Problem Set 2

Jacob Light

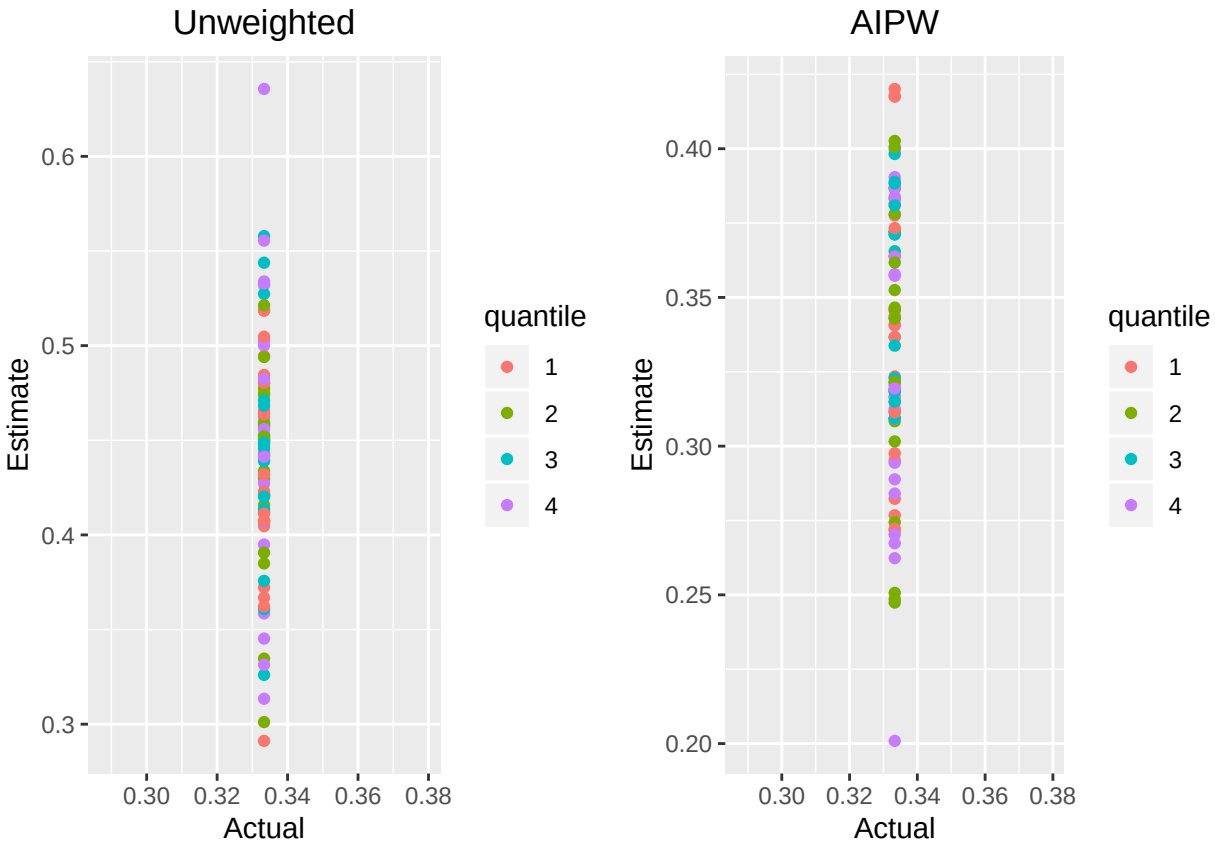
5/16/2020

Evaluating Treatment Heterogeneity

1. Per the instructions, I generate 20 simulated datasets using the data generating process provided in the problem set. For this data generating process, the true ATE treatment effect τ is a constant **1/3**. In each iteration, I estimate a causal forest on the simulated data. Using the causal forest, I estimate out-of-bag treatment effects for each observation. The out-of-bag treatment effect estimates are estimated on the training dataset, rather than a test dataset withheld from the causal forest estimation. I then group the treatment effect estimates into quartiles and calculate two treatment effect estimates for each quartile. First, I calculate sample ATEs: the difference in mean treatment effect estimates within each quartile. Second, I calculate an augmented IPW ATE - an average of the doubly robust estimators which allows us to correct for correlation between X_i and W_i. In the figure below, I plot within-quartile treatment effect estimates and standard errors separately for the sample ATE and AIPW ATEs. There is no detectable correlation between the quartile and treatment effect estimate for either the CATE or AIPW. In the table below, I calculate a simple average of the 20 iterations across quartiles by method. The AIPW estimates are generally more accurate and more precise than the CATE estimates. The standard errors for the quantile average treatment effect are nearly identical across quantiles, and the AIPW standard errors are smaller than the CATE standard errors.

