10919  0.02585  0.19254
```

AIPW

```
aipw_ols <- function(dataset, p) {
  dataset = dataset[,-which(names(dataset) %in% c("p_logistic"))]
  ols.fit = lm(Y ~ W * ., data = dataset)
  # predict on dataset as if everyone was treated
  dataset.treatall = copy(dataset); dataset.treatall$W = 1
  μ1 = predict(ols.fit, dataset.treatall)
  # predict on dataset as if no one was treated
  dataset.treatnone = copy(dataset); dataset.treatnone$W = 0
  μ0 = predict(ols.fit, dataset.treatnone)
  # predict on actual status
  actual_pred = predict(ols.fit, dataset)
  G <- μ1 - μ0+
    ((dataset$W - p) * (dataset$Y - actual_pred)) / (p * (1 - p))
  tau.hat <- mean(G)
  se.hat <- sqrt(var(G) / (length(G) - 1))
  c(ATE=tau.hat,
    lower_ci = tau.hat - signif.level * se.hat,
    upper_ci = tau.hat + signif.level * se.hat)
}
(tauhat_lin_logistic_aipw <- aipw_ols(df_mod, df_mod$p_logistic))
```

```
##      ATE lower_ci upper_ci
##  0.07458  0.01020  0.13896
```

AIPW using forests

```
library(grf)

cf <- causal_forest(Xmod, Ymod, Wmod, num.trees = 500)
ate_cf_aipw <- average_treatment_effect(cf)
ate_cf_aipw %>% summary
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0281  0.0447  0.0613  0.0613  0.0779  0.0945
```

```
p_rf <- as.data.frame(cf$W.hat) %>% rename(p_cf='cf$W.hat')
(tauhat_ols_rf_aipw <- aipw_ols(df_mod, p_rf$p_cf))
```

```
##      ATE lower_ci upper_ci
##  0.08653  0.03350  0.13956
```

DoubleML

```r
library("hdm"); library("AER"); library("randomForest"); library(glmnet)
data(GrowthData)                      # Barro-Lee growth data
y= as.matrix(GrowthData[,1])          # outcome: growth rate
d= as.matrix(GrowthData[,3])          # treatment: initial wealth
x= as.matrix(GrowthData[,-c(1,2,3)]) # controls: country characteristics


DML2.for.PLM <- function(x, d, y, dreg, yreg, nfold=2) {
  nobs <- nrow(x) #number of observations
  foldid <- rep.int(1:nfold, times = ceiling(nobs/nfold))[sample.int(nobs)] #define folds indices
  I <- split(1:nobs, foldid)  #split observation indices into folds
  ytil <- dtil <- rep(NA, nobs)
  cat("fold: ")
  for(b in 1:length(I)){
    dfit <- dreg(x[-I[[b]],], d[-I[[b]]]) #take a fold out
    yfit <- yreg(x[-I[[b]],], y[-I[[b]]]) # take a fold out
    dhat <- predict(dfit, x[I[[b]],], type="response") #predict the left-out fold
    yhat <- predict(yfit, x[I[[b]],], type="response") #predict the left-out fold
    dtil[I[[b]]] <- (d[I[[b]]] - dhat) #record residual for the left-out fold
    ytil[I[[b]]] <- (y[I[[b]]] - yhat) #record residual for the left-out fold
    cat(b," ")
        }
  rfit <- lm(ytil ~ dtil)    #estimate the main parameter by regressing one residual on the other
  coef.est <- coef(rfit)[2]  #extract coefficient
```

DML with different learners

```
set.seed(1)
dreg <- function(x,d){ glmnet(x, d, lambda = 0) } #ML method= OLS using glmnet; using lm gives bugs
yreg <- function(x,y){ glmnet(x, y, lambda = 0) } #ML method = OLS
DML2.OLS = DML2.for.PLM(x, d, y, dreg, yreg, nfold=10)
```

```
## fold: 1  2  3  4  5  6  7  8  9  10
## coef (se) = 0.01013 (0.0167061)
```

```
dreg <- function(x,d){ rlasso(x,d, post=FALSE) } #ML method= lasso from hdm
yreg <- function(x,y){ rlasso(x,y, post=FALSE) } #ML method = lasso from hdm
DML2.lasso = DML2.for.PLM(x, d, y, dreg, yreg, nfold=10)
```

```
## fold: 1  2  3  4  5  6  7  8  9  10
## coef (se) = -0.0417523 (0.016092)
```

```
dreg <- function(x,d){ randomForest(x, d) } #ML method=Forest
yreg <- function(x,y){ randomForest(x, y) } #ML method=Forest
DML2.RF = DML2.for.PLM(x, d, y, dreg, yreg, nfold=10)
```

