Description of Analysis

Replication of ITT Effects

The following code supports the ITT estimation in our report. We implement three methods for estimating average treatment effects: * First, we estimate direct conditional ATEs for the assigned treatment group. The authors of the study we are replicating designed treatment assignment to be unconfounded, so we might expect that direct ATE estimates are a reasonable estimate of the underlying treatment effect. The estimated treatment effect is the average of $\tau(X)$, where

$$\tau(X) = E[Y_i(1) - Y_i(0)|X_i = x]$$

* Second, we estimate a weighted average treatment effect using propensity weights. In a non-randomized study, we might be concerned that treatment is correlated with certain features X. In our context, we know that assignment balanced school-level characteristics but we might be worried that student-level characteristics in particular are unbalanced in the treatment versus control groups. If unconfoudnedness holds, then it should be the case that

$$Y_i(1), Y_i(0) \perp W_i|e(X_i)$$

We use inverse propensity weighting to balance the influence of treated and control observations for a given set of features X_i, then calculate a weighted average treatment effect. Our results confirm randomization - there is substantial overlap between the propensity scores of treated and control individuals, so the IPW estimated ATE ends up close to the unweighted ATE. * Third, we calculate an augmented inverse-propensity weighted ATE. The AIPW estimator combines elements of conditional means and inverse propensity weighting to correct for possible bias due to misspecification. The estimated treatment effect for the AIPW estimator is

$$\tau = \mathbb{E}\left[W_i \frac{Y_i - \tau\left(1, X_i\right)}{e\left(X_i\right)} + \left(1 - W_i\right) \frac{Y_i - \tau\left(0, X_i\right)}{\left(1 - e\left(X_i\right)\right)} + \tau\left(1, X_i\right) - \tau\left(0, X_i\right)\right]$$