Table 1: Compare Unweighted vs. AIPW - Constant Effect

quantile	simple	aipw
1	0.432	0.337
2	0.434	0.332
3	0.451	0.348
4	0.454	0.324

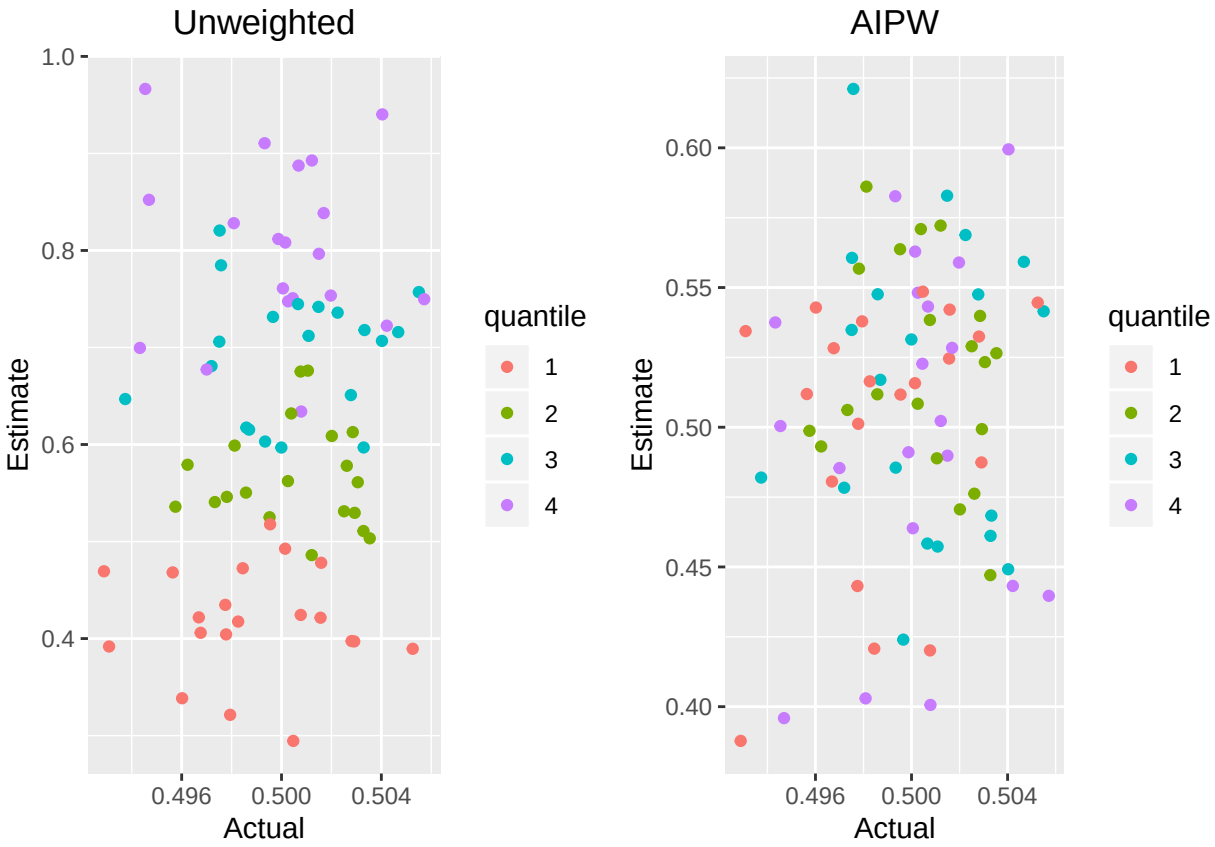



```
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

2. I repeat the analysis above using a the new treatment effect function. In this simulation exercise, the treatment effect is heterogeneous and varies with X2. The correlation between quartile and the average of the estimated CATE in among observations in the quartile is much stronger in the heterogeneous treatment effect simulation - the treatment effect estimates among first-quartile observations are noticeably lower than upper-quartile observations. The correlation is apparent only in the CATE estimates - the AIPW estimates are uncorrelated with quartile, similar to the constant treatment effect case.

Table 2: Compare CATE vs. AIPW - Heterogeneous Effect

quantile	simple	aipw
1	0.418	0.502
2	0.567	0.520
3	0.694	0.514
4	0.801	0.500




