Project 4: Causal Inference Algorithms Evaluation

Group 8

03/31/2021

Setup

First, we set working directories as needed, install required libraries and import the data.

Introduction

In this project, we are looking for the best algorithm for causal inference of propensity scores to see how close the estimated average treatment effects (ATEs) are to the true ATEs. For the estimation of propensity scores, we use regression trees. The algorithms we use to estimate ATEs in this project are propensity matching, stratification and weighted regression. We also compare the run times of different algorithms and propensity score estimations across both data sets.

About the Data

There are two attached data sets for this project, named *highDim_data set* and *lowDim_data set*, which were from the Atlantic Causal Inference Conference (ACIC) Data Challenge. For the high dimensional data, there are 2000 observations, 185 variables ($V1$-$V185$), 1 treatment indicator variable (A) and 1 continuous response variable (Y). For the low dimensional data, there are 475 observations, 22 variables ($V1$-$V22$), 1 treatment indicator variable (A) and 1 continuous response variable (Y).

```
## [1] "High Dimensional Data"
```

```
##
##      0      1
## 0.6785 0.3215
```

```
## [1] "Low Dimensional Data"
```

```
##
##     0     1
## 0.788 0.212
```

From the tables above, we see that the control groups are not balanced for both data sets. For the high dimension data, the treated group consists of 67.85% of the samples. For the low dimension data, the treated group consists of 78.8% of the samples.

For the purpose of comparing the different algorithms, we chose to add weights to the observations while estimating the propensity scores using classification/regression trees.

Background

Regression Trees

The mathematical formula for a regression tree is shown below.

$$\hat{f}(x) = \sum_{m=1}^{M} c_m I\{x \in R_m\}$$

Here, R_m is a specific region, M is the number of regions, and c_m is the value associated with a region.

The way regression trees work is that the space is split into multiple regions based on some set of decisions. This process keeps repeating until a stopping rule is applied. In regression, we use **squared error loss** to find the optimal tree model, but in classification, as with the case for estimating propensity scores, we use a measure of impurity. In our case, we use one called the **Gini index**. In R, we use "rpart" library to run the regression tree algorithm. The parameters of the function includes **minsplit**, **minbucket** and **cp**. The parameter of **cp** indicates the complexity of the model, we get more complex trees with lower values of "cp".

We can visualize the tree and the decision rules at each split, using the **plot.rpart** function in R. The decision rules help to determine which variables are of the highest importance when estimating the propensity scores.

Propensity Scores

The propensity score is defined as follows:

$$e(x) = Pr(T = 1|X = x),\ 0 < e(x) < 1$$

Based on the formula above, given the (multiple) covariates (x), the propensity score is the probability that the observation is in the treatment group (in our case, observations where $A = 1$). Since, our data is based on observational studies, we can use propensity scores to make causal inferences.

Average Treatment Effect (ATE)

The average treatment effect is defined as follows:

$$\Delta_t = E(Y_1 - Y_0|T = 1)$$

ATE is defined as the difference in the average outcomes between observations assigned to treatment group and the control group. This allows us to measure the effect a treatment had on each group.

Cross-Validation

We perform five fold cross-validation for the high dimension and low dimension data sets. The main objective of the cross validation is to tune the "cp" parameter to avoid overfitting.

Step 1: Set Controls and Establish Hyperparameters

We set up the controls to start the the cross validation process

```
K <- 5  # number of CV folds
sample.reweight <- TRUE # run sample reweighting in model training

# setting the following to false loads data generated from a previous run
# this data is the same in each run due to a set seed

run.cv.trees_high <- FALSE # run cross-validation on the training set for trees on high dim data

run.cv.trees_low <- FALSE # run cross-validation on the training set for trees on low dim data
```

We choose a set of "cp" values here to cross validate in order to find the optimal "cp" value for each data set; here we choose to do powers of two.

```
# hyperparameters for trees
hyper_grid_trees <- expand.grid(
  cp = c(2^(0), 2^(-1), 2^(-2), 2^(-3), 2^(-4),
         2^(-5), 2^(-6), 2^(-7), 2^(-8), 2^(-9),
         2^(-10), 2^(-11), 2^(-12), 2^(-13), 2^(-14),
         2^(-15), 2^(-16), 2^(-17), 0, -2^(0))
)
```

Step 2: Cross-Validate the Hyperparameters

We source the library functions that we created to help cross validate the "cp" hyperparamter.

```
# data pre-processing

# features are the predictors: V1 - Vp
# column 1 is the response Y
# column 2 is the treatment A

feature_train_high = df_high[, -1:-2]
label_train_high = df_high[, 2]

feature_train_low = df_low[, -1:-2]
label_train_low = df_low[, 2]
```

High Dimensional Data We run the cross validation algorithm on the high dimensional data.

```
set.seed(5243)

if(run.cv.trees_high){
  res_cv_trees_high <- matrix(0, nrow = nrow(hyper_grid_trees), ncol = 4)
  for(i in 1:nrow(hyper_grid_trees)){
    cat("complexity = ", hyper_grid_trees$cp[i], "\n", sep = "")
    res_cv_trees_high[i,] <- cv.function(features = feature_train_high,
                                         labels = label_train_high,
                                         cp = hyper_grid_trees$cp[i],
                                         K, reweight = sample.reweight)
  save(res_cv_trees_high, file = "../output/res_cv_trees_high.RData")
  }
```

```
} else{
  load("../output/res_cv_trees_high.RData")
}
```

Low Dimensional Data We run the cross validation algorithm on the high dimensional data.

```
set.seed(5243)

if(run.cv.trees_low){
  res_cv_trees_low <- matrix(0, nrow = nrow(hyper_grid_trees), ncol = 4)
  for(i in 1:nrow(hyper_grid_trees)){
    cat("complexity = ", hyper_grid_trees$cp[i], "\n", sep = "")
    res_cv_trees_low[i,] <- cv.function(features = feature_train_low,
                                        labels = label_train_low,
                                        cp = hyper_grid_trees$cp[i],
                                        K, reweight = sample.reweight)
  save(res_cv_trees_low, file="../output/res_cv_trees_low.RData")
  }
}else{
  load("../output/res_cv_trees_low.RData")
}
```

Step 3: Visualize CV Error and AUC

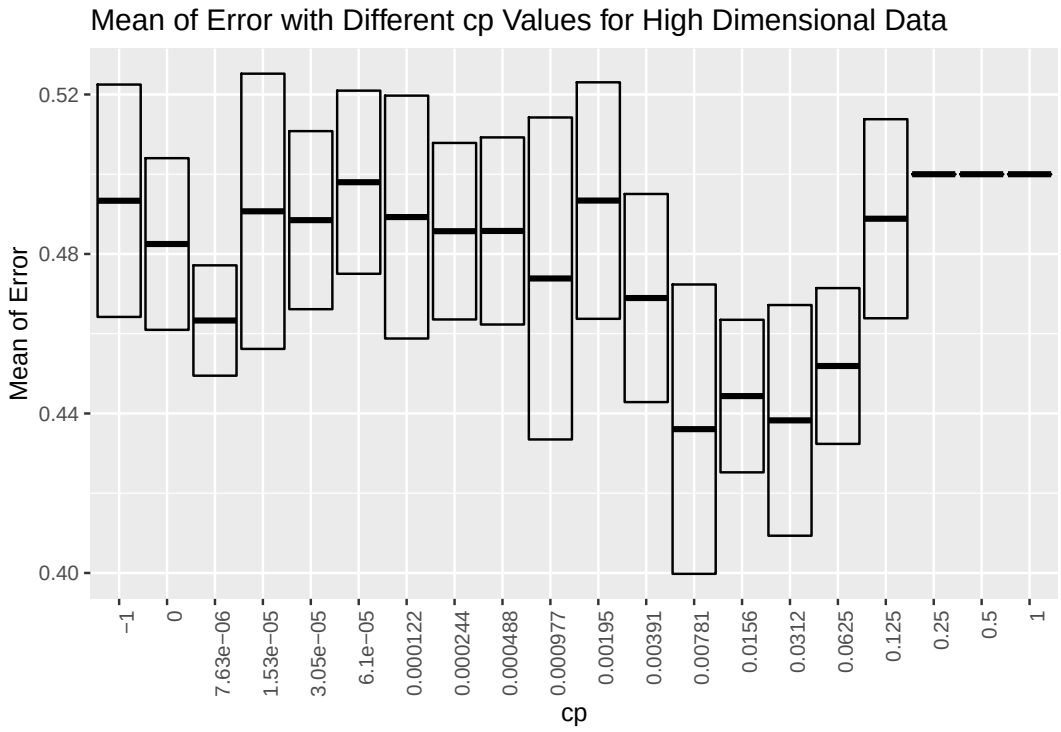
After cross validating, we obtain the mean error and the AUC values for each potential "cp" value for both data sets. We display these values and associated standard errors in the plots below.

