Project 6: Randomization and Matching

Alex Schulte

April 4, 2024

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.

```
# Load tidyverse and MatchIt
# Feel free to load other libraries as you wish
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## 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 errors
```

```
library(MatchIt)

library(ggplot2)

# Load ypsps data
ypsps <- read_csv('data/ypsps.csv')
```

```
## 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.
```

```
head(ypsps)
```

```
## # 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>, ...
```

Randomization

Matching is usually used in observational studies to 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.

```
# first, I want to see the oucome values for these variables to see which are binary:

unique_values_student_vote <- distinct(ypsps, student_vote)
unique_values_student_meeting <- distinct(ypsps, student_meeting)
unique_values_student_money <- distinct(ypsps, student_money)
unique_values_Gen <- distinct(ypsps, student_Gen)

print(unique_values_student_vote)
```

```
## # A tibble: 2 x 1
##   student_vote
##          <dbl>
## 1            1
## 2            0
```

```
print(unique_values_student_meeting)
```

```
## # A tibble: 2 x 1
##   student_meeting
##             <dbl>
## 1               0
## 2               1
```

```
print(unique_values_student_money)
```

```
## # A tibble: 2 x 1
##   student_money
##           <dbl>
## 1             0
## 2             1
```

```
print(unique_values_Gen)
```

```
## # A tibble: 2 x 1
##   student_Gen
##         <dbl>
## 1           0
## 2           1
```

```
# Generate a vector that randomly assigns each unit to treatment/control

set.seed(123) # For reproducibility
ypsps <- ypsps %>%
```

```
  mutate(treatment_random = sample(c("Treatment", "Control"),
                                   n(), # n() gets the number of rows in the dataframe
                                   replace = TRUE))

colnames(ypsps)
```

```
##   [1] "interviewid"                "college"
##   [3] "student_vote"               "student_meeting"
##   [5] "student_other"              "student_button"
##   [7] "student_money"              "student_communicate"
##   [9] "student_demonstrate"        "student_community"
##  [11] "student_ppnscal"            "student_PubAff"
##  [13] "student_Newspaper"          "student_Radio"
##  [15] "student_TV"                 "student_Magazine"
##  [17] "student_FamTalk"            "student_FrTalk"
##  [19] "student_AdultTalk"          "student_PID"
##  [21] "student_SPID"               "student_GovtOpinion"
##  [23] "student_GovtCrook"          "student_GovtWaste"
##  [25] "student_TrGovt"             "student_GovtSmart"
##  [27] "student_Govt4All"           "student_Cynic"
##  [29] "student_LifeWish"           "student_GLuck"
##  [31] "student_FPlans"             "student_EgoA"
##  [33] "student_WinArg"             "student_StrOpinion"
##  [35] "student_MChange"            "student_EgoB"
##  [37] "student_TrOthers"           "student_OthHelp"
##  [39] "student_OthFair"            "student_Trust"
##  [41] "student_Senate"             "student_Tito"
##  [43] "student_Court"              "student_Govern"
##  [45] "student_CCamp"              "student_FDR"
##  [47] "student_Knowledge"          "student_NextSch"
##  [49] "student_GPA"                "student_SchOfficer"
##  [51] "student_SchPublish"         "student_Hobby"
##  [53] "student_SchClub"            "student_OccClub"
##  [55] "student_NeighClub"          "student_RelClub"
##  [57] "student_YouthOrg"           "student_MiscClub"
##  [59] "student_ClubLev"            "student_Phone"
##  [61] "student_Gen"                "student_Race"
##  [63] "parent_Newspaper"           "parent_Radio"
##  [65] "parent_TV"                  "parent_Magazine"
##  [67] "parent_LifeWish"            "parent_GLuck"
##  [69] "parent_FPlans"              "parent_WinArg"
##  [71] "parent_StrOpinion"          "parent_MChange"
##  [73] "parent_TrOthers"            "parent_OthHelp"
##  [75] "parent_OthFair"             "parent_PID"
##  [77] "parent_SPID"                "parent_Vote"
##  [79] "parent_Persuade"            "parent_Rally"
##  [81] "parent_OthAct"              "parent_PolClub"
##  [83] "parent_Button"              "parent_Money"
##  [85] "parent_Participate1"        "parent_Participate2"
##  [87] "parent_ActFrq"              "parent_GovtOpinion"
##  [89] "parent_GovtCrook"           "parent_GovtWaste"
##  [91] "parent_TrGovt"              "parent_GovtSmart"
##  [93] "parent_Govt4All"            "parent_Employ"
##  [95] "parent_EducHH"              "parent_EducW"
```

