Project 6: Randomization and Matching

April 05, 2024, 03:19

Introduction

In this project, you will explore the question of whether college education causally affects political participation. Specifically, you will use replication data from Who Matches? Propensity Scores and Bias in the Causal Effects of Education on Participation by former Berkeley PhD students John Henderson and Sara Chatfield. Their paper is itself a replication study of Reconsidering the Effects of Education on Political Participation by Cindy Kam and Carl Palmer. In their original 2008 study, Kam and Palmer argue that college education has no effect on later political participation, and use the propensity score matching to show that pre-college political activity drives selection into college and later political participation. Henderson and Chatfield in their 2011 paper argue that the use of the propensity score matching in this context is inappropriate because of the bias that arises from small changes in the choice of variables used to model the propensity score. They use genetic matching (at that point a new method), which uses an approach similar to optimal matching to optimize Mahalanobis distance weights. Even with genetic matching, they find that balance remains elusive however, thus leaving open the question of whether education causes political participation.

You will use these data and debates to investigate the benefits and pitfalls associated with matching methods. Replication code for these papers is available online, but as you'll see, a lot has changed in the last decade or so of data science! Throughout the assignment, use tools we introduced in lab from the tidyverse and the MatchIt packages. Specifically, try to use dplyr, tidyr, purrr, stringr, and ggplot instead of base R functions. While there are other matching software libraries available, MatchIt tends to be the most up to date and allows for consistent syntax.

Data

The data is drawn from the Youth-Parent Socialization Panel Study which asked students and parents a variety of questions about their political participation. This survey was conducted in several waves. The first wave was in 1965 and established the baseline pre-treatment covariates. The treatment is whether the student attended college between 1965 and 1973 (the time when the next survey wave was administered). The outcome is an index that calculates the number of political activities the student engaged in after 1965. Specifically, the key variables in this study are:

- **college**: Treatment of whether the student attended college or not. 1 if the student attended college between 1965 and 1973, 0 otherwise.
- **ppnscal**: Outcome variable measuring the number of political activities the student participated in. Additive combination of whether the student voted in 1972 or 1980 (student_vote), attended a campaign rally or meeting (student_meeting), wore a campaign button (student_button), donated money to a campaign (student_money), communicated with an elected official (student_communicate), attended a demonstration or protest (student_demonstrate), was involved with a local community event (student_community), or some other political participation (student_other)

Otherwise, we also have covariates measured for survey responses to various questions about political attitudes. We have covariates measured for the students in the baseline year, covariates for their parents in the baseline year, and covariates from follow-up surveys. **Be careful here**. In general, post-treatment covariates will be clear from the name (i.e. student_1973Married indicates whether the student was married in the 1973 survey). Be mindful that the baseline covariates were all measured in 1965, the treatment occurred between 1965 and 1973, and the outcomes are from 1973 and beyond. We will distribute the Appendix from Henderson and Chatfield that describes the covariates they used, but please reach out with any questions if you have questions about what a particular variable means.

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
## here() starts at /Users/brenda/github/Computational-Social-Science-Projects
##
##
## Attaching package: 'gridExtra'
##
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
##
##  cobalt (Version 4.5.5, Build Date: 2024-04-02)
##
##
## Attaching package: 'cobalt'
##
##
## The following object is masked from 'package:MatchIt':
##
##     lalonde
##
##
## Rows: 1254 Columns: 174
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (174): interviewid, college, student_vote, student_meeting, student_othe...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 6 x 174
##   interviewid college student_vote student_meeting student_other student_button
##         <dbl>   <dbl>        <dbl>           <dbl>         <dbl>          <dbl>
## 1           1       1            1               0             0              0
## 2           2       1            1               1             1              1
## 3           3       1            1               0             0              1
## 4           4       0            0               0             0              0
## 5           5       1            1               1             0               0
## 6           6       1            1               0             0               0
## # i 168 more variables: student_money <dbl>, student_communicate <dbl>,
## #   student_demonstrate <dbl>, student_community <dbl>, student_ppnscal <dbl>,
## #   student_PubAff <dbl>, student_Newspaper <dbl>, student_Radio <dbl>,
## #   student_TV <dbl>, student_Magazine <dbl>, student_FamTalk <dbl>,
## #   student_FrTalk <dbl>, student_AdultTalk <dbl>, student_PID <dbl>,
## #   student_SPID <dbl>, student_GovtOpinion <dbl>, student_GovtCrook <dbl>,
## #   student_GovtWaste <dbl>, student_TrGovt <dbl>, student_GovtSmart <dbl>, ...
```

```
## [1] 1254
```

```
## Rows: 1254 Columns: 174
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (174): interviewid, college, student_vote, student_meeting, student_othe...
##
## i Use 'spec()' to retrieve the full column specification for this data.
## i Specify the column types or set 'show_col_types = FALSE' to quiet this message.
```

Randomization

Matching is usually used in observational studies to approximate random assignment to treatment. But could it be useful even in randomized studies? To explore the question do the following:

1. Generate a vector that randomly assigns each unit to either treatment or control
2. Choose a baseline covariate (for either the student or parent). A binary covariate is probably best for this exercise.
3. Visualize the distribution of the covariate by treatment/control condition. Are treatment and control balanced on this covariate?
4. Simulate the first 3 steps 10,000 times and visualize the distribution of treatment/control balance across the simulations.

(1) Generate a vector that randomly assigns each unit to treatment/control

```r
## set seed
set.seed(42)
ypsps$treatment <- sample(c("Control","Treatment"), nrow(ypsps),
                          replace = TRUE, prob = c(0.50,0.50))
table(ypsps$treatment)
```

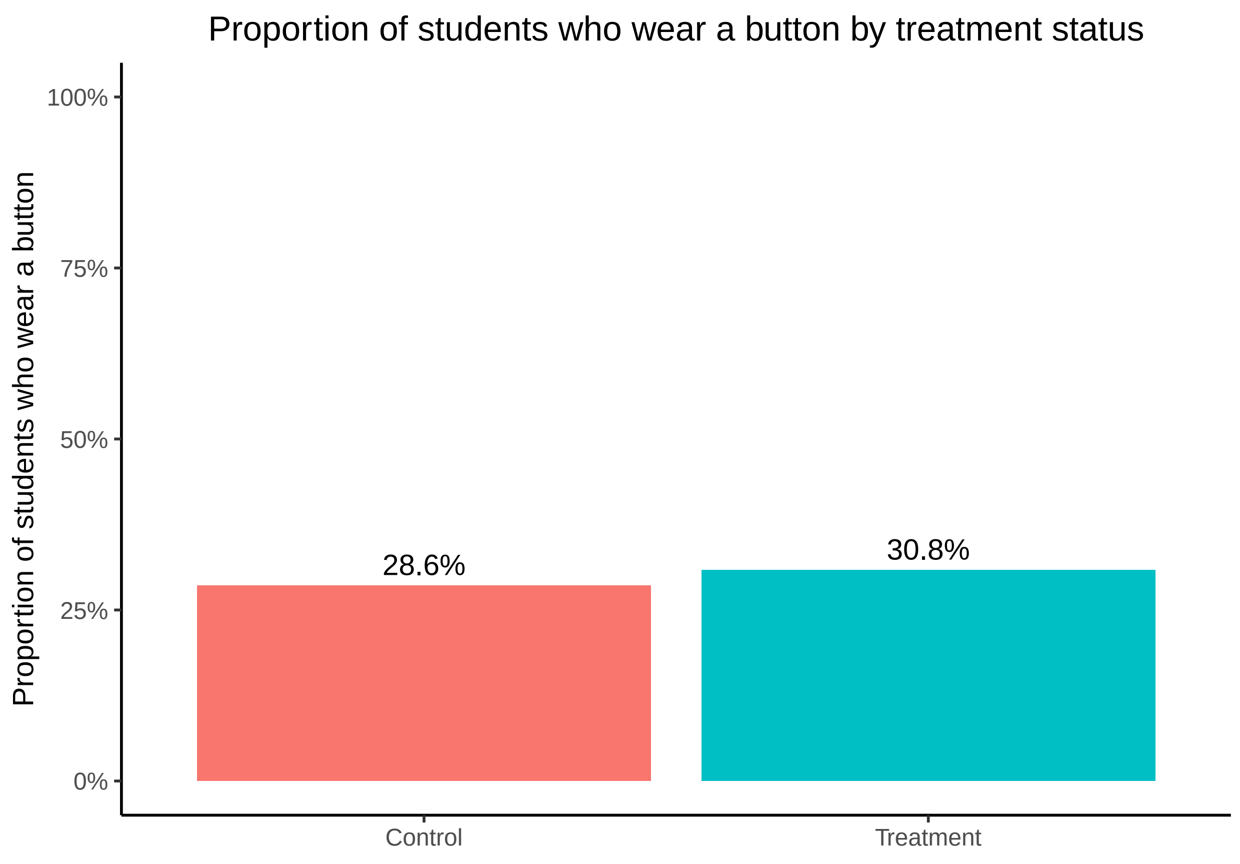
```
##
##   Control Treatment
##       602       652
```

(2) Choose a baseline covariate for either the student or the parent and (3) Visualize the distribution of the covariate by treatment/control condition.