```
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

3. Following the tutorial, the test_calibration function allows us to evaluate the fit and heterogeneity of a model estimated using a causal forest. The test_calibration function estimates the model:

$$Y_i - \hat{m}^{-i}(X_i) = \alpha\bar{\tau}\left(W_i - \hat{e}^{-i}(X_i)\right) + \beta\left(\hat{\tau}^{-i}(X_i) - \bar{\tau}\right)\left(W_i - \hat{e}^{-i}(X_i)\right) + \epsilon \qquad \bar{\tau} := \frac{1}{n}\sum_{i=1}^{n}\hat{\tau}^{-i}(X_i)$$

where $a = 1$ confirms the average prediction of the forest and $\beta = 1$ confirms that the forest captures underlying heterogeneity. The β estimate allows us to test for heterogeneity - $\beta > 0$ allows us to reject treatment homogeneity. The table below summarizes averages of the point estimates for α and β, standard error of the mean parameter estimates, and p-values. For both treatment effect processes,

Table 3: Calibration for CATE case 1

type	mean	se	t_stat	pval
alpha	1.00	0.004	226.01	0.000
beta	-0.05	0.429	-0.11	0.543

Table 4: Calibration for HTE case 2

type	mean	se	t_stat	pval
alpha	1.00	0.003	331.55	0
beta	0.73	0.196	3.73	0

we confirm that the average prediction of the forest is correct (α significantly differs from 0 in the one-sided t-test, close to 1). In the constant treatment effect case ($\tau = 1/3$), we fail to reject the null hypothesis of a constant treatment effect. In contrast, in the case of the heterogeneous treatment effect, we are able to reject the null hypothesis of a CTE.

4. In this exercise, we used a causal forest to split simulated data into subgroups ordered by the treatment effect estimate, then, in each subgroup, estimate a subgroup treatment effect. When the treatment effect is truly constant, as in the first part of the problem, the unweighted and AIPW methods correctly recover that the treatment effect does not vary across subgroups. In the second part of the problem, we introduce heterogeneity in the treatment effect. Because the size of the treatment effect is correlated with the feature space, splitting and calculating an unweighted average treatment effect will lead to correlation between the subgroup and subgroup average treatment effect. This effect goes away when we propensity weight. Ultimately, in some applications, we might be interested in identifying heterogeneous treatment effects in an experiment. Grouping by well-defined features allows for estimation of CATEs local to the subgroup. If we want to use the data to split subgroups, we want to be sure to address potential confoundedness through some sort of propensity score correction.

HTEs in Observational Studies

For this problem, I use output from the STAR experiment. As in the last problem set (but implemented *correctly* in this problem set), I confound treatment assignment by free/reduced lunch status and gender: I drop free/reduced lunch status students from the control group and non-free/reduced lunch status from the treatment group, girls from the control group and boys from the treatment group. Based on previous studies, I would expect that the confounded assignment would increase a naive treatment effect estimate compared to an estimate that controls for confounded assignment.

I run a series of estimation procedures on different subsets of the data. The subsets play with the total number of observations and the balance between treated and control observations in the dataset. I will first briefly summarize the methods used in estimation, then compare output across methods and data subsamples.