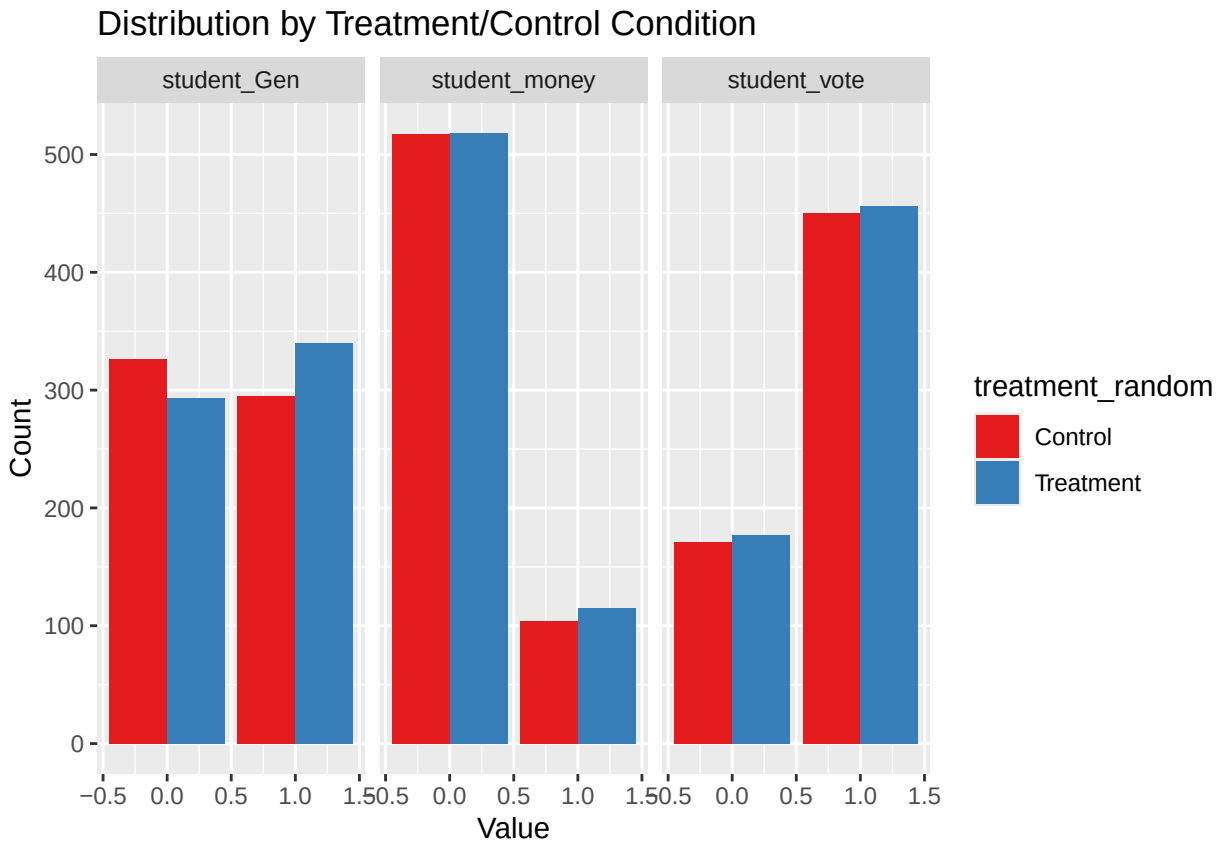
```
##  [97] "parent_ChurchOrg"             "parent_FratOrg"
##  [99] "parent_ProOrg"                "parent_CivicOrg"
## [101] "parent_CLOrg"                 "parent_NeighClub"
## [103] "parent_SportClub"             "parent_InfClub"
## [105] "parent_FarmGr"                "parent_WomenClub"
## [107] "parent_MiscClub"              "parent_ClubLev"
## [109] "parent_FInc"                  "parent_HHInc"
## [111] "parent_OwnHome"               "parent_Senate"
## [113] "parent_Tito"                  "parent_Court"
## [115] "parent_Govern"                "parent_CCamp"
## [117] "parent_FDR"                   "parent_Knowledge"
## [119] "parent_Gen"                   "parent_Race"
## [121] "parent_GPHighSchoolPlacebo"   "parent_HHCollegePlacebo"
## [123] "student_1973Married"          "student_1973Military"
## [125] "student_1973Drafted"          "student_1973Unemployed"
## [127] "student_1973NoEmployers"      "student_1973OwnHome"
## [129] "student_1973NoResidences"     "student_1973VoteNixon"
## [131] "student_1973VoteMcgovern"     "student_1973CollegeDegree"
## [133] "student_1973CurrentCollege"   "student_1973CollegeYears"
## [135] "student_1973HelpMinority"     "student_1973Busing"
## [137] "student_1973GovChange"        "student_1973VietnamRight"
## [139] "student_1973VietnamApprove"   "student_1973Trust"
## [141] "student_1973Luck"             "student_1973SureAboutLife"
## [143] "student_1973CurrentSituation" "student_1973FutureSituation"
## [145] "student_1973ThermMilitary"    "student_1973ThermRadical"
## [147] "student_1973ThermDems"        "student_1973ThermRep"
## [149] "student_1973ThermBlack"       "student_1973ThermWhite"
## [151] "student_1973ThermNixon"       "student_1973ThermMcgovern"
## [153] "student_1973Newspaper"        "student_1973PubAffairs"
## [155] "student_1973GovtEfficacy"     "student_1973GovtNoSay"
## [157] "student_1973PartyID"          "student_1973IncSelf"
## [159] "student_1973HHInc"            "student_1973ChurchAttend"
## [161] "student_1973Knowledge"        "student_1973Ideology"
## [163] "student_1982vote76"           "student_1982vote80"
## [165] "student_1982meeting"          "student_1982other"
## [167] "student_1982button"           "student_1982money"
## [169] "student_1982communicate"      "student_1982demonstrate"
## [171] "student_1982community"        "student_1982IncSelf"
## [173] "student_1982HHInc"            "student_1982College"
## [175] "treatment_random"
```