Confirming our intuition, the AIPW estimator is reasonably close to the IPW and unweighted ATE. When we omit group fixed effects from the estimate,

```
############################
# LOAD DATA
############################
  df <- read_csv('../sample.csv') %>%
    select(-X1) %>%
    mutate(gender = gender - 1) %>%
    filter(!is.na(NGO_donations),
           !is.na(student_wealth))
  itt_out <- list()


############################
# REPLICATE TABLE 3 OF ROMERO ET AL.
############################
  rf1 <- paste0('IRT_score ~ ',
                    # Group fixed effect
```

```
                    'factor(groupid) + ',
                    # Treatment
                    'W + ',
                    # Student fixed characteristics
                    paste('age', 'gender', 'student_wealth', 'factor(grade)', sep = ' + '),
                    ' + ',
                    # School fixed characteristics
                    paste('school_facilities', 'time_to_bank', 'enrollment_2015',
                          'rural', 'NGO_donations', sep = ' + '))

  # Regression function with no group fixed effect
  rf2 <- paste0('IRT_score ~ ',
                  # Treatment
                  'W + ',
                  # Student fixed characteristics
                  paste('age', 'gender', 'student_wealth', 'factor(grade)', sep = ' + '),
                  ' + ',
                  # School fixed characteristics
                  paste('school_facilities', 'time_to_bank',
                        'enrollment_2015', 'rural', 'NGO_donations', sep = ' + '))

############################
# ITT FUNCTION
############################
ITT_est <- function(reg_fun) {
  ############################
  # Calculate ITT OLS, cluster standard errors at school level
  ############################
    lm_out <- lm(reg_fun, data = df)
    itt_unweighted <- coef_test(lm_out, vcov = "CR1",
                                cluster = df$schoolid, test = "naive-t")['W', ]
    itt_unweighted <- tibble(type = 'unweighted',
                      estimate = itt_unweighted$beta,
                      se = itt_unweighted$SE,
                      ci_low = itt_unweighted$beta - 1.96 * itt_unweighted$SE,
                      ci_high = itt_unweighted$beta + 1.96 * itt_unweighted$SE)

  ############################
  # 2 - IPW Estimator
  ############################
    W <- as.matrix(df %>% select(W))

      # Manually create indicator variables for X$grade
      grs <- matrix(nrow = nrow(W), ncol = length(unique(df$grade)), 0L)
      grades <- unique(df$grade)
      for(i in 1:length(grades)) {
        grs[ , i] <- as.numeric(grades[[i]] == df$grade)
      }
      X <- cbind(grs,
                 as.matrix(df %>%
                             select(school_facilities, time_to_bank, enrollment_2015,
                                    rural, NGO_donations, gender, age, student_wealth)))
    # Estimate propensity weights
```

```
  p <- glm(W ~ X, family = "binomial") %>%
      predict(type = 'response')

  # Append propensity weights to df
  df_ipw <- df %>%
    # Calculate propensity weights
    mutate(weight = (W / p) + ((1 - W) / (1 - p)))

  # Estimate IPW regression
  lm_out <- lm(reg_fun, data = df_ipw, weights = weight)
  itt_ipw <- coef_test(lm_out, vcov = "CR1",
                               cluster = df_ipw$schoolid, test = "naive-t")['W', ]
  itt_ipw <- tibble(type = 'IPW',
                    estimate = itt_ipw$beta,
                    se = itt_ipw$SE,
                    ci_low = itt_ipw$beta - 1.96 * itt_ipw$SE,
                    ci_high = itt_ipw$beta + 1.96 * itt_ipw$SE)

############################
# 3 - Double robust estimator
############################
  aipw_fun <- paste0('IRT_score ~ ',
                    # Group fixed effect
                    'factor(groupid) + ',
                    # School fixed characteristics
                    paste('school_facilities', 'time_to_bank', 'enrollment_2015',
                          'rural', 'NGO_donations', sep = ' + '),
                    # Interaction
                    ' + W * (',
                    # Student fixed characteristics
                    paste('age', 'gender', 'student_wealth', 'factor(grade)', sep = ' + '),
                    ')')
  lm_out <- lm(aipw_fun, data = df)

    # Predict - all treated
    df_treatall <- df %>%
      mutate(W = 1)
    y_treatall <- predict(lm_out, df_treatall)

    # Predict - all control
    df_treatnone <- df %>%
      mutate(W = 0)
    y_treatnone <- predict(lm_out, df_treatnone)

    # Predict actual
    actual_pred = predict(lm_out, df)

    # Calculate AIPW estimate
    G <- y_treatall - y_treatnone +
      ((df$W - p) * (df$IRT_score - actual_pred)) / (p * (1 - p))
    tau.hat <- mean(G)
    se.hat <- sqrt(var(G) / (length(G) - 1))
```

```
    # Format output
    itt_aipw <- tibble(type = 'aipw',
                       estimate = tau.hat,
                       se = se.hat,
                       ci_low = tau.hat - 1.96 * se.hat,
                       ci_high = tau.hat + 1.96 * se.hat)

    # Only plot propensity score histogram if reg function contains
    # group fixed effects
    if(str_detect(reg_fun, 'group_id')) {
        plot_tib <- tibble(treatment = factor(W), score = p)
        ggplot(plot_tib) +
            geom_histogram(aes(x = score, y = stat(density), fill = treatment),
                           alpha = 0.3, position = 'identity') +
          labs(x = 'Propensity Score',
               y = 'Density',
               title = 'Logit Propensity Scores')
        ggsave(filename = 'Output/Propensity Histogram.png', device = 'png')
    }

    # Return output
    return(bind_rows(itt_unweighted,
                     itt_ipw,
                     itt_aipw))

}

############################
# FORMAT OUTPUT - ITT
############################
  # 1 - Propensity score overlap
  it1 <- ITT_est(rf1) %>%
    select(-contains('ci'))
  it2 <- ITT_est(rf2)  %>%
    select(-c(contains('ci'), type))
  itt_table <- cbind(it1, it2)


  # 2 - Table of ITT Estimates
  kable(itt_table, digits = 3, format = 'latex') %>%
    add_header_above(c(" " = 1, "Include Group FE" = 2, "Omit Group FE" = 2))
```

Heterogeneous Treatment Effects

We are interested in using machine learning methods to estimate treatment effects in the PSL setting. Machine learning methods enable more flexible estimation of treatment effects, where the non-parametric flexibility allows the models to identify effects that might not have been captured under assumptions driving the previous analysis. We implement the following four methods to estimate treatment effects in the PSL setting:

- **S-Learner**: Estimate $Y(0)$, $Y(1)$ with a single model. In this application, I learn a single model $\hat{\mu}(z)$ using a single random forest that predicts Y_i from $Z_i = (X_i, W_i)$, then estimates the treatment effect

for some feature vector $\hat{\tau}(x) = \hat{\mu}(x,1) - \hat{\mu}(x,0)$. In general, S-learners work well when groups are of substantially different size because the learner pools information about both groups.

- **T-Learner**: The T-learner first separate models $\hat{\mu}_{(i)}(x)$ for treated and control individual $i \in \{0,1\}$, then calculates a treatment effect as the difference $\hat{\tau}(x) = \hat{\mu}_{(1)}(x) - \hat{\mu}_{(0)}(x)$. To develop accurate models across the entire feature space, we need similarly distributions of treated and control observations across $\mathcal{X}$. The T-learner will fail if the density of treated and control observations differ substantially