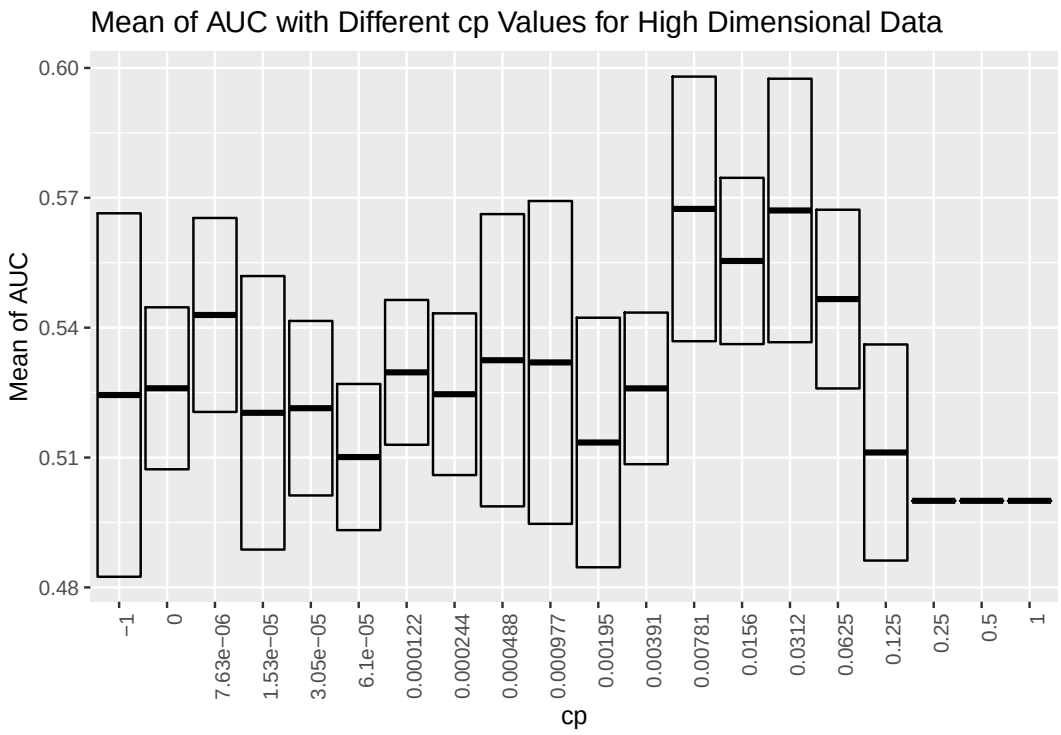
Because of the imbalances in the groups for both datasets, we choose to not only weigh our observations, but to also focus on mean AUC when selecting the optimal hyperparameter.

High Dimensional Data Based on the plots and table below, we find that the model with the highest mean AUC value has a "cp" value of 2^{-7}.

```
# create data frame to organize results
res_cv_trees_high <- as.data.frame(res_cv_trees_high)
colnames(res_cv_trees_high) <- c("mean_error", "sd_error", "mean_AUC", "sd_AUC")
cv_results_trees_high = data.frame(hyper_grid_trees, res_cv_trees_high)

# look at top 5 models with highest AUC
cv_results_trees_high[order(cv_results_trees_high$mean_AUC, decreasing = TRUE), ][1:5, ]
```

```
##              cp mean_error   sd_error  mean_AUC     sd_AUC
## 8  7.812500e-03  0.4360668 0.03628091 0.5674343 0.03056252
## 6  3.125000e-02  0.4382665 0.02892494 0.5670732 0.03043522
## 7  1.562500e-02  0.4443627 0.01912652 0.5554088 0.01920727
## 5  6.250000e-02  0.4519064 0.01953579 0.5466031 0.02064263
## 18 7.629395e-06  0.4633120 0.01383644 0.5429351 0.02240466
```





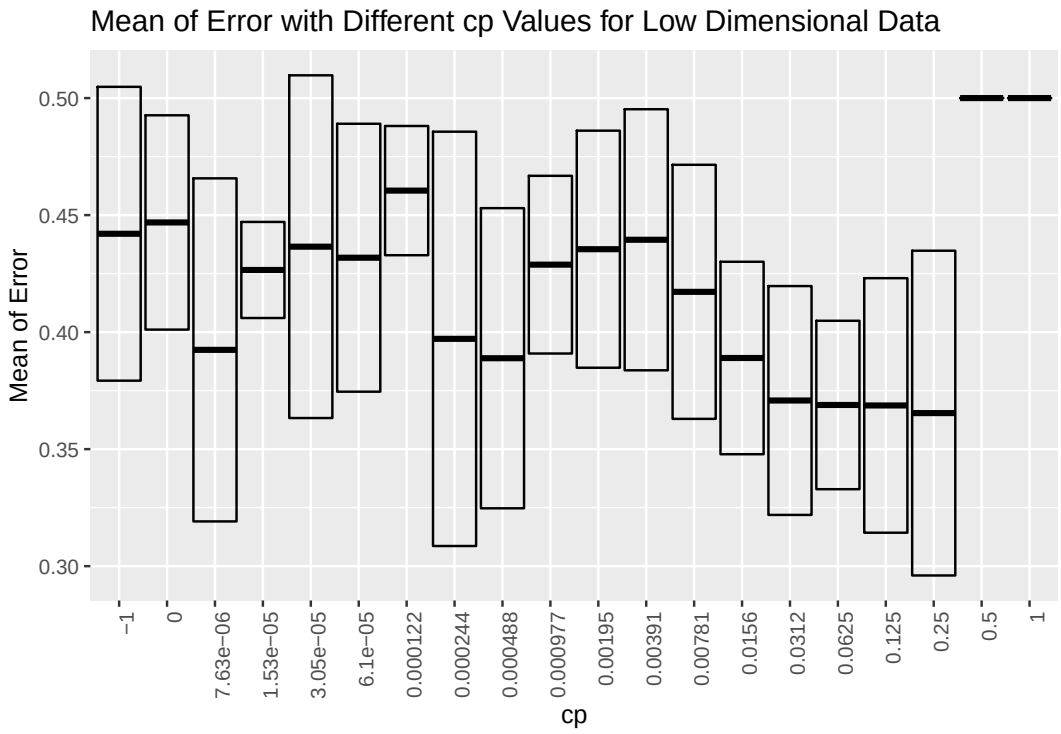

```
best_cp_high <- cv_results_trees_high$cp[cv_results_trees_high$mean_AUC ==
                                          max(cv_results_trees_high$mean_AUC)]

best_cp_high
```

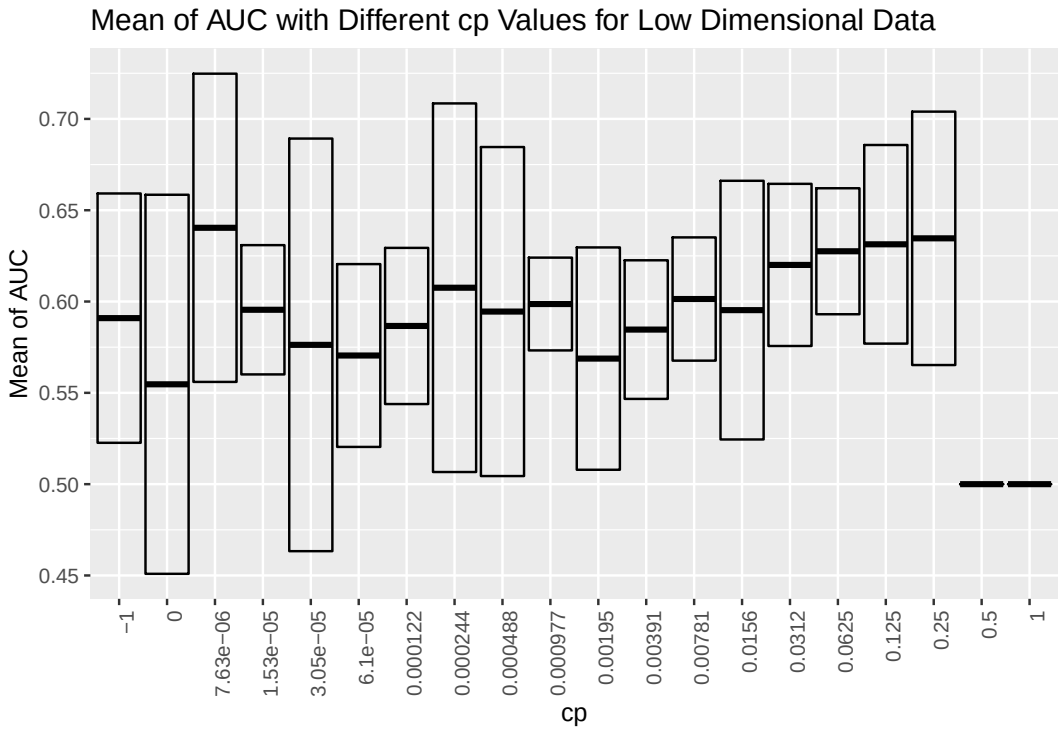
```
## [1] 0.0078125
```

Low Dimensional Data Based on the plots and table below, we find that the model with the highest mean AUC value has a "cp" value of 2^{-18}.

```
# create data frame to organize results
res_cv_trees_low <- as.data.frame(res_cv_trees_low)
colnames(res_cv_trees_low) <- c("mean_error", "sd_error", "mean_AUC", "sd_AUC")
cv_results_trees_low = data.frame(hyper_grid_trees, res_cv_trees_low)

# look at top 5 models with lowest AUC
cv_results_trees_low[order(cv_results_trees_low$mean_AUC, decreasing = TRUE), ][1:5, ]
```

```
##              cp mean_error   sd_error  mean_AUC     sd_AUC
## 18 7.629395e-06  0.3924126 0.07329778 0.6403787 0.08438962
## 3  2.500000e-01  0.3653965 0.06938198 0.6346035 0.06938198
## 4  1.250000e-01  0.3686625 0.05437925 0.6313375 0.05437925
## 5  6.250000e-02  0.3688654 0.03598451 0.6275346 0.03449488
## 6  3.125000e-02  0.3707763 0.04889355 0.6200507 0.04439815
```






```
best_cp_low <- cv_results_trees_low$cp[cv_results_trees_low$mean_AUC ==
                                         max(cv_results_trees_low$mean_AUC)]

best_cp_low
```

```
## [1] 7.629395e-06
```