- **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

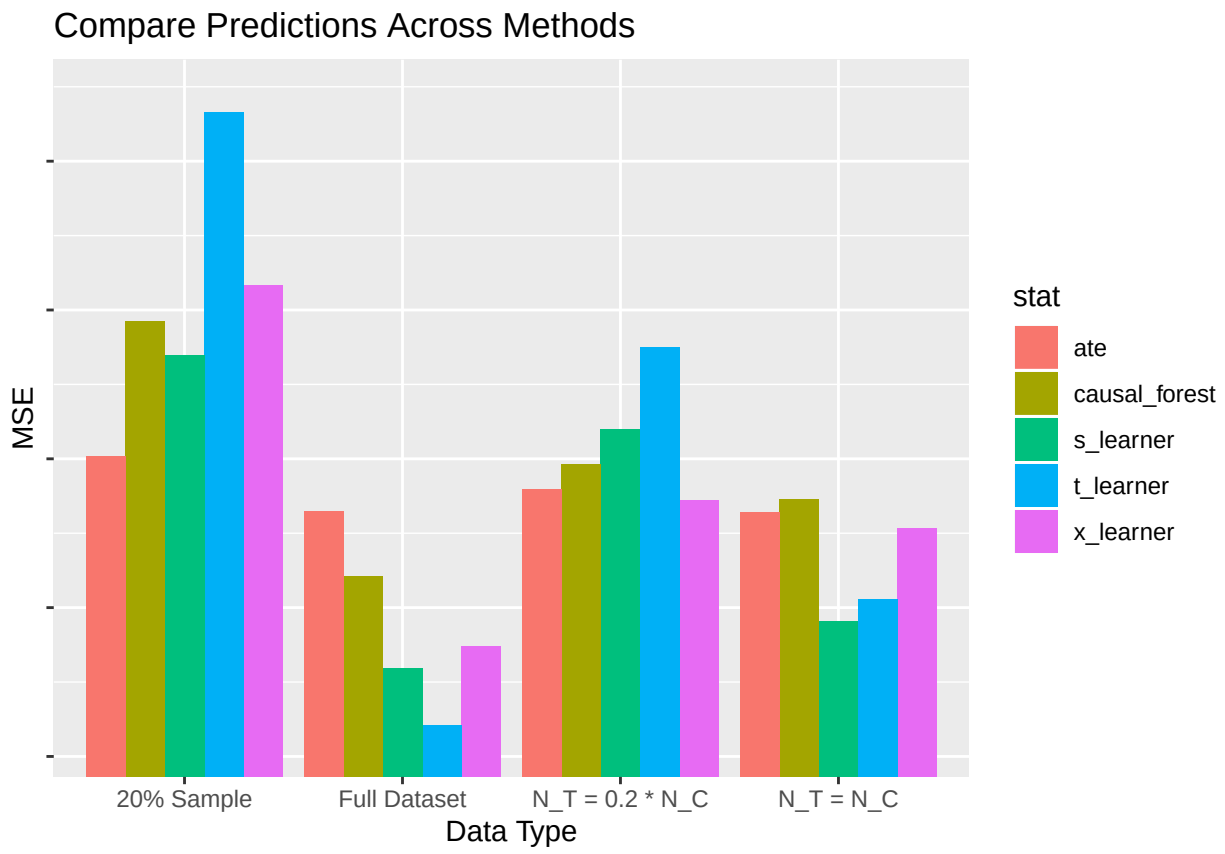
- **X-Learner**: The X-learner models $Y(0)$ and $Y(0)$ to estimate conditional average treatment effects on the treated and control observations. The model extracts the relationship between outcomes and features in separate forests for treated and control individuals, then uses the model to predict 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.

Based on the summaries above, we might expect that the T-, S-, and X-Learners will all struggle to estimate treatment effects due to confounded assignment. We might also expect that the S-Learner will perform best when the treatment and control groups are of dissimilar size, while the T-Learner will perform best when the treatment and control groups are of similar density in the feature space.

To compare predictions, I follow the method outlined in the tutorial - I test the out of sample fit on each of the methods, then compare the MSE between Y_i^* and $\hat{\tau}(X_i)$ for each of the different methods, with a naive ATE as the baseline estimate. In the full dataset, the T-learner performs the worst due to an imbalance in density of features between the treated and control variables - a mechanical consequence of the manual confounding exercise. The T-learner issue persists when I trim the data to a 20% stratified sample. As we expected, the S-learner performs relatively well in the case where there is a large imbalance between the number of treated and control observations. The X-learner inherits some issues from the T- and S-learners, so it seems reasonable that the X-learner generally performs between the two.

	ATE		S-Learner		T-Learner		X-Learner		Forest	
Data Type	Mean	SE	Mean	SE	Mean	SE	Mean	SE	Mean	SE
Full Dataset	321.6	835.9	319.0	830.8	318.0	825.6	320.5	830.0	319.3	828.0
20% Sample	322.5	832.4	324.2	839.1	328.3	845.2	324.8	839.6	325.4	839.6
N_T = N_C	321.6	827.8	319.8	821.7	320.1	814.6	321.8	821.1	321.3	819.9
N_T = 0.2 * N_C	322.0	828.8	323.0	832.5	324.4	837.5	322.4	829.0	321.8	830.7