- **X-Learner**: The X-learner models $Y(0)$ and $Y(0)$ to estimate conditional average treatment effects on the treated and control observations. The model extracts the relationship between outcomes and features in separate forests for treated and control individuals, then uses the model to predict counterfactual outcomes $\hat{\mu}$. We then estimate treatment effects by regressing the difference of individual treatment effects on the covariates. The final estimate is a convex combination of the estimated CATE on treated and CATE on control observations, weighted by the estimated propensity score. The X-learner combines some of the benefits of the S- and T-learners: it fits models for treated and control observations but overcomes regularization bias by regressing the predicted treatment effect on the features. However, the X-learner does not learn treatment effect estimates using propensity scores, so it is still vulnerable to confounding

- **Causal Forest**: Estimates average treatment effects by learning multiple causal trees (partition feature space to maximize contribution to loss function, estimate constant ATE within each leaf, then average over trees to smooth). One of the benefits of the causal forest is the incorporation of propensity weighting to address confoundedness - of particular relevance in this problem.

Having laid out the benefits and disadvantages of each method above, we can predict which methods will perform well in this setting and which methods will struggle. In particular, given strong balance on observables in the treatment and control groups, we might expect that the T-learner will outperform the S- or X-learners.

We learn each model using the entire dataset, given that we are not concerned about out of sample predictions for this exercise (our objective is to recreate within-group average treatment effects without sharing group ids with the model).

```
############################
# PREPARE DATA
############################
  # Split data frame into outcome, features, treatment
  W <- df$W
  Y <- df$IRT_score
  X <- select(df, -W, -IRT_score,-studentid) %>%
    fastDummies::dummy_cols(select_columns = c('grade','schoolid'),
                            remove_first_dummy = TRUE) %>%
    select(-grade,-schoolid, groupid) %>% as.matrix() %>%
    as('dgCMatrix')

############################
# S LEARNER
############################
  # 1 - Calculate S-Learner (see slide 22 of Lecture 4)
  s_learn <- regression_forest(cbind(W, X), Y)
  pred_s_0 <- predict(s_learn, cbind(0, X))$predictions
  pred_s_1 <- predict(s_learn, cbind(1, X))$predictions
  pred_s_oob <- predict(s_learn)$predictions
  pred_s_0[W == 0] <- pred_s_oob[W == 0]
  pred_s_1[W == 1] <- pred_s_oob[W == 1]
```

```
  pred_s <- pred_s_1 - pred_s_0


############################
# T LEARNER
############################
  # 2 - Calculate T-Learner (see slide 21 of Lecutre 4)
  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
  pred_t <- tf.preds.1 - tf.preds.0

############################
# X LEARNER
############################
  # A - Predict Y_i from X_i when W_i == 0 (use T-learner forest 0),
  # Learn \tau_1 by predicting delta from X_i when W_i == 1
  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

  # B - Swap: Predict Y_i from X_i when W_i == 1 (use T-learner forest 1),
  # Learn \tau_0
  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

  # C - Estimate the propensity score - regression forest
  # of W on X
  propf = regression_forest(X, W)  # , tune.parameters = TRUE)
  ehat = predict(propf)$predictions

  # D - Estimate \hat{\tau}(x)
  pred_x = (1 - ehat) * xf.preds.1 + ehat * xf.preds.0


############################
# CAUSAL FOREST
############################
  cf <- causal_forest(X, Y, W, num.trees = dim(X)[1])
  pred_cf <- predict(cf)$predictions

############################
# WITHIN-GROUP ATE
############################
  # Estimate ATE within each matched pair of schools
  group_ate <- df %>%
    # Collapse to pair-treatment group level
```

```
    group_by(groupid, W) %>%
    mutate(avg_score = mean(IRT_score),
           sq_error = (avg_score - IRT_score) ^ 2) %>%
    summarize(avg_score = mean(avg_score),
              count = n(),
              sum_error = sum(sq_error)) %>%
    ungroup() %>%
    # Collapse to pair level to estimate pair treatment effect, error
    mutate(avg_score = if_else(W == 0, -1 * avg_score, avg_score),
           st_err = sum_error / (count - 1)) %>%
    group_by(groupid) %>%
    summarize(treatment_effect = sum(avg_score),
              st_err = sum(st_err),
              count = sum(count)) %>%
    ungroup() %>%
    mutate(st_err = sqrt(st_err))

  ############################
  # Average treatment effects in group
  ############################
    ate_est <- tibble(groupid = df$groupid,
                      ate_s = pred_s,
                      ate_t = pred_t,
                      ate_x = pred_x,
                      ate_cf = pred_cf)
    ate_est <- ate_est %>%
      group_by(groupid) %>%
      summarize_all(mean) %>%
      inner_join(group_ate %>%
                   select(groupid, count, treatment_effect) %>%
                   rename(true_ate = treatment_effect),
                 by = 'groupid') %>%
      select(groupid, count, true_ate, everything())

  # Export output
  write.csv(ate_est, '../Intermediate/2020.06.04 compare ate estimates.csv')
```