```r
# Baseline covariate: student_button
table(ypsps$student_button)
```

```
##
##   0   1
## 881 373
```

```r
# Visualize the distribution by treatment/control (ggplot)
prop_table <- ypsps %>%
  group_by(treatment) %>%
  summarise(prop_student_button = mean(student_button, na.rm = TRUE))

# Bar plot to visualize covariate distribution by treatment group
ggplot(prop_table, aes(x = treatment, y = prop_student_button, fill = treatment,
                       label = scales::percent(prop_student_button))) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(position = position_dodge(width = 0.9), vjust = -0.5) +
  labs(title = "Proportion of students who wear a button by treatment status",
       x = NULL,
       y = "Proportion of students who wear a button") +
  scale_y_continuous(labels = scales::percent_format(scale = 100),
                     limits = c(0, 1)) +
  theme_classic() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5))
```



```r
## test whether the difference between treatment and control it's significant and it's not
chisq.test(table(ypsps$treatment, ypsps$student_button))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table(ypsps$treatment, ypsps$student_button)
## X-squared = 0.65868, df = 1, p-value = 0.417
```

Are treatment/control balanced?

At baseline, students who wear a button or post to any campaign sites are balanced between the treatment and control group (28% vs 30%, p-value = 0.417).

(4) Simulate the first three steps 10,000 times and visualize the distribution of treatment/control balance across simulations.

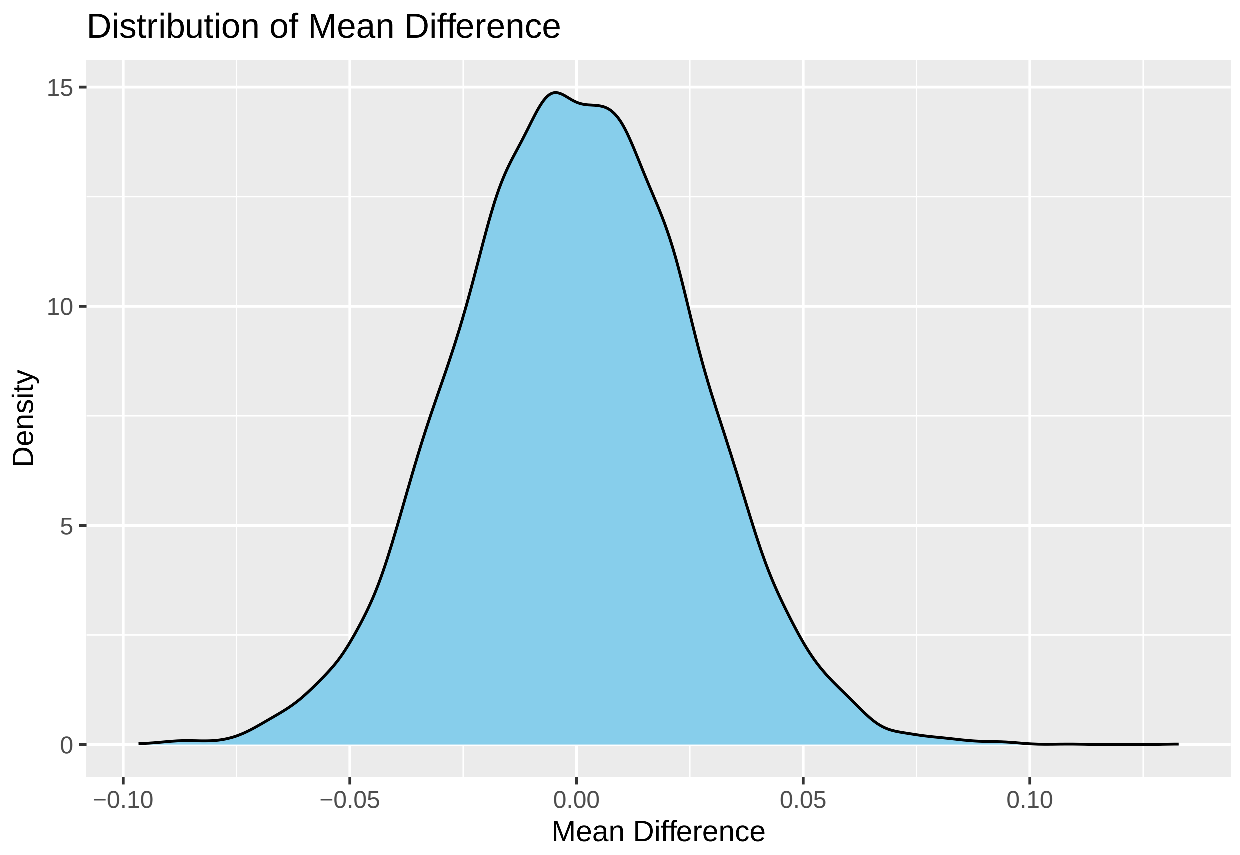
```r
# Function to simulate samples, perform t-test, and store results
simulate_and_visualize <- function(ypsps, num_simulations = 10000) {

  # Initialize empty vectors to store results
  p_values <- numeric(num_simulations)
  diff_means <- numeric(num_simulations)

  # Simulate samples, perform t-test, and store results
  for (i in 1:num_simulations) {
    # Draw sample from df
    ypsps$treatment <- sample(c("Control", "Treatment"), nrow(ypsps),
                              replace = TRUE, prob = c(0.5, 0.5))

    # Perform t-test
    t_test_result <- t.test(ypsps$student_button ~ ypsps$treatment)

    # Extract p-value and difference in means from t-test result
    p_values[i] <- t_test_result$p.value
    diff_means[i] <- t_test_result$estimate[2] - t_test_result$estimate[1]

}

# Create a dataframe to store simulation results
  results_df <- data.frame(
    simulation = 1:num_simulations,
    p_value = p_values,
    diff_in_means = diff_means
  )

  # Return the results dataframe
  return(results_df)
}

simulation_results <- simulate_and_visualize(ypsps, num_simulations = 10000)
head(simulation_results, 20)
```

```
##    simulation    p_value diff_in_means
## 1           1 0.52494591  -0.016427999
## 2           2 0.86106376  -0.004523252
## 3           3 0.47589335  -0.018425136
## 4           4 0.11097401   0.041165024
## 5           5 0.97196887  -0.000908182
## 6           6 0.24506592   0.030030130
## 7           7 0.38455609  -0.022463503
## 8           8 0.05489866   0.049658894
## 9           9 0.40701739  -0.021422411
## 10         10 0.17535152  -0.034990895
## 11         11 0.23300643  -0.030842408
## 12         12 0.86124115   0.004527177
## 13         13 0.92902236   0.002305135
## 14         14 0.93769325   0.002020562
## 15         15 0.25028798   0.029698840
## 16         16 0.49276033  -0.017724691
## 17         17 0.53153611  -0.016176190
## 18         18 0.23947830  -0.030458716
## 19         19 0.69262594   0.010215502
## 20         20 0.58078099  -0.014276300
```

```
# Estimate mean of diff in means across simulatins
summary(simulation_results$diff_in_means, na.rm = T)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.0966107 -0.0177375 -0.0002214 -0.0001614  0.0174938  0.1328211
```

```
# Plot distribution of diff-in-means across simulations
ggplot(simulation_results, aes(x = diff_in_means)) +
  geom_density(fill = "skyblue", color = "black") +
  labs(title = "Distribution of Mean Difference",
       x = "Mean Difference",
       y = "Density")
```



```r
# Estimate and print proportion of significant p-values across simulations
sum(simulation_results$p_value < 0.05) / nrow(simulation_results) ## 5%
```