Propensity Score Estimation

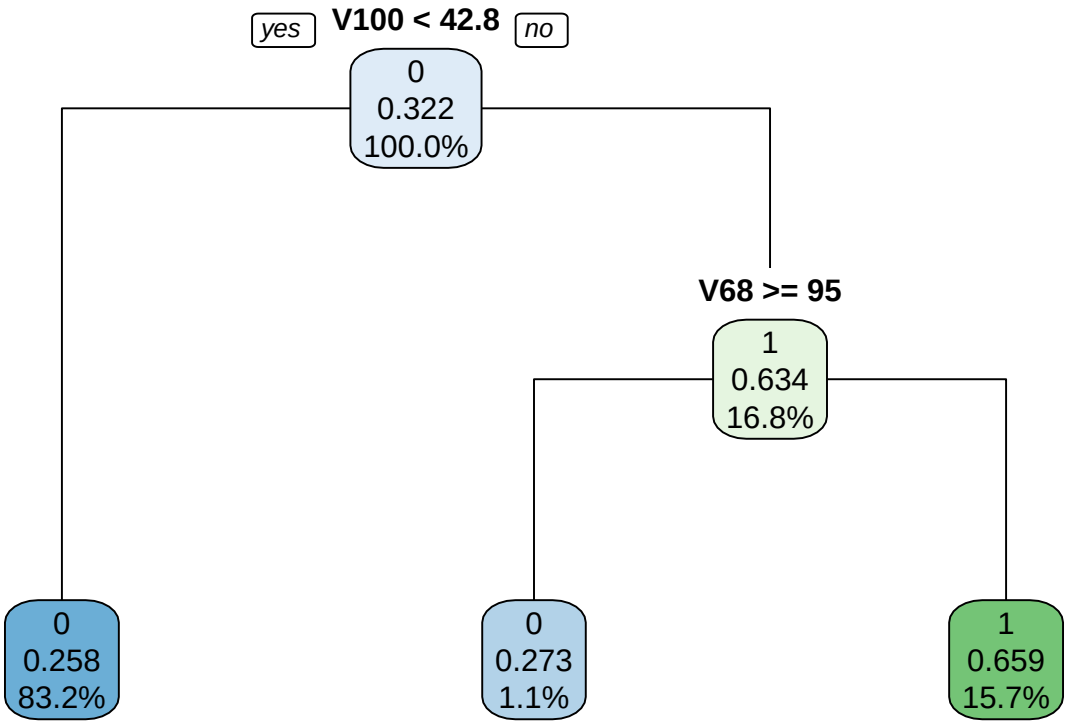
With the optimal "cp" parameters for each dataset, we now estimate the propensity scores using a weighted classification tree model.

```
# imbalanced dataset requires weights
# to be used in the trained model

weights_high <- rep(NA, length(df_high$A))
for (v in unique(df_high$A)){
  weights_high[df_high$A == v] = 0.5 * length(df_high$A) / length(df_high$A[df_high$A == v])
}


weights_low <- rep(NA, length(df_low$A))
for (v in unique(df_low$A)){
  weights_low[df_low$A == v] = 0.5 * length(df_low$A) / length(df_low$A[df_low$A == v])
}
```

```
start.time_propensity_score_high <- Sys.time()

# create tree model for high dimensional data with best cp parameter
tree_high <- rpart(A ~ . - Y, method = "class", data = df_high, cp = best_cp_high)

# calculate propensity scores
prop_score_high <- predict(tree_high, newdata = df_high[, -2], type = "prob")[, 2]

end.time_propensity_score_high <- Sys.time()
time_propensity_score_high <- end.time_propensity_score_high - start.time_propensity_score_high
time_propensity_score_high
```

High Dimensional Data

```
## Time difference of 1.203878 secs
```

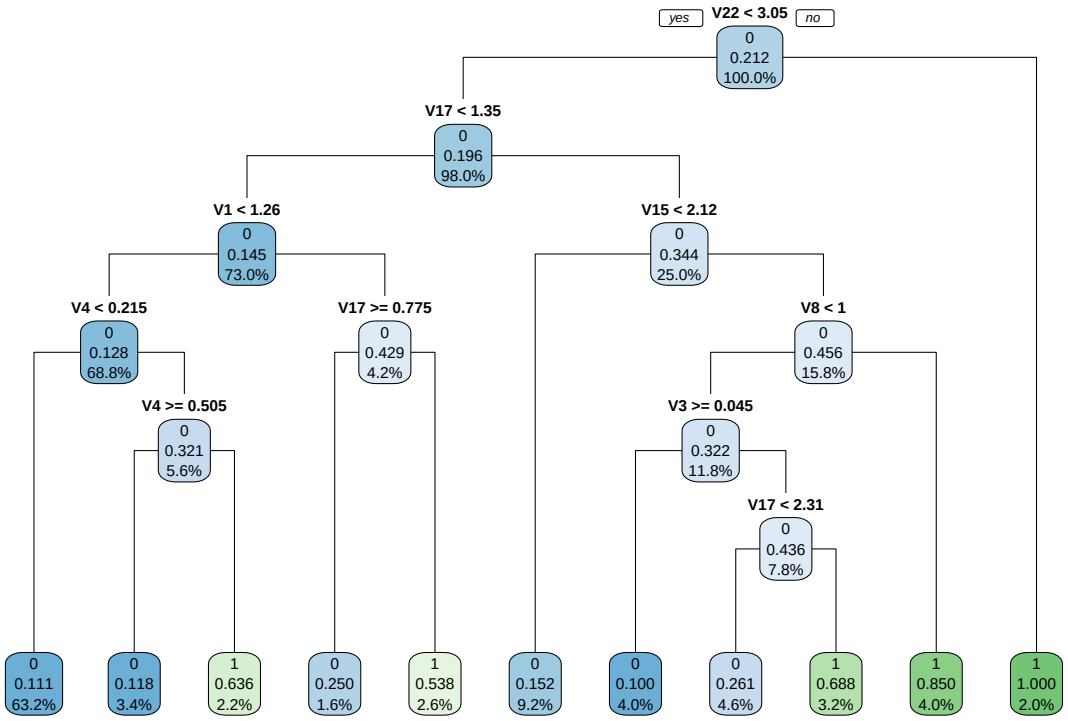



```
start.time_propensity_score_low <- Sys.time()

# create tree model for low dimensional data with best cp parameter
tree_low <- rpart(A ~ . - Y, method = "class", data = df_low, cp = best_cp_low)

# calculate propensity scores
prop_score_low <- predict(tree_low, newdata = df_low[, -2], type = "prob")[, 2]

end.time_propensity_score_low <- Sys.time()
```

```
time_propensity_score_low <- end.time_propensity_score_low - start.time_propensity_score_low
time_propensity_score_low
```

Low Dimensional Data

```
## Time difference of 0.0357759 secs
```



ATE Estimation

With the estimated propensity scores on hand, we propose, explain, and discuss the pros and cons of three different ATE estimation algorithms: propensity matching, stratificationand weighted regression.

Stratification

In the stratification method, we are trying to achieve groups where propensity scores hold approximately. For choosing number of strata, K, five (5) was advocated by Rosenbaum and Rubin (1984). However, in our case, choosing $K = 5$ will lead to empty stratum for both high and low dimensional data.

```
# when K = 5, the second stratus of high and low dimensional data are both empty
# but we need to make sure each stratum consists at least one object

summary_high_k5
```

```
##    A stratum    n   prop     avg_y
## 1  0       1    0 0.0000   0.00000
## 2  1       1    0 0.0000   0.00000
## 3  0       2    0 0.0000   0.00000
```

```
## 4  1       2    0 0.0000   0.00000
## 5  0       3 1234 0.6170  35.35960
## 6  1       3  430 0.2150 -28.97080
## 7  0       4    0 0.0000   0.00000
## 8  1       4    0 0.0000   0.00000
## 9  0       5  123 0.0615 -29.74445
## 10 1       5  213 0.1065 -79.51132
```

```
summary_low_k5
```

```
##    A stratum   n  prop    avg_y
## 1  0       1  18 0.036 28.28227
## 2  1       1   2 0.004 32.00448
## 3  0       2   0 0.000  0.00000
## 4  1       2   0 0.000  0.00000
## 5  0       3 281 0.562 12.92826
## 6  1       3  35 0.070 16.53301
## 7  0       4  60 0.120 24.57590
## 8  1       4  11 0.022 23.53773
## 9  0       5  35 0.070 26.03157
## 10 1       5  58 0.116 34.11020
```

Therefore we choose $K = 3$ strata, which is the highest value of K that do not produce empty stratum. Then we can estimate ATE between treated and untreated subgroups by following formula:

$$\hat{\Delta}_S = \sum_{j=1}^{K} \frac{N_j}{N} \{N_{1j}^{-1} \sum_{i=1}^{N} T_i Y_i I(\hat{e}_i \in \hat{Q}_j) - N_{0j}^{-1} \sum_{i=1}^{N} (1 - T_i) Y_i I(\hat{e}_i \in \hat{Q}_j)\}$$

where K is the number of strata, $\hat{e}$ is the estimated propensity score, Y is the response for each observation, and T is the treatment variable (either 0 or 1). N_j is the number of individuals in stratum j. N_{1j} is the number of "treated" individuals in stratum j, while N_{0j} is the number of "controlled" individuals in stratum j. $\hat{Q}_j = (\hat{q_{j-1}}, \hat{q_j}]$ where $\hat{q_j}$ is the jth sample quantile of the estimated propensity scores.