HTEs in Randomized Experiments

1. Following the lecture notes and instructions, I split the unconfounded STAR dataset into three subsamples: I allocate 40% of observations to a training dataset, 40% to an estimate dataset, and 20% to a test dataset. I then estimate an honest causal tree using the train and estimate datasets. I needed to decrease the minimum leaf size with comparison to the tutorial: I restrict the tree to leaves with at least 10 treated and 10 control observations per leaf. To avoid over-fitting, I cross-validate with the estimation sample. The cross-validation procedure selects a tuning parameter to penalize over-fitting. I then prune the tree using the penalty term that comes from the cross-validation exercise. For each of the train, estimate, and test datasets, I predict point estimates and standard errors in a regression of test score on dummies for each leaf and the interaction of the leaf-treatment dummy. I estimate the regressions separately for each dataset and present the point estimates and robust standard errors in the table below. The leaf-treatment interaction term captures the ATE in each leaf. I observe wide variation in the ATE estimates across leaves, with particularly wide variation in the train sample and substantially smaller variation in the test sample. The standard errors, on the other hand, are larger in the test sample because the data are more sparse. Standard errors on the ATEs are lowest in the estimate sample.

```
## [1] 2
## [1] "CT"
```

2. Next, I estiamte heterogeneous treatment effects on the STAR data using a causal forest. I split the STAR data into train (80%) and test (20%) samples. I estimate causal forests on the entire train dataset, then draw random 90%, 70%, and 50% samples of the train dataset. I estimate causal forests on

Table 5: Compare Leaf x Treatment Interaction

	Train		Est		Test	
	Estimate	Std. Error	Estimate	Std. Error	Estimate	Std. Error
leaf1:treatment	0.41	3.02	-1.43	2.03	-3.43	4.33
leaf2:treatment	-15.63	5.51	2.60	4.85	4.69	6.04
leaf3:treatment	4.37	4.21	3.74	4.58	0.75	5.08
leaf4:treatment	3.41	1.12	4.25	1.09	7.28	1.52
leaf5:treatment	21.75	6.69	17.36	6.01	8.00	6.96

Table 6: STAR HTE for full and subsample

predictions	variance.estimates	frac
Group 1		
-7.6540645	30.788975	1.0
-5.8727920	35.405526	0.9
0.7489043	12.390947	0.7
-5.9393302	35.139420	0.5
Group 2		
3.9500259	5.721035	1.0
3.6620654	4.043090	0.9
1.0473716	4.432727	0.7
1.9359922	4.142381	0.5
Group 3		
2.9522422	3.775358	1.0
3.4998274	6.054090	0.9
1.3261254	2.409771	0.7
4.0652450	3.096950	0.5