```
## [1] 0.0494
```

Questions

1. **What do you see across your simulations? Why does independence of treatment assignment and baseline covariates not guarantee balance of treatment assignment and baseline covariates?**

We observe that the difference in means between the treatment and control changes from simulation to simulations. For instance, the minimum diff-in-means of -0.90 whereas the maximum is 0.13, so we can see a spectrum even if the mean is almost zero -0.0001614. This means that some simulations, just randomly, will get samples that are a bit more different than others, and this is more likely to happen in small samples. The difference in means between them is statistically significant in 5% of the simulations ran, which seems consistent with statistics.

Propensity Score Matching

One Model

Select covariates that you think best represent the "true" model predicting whether a student chooses to attend college, and estimate a propensity score model to calculate the Average Treatment Effect on the

Treated (ATT). Plot the balance of the top 10 (or fewer if you select fewer covariates). Report the balance of the p-scores across both the treatment and control groups, and using a threshold of standardized mean difference of p-score $\leq .1$, report the number of covariates that meet that balance threshold.

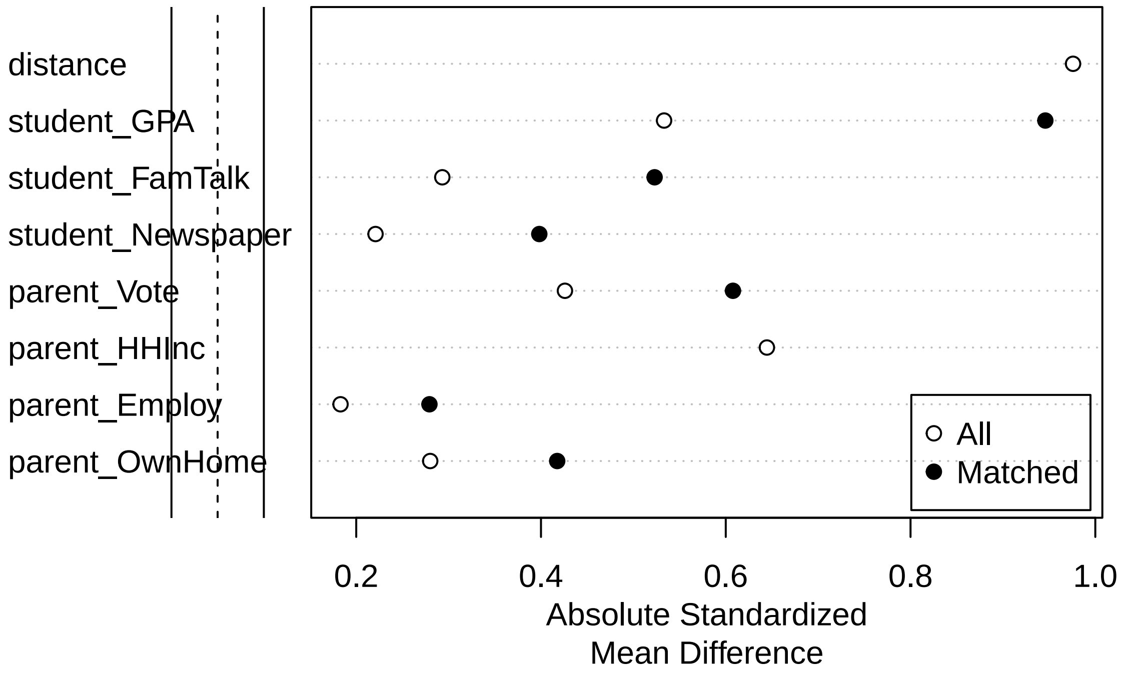
```
covariates_logistic <- c("student_GPA", "student_FamTalk",
                         "student_Newspaper", "parent_Vote",
                         "parent_HHInc", "parent_Employ", "parent_OwnHome")

formula_logistic <- as.formula(paste("college ~",
                                     paste(covariates_logistic, collapse = " + ")))

match_nearest <- matchit(formula = formula_logistic,
                         data = ypsps,
                         method = "nearest",
                         distance = "glm",
                         link = "logit",
                         discard = "control",
                         replace = FALSE,
                         ratio = 2)
bal.tab(match_nearest, un = TRUE, stats = c("m", "v", "ks"))
```

```
## Balance Measures
##                       Type Diff.Un V.Ratio.Un  KS.Un Diff.Adj V.Ratio.Adj
## distance          Distance  0.9759     0.6875 0.3604   1.6097      0.1046
## student_GPA        Contin. -0.5332     0.9101 0.2328  -0.9459      0.7102
## student_FamTalk    Contin. -0.2932     0.7770 0.1018  -0.5230      0.5683
## student_Newspaper  Contin. -0.2209     0.8638 0.1240  -0.3982      0.5831
## parent_Vote         Binary  0.1350          . 0.1350   0.1927           .
## parent_HHInc       Contin.  0.6445     0.9501 0.2613   1.0709      0.4514
## parent_Employ       Binary  0.0842          . 0.0842   0.1284           .
## parent_OwnHome      Binary  0.1015          . 0.1015   0.1514           .
##                   KS.Adj
## distance          0.7706
## student_GPA       0.4266
## student_FamTalk   0.2041
## student_Newspaper 0.1812
## parent_Vote       0.1927
## parent_HHInc      0.4472
## parent_Employ     0.1284
## parent_OwnHome    0.1514
##
## Sample sizes
##           Control Treated
## All           451     803
## Matched       436     436
## Unmatched       0     367
## Discarded      15       0
```

```
# estimate ATT
matching_att <- match.data(match_nearest)
dependent_variable <- "student_ppnscal"
independent_variables <- c("college", covariates_logistic)
formula_att <- as.formula(paste(dependent_variable, "~",
                                paste(independent_variables, collapse = " + ")))

lm_att <- lm(formula = formula_att,
                      data = matching_att,
                      weights = weights)

## get ATT
ATT_nearest <- coef(lm_att)["college"] # Extract ATT
ATT_nearest
```

```
##  college
## 1.297872
```

```
# Plot the balance for the top 10 covariates
plot(summary(match_nearest))
```



```
## love.plot(match_nearest)
## I cannot make this package work, I get the error:
## Error in scale_override_call(call): could not find function "scale_override_call".

# Report the balance of the p-scores across both the treatment and control groups,
## and using a threshold of standardized mean difference of p-score <= .1,
## report the number of covariates that meet the balance threshold
summary(match_nearest)$sum.matched %>% as.data.frame()
```

```
##                     Means Treated Means Control Std. Mean Diff. Var. Ratio
## distance                0.8167649     0.5517196       1.6097397  0.1046316
## student_GPA             1.9816514     2.6009174      -0.9459284  0.7101771
## student_FamTalk         1.6123853     2.0940367      -0.5229888  0.5683319
## student_Newspaper       1.6857798     2.2087156      -0.3981687  0.5830771
## parent_Vote             0.9655963     0.7729358       0.6077817         NA
## parent_HHInc            8.3302752     6.0344037       1.0709025  0.4513959
## parent_Employ           0.7568807     0.6284404       0.2792648         NA
## parent_OwnHome          0.9059633     0.7545872       0.4175449         NA
##                     eCDF Mean  eCDF Max Std. Pair Dist.
## distance            0.4013761 0.7706422       1.6097397
## student_GPA         0.1238532 0.4266055       1.0650453
## student_FamTalk     0.1204128 0.2041284       1.0559393
## student_Newspaper   0.1045872 0.1811927       0.9395385
## parent_Vote         0.1926606 0.1926606       0.7669627
## parent_HHInc        0.2295872 0.4472477       1.2206790
## parent_Employ       0.1284404 0.1284404       0.8677158
## parent_OwnHome      0.1513761 0.1513761       0.7844782
```

Report the balance of the p-scores across both the treatment and control groups, and using a threshold of standardized mean difference of p-score $\leq .1$, report the number of covariates that meet that balance threshold.

```
summary_df <- summary(match_nearest)
df <- as.data.frame(summary_df$sum.matched)

## report the balance of the p-scores across both the treatment and control groups,
ypsps$pscore <- match_nearest$distance
ypsps %>% group_by(college) %>% summarize(mean(pscore))
```

```
## # A tibble: 2 x 2
##   college `mean(pscore)`
##     <dbl>          <dbl>
## 1       0          0.537
## 2       1          0.698
```

```
## Using a threshold of standardized mean difference of p-score $\leq .1$,
## report the number of covariates that meet that balance threshold.
num_balanced_covariates  <- nrow(df %>% filter(`Std. Mean Diff.` <= 0.1))
num_balanced_covariates ## 3
```

```
## [1] 3
```

Simulations

Henderson/Chatfield argue that an improperly specified propensity score model can actually *increase* the bias of the estimate. To demonstrate this, they simulate 800,000 different propensity score models by choosing different permutations of covariates. To investigate their claim, do the following:

- Using as many simulations as is feasible (at least 10,000 should be ok, more is better!), randomly select the number of and the choice of covariates for the propensity score model.

- For each run, store the ATT, the proportion of covariates that meet the standardized mean difference $\leq .1$ threshold, and the mean percent improvement in the standardized mean difference. You may also wish to store the entire models in a list and extract the relevant attributes as necessary.
- Plot all of the ATTs against all of the balanced covariate proportions. You may randomly sample or use other techniques like transparency if you run into overplotting problems. Alternatively, you may use plots other than scatterplots, so long as you explore the relationship between ATT and the proportion of covariates that meet the balance threshold.
- Finally choose 10 random models and plot their covariate balance plots (you may want to use a library like gridExtra to arrange these)

Note: There are lots of post-treatment covariates in this dataset (about 50!)! You need to be careful not to include these in the pre-treatment balancing. Many of you are probably used to selecting or dropping columns manually, or positionally. However, you may not always have a convenient arrangement of columns, nor is it fun to type out 50 different column names. Instead see if you can use dplyr 1.0.0 functions to programatically drop post-treatment variables (here is a useful tutorial).