The advantage of stratification is that it controls systematic differences between the control and treated groups. However, as we mentioned earlier, stratification has imbalance issues in each stratum. In some extreme cases, where some strata may contain subjects from only the treated group or control group, it becomes impossible to estimate treatment effect. What's more, Lunceford and Davidian (2004) demonstrated that stratification results in estimates of average treatment effects with greater bias than does a variety of weighted estimators.

```
K = 3
strata <- seq(0, 1, by = 1/K)
```

```
start.time_stratification_high <- Sys.time()

df_high <- cbind(df_high, prop_score_high)
stratum_values_high <- rep(NA, length(strata))

for (i in 1:length(strata)){
  stratum_values_high[i] <- quantile(prop_score_high, strata[i])
```

```
}

# values of strata for high data
stratum_values_high
```

High Dimensional Data

```
## [1] 0.2584135 0.2584135 0.2584135 0.6592357
```

```
df_high$stratum_class_high <- rep(NA, nrow(df_high))

# assign stratum class to each observation
for (i in 1:nrow(df_high)){
  if ((stratum_values_high[1] <= df_high$prop_score_high[i]) &
      (df_high$prop_score_high[i] < stratum_values_high[2])) {
    df_high$stratum_class_high[i] <- 1
  } else if ((stratum_values_high[2] <= df_high$prop_score_high[i]) &
             (df_high$prop_score_high[i] < stratum_values_high[3])) {
    df_high$stratum_class_high[i] <- 2
  } else if ((stratum_values_high[3] <= df_high$prop_score_high[i]) &
             (df_high$prop_score_high[i] <= stratum_values_high[4])) {
    df_high$stratum_class_high[i] <- 3
  }
}

summary_high = expand.grid(
  A = c(0, 1),
  stratum = seq(1, K, by = 1),
  n = NA,
  prop = NA,
  avg_y = NA
)

for (i in 1:nrow(summary_high)) {
  subset <- df_high[(df_high$A == summary_high$A[i]) &
                      (df_high$stratum_class_high == summary_high$stratum[i]), ]
  summary_high$n[i] = nrow(subset)
  summary_high$prop[i] = summary_high$n[i]/nrow(df_high)
  summary_high$avg_y[i] = mean(subset$Y)
}


for (i in 1:nrow(summary_high)) {
  if (is.nan(summary_high$avg_y[i]) == TRUE) {
    summary_high$avg_y[i] <- 0
  }
}

# this table records the mean response in each stratum; needed for stratification
summary_high
```

```
##   A stratum    n   prop     avg_y
## 1 0       1    0 0.0000   0.00000
```

```
## 2 1       1    0 0.0000   0.00000
## 3 0       2    0 0.0000   0.00000
## 4 1       2    0 0.0000   0.00000
## 5 0       3 1357 0.6785  29.45850
## 6 1       3  643 0.3215 -45.71284
```

```
stratum_prop_high <- summary_high %>% group_by(stratum) %>% summarise(sum = sum(n)/nrow(df_high))

# this table records the proportions for each stratum; also needed for stratification
stratum_prop_high
```

```
## # A tibble: 3 x 2
##   stratum   sum
## *   <dbl> <dbl>
## 1       1     0
## 2       2     0
## 3       3     1
```

```
ATE_stratification_high = stratum_prop_high$sum[1]*(summary_high$avg_y[2] - summary_high$avg_y[1]) +
  stratum_prop_high$sum[2]*(summary_high$avg_y[4] - summary_high$avg_y[3]) +
  stratum_prop_high$sum[3]*(summary_high$avg_y[6] - summary_high$avg_y[5])

ATE_stratification_high
```

```
## [1] -75.17133
```

```
end.time_stratification_high <- Sys.time()
time_stratification_high <- end.time_stratification_high - start.time_stratification_high
time_stratification_high
```

```
## Time difference of 0.2523232 secs
```

We find that the ATE for the high dimensional dataset was -75.171 with a run time of 0.252 seconds.

```
start.time_stratification_low <- Sys.time()

df_low <- cbind(df_low, prop_score_low)
stratum_values_low <- rep(NA, length(strata))

for (i in 1:length(strata)){
  stratum_values_low[i] <- quantile(prop_score_low, strata[i])
}

# values of strata for low data
stratum_values_low
```

Low Dimensional Data

```
## [1] 0.1000000 0.1107595 0.1107595 1.0000000
```

```r
df_low$stratum_class_low <- rep(NA, nrow(df_low))

# assign stratum class to each observation
for (i in 1:nrow(df_low)){
  if ((stratum_values_low[1] <= df_low$prop_score_low[i]) &
      (df_low$prop_score_low[i] < stratum_values_low[2])) {
    df_low$stratum_class_low[i] <- 1
  } else if ((stratum_values_low[2] <= df_low$prop_score_low[i]) &
             (df_low$prop_score_low[i] < stratum_values_low[3])) {
    df_low$stratum_class_low[i] <- 2
  } else if ((stratum_values_low[3] <= df_low$prop_score_low[i]) &
             (df_low$prop_score_low[i] <= stratum_values_low[4])) {
    df_low$stratum_class_low[i] <- 3
  }
}


summary_low = expand.grid(
  A = c(0, 1),
  stratum = seq(1, K, by = 1),
  n = NA,
  prop = NA,
  avg_y = NA
)

for (i in 1:nrow(summary_low)) {
  subset <- df_low[(df_low$A == summary_low$A[i]) &
                     (df_low$stratum_class_low == summary_low$stratum[i]), ]
  summary_low$n[i] = nrow(subset)
  summary_low$prop[i] = summary_low$n[i]/nrow(df_low)
  summary_low$avg_y[i] = mean(subset$Y)
}


for (i in 1:nrow(summary_low)) {
  if (is.nan(summary_low$avg_y[i]) == TRUE) {
    summary_low$avg_y[i] <- 0
  }
}

# this table records the mean response in each stratum; needed for stratification
summary_low
```

```
##   A stratum   n  prop    avg_y
## 1 0       1  18 0.036 28.28227
## 2 1       1   2 0.004 32.00448
## 3 0       2   0 0.000  0.00000
## 4 1       2   0 0.000  0.00000
## 5 0       3 376 0.752 16.00665
## 6 1       3 104 0.208 27.07655
```

```
stratum_prop_low <- summary_low %>% group_by(stratum) %>% summarise(sum = sum(n)/nrow(df_low))

# this table records the proportions for each stratum; also needed for stratification
stratum_prop_low
```

```
## # A tibble: 3 x 2
##   stratum   sum
## *   <dbl> <dbl>
## 1       1  0.04
## 2       2  0
## 3       3  0.96
```

```
ATE_stratification_low = stratum_prop_low$sum[1]*(summary_low$avg_y[2] - summary_low$avg_y[1]) +
  stratum_prop_low$sum[2]*(summary_low$avg_y[4] - summary_low$avg_y[3]) +
  stratum_prop_low$sum[3]*(summary_low$avg_y[6] - summary_low$avg_y[5])

ATE_stratification_low
```

```
## [1] 10.776
```

```
end.time_stratification_low <- Sys.time()
time_stratification_low <- end.time_stratification_low - start.time_stratification_low
time_stratification_low
```

```
## Time difference of 0.1022029 secs
```

We find that the ATE for the low dimensional dataset was 10.776 with a run time of 0.102 seconds.

Propensity Matching using Propensity Score from Regression Trees (Full Matching)

Full matching creates a series of matched sets, where each matched set contains at least one treated individual and at least one control individual (and each matched set may have many from either group). Full matching forms these matched sets in an optimal way, such that treated individuals who have many comparison individuals who are similar (on the basis of the propensity score) will be grouped with many comparison individuals, whereas treated individuals with few similar comparison individuals will be grouped with relatively fewer comparison individuals.(See Stuart (2010))

Our group is assigned the task of using propensity score for Full Matching. The distance of Propensity Score is defined as:

$$D_{ij} =\mid e_i - e_j \mid$$

where e_k is the propensity score for individual k.

After the matched sets are obtained, calculate a "subclass effects" for each matched set/subclass, and then estimate overall ATE by an weighted average of the subclass effects where weights would be the number of individuals in each subclass.

```
source("../lib/propensity_matching.R")
match_High_Dim <- get_match_obj(read.csv(highDim_csv))
```

Getting the Data Loaded for High and Low Dimension Data

```
## Full matching...
## Calculating matching weights... Done.
```

```
match_Low_Dim <- get_match_obj(read.csv(lowDim_csv))
```

```
## Full matching...
## Calculating matching weights... Done.
```

```
#g.matches <- get_matches(matchit.out, data = Any_Dim_Data, distance = prop_score_column_name)
# creates a dataset with one row per unit. It will be identical to the dataset supplied except
# that several new columns will be added containing information related to the matching
g.matches_high <- match.data(match_High_Dim, data = read.csv(highDim_csv), distance = "prop_score")
g.matches_low <- match.data(match_Low_Dim, data = read.csv(lowDim_csv), distance = "prop_score")
```