each of the samples. I select a constant set of unique X_i vectors from the test sample for out-of-sample predictions. I do not report the features for readability but produce the point estimates and variance estimates in the table below, where rows with the same 'id' correspond to the same characteristics vector. I find the results somewhat surprising - the point estimates and variance estimates bounce around as I reduce the sample and there is no clear monotonicity in the variance estimates as I shrink the sample. The variation in point estimates suggests sensitivity of the causal forest to the sample and may reflect instability of the estimation. I am surprised that variance is not monotonically increasing in sample size. I do not have a strong explanation for this outcome.

```
##############################
# CODE FOR PROBLEM 1
##############################

run1 <- function(x, tau_fn) {
    # Data generating process
    n = 8000; p = 6
    X = matrix(rnorm(n*p), n, p)
    tau = apply(X, 1, tau_fn)
    W = rbinom(n, 1, 1/(1 + exp(-X[,1] / 2)))
    Y = pmax(0, X[,1] / 2 + X[,2]) + W * tau + rnorm(n)

    # 1 - train a causal forest, predict CATE
```

```
  dat <- data.frame(cbind(X, Y, W))
  colnames(dat) <- c('X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'Y', 'W')
  for_test <- causal_forest(dat[ , 1:6], dat$Y, dat$W)
  cate <- predict(for_test, estimate.variance = TRUE)

  # 2 - sort CATEs by quartile
  ols_sample_ate <- tibble(Y = Y,
                           W = W,
                           oob_cate = cate$predictions,
                           oob_error = cate$excess.error,
                           quantile = as.factor(ntile(cate$predictions, 4))) %>%
    arrange(quantile)
  ols_out <- lm(Y ~ -1 + quantile + quantile:W, data = ols_sample_ate)
  ate_estimate <- coef(summary(ols_out)) %>%
    as.data.frame() %>%
    # Restrict to interaction terms, pull point estimate and standard error
    select(Estimate, 'Std. Error') %>%
    mutate(val = rownames(.)) %>%
    filter(str_detect(val, 'W'))

  # 3 - AIPW
  aipw_ate <- lapply(unique(ols_sample_ate$quantile),
                     function(w) average_treatment_effect(for_test,
                                                          subset = ols_sample_ate$quantile == w))
  aipw_ate <- data.frame(do.call(rbind, aipw_ate))

  # 4 - Actual CATE
  actual <- tibble(tau = tau,
                   quantile = ols_sample_ate$quantile) %>%
    group_by(quantile) %>%
    summarize(cate = mean(tau)) %>%
    ungroup()

  # 4 - Return
  compare_cate <- tibble(quantile = unique(ols_sample_ate$quantile),
                         actual = actual$cate,
                         ols_estimate = ate_estimate$Estimate,
                         aipw_estimate = aipw_ate$estimate)

  # 5 - Calibration test
  tc <- test_calibration(for_test)

  return(list(iter = x,
              datagen = dat,
              ols = ols_out,
              forest = for_test,
              output = compare_cate,
              calibr = tc))
}

# Test two tau functions
tau1 <- function(x) {1/3}
tau2 <- function(x) { 1 / (1 + exp(-x[2]/2))}
```

```r
  # Run 20 random forests using each tau function
  out1 <- list()
  NUM <- 20
  out1[['tau1']] <- lapply(1:NUM, run1, tau_fn = tau1)
  out1[['tau2']] <- lapply(1:NUM, run1, tau_fn = tau2)

  # Produce figures 1-2, table 1
  plot_fig1 <- function(df) {
    df <- df %>%
      lapply(., function(x) x[['output']]) %>%
      bind_rows()

    ret <- list()
    f <- list()
    f[['simple_plot']] <- ggplot(data = df, aes(x = actual, y = ols_estimate)) +
      geom_point(aes(color = quantile)) +
      labs(y = "Estimate",
           x = "Actual",
           title = "Unweighted") +
    theme(plot.title = element_text(hjust = 0.5))
    f[['aipw_plot']] <- ggplot(data = df, aes(x = actual, y = aipw_estimate)) +
      geom_point(aes(color = quantile)) +
      labs(y = "Estimate",
           x = "Actual",
           title = "AIPW") +
      theme(plot.title = element_text(hjust = 0.5))
    ret[['figure']] <- grid.arrange(f[[1]], f[[2]], nrow = 1)
    ret[['table']] <- df %>%
      group_by(quantile) %>%
      summarize(simple = mean(ols_estimate),
                aipw = mean(aipw_estimate))

    return(ret)

##############################
# CODE FOR PROBLEM 2
##############################
    run2 <- function(df, test) {
    # Remove student ID from data frame
    df <- df %>%
      select(-student_id_char) %>%