```r
## I do the first part with the simulation in another R script and import the function here
head(result_nearest$results_df,10)
```

```
##    simulation num_covariates prop_balanced_covariates       att
## 1           1             12                0.0000000 0.4742426
## 2           2             11                0.2857143 0.7772062
## 3           3              8                0.3636364 0.9161596
## 4           4             10                0.3846154 0.7097679
## 5           5             17                0.2500000 1.0387742
## 6           6             13                0.3125000 1.0588856
## 7           7             13                0.1875000 0.9734578
## 8           8             12                0.3333333 0.8408319
## 9           9             17                0.3500000 0.6650932
## 10         10              5                0.1250000 1.2177053
##    mean_pct_improvement
## 1               0.00000
## 2              38.96903
## 3              53.22331
## 4              49.95281
## 5              32.38192
## 6              41.20184
## 7              41.30201
## 8              46.56337
## 9              36.87070
## 10             48.85829
```

```r
## Number of simulations for which the proportion of balanced covariates is above 70%
sum(result_nearest$results_df$prop_balanced_covariates > 0.70)
```

```
## [1] 2
```

```r
## Number of simulations for which the proportion of balanced covariates is above 60%
sum(result_nearest$results_df$prop_balanced_covariates > 0.60)
```

```
## [1] 15
```

```
## Number of simulations for which the proportion of balanced covariates is above 50%
sum(result_nearest$results_df$prop_balanced_covariates > 0.50)
```

```
## [1] 60
```

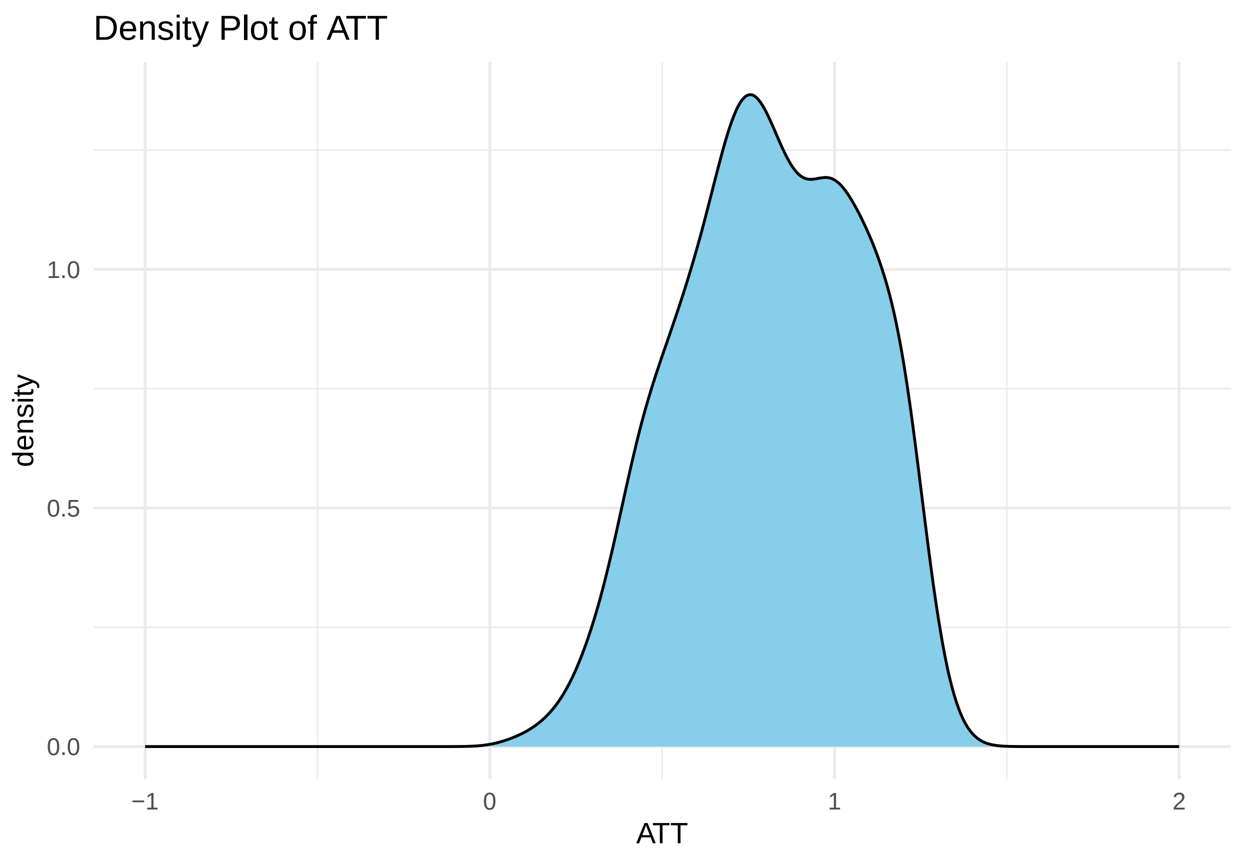
```
print("ATT summary across simulations for results_nearest")
```

```
## [1] "ATT summary across simulations for results_nearest"
```

```
summary(result_nearest$results_df$att)
```

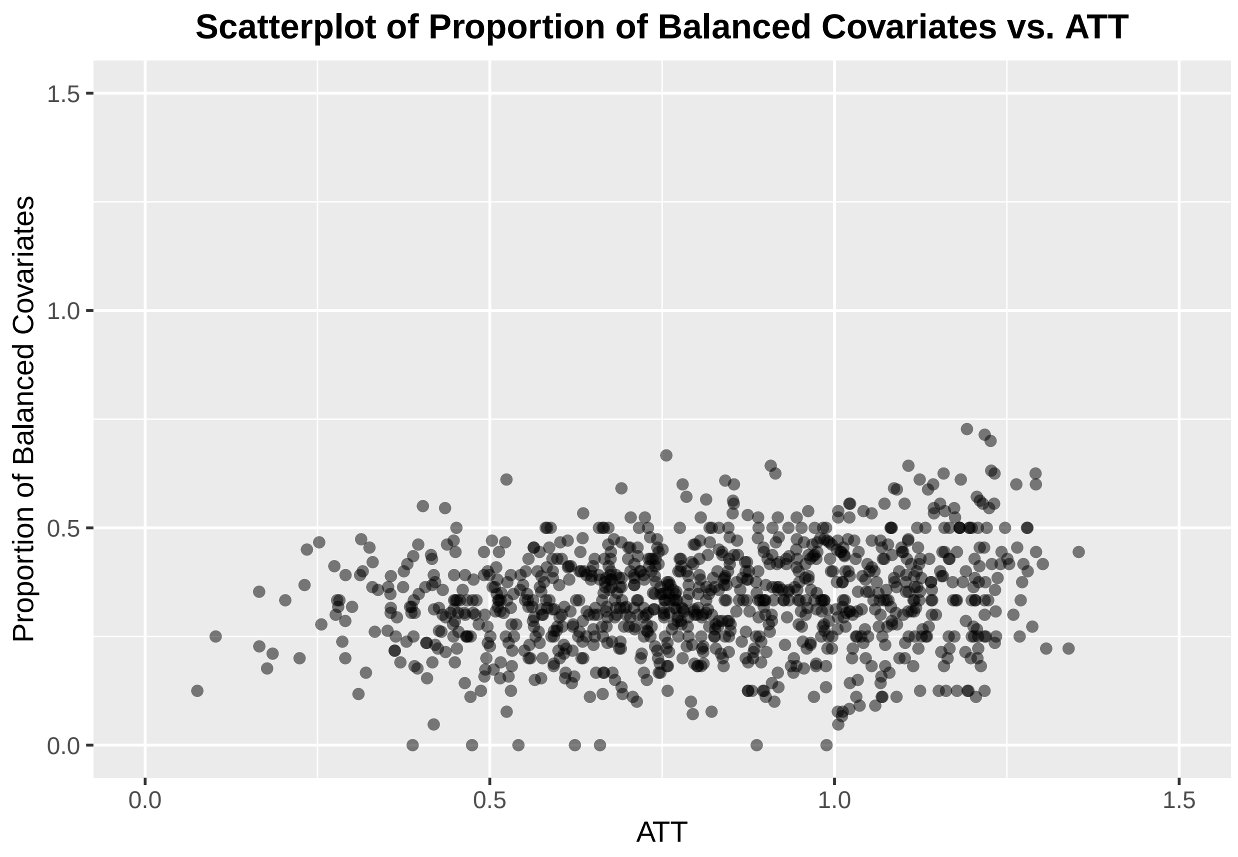
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.07575 0.62755 0.81367 0.81357 1.01973 1.35433
```

```
# Plotting distribution of ATT across simulations
ggplot(result_nearest$results_df, aes(x = att)) +
  geom_density(fill = "skyblue", color = "black") +
  labs(title = "Density Plot of ATT", x = "ATT") +
  theme_minimal() +
   xlim(-1, 2)
```



Plot all of the ATTs against all of the balanced covariate proportions. You may randomly sample or use other techniques like transparency if you run into overplotting problems. Alternatively, you may use plots other than scatterplots, so long as you explore the relationship between ATT and the proportion of covariates that meet the balance threshold.