###Estimating Effects After Matching

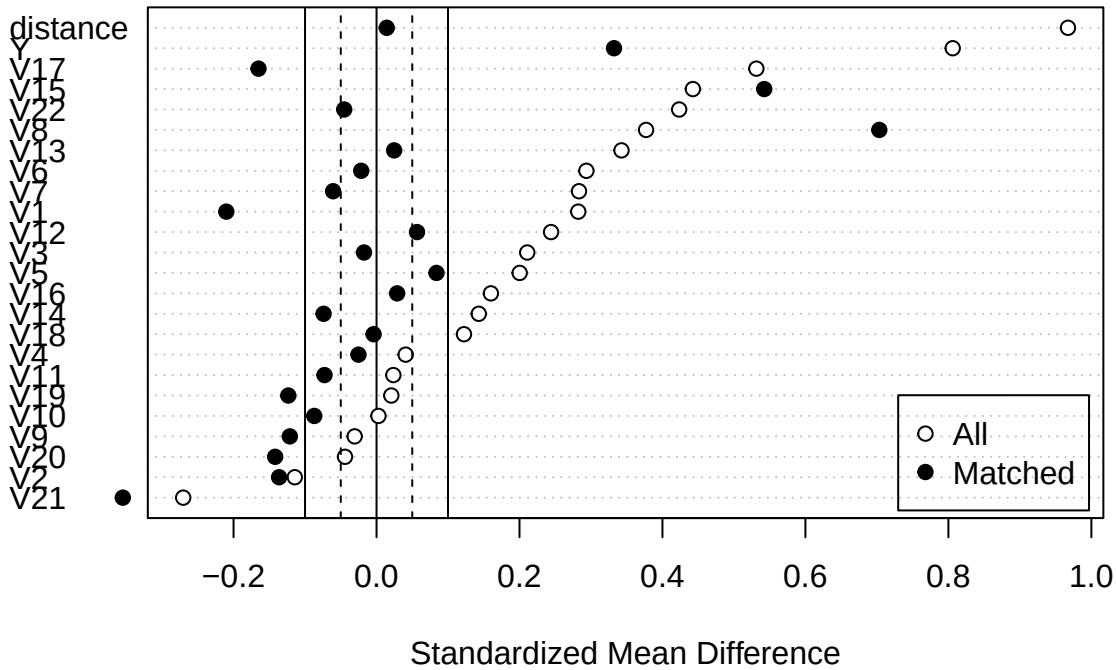
We diagnose the quality of the resulting matched samples. We would would like the treatment to be unrelated to the covariates such that:

$$\tilde{p}(X|T=1) = \tilde{p}(X|T=0)$$

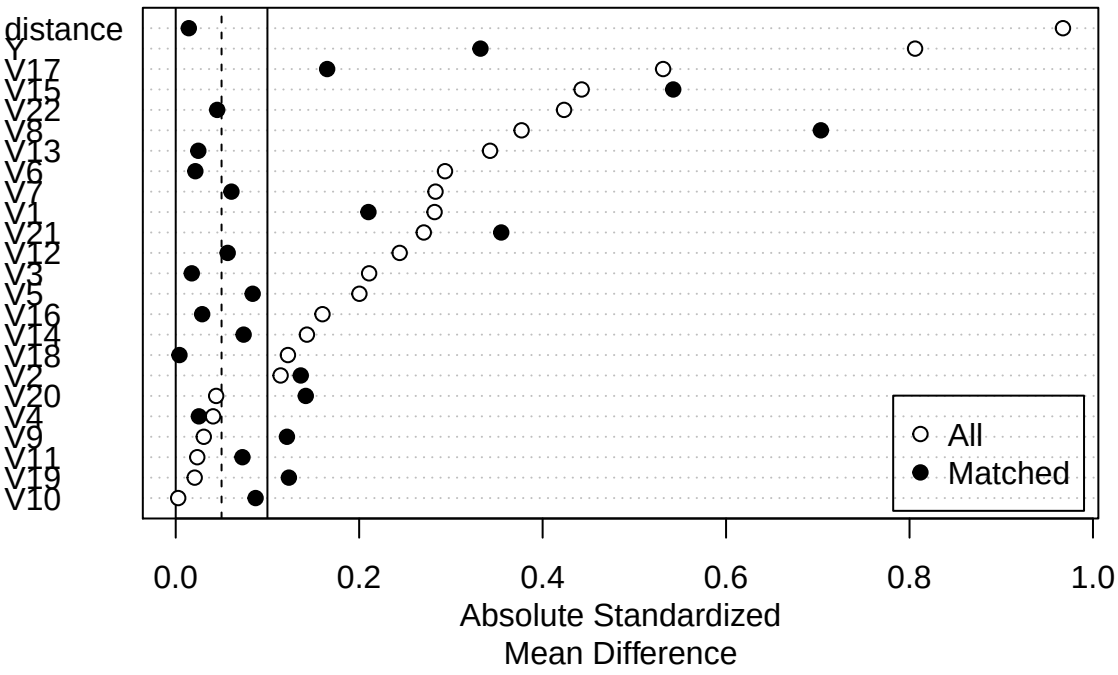
where $\tilde{p}$ denotes the empirical distribution.

a plot of the standardized differences of means, gives us a quick overview of whether balance has improved for individual covariates

```
#plot(summary(match_Low_Dim, interactions = TRUE), var.order = "unmatched")
plot(summary(match_Low_Dim, subclass = TRUE),
     var.order = "unmatched", abs = FALSE)
```




```
plot(summary(match_Low_Dim, subclass = TRUE),
     var.order = "unmatched", abs = TRUE)
```



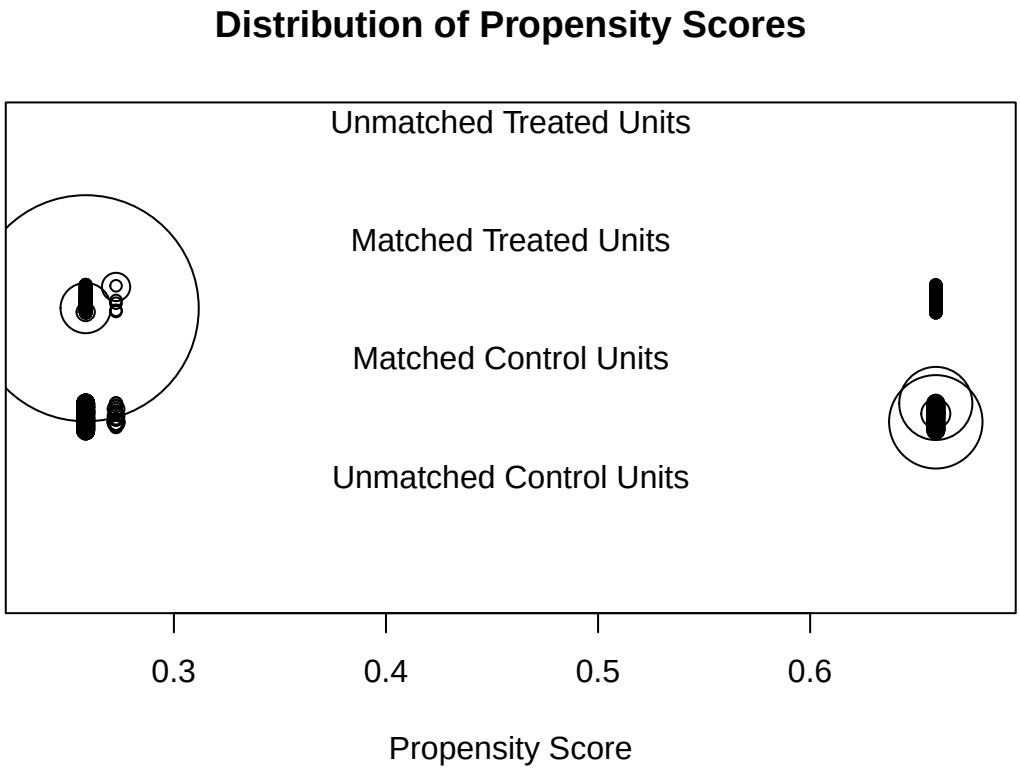
the standardized difference of means of each covariate has decreased after matching For regression adjustment to be trustworthy, the absolute standardized differences of means should be less than 0.25

For continuous covariates, we can also examine quantile–quantile (QQ) plots, which compare the empirical distributions of each variable

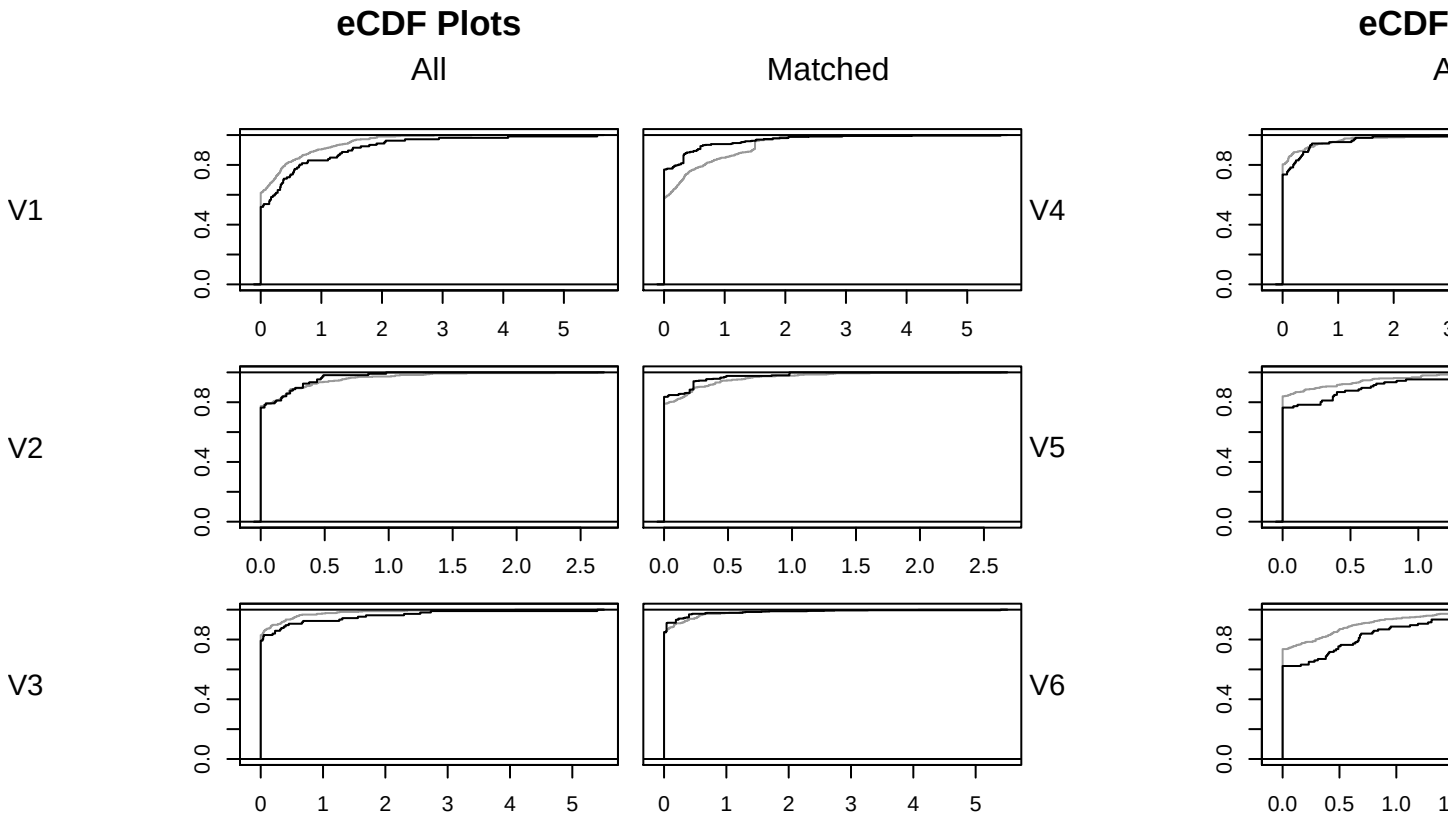
QQ plots compare the quantiles of a variable in the treatment group against the corresponding quantiles in the control group. If the two groups have identical empirical distributions, all points would lie on the 45 degree line