```
## fold: 1  2  3  4  5  6  7  8  9  10
## coef (se) = -0.038251 (0.0149232)
```

Latent Treatments

Fong and Grimmer 2021

Text as Treatment (Fong and Grimmer (2016, 2021))

- Goal: discover treatments and estimate their effects
    - Fong and Grimmer 2016 - identify treatments and estimate their Average Marginal Component specific Effect (AMCE)
- Text $\mathbf{T}_i$, potential outcome $Y_i(\mathbf{T}_i)$
- Measured treatment $g(\mathbf{T}_i) =: Z_i$
- Unmeasured treatment $h(\mathbf{T}_i) =: B_i$

1. SUTVA
2. Random Assignment of Texts
3. Measured and Unmeasured representation
4. One of two
    - Measured and unmeasured latent treatments independent
    - Unmeasured treatments unrelated to outcome

Estimand and Estimator

$$\mathrm{ATE} = \sum_{b \in B} \left(\mathbb{E}\left[Y_i(Z_i = 1, \mathbf{B}_i = b) \right] - \mathbb{E}\left[Y_i(Z_i = 0, \mathbf{B}_i = \mathbf{b}) \right] \right) \mathbf{Pr}\left(B_i = b \right)$$

$$\widehat{\mathrm{ATE}} = \mathbb{E}\left[Y_i(\mathbf{T}_i | g(\mathbf{T}_i = 1)) \right] - \mathbb{E}\left[Y_i(\mathbf{T}_i | g(\mathbf{T}_i = 0)) \right]$$

Trump tweets experiment (Section 5.2)

```
library(tidytext); library(texteffect); library(textdata); library(car)
library(knitr); library(modelsummary)
dat <- read.csv("grimmerFong/trumpdt.csv")

Y <- dat[,1]
G <- dat[,2:4]
X <- dat[,5:ncol(dat)]
rm(dat)
```

Sample Splitting

```
s = 12082017
set.seed(s)
training.tweets <- sample(1:(nrow(X)/3), nrow(X)/3*.5)
train.ind <- c()
for (i in 1:length(training.tweets)){
  train.ind <- c(train.ind, 3*(training.tweets[i]-1)+(1:3))
}
```

Supervised Indian Buffet Process (Implementation)

- Infer Treatments

```
## Fit sIBP with many different parameter figurations so the analyst can choose
## the most substantively interesting run
## Note: This will take a while to run (approx 20 minutes)
sibp.search <- sibp_param_search(X, Y, K = 5, alphas = c(2,3,4), sigmasq.ns = c(0.5, 0.75, 1),
                                 iters = 5, train.ind = train.ind, G = G, seed = s)
```

Identified Latent Treatments

```
load("grimmerFong/sibp_search.rds")
sibp.fit <- sibp.search[["3"]][["0.5"]][[1]]
sibp_top_words(sibp.fit, colnames(X), verbose = TRUE) %>% str
```

```
## [1] "Frequency of treatments: "
## [1] 338.5 228.0 322.9 173.9 185.0
## [1] "Relation between top words and treatments"
##        [,1]  [,2]   [,3]  [,4]  [,5]
## [1,] 0.9626 1.414 1.1729 1.715 2.149
## [2,] 0.9616 1.414 0.8814 1.715 2.149
## [3,] 0.8589 1.149 0.8222 1.486 1.458
## [4,] 0.7946 1.149 0.7689 1.433 1.319
## [5,] 0.7428 1.149 0.7144 1.412 1.222
## [6,] 0.7428 1.020 0.7073 1.391 1.175
##  chr [1:10, 1:5] "minister" "prime" "join" "united" "behalf" "vp" "hunt" ...
```

Effect estimates by group

```
## Infer values of latent treatments in the test-set
set.seed(12092017)
X.test <- t(apply(X[sibp.fit$test.ind,], 1, function(x) (x - sibp.fit$meanX)/sibp.fit$sdX))
nu.test <- infer_Z(sibp.fit, X)
Z.train <- matrix(as.numeric(sibp.fit$nu >= 0.5), ncol = 5)
Z.test <- matrix(as.numeric(nu.test >= 0.5), ncol = 5)

## Construct data for anaysis split out by partisanship
dat2 <- data.frame(Y[sibp.fit$test.ind], G[sibp.fit$test.ind,])
colnames(dat2) <- c("Y", "ind", "dem", "rep")
dat2$Z1 <- Z.test[,1]
dat2$Z2 <- Z.test[,2]
dat2$Z3 <- Z.test[,3]
dat2$Z4 <- Z.test[,4]
dat2$Z5 <- Z.test[,5]

## Get sentiment scores via AFINN and dichotomize into positive or negative
start<- as.matrix(get_sentiments('afinn'))
use0 <- match(colnames(X), start[,1])
use <- use0[which(!is.na(use0))]
use_col<- which(is.na(match(colnames(X), start[,1]))==F)
sents<- as.matrix(X[sibp.fit$test.ind,use_col])%*%as.numeric(start[use,2])
dat2$sents <- I(sents > 0)
```

Estimates

```
modelsummary(list(m6, m5, m4, m3, m2, m1))
```

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
(Intercept)	-83.083	-1.459	95.318	-93.047	-8.684	90.624
	(1.680)	(1.289)	(1.030)	(2.096)	(1.614)	(1.304)
Z1	36.164	22.496	10.080	29.113	17.383	6.758
	(5.976)	(4.585)	(3.662)	(5.842)	(4.501)	(3.635)
Z2	-27.639	-24.143	-13.908	-22.959	-20.750	-11.703
	(7.080)	(5.432)	(4.339)	(6.860)	(5.284)	(4.268)
Z3	-20.490	-17.099	-1.248	-19.148	-16.127	-0.616
	(6.895)	(5.290)	(4.226)	(6.656)	(5.127)	(4.141)
Z4	-19.999	-10.339	0.275	-16.440	-7.758	1.952