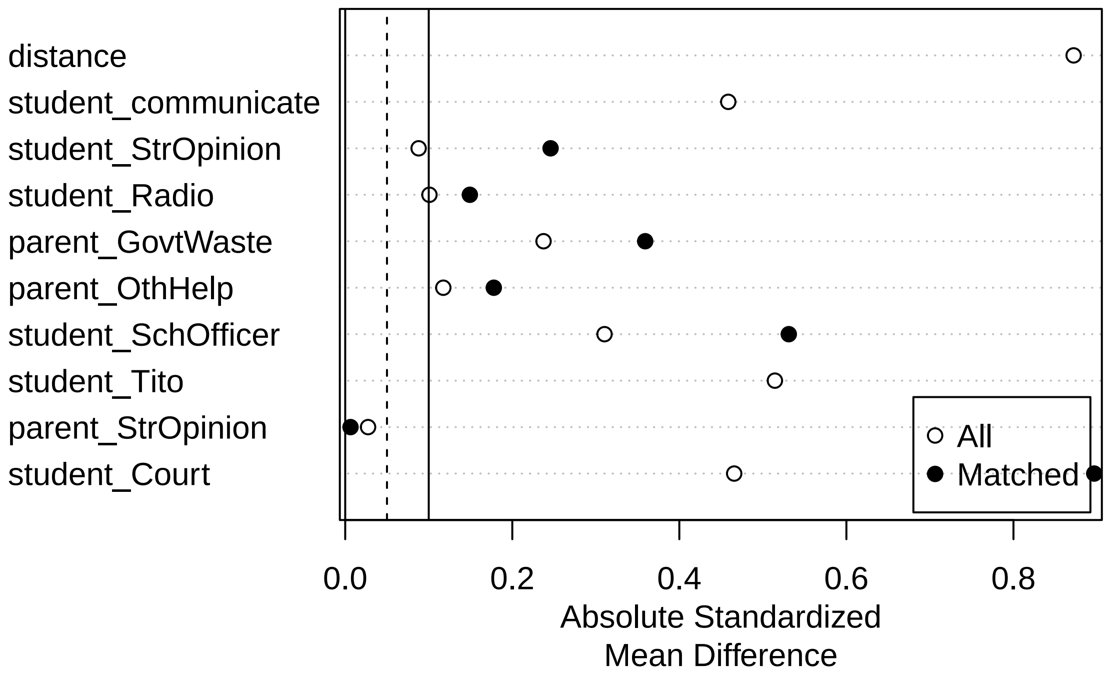
```r
# Create a data frame for plotting
plot_data <- data.frame(
  ATT = result_nearest$results_df$att,
  Balanced_Covariate_Proportion = result_nearest$results_df$prop_balanced_covariates
)

# Create a scatterplot
ggplot(plot_data, aes(x = ATT, y = Balanced_Covariate_Proportion)) +
  geom_point(alpha = 0.5) +
  labs(x = "ATT", y = "Proportion of Balanced Covariates") +
  ggtitle("Scatterplot of Proportion of Balanced Covariates vs. ATT") +
  xlim(0, 1.5) +
  ylim(0, 1.5) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```

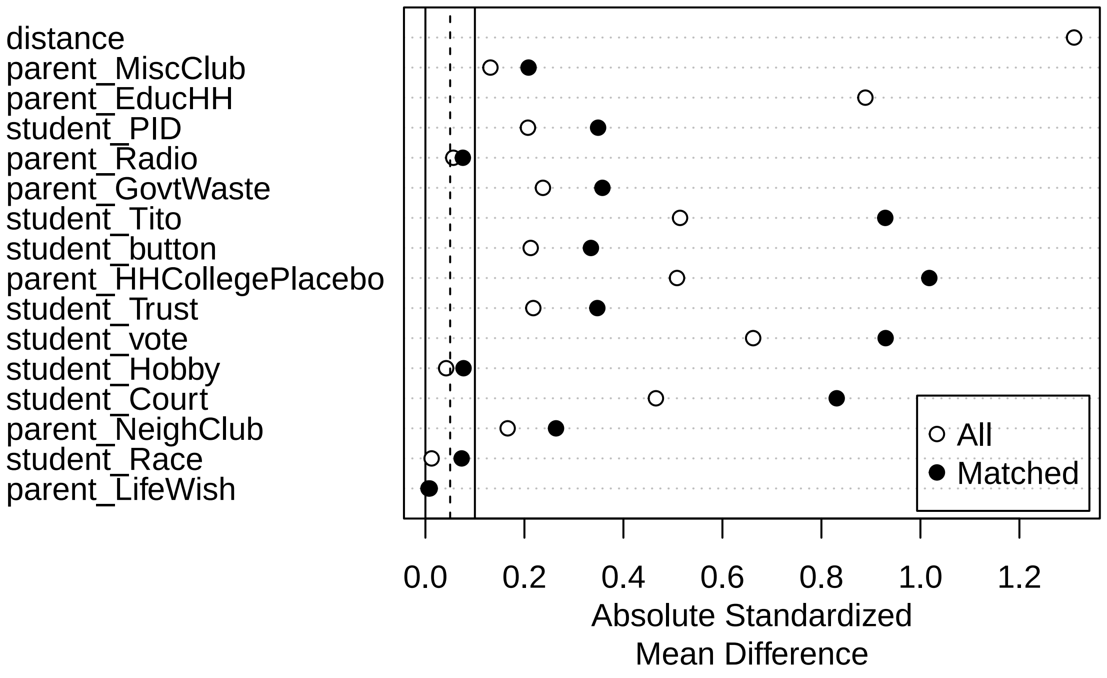


Finally choose 10 random models and plot their covariate balance plots (you may want to use a library like gridExtra to arrange these)

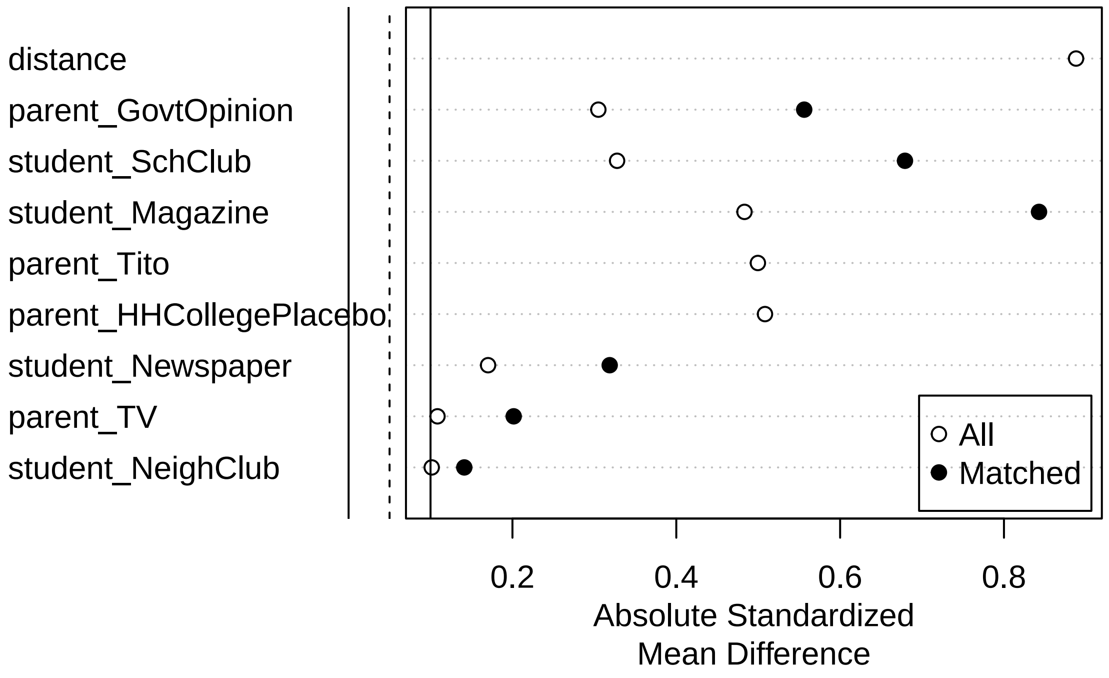
```r
# Randomly sample 10 indices from the list of models
sampled_indices <- sample(seq_along(result_nearest$match_nearest), size = 10,
                          replace = FALSE)
random_models <- result_nearest$match_nearest[sampled_indices]
```

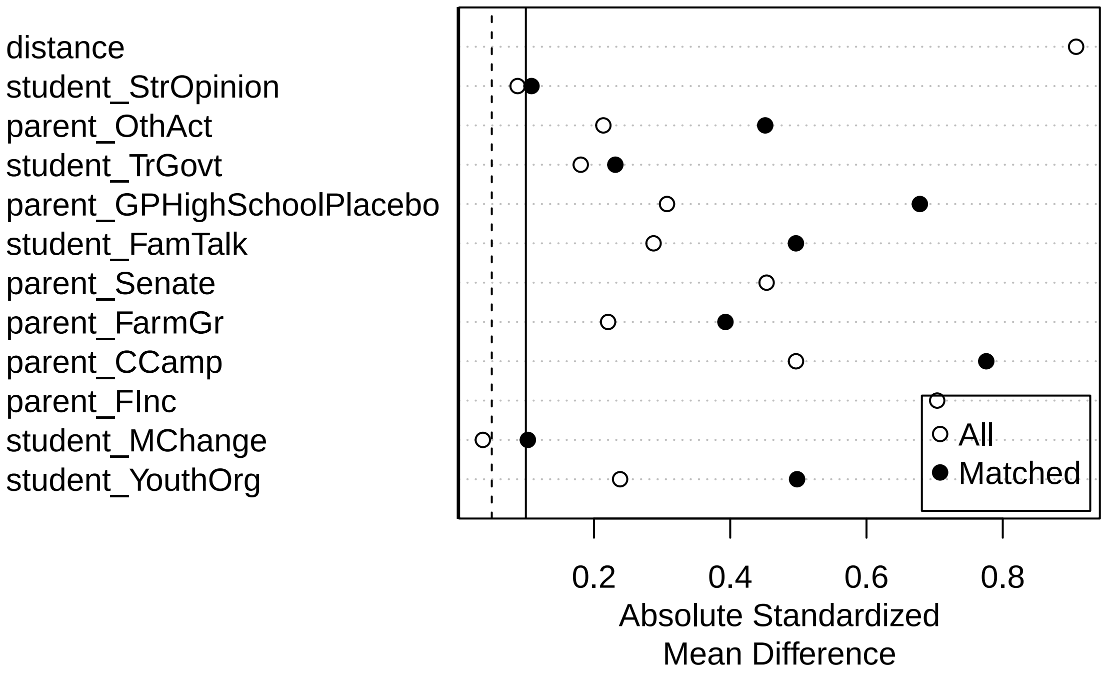
```r
# Function to plot the balance of covariates in each model
plot_summary <- function(model) {
  plot(summary(model))
}