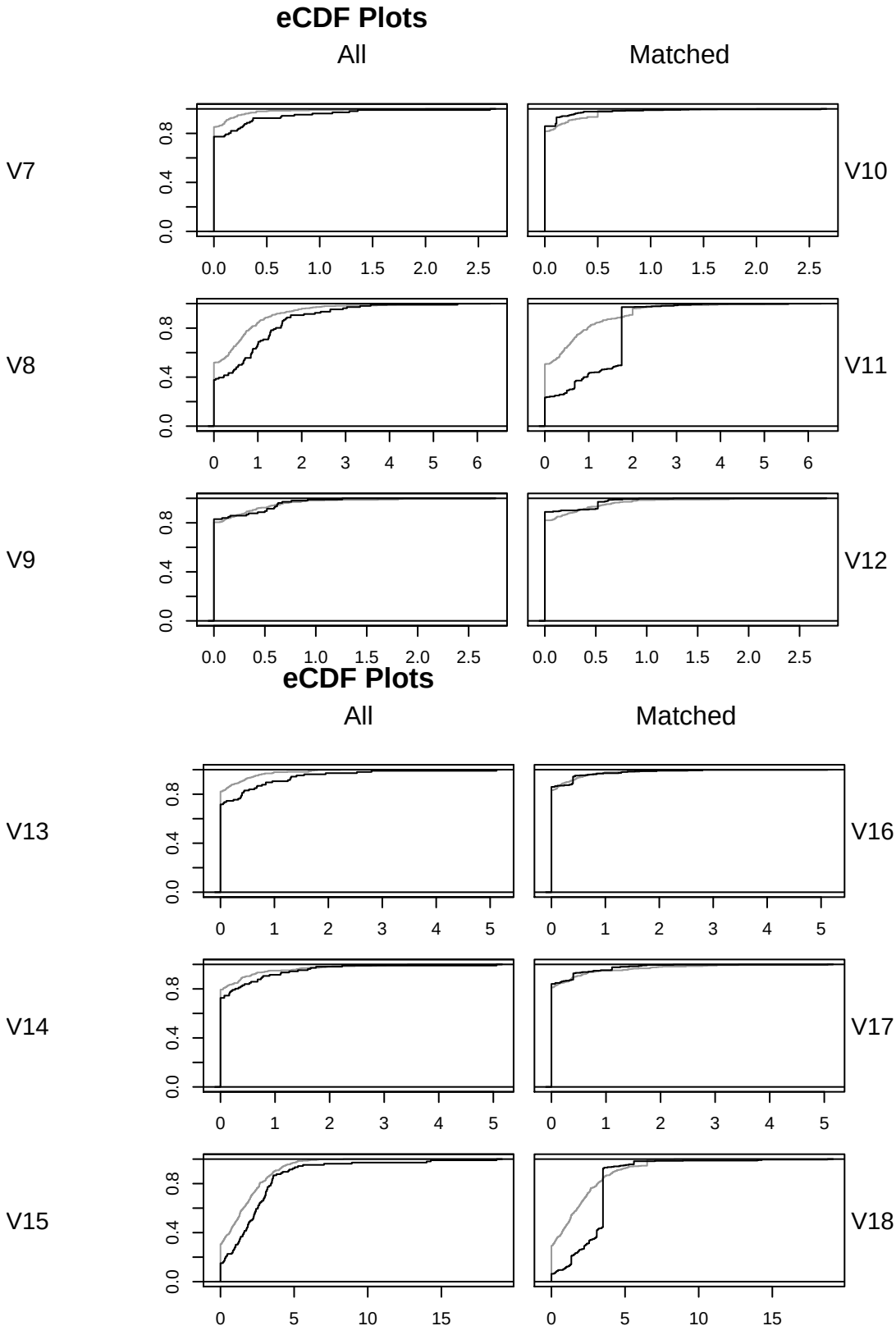
Statistics related to the difference in the empirical cumulative density functions (eCDFs) of each covariate between groups allow assessment of imbalance across the entire covariate distribution of that covariate rather than just its mean or variance.

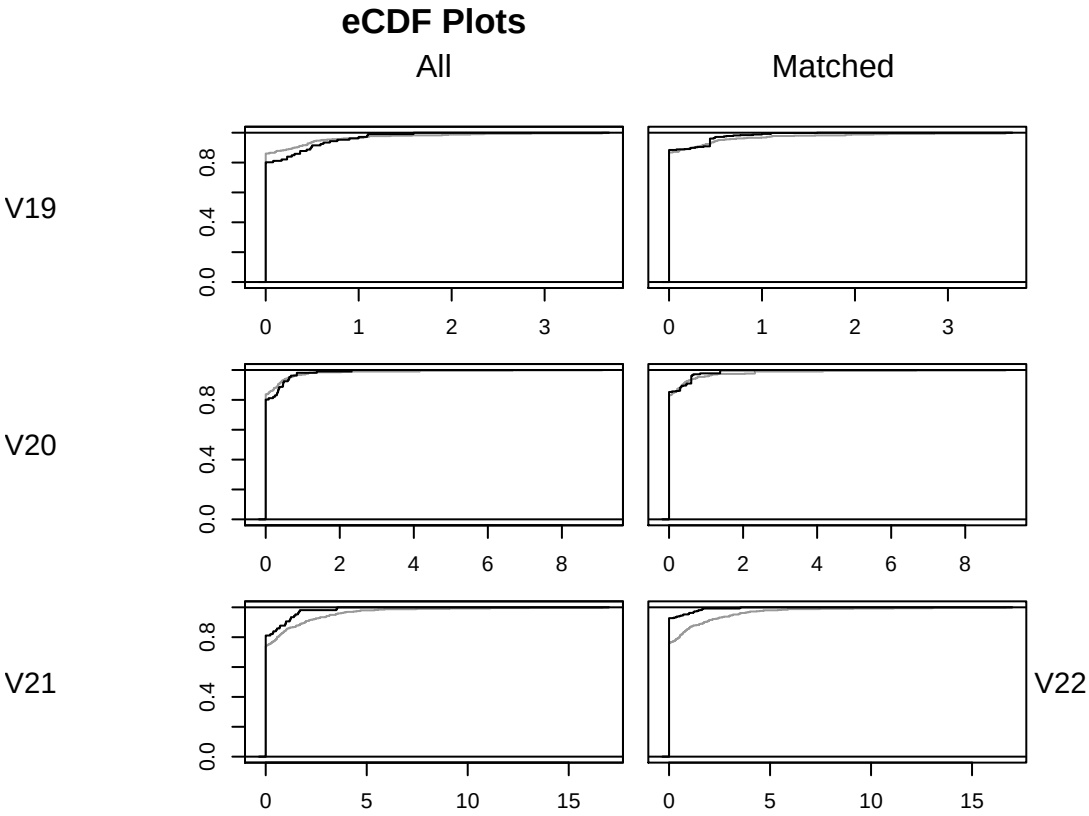
```
plot(match_High_Dim, type = "jitter", interactive = FALSE)
```




```
plot(match_Low_Dim, type = "ecdf", interactive = FALSE)
```