Plot Causal Tree

```
#STEP 1. Split the dataset
# Diving the data 40%-40%-20% into splitting, estimation and validation samples
split_size <- floor(nrow(df) * 0.5)
split_idx <- sample(nrow(df), replace=FALSE, size=split_size)

# Make the splits
df_split <- df[split_idx,]
df_est <- df[-split_idx,]

#STEP 2. Fit the tree
#use reg_fun for model
```

```r
ct_unpruned <- honest.causalTree(
  formula=rf2,            # Define the model
  data=df_split,              # Subset used to create tree structure
  est_data=df_est,            # Which data set to use to estimate effects

  treatment=df_split$W,       # Splitting sample treatment variable
  est_treatment=df_est$W,     # Estimation sample treatment variable

  split.Rule="CT",            # Define the splitting option
  cv.option="TOT",            # Cross validation options
  cp=0,                       # Complexity parameter

  split.Honest=TRUE,          # Use honesty when splitting
  cv.Honest=TRUE,             # Use honesty when performing cross-validation

  minsize=10,                 # Min. number of treatment and control cases in each leaf
  HonestSampleSize=nrow(df_est)) # Num obs used in estimation after building the tree

#STEP 3. Cross-Validate
# Table of cross-validated values by tuning parameter.
ct_cptable <- as.data.frame(ct_unpruned$cptable)

# Obtain optimal complexity parameter to prune tree.
selected_cp <- which.min(ct_cptable$xerror)
optim_cp_ct <- ct_cptable[selected_cp, "CP"]

# Prune the tree at optimal complexity parameter.
ct_pruned <- prune(tree=ct_unpruned, cp=optim_cp_ct)

#STEP 4. predict point estimates
tauhat_ct_est <- predict(ct_pruned, newdata=df_est)

#STEP 5. Compute standard errors
# Create a factor column 'leaf' indicating leaf assignment
num_leaves <- length(unique(tauhat_ct_est))  # There are as many leaves as there are predictions
df_est$leaf <- factor(tauhat_ct_est, labels = seq(num_leaves))

# Run the regression
# ols_ct <- lm_robust(IRT_score ~ 0 + leaf + W:leaf, data=df_est)

# ols_ct_summary <- summary(ols_ct)
# te_summary <- coef(ols_ct_summary)[(num_leaves+1):(2*num_leaves), c("Estimate", "Std. Error")]

#STEP 6. Predict point estimates on test set
# tauhat_ct_test <- predict(ct_pruned, newdata=df_test)


rpart.plot(
  x=ct_pruned,        # Pruned tree
  type=3,             # Draw separate split labels for the left and right directions
  fallen=TRUE,        # Position the leaf nodes at the bottom of the graph
  leaf.round=1,       # Rounding of the corners of the leaf node boxes
  extra=100,          # Display the percentage of observations in the node
```

```
  branch=.1,           # Shape of the branch lines
  box.palette="RdBu") # Palette for coloring the node
```

Histograms of treatments by different methodologies

```
par(mfrow=c(2,3))
hist(ate_est$ate_cf, main="Causal Forest: Group-level average of CATE's")
hist(ate_est$ate_t, main="T-learner: Group-level average of CATE's")
hist(ate_est$ate_s, main="S-learner: Group-level average of CATE's")
hist(ate_est$ate_x, main="X-leaner: Group-level Difference-in-means")
hist(ate_est$true_ate, main="Matched pairs: Group-level Difference-in-means")
# computing difference in means by group
# ATE for each group using causal forest -> histogram
# Compare the distribution using
```

Comparison of covariates by decile on groups (which are pairs of schools (which have many observations each))

```
# set number of quantiles
ntiles = 10
#define covariates to get summary statistics
covariates = c('age', 'gender', 'student_wealth', 'school_facilities', 'time_to_bank', 'enrollment_2015

# merge
df_group <- ate_est %>% inner_join(df
                      %>% group_by(groupid) %>% summarise_at(covariates, mean))

#create ranks
df_group$t_decile = ntile(df_group$ate_t, ntiles)
df_group$x_decile = ntile(df_group$ate_x, ntiles)
df_group$decile = ntile(df_group$true_ate, ntiles)


#generate summary statistics
sum_stats_t <- df_group %>% group_by(t_decile) %>% summarise_at(covariates, mean)
sum_stats_x <- df_group %>% group_by(x_decile) %>% summarise_at(covariates, mean)
sum_stats_a <- df_group %>% group_by(decile) %>% summarise_at(covariates, mean)

#print tables
print(sum_stats_t)
print(sum_stats_x)
print(sum_stats_a)

#print tables
xtable(sum_stats_t, digits = 3, type = 'latex')
xtable(sum_stats_x, digits = 3, type = 'latex')
xtable(sum_stats_a, digits = 3, type = 'latex')
```