# Apply the plot_summary function to each model in random_models
plots_list <- lapply(random_models, plot_summary)
```



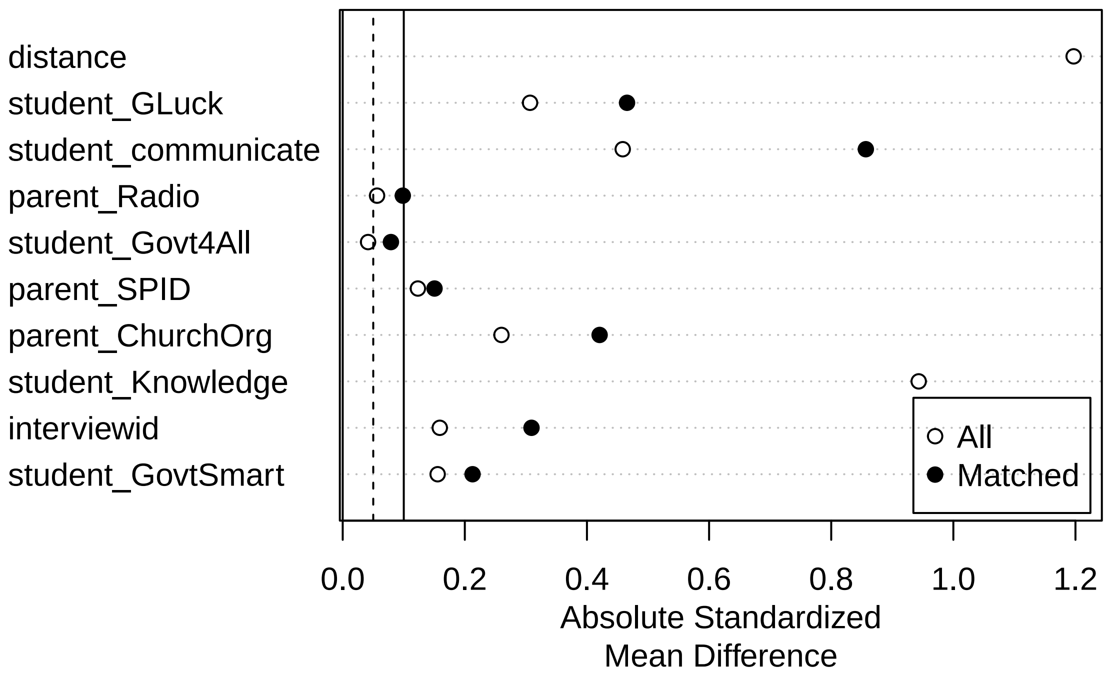


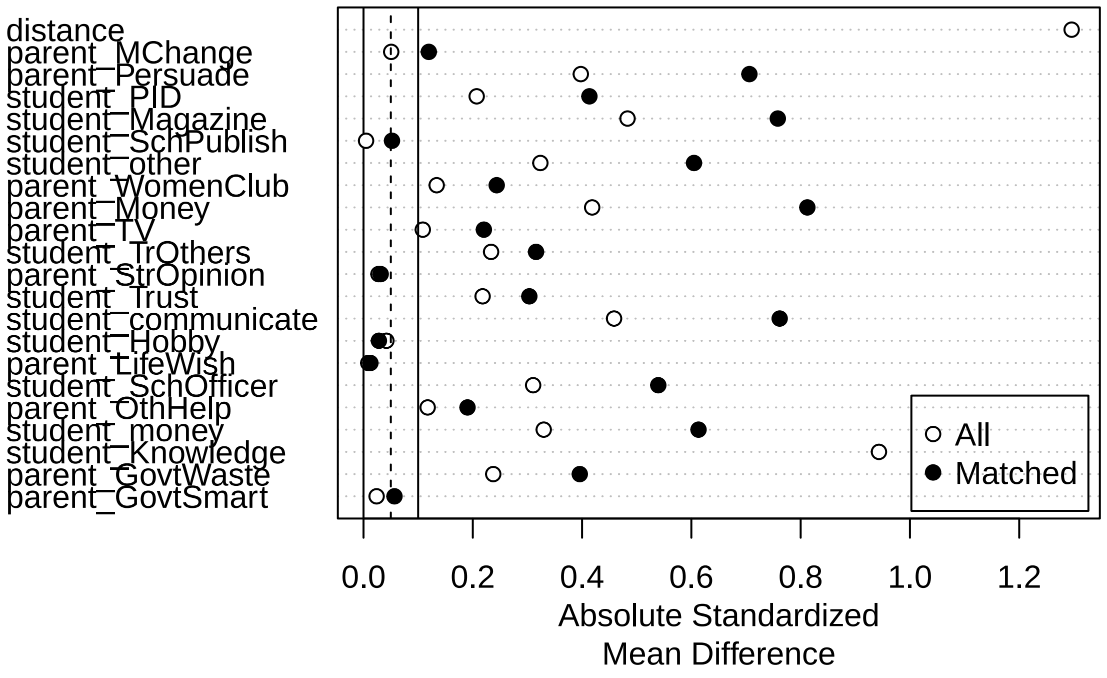
distance
parent_MiscClub
parent_EducHH
student_PID
parent_Radio
parent_GovtWaste
student_Tito
student_button
parent_HHCollegePlacebo
student_Trust
student_vote
student_Hobby
student_Court
parent_NeighClub
student_Race
parent_LifeWish
○ All
● Matched
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Absolute Standardized Mean Difference


distance
parent_GovtOpinion
student_SchClub
student_Magazine
parent_Tito
parent_HHCollegePlacebo
student_Newspaper
parent_TV
student_NeighClub
○ All
● Matched
0.2
0.4
0.6
0.8
Absolute Standardized Mean Difference

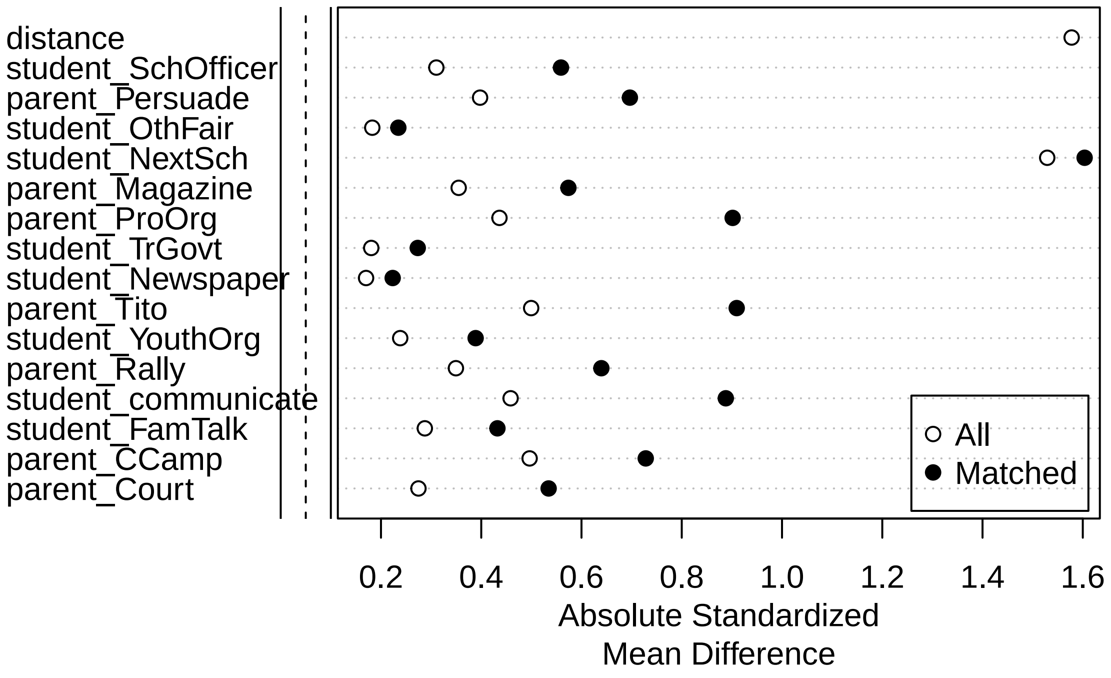


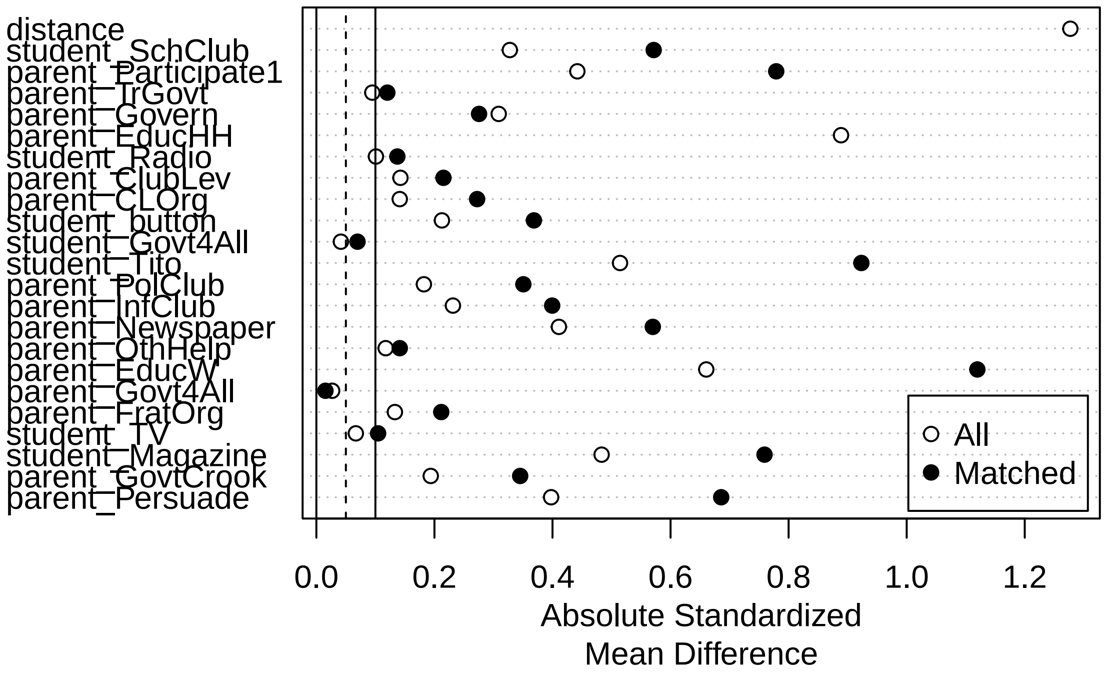
distance
student_StrOpinion
parent_OthAct
student_TrGovt
parent_GPHighSchoolPlacebo
student_FamTalk
parent_Senate
parent_FarmGr
parent_CCamp
parent_FInc
student_MChange
student_YouthOrg
All
Matched
0.2
0.4
0.6
0.8
Absolute Standardized Mean Difference

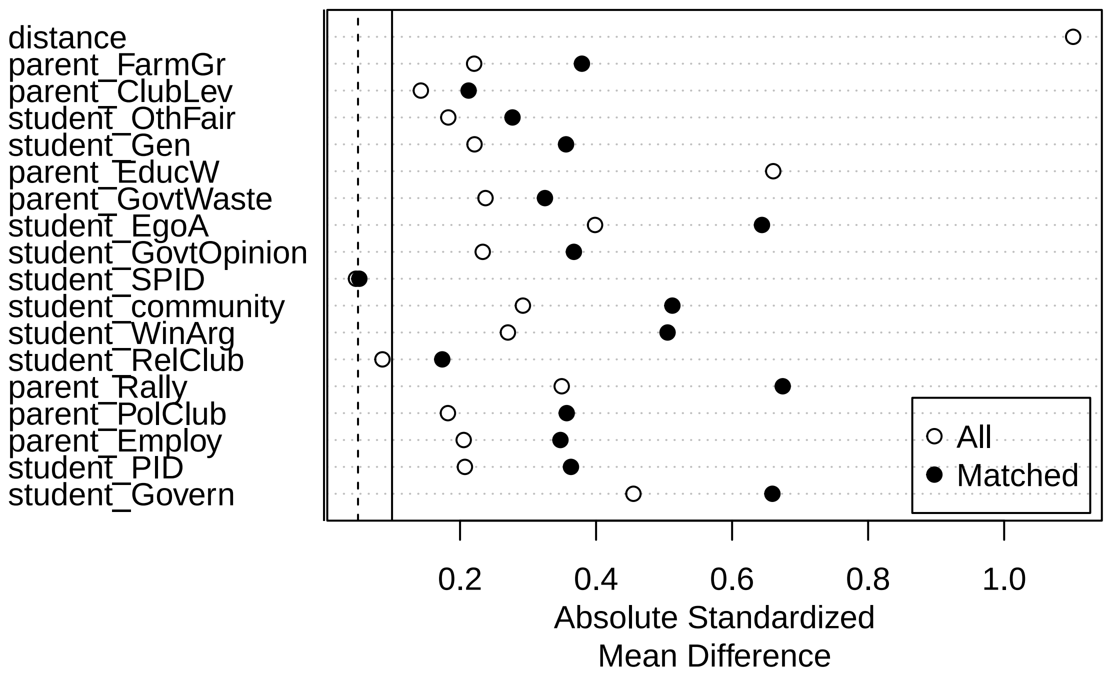

distance
student_GLuck
student_communicate
parent_Radio
student_Govt4All
parent_SPID
parent_ChurchOrg
student_Knowledge
interviewid
student_GovtSmart
All
Matched
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Absolute Standardized
Mean Difference

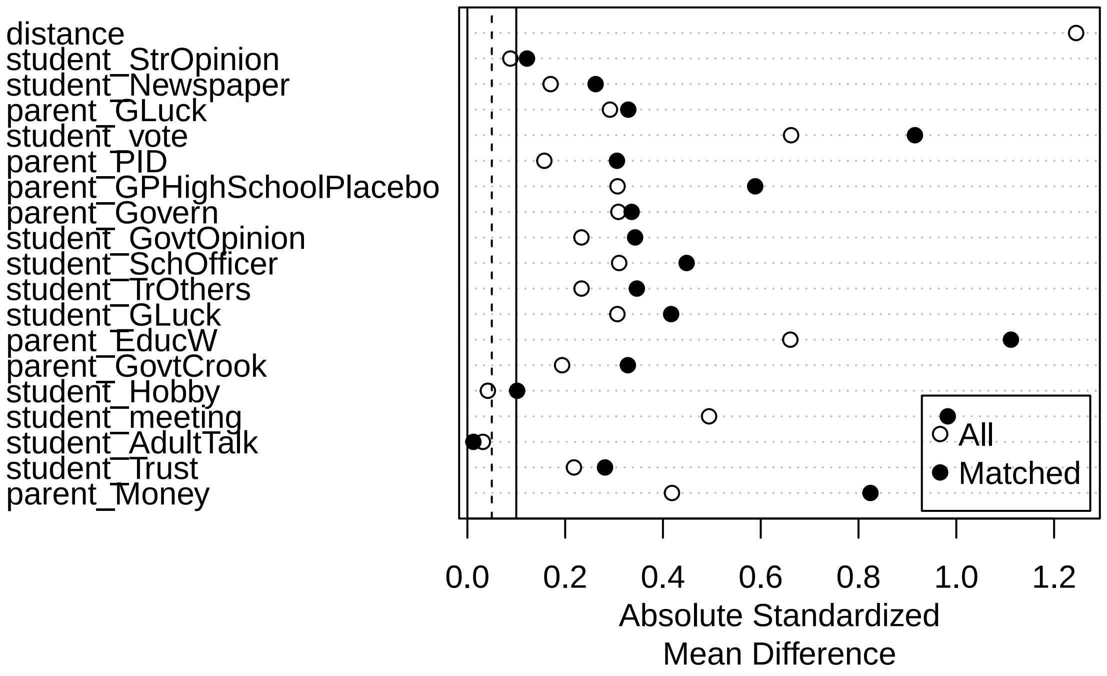


distance
parent_MChange
parent_Persuade
student_PID
student_Magazine
student_SchPublish
student_other
parent_WomenClub
parent_Money
parent_TV
student_TrOthers
parent_StrOpinion
student_Trust
student_communicate
student_Hobby
parent_LifeWish
student_SchOfficer
parent_OthHelp
student_money
student_Knowledge
parent_GovtWaste
parent_GovtSmart
All
Matched
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Absolute Standardized Mean Difference


distance
student_SchOfficer
parent_Persuade
student_OthFair
student_NextSch
parent_Magazine
parent_ProOrg
student_TrGovt
student_Newspaper
parent_Tito
student_YouthOrg
parent_Rally
student_communicate
student_FamTalk
parent_CCamp
parent_Court
All
Matched
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Absolute Standardized
Mean Difference


distance
student_SchClub
parent_Participate1
parent_TrGovt
parent_Govern
parent_EducHH
student_Radio
parent_ClubLev
parent_CLOrg
student_button
student_Govt4All
student_Tito
parent_PolClub
parent_InfClub
parent_Newspaper
parent_OthHelp
parent_EducW
parent_Govt4All
parent_FratOrg
student_TV
student_Magazine
parent_GovtCrook
parent_Persuade
All
Matched
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Absolute Standardized
Mean Difference


distance
parent_FarmGr
parent_ClubLev
student_OthFair
student_Gen
parent_EducW
parent_GovtWaste
student_EgoA
student_GovtOpinion
student_SPID
student_community
student_WinArg
student_RelClub
parent_Rally
parent_PolClub
parent_Employ
student_PID
student_Govern
All
Matched
0.2
0.4
0.6
0.8
1.0
Absolute Standardized Mean Difference