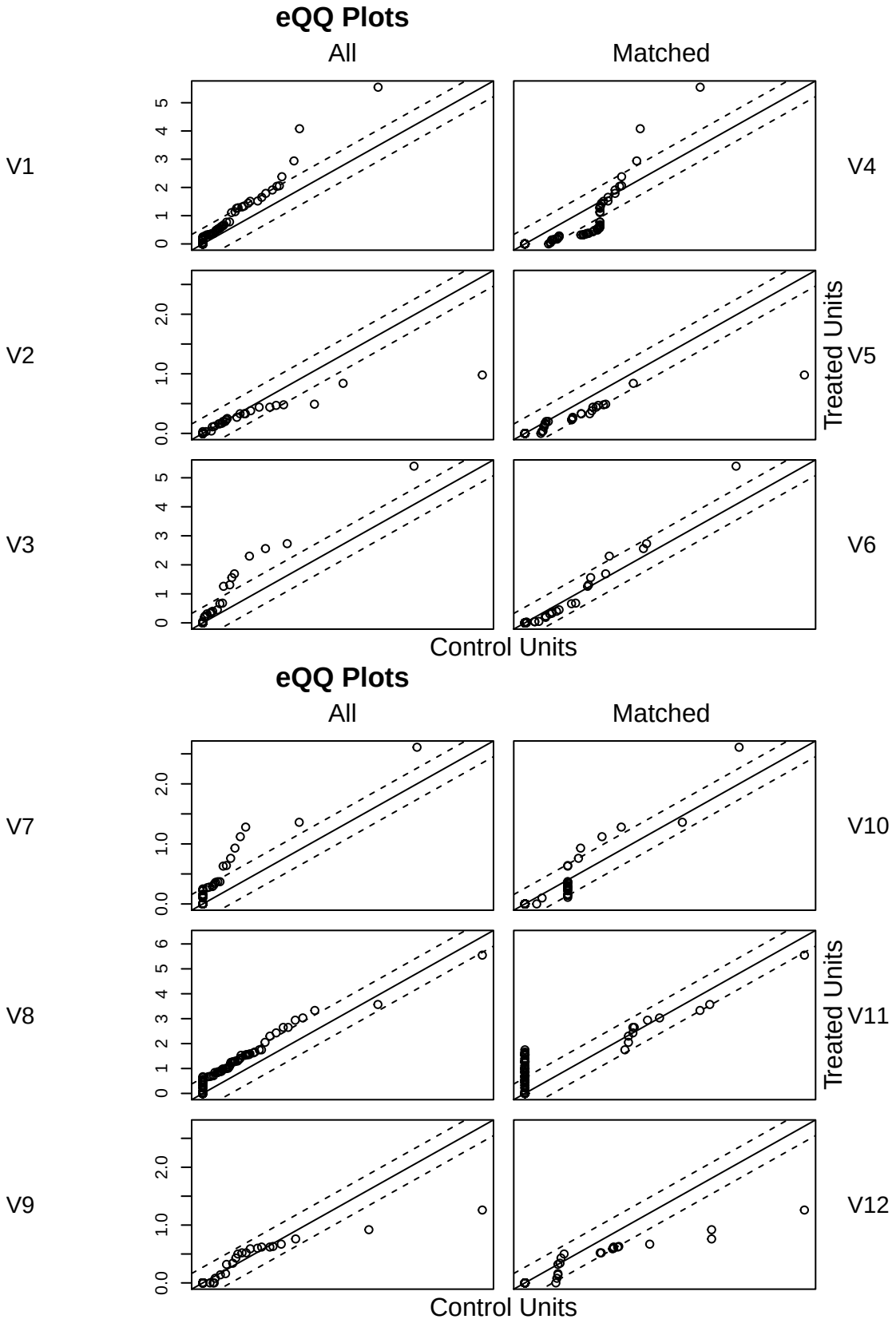


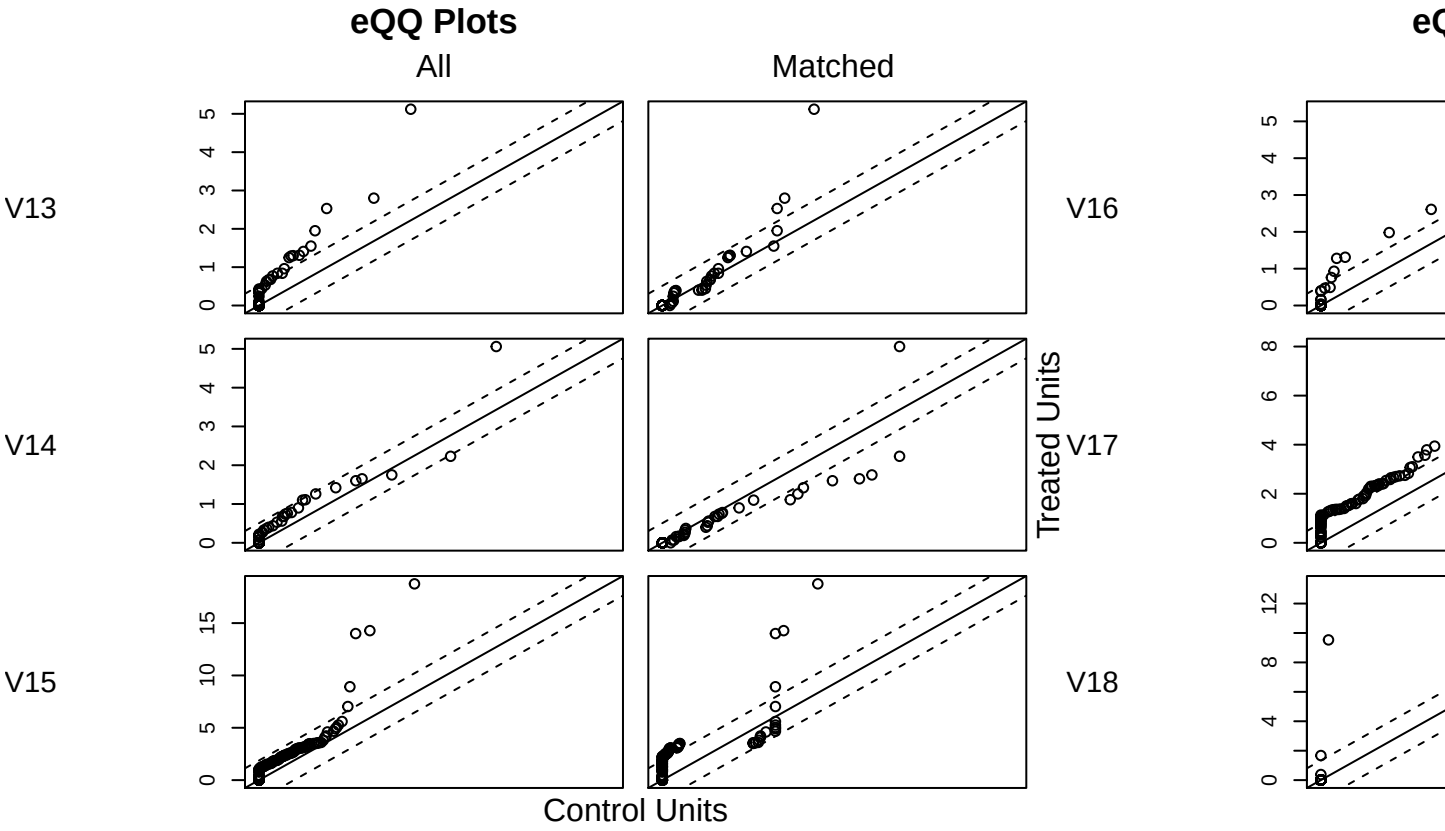

eCDF Plots
All
Matched
V7
V10
V8
V11
V9
V12
eCDF Plots
All
Matched
V13
V16
V14
V17
V15
V18

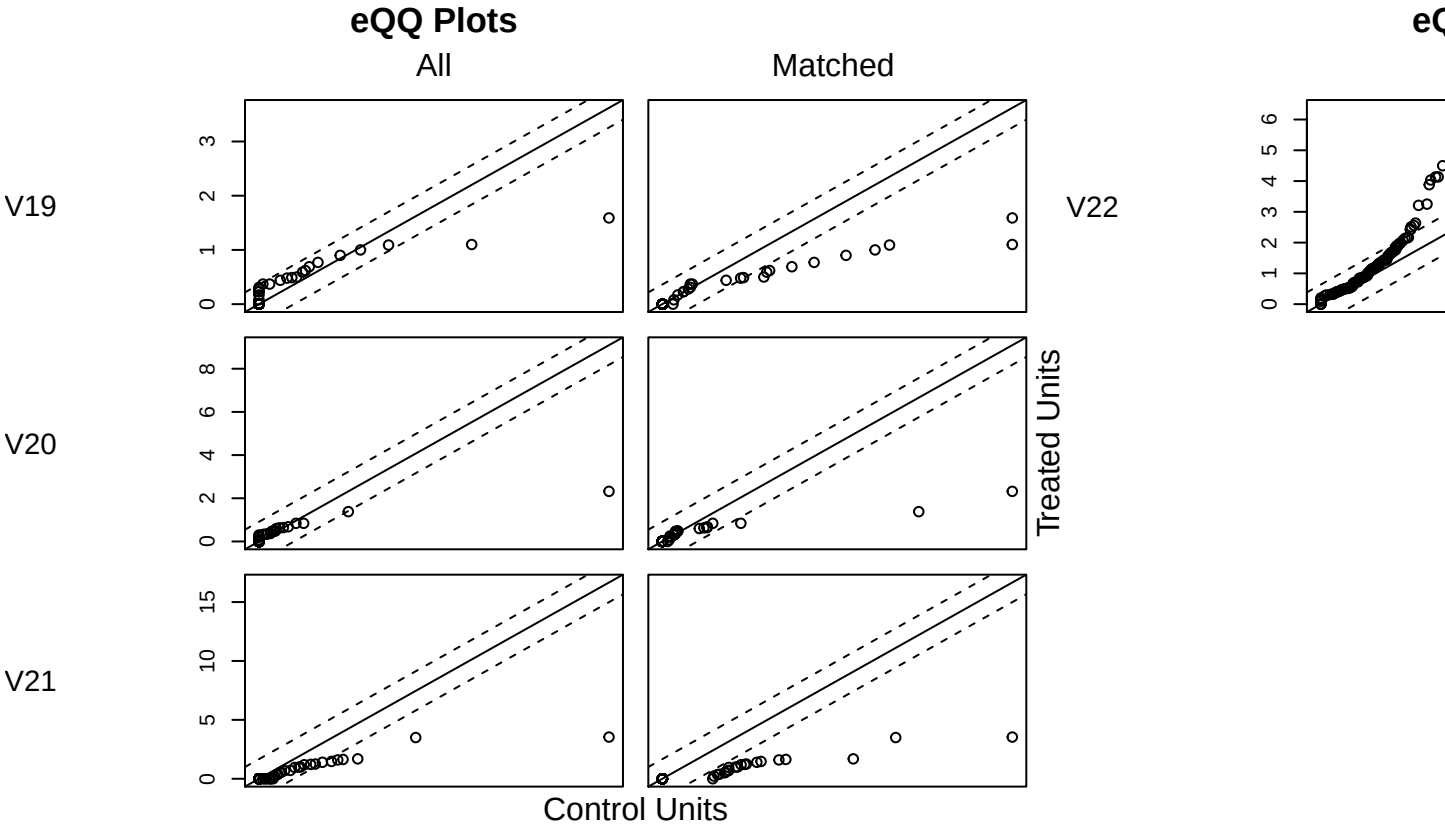


The y-axis displays the value of the covariate for the treated units, and the x-axis displays the the value of the covariate at the corresponding quantile in the control group. When values fall on the 45 degree line, the groups are balanced.

```
#eQQ plot
plot(match_Low_Dim, type = "qq", interactive = FALSE)
```


eQQ Plots
All
Matched
V1
V2
V3
V4
V5
V6
Treated Units
Control Units
eQQ Plots
All
Matched
V7
V8
V9
V10
V11
V12
Treated Units
Control Units