      # Restrict to numeric columns
      select(-contains('fac'))
    test <- test %>%
      select(-student_id_char) %>%
      # Restrict to numeric columns
      select(-contains('fac'))


    # Split data frame into outcome, features, treatment
    W <- df$treatment
    Y <- df$score
    X <- df %>%
```

```
  select(-c(treatment, score))

# Repeat for test set
W_test <- test$treatment
Y_test <- test$score
X_test <- test %>%
  select(-c(treatment, score))

# 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

  # Predict on test data
  tauhat_s <- predict(s_learn, cbind(1, X_test))$predictions -
    predict(s_learn, cbind(0, X_test))$predictions


# 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

  # Predict on test data
  tauhat_t <- predict(tf1, X_test)$predictions - predict(tf0, X_test)$predictions


# 3 - Calculate 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

    # E - Predict
    ehat.test <- predict(propf, X_test)$predictions
    xf.preds.1.test <- predict(xf1, X_test)$predictions
    xf.preds.0.test <- predict(xf0, X_test)$predictions
    tauhat_xl_test <- (1 - ehat.test) * xf.preds.1.test + ehat.test * xf.preds.0.test

  # 4 - Estimate Causal Forest
  cf <- causal_forest(X, Y, W, num.trees = dim(X)[1])
  pred_cf <- predict(cf)$predictions
  cf_test <- predict(cf, newdata = X_test)$predictions

  # 5 - For comparison, calculate ATE
  tauhat_sample_ate <- with(df, mean(Y[W==1]) - mean(Y[W==0]))

  # 6 - Compare predictions through R-loss
  Y.forest.test = regression_forest(X = as.matrix(X_test), Y = Y_test)
  Y.hat.test = predict(Y.forest.test)$predictions
  W.forest.test = regression_forest(X = as.matrix(X_test), Y = W_test)
  W.hat.test = predict(W.forest.test)$predictions

  mse_rloss <- data.frame(
    ate = (Y_test - Y.hat.test - (W_test - W.hat.test) * tauhat_sample_ate)^2,
    s_learner = (Y_test - Y.hat.test - (W_test - W.hat.test) * tauhat_s)^2,
    t_learner = (Y_test - Y.hat.test - (W_test - W.hat.test) * tauhat_t)^2,
    causal_forest = (Y_test - Y.hat.test - (W_test - W.hat.test) * cf_test)^2,
    x_learner = (Y_test - Y.hat.test - (W_test - W.hat.test) * tauhat_xl_test)^2
  )

# Prep output
out <- bind_rows(mse_rloss %>%
    summarize_all(mean) %>%
    mutate(stat = 'mean'),
    mse_rloss %>%
    summarize_all(sd) %>%
    mutate(stat = 'standard error')) %>%
  select(stat, everything())

  # 5 - Package and return
  return(out)
  }

# Apply run2 to the confounded star data
 samp <- sample_frac(data.frame(id = 1:nrow(data_confound)), 0.2)
 df_conf_test <- data_confound[samp$id, ]
 df_conf_train <- data_confound[-samp$id, ]

 # APPLY RUN2 TO SUBSAMPLES
 out2 <- list()
```

```r
# A - Full confounded dataset
out2[['full']] <- run2(df_conf_train, df_conf_test)
# B - Stratified 20% sample
out2[['sample_20']] <- run2(df_conf_train %>%
                            group_by(treatment) %>%
                            sample_frac(0.2) %>%
                            ungroup(), df_conf_test)
# C - Equal number of treated and control
out2[['equal_n']] <- run2(df_conf_train %>%
                            group_by(treatment) %>%
                            sample_n(min(sum(df_conf_train$treatment),
                                         sum(1 - df_conf_train$treatment))) %>%
                            ungroup(), df_conf_test)
# D - More control
out2[['more_control']] <- run2(rbind(df_conf_train %>%
                                       filter(treatment == 0),
                                   df_conf_train %>% filter(treatment == 1) %>%
                                     sample_n(floor(sum(1 - data_confound$treatment) / 5))),
                               df_conf_test)
quick_fn <- function(d) {
  d %>%
    mutate(id = 1,
           stat = str_replace_all(stat, ' ', '_')) %>%
    pivot_wider(id_cols = id,
                names_from = stat,
                values_from = colnames(d)[!(colnames(d) %in% c('stat', 'id'))]) %>%
    select(-id)
}
out2_table <- bind_rows(lapply(out2, quick_fn)) %>%
  mutate(descr = c('Full Dataset', '20% Sample', 'N_T = N_C', 'N_T = 0.2 * N_C')) %>%
  select(descr, everything())
kable(out2_table,
      digits = 1,
      col.names = c('Data Type', 'Mean', 'SE', 'Mean', 'SE', 'Mean',
                    'SE', 'Mean', 'SE', 'Mean', 'SE')) %>%
  add_header_above(c(" " = 1, "ATE" = 2, "S-Learner" = 2, 'T-Learner' = 2,
                     'X-Learner' = 2, 'Forest' = 2)) %>%
  kable_styling("striped")

out2_long <- out2_table %>%