```
## I read that supposedly extragrid converts your plots to grobs (https://stackoverflow.com/questions/3
##grid.arrange(grobs = plots_list)
##grid.arrange(grobs = plots_list, ncol = 10)
```

Questions

1. **How many simulations resulted in models with a higher proportion of balanced covariates? Do you have any concerns about this?**. Half of the models resulted in a proportion of balanced covariates of 37

2. **Analyze the distribution of the ATTs. Do you have any concerns about this distribution?**. Yes, we observe that the ATT varies, as in some simulations it is negative and in other simulations it goes above 1. So different random simulations would give you very different views about the relationships of the variables you're analyzing.

3. **Do your 10 randomly chosen covariate balance plots produce similar numbers on the same covariates? Is it a concern if they do not?**. Not necessarily. For example for the covariate parent CivicOrg they do, but for the covariate student Radio they are very different in different plots. I guess the post-matching balance for each covariate is conditional on the set of covariates that you choose in the model. I think maybe this is why context knowledge is important, and that ultimately you choose variables that make sense theory-wise.

Matching Algorithm of Your Choice

Simulate Alternative Model

Henderson/Chatfield propose using genetic matching to learn the best weights for Mahalanobis distance matching. Choose a matching algorithm other than the propensity score (you may use genetic matching if you wish, but it is also fine to use the greedy or optimal algorithms we covered in lab instead). Repeat the same steps as specified in Section 4.2 and answer the following questions:

```
head(result_optimal$results_df, 10)
```

```
##    simulation num_covariates prop_balanced_covariates                      att
## 1           1             72                0.0000000  0.191530887672607119665
## 2           2            112                0.6173913  0.000000000000000675458
## 3           3             47                0.6600000  0.023334601524795883786
## 4           4            108                0.5945946 -0.000000000000000304931
## 5           5            121                0.5950413  0.000000000000000283365
## 6           6             75                0.7307692  0.132045708221755514477
## 7           7             56                0.5423729 -0.030339795693779217367
## 8           8             82                0.5294118  0.320850298789045307312
## 9           9             26                0.6551724  0.422704183080417406070
## 10         10             31                0.7058824  0.211859003644192217530
##    mean_pct_improvement
## 1               0.00000
## 2            -152.64336
## 3            -121.16033
## 4            -169.00777
## 5            -146.40634
## 6            -231.86599
## 7            -109.96263
## 8            -107.68256
## 9             -88.91473
## 10           -196.28616
```

```
# Print summary for result_nearest
print("Summary proportion of balanced covariates for result_nearest:")
```

```
## [1] "Summary proportion of balanced covariates for result_nearest:"
```

```
summary(result_nearest$results_df$prop_balanced_covariates)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.2500  0.3333  0.3354  0.4167  0.7273
```

```
# Print summary for result_optimal
print("Summary proportion of balanced covariates for result_optimal:")
```

```
## [1] "Summary proportion of balanced covariates for result_optimal:"
```

```
summary(result_optimal$results_df$prop_balanced_covariates)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.0000  0.5843  0.6250  0.6463  0.6855  1.0000 
```

```
# Print summary for result_nearest
print("Summary or mean percent improvement for result_nearest:")
```

```
## [1] "Summary or mean percent improvement for result_nearest:"
```

```
print(summary(result_nearest$results_df$mean_pct_improvement))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  -83.12   37.93   42.15   42.57   47.13  285.45 
```

```
# Print summary for result_optimal
print("Summary of mean percent improvement for result_optimal:")
```

```
## [1] "Summary of mean percent improvement for result_optimal:"
```

```
print(summary(result_optimal$results_df$mean_pct_improvement))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
## -77170.1   -181.6   -146.4   -307.6   -123.7  13985.8 
```

Questions

1. **Does your alternative matching method have more runs with higher proportions of balanced covariates?**. No, both methods show an average across simulations of 36
2. **Use a visualization to examine the change in the distribution of the percent improvement in balance in propensity score matching vs. the distribution of the percent improvement in balance in your new method. Which did better? Analyze the results in 1-2 sentences.**. The mean mean percent improvement across simulations is basically the same (36

Optional: Looking ahead to the discussion questions, you may choose to model the propensity score using an algorithm other than logistic regression and perform these simulations again, if you wish to explore the second discussion question further.

Discussion Questions

1. **Why might it be a good idea to do matching even if we have a randomized or as-if-random design?**. Matching can help reduce the influence of confounding variables, because by matching you make these variables balanced across the treatment and control group. It can also achieve more efficient estimates by reducing variability within matched pairs. By knowing that you're actually comparing someone in the treatment group who's very similar to someone in the control group, it might be easier for a non-academic audience to understand what you're doing and interpreting results. You could also use matching as a robustness check that your results are valid. Also, even if you have a randomized study, sometimes randomization is not perfect, and matching can help control for selection bias.

2. **The standard way of estimating the propensity score is using a logistic regression to estimate probability of treatment. Given what we know about the curse of dimensionality, do you think there might be advantages to using other machine learning algorithms (decision trees, bagging/boosting forests, ensembles, etc.) to estimate propensity scores instead?**. Yes. In propensity score matching, you're selecting a number of covariates to predict the likelihood that someone would get treated. This selection has to be smart, and here is where maching learning could make a difference. Logistic regression, which is what is generally used to estimate the propensity score, addresses linear relationships, so for instance machine learning methods can help model nonlinear relationships between covariates and treatment assignment. In this exercise we selected covariates arbitrarily first and then we ran simulations with different random groups of covariates. In this case, machine learning models can perform feature selection and regularization techniques and keep the most relevant variables to estimate the propensity score.