The most common and recommended way to estimate effects after full matching is to use the computed matching weights to estimate weighted effects.

```r
#fit <- lm(re78 ~ A, data = g.matches_low, weights = weights)
#coeftest(fit, vcov. = vcovCL, cluster = ~subclass)
#Linear model without covariates
fit1 <- lm(Y ~ A, data = g.matches_low, weights = weights)
coeftest(fit1, vcov. = vcovCL, cluster = ~subclass)["A",,drop=FALSE]
```

```
##   Estimate Std. Error  t value    Pr(>|t|)
## A 4.369954   1.437233 3.040532 0.002485969
```

including interactions between the treatment and covariates can be beneficial when effect modification by the covariates may be present In order to interpret the coefficient on treatment as a marginal effect estimate, we need to center the covariates at their means in the target population. Below we use the strategy of centering the covariates at their means.

```r
#Estimating a covariate-adjusted marginal effect
#with treatment-covariate interactions
#Create a new dataset for centered variables
md_cen <- g.matches_low
covs_to_center <- c("V1", "V2", "V3", "V4", "V5",
                    "V6", "V7", "V8", "V9", "V10",
                    "V11", "V12", "V13", "V14","V16",
                    "V17", "V18","V19","V20","V21", "V22")
md_cen[covs_to_center] <- scale(md_cen[covs_to_center],
                                scale = FALSE)
#Fit the model with every covariate interacting with treatment
fit2 <- lm(Y ~ A * (V1 + V2 + V3 + V4 + V5 +
                      V6 + V7 + V8 + V9 + V10 + V11+
                      V12 + V13 + V14 + V15 + V16 + V17 +
                      V18 + V19 + V20 + V21 + V22),
           data = md_cen, weights = weights)
#Only output the intercept and coefficient on treatment
coeftest(fit2, vcov. = vcovCL, cluster = ~subclass)[1:2,]
```

```
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 13.263731 0.06563455 202.0846 0.000000e+00
## A            1.988692 0.15374739  12.9348 8.440227e-33
```

Weighted Regression

Weighted least square estimation of the regression function:

$$Y_i = \alpha_0 + \tau * T_i + \alpha_1^{'} * Z_i + \alpha_2^{'} * (Z_i - \bar{Z}) * T_i + \varepsilon_i$$

The weights are the same as the ones of "Inverse Propensity Weighting". The Z_i are a subset of the covariates X_i; with sample average $\bar{Z}$. τ is an estimate for ATE. For the method of selecting Z, we get the estimation using linear regressions:

$$Y_i = \beta_{k0} + \beta_{k1} * T_i + \beta_{k2} * X_{ik} + \varepsilon_i$$

We calculate the t-statistic for the test of the null hypothesis that the slope coefficient β_{k2} is equal to zero in each of these regressions, and now select for Z all the covariates with a t-statistic larger in absolute value than t_{reg}. Thus, we include in the final regression all covariates which have substantial correlation with the outcome conditional on the treatment. (See Hirano and Imbens (2001))

```
set.seed(0)
X_low <- df_low %>% select(-Y, -A) %>% as.matrix
A_low <- df_low %>% select(A) %>% as.matrix
```

```
X_high <- df_high %>% select(-Y, -A) %>% as.matrix
A_high <- df_high %>% select(A) %>% as.matrix
```

Loading the Data and Data Preprocessing

```
weight_low <- cbind(as.numeric(A_low), prop_score_low) %>%
  as_tibble %>%
  mutate(weights = (V1/prop_score_low + (1-V1)/(1-prop_score_low))) %>%
  select(weights)
```

Finding weights

```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if `.name_repair` is
## Using compatibility `.name_repair`.
```

```
weight_high <- cbind(as.numeric(A_high), prop_score_low) %>%
  as_tibble %>%
  mutate(weights = (V1/prop_score_high + (1-V1)/(1-prop_score_high))) %>%
  select(weights)
```

```
filter_low <- summary(lm(Y~., data = df_low))$coef[,4][3:24]<0.05
Z_low <- cbind(A_low, X_low[,filter_low])
```

```
filter_high <- summary(lm(Y~., data = df_high))$coef[,4][3:ncol(X_high)]<0.05
Z_high <- cbind(A_high, X_high[,filter_high])
```

Linear regression for selecting covarites

```
Z_low <- Z_low %>% apply(2, as.numeric)
Z_high <- Z_high %>% apply(2, as.numeric)
```

Modify the data

```
Y_low <- df_low$Y
Y_high <- df_high$Y
```

```
weighted_low <- lm(Y_low ~ Z_low, weights = as.numeric(unlist(weight_low)))
ATE_weightedreg_low <- coef(weighted_low)[2]
```

```
weighted_high <- lm(Y_high ~ Z_high, weights = as.numeric(unlist(weight_high)))
ATE_weightedreg_high <- coef(weighted_high)[2]
```

Final Regression for ATE

```
ATE_weightedreg_low
```

Summarizing Final Results

```
##   Z_lowA
## 2.236732
```

```
ATE_weightedreg_high
```

```
##  Z_highA
## -56.9649
```

Results

We compare the accuracy and performance of the three ATE Estimation procedures below.

ATE Results

We are provided the true ATE values of -54.8558 for the high dimensional data and 2.0901 for the low dimensional data.

From the table above, we see that regression adjustment performed the best for the high dimensional data and stratification performed the best for the low dimensional data.

Run Time Results

Given the nature of trees, propensity score estimations are quickly calculated once we have the proper hyperparamters selected from cross-validation–even for the high dimensional data, propensity score estimations did not take more than two seconds.

It is also no surprise that, given the sizes of our two datasets, that regression adjustment was the fastest method. However, with larger datasets with more observations, this may not be the case. Stratification

took the longest time, mainly due to the many intermediate calculations requited. Lastly, the combination method of both stratification and regression adjustment had a run time between the two former methods.

However, we want to note that this .Rmd file was knitted using a computer with a NVMe SAMSUNG SSD with 16 GB RAM. Run times may vary from device to device and with each iteration. Descriptions of run times are based on average run times we saw through numerous iterations.

Conclusion

Overall, we believe that using classification/regression trees for propensity scores was not the ideal approach for either dataset. While we cross-validated the complexity hyperparameter, cp, to help avoid with over-fitting, our models for both the high dimensional and low dimensional datasets ended up estimating the same propensity score value for over half of the entire dataset. This would not be a very helpful model in differentiating our observations and of course affect our ATE estimations regardless of the method used.

We see this most prominently in stratification, in which different values of K, that is, the number of strata, resulted in an empty stratum in our results. Even after choosing a value of K which would present no empty strata, we saw that each stratum tend to have imbalanced classes. In the case of the low dimensional dataset, one stratum only consisted of observations from the control group. These complications may explain why the stratification plus regression adjustment method would not have performed the best.

However, the results were relatively consistent among all three methods–there were no large deviations from the true value. In particular, the ATE for stratification was actually quite close to the true value for the low dimensional data. Additionally, compared to other methods, we note the relative ease of interpretation and fast run times for not only the propensity score estimations but also for the ATE estimations as well. While we may not advocate for these estimation methods for their accuracy (and validity in certain cases), but these methods here show a fast and easy way to get a general sense of the average treatment effect.

References

- Atkinson, Beth. "Recursive Partitioning And Regression Trees." R Documentation, DataCamp, www.rdocumentation.org/packages/rpart/versions/4.1-15/topics/rpart.
- Austin, Peter C. 2011. "An Introduction to Propensity Score Methods for Reducing the Effects of Confounding in Observational Studies." Multivariate Behavioral Research 46 (3): 399–424.
- Chan, David & Ge, Rong & Gershony, Ori & Hesterberg, Tim & Lambert, Diane. (2010). Evaluating online ad campaigns in a pipeline: Causal models at scale. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 7-16. 10.1145/1835804.1835809.
- D'Agostino RB Jr. Propensity score methods for bias reduction in the comparison of a treatment to a non-randomized control group. Stat Med. 1998 Oct 15;17(19):2265-81. doi: 10.1002/(sici)1097-0258(19981015)17:19<2265::aid-sim918>3.0.co;2-b. PMID: 9802183.
- Hastie, Trevor,, Robert Tibshirani, and J. H Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. New York: Springer, 2009. Print.
- Lunceford, Jared K, and Marie Davidian. 2004. "Stratification and Weighting via the Propensity Score in Estimation of Causal Treatment Effects a Comparative Study." Statistics in Medicine 23 (19): 2937–60.
- Rosenbaum PR, Rubin DB. The central role of the propensity score in observational studies for causal efiects.Biometrika 1983; 70:41–55.