  pivot_longer(cols = colnames(out2_table)[colnames(out2_table) != 'descr'],
               names_to = 'stat',
               values_to = 'val') %>%
  filter(str_detect(stat, 'mean')) %>%
  mutate(stat = str_replace(stat, '_mean', ''))
ggplot(out2_long, aes(x = descr, y = val, fill = stat)) +
  geom_bar(position="dodge", stat="identity") +
  coord_cartesian(ylim = c(min(out2_long$val) * .999 , max(out2_long$val) * 1.001)) +
  labs(y = 'MSE',
       x = 'Data Type',
       title = 'Compare Predictions Across Methods') +
  theme(axis.text.y = element_blank())
```

```r
################################
# CODE FOR PROBLEM 3
################################
    # Remove factor variables and identifier from star_data
    star_data <- star_data %>%
      select(-contains('fac'), -student_id_char)

    # Split data frame into train, test, est subsamples
    splits <- floor(runif(nrow(star_data)) * 2.5)
    df_tr <- star_data[splits == 0, ]
    df_est <- star_data[splits == 1, ]
    df_test <- star_data[splits == 2, ]

    # Honest causal tree
    formul <- paste0('score ~ ', paste0(colnames(star_data %>% select(-score)), collapse = " + "))
    ct_unpruned <- honest.causalTree(
      formula = formul,
      data = df_tr,
      est_data = df_est,
      treatment = df_tr$treatment,
      est_treatment = df_est$treatment,
      split.Rule = 'CT',
      cv.option = 'TOT',
      split.Honest = TRUE,
      cv.Honest = TRUE,
      minsize = 20,
      HonestSampleSize = nrow(df_est)
    )

    # Cross-Validation
    ct_cptable <- as.data.frame(ct_unpruned$cptable)

    # Obtain optimal complexity parameter to prune tree.
    selected_cp <- which.min(ct_cptable$xerror[2:nrow(ct_cptable)]) + 1
    optim_cp_ct <- ct_cptable[selected_cp, "CP"]

    # Prune
    ct_pruned <- prune(tree = ct_unpruned, cp = optim_cp_ct)

    # Function - Create a factor variable for the leaves, run linear regression to
    # estimate treatment effect magnitudes and standard errors for each leaf
    leaf_error <- function(df) {
      # Predict treatment effect on estimation subsample
      tauhat_ct_est <- predict(ct_pruned, newdata = df)

      # Calculate standard errors
      num_leaves <- length(unique(tauhat_ct_est))
      df$leaf <- factor(tauhat_ct_est, labels = seq(num_leaves))

      # Run the regression
      ols_ct <- lm_robust(score ~ 0 + leaf + treatment:leaf, data = df, se_type = 'HC1')
      ols_ct_summary <- summary(ols_ct)
      te_summary <- coef(ols_ct_summary)[(num_leaves+1):(2*num_leaves), c("Estimate", "Std. Error")]
```

```
    # Return
    return(te_summary)
  }

  # Leaf ATEs
  leaf_te <- list()
  leaf_te[['train']] <- leaf_error(df_tr)
  leaf_te[['est']] <- leaf_error(df_est)
  leaf_te[['test']] <- leaf_error(df_test)
  leaf_table <- cbind(leaf_te[[1]], leaf_te[[2]], leaf_te[[3]])

  # Print output
  kable(leaf_table,
        digits = 2,
        caption = 'Compare Leaf x Treatment Interaction') %>%
      add_header_above(c(" " = 1, "Train" = 2, "Est" = 2, "Test" = 2)) %>%
    kable_styling("striped")

# USE CAUSAL FOREST TO ESTIMATE HTEs
  df_tr <- rbind(df_tr, df_est)

  # Select a few covariates from the test dataset to feed into tree
  cf_test <- df_test %>%
    select(-c(treatment, score)) %>%
    unique() %>%
    sample_n(20) %>%
    mutate(id = row.names(.))

  # CF estimator
  cf_estimator <- function(d, points) {
    # Estimate causal forest
    star_cf <- causal_forest(X = d %>% select(-c(score, treatment)),
                             W = d$treatment,
                             Y = d$score,
                             num.trees = nrow(d))
    # Prediction OOB
    # oob_star <- predict(star_cf, estimate.variance = TRUE)

    # Predict on test set
    test_star <- predict(star_cf,
                         newdata = as.matrix(points %>% select(-id)),
                         estimate.variance = TRUE) %>%
      mutate(id = points$id)
    return(test_star)
  }

  fracs <- c(1, .9, .7, .5)
  out <- lapply(lapply(fracs, function(x) sample_frac(df_tr, x)), cf_estimator, cf_test)
  for(i in 1:length(out)) {
    out[[i]] <- out[[i]] %>% mutate(frac = fracs[i])
  }
  out <- bind_rows(out) %>%
    arrange(id, -frac)
```

```r
kable(out %>% filter(id %in% sample(1:max(out$id), 3)) %>% select(-id),
      caption = "STAR HTE for full and subsample") %>%
  kable_styling("striped") %>%
  group_rows("Group 1", 1, 4) %>%
  pack_rows("Group 2", 5, 8) %>%
  pack_rows('Group 3', 9, 12)
```