In the PDF, 3.4 step says to simulate the first 3 steps 10K times, so I wasn't sure if that meant choose 3 different variables in step 3.2. Here I'll. In this step, I'll look at the distribution of 3 different baseline binary covariates.I'm assuming any covariates without 1973 or 1982 in the variable name are measured at the baseline.

```
# for 3 baseline covariates: student_Gen, student_vote, student_money (assuming 1 for yes, 0 for no)
# Visualize the distribution by treatment/control (ggplot)

ypsps %>%
  gather(key = "variable", value = "value", student_Gen, student_vote, student_money) %>%
  ggplot(aes(x = value, fill = treatment_random)) +
  geom_bar(position = "dodge") +
  facet_wrap(~variable, scales = "free_x") +
  labs(title = "Distribution by Treatment/Control Condition", x = "Value", y = "Count") +
```

```
  scale_fill_brewer(palette = "Set1")
```



Interpretation: the count looks pretty even in the control vs treatment groups

```
# Simulate this 1,000 times (my computer wouldn't run 10K)
# only going to do for student_vote, not sure if you want me to do it for all 3 variables above

# simulate random assignment and calculate proportion of student_vote
simulate_assignment <- function(data) {
  data %>%
    mutate(treatment_random = sample(c("Treatment", "Control"),
                                     n(),
                                     replace = TRUE)) %>%
    group_by(treatment_random) %>%
    summarize(proportion_vote = mean(student_vote, na.rm = TRUE),
              .groups = 'drop') %>%
    pivot_wider(names_from = treatment_random, values_from = proportion_vote)
}

set.seed(123) # reproducibility

# Perform the simulation
results <- replicate(1000, simulate_assignment(ypsps), simplify = FALSE) %>%
  bind_rows() %>%
  mutate(simulation_id = row_number())

# make sure it looks correct
head(results)
```

```
## # A tibble: 6 x 3
##   Control Treatment simulation_id
##     <dbl>     <dbl>         <int>
## 1   0.725     0.720             1
## 2   0.727     0.719             2
## 3   0.725     0.720             3
## 4   0.723     0.722             4
## 5   0.746     0.698             5
## 6   0.706     0.739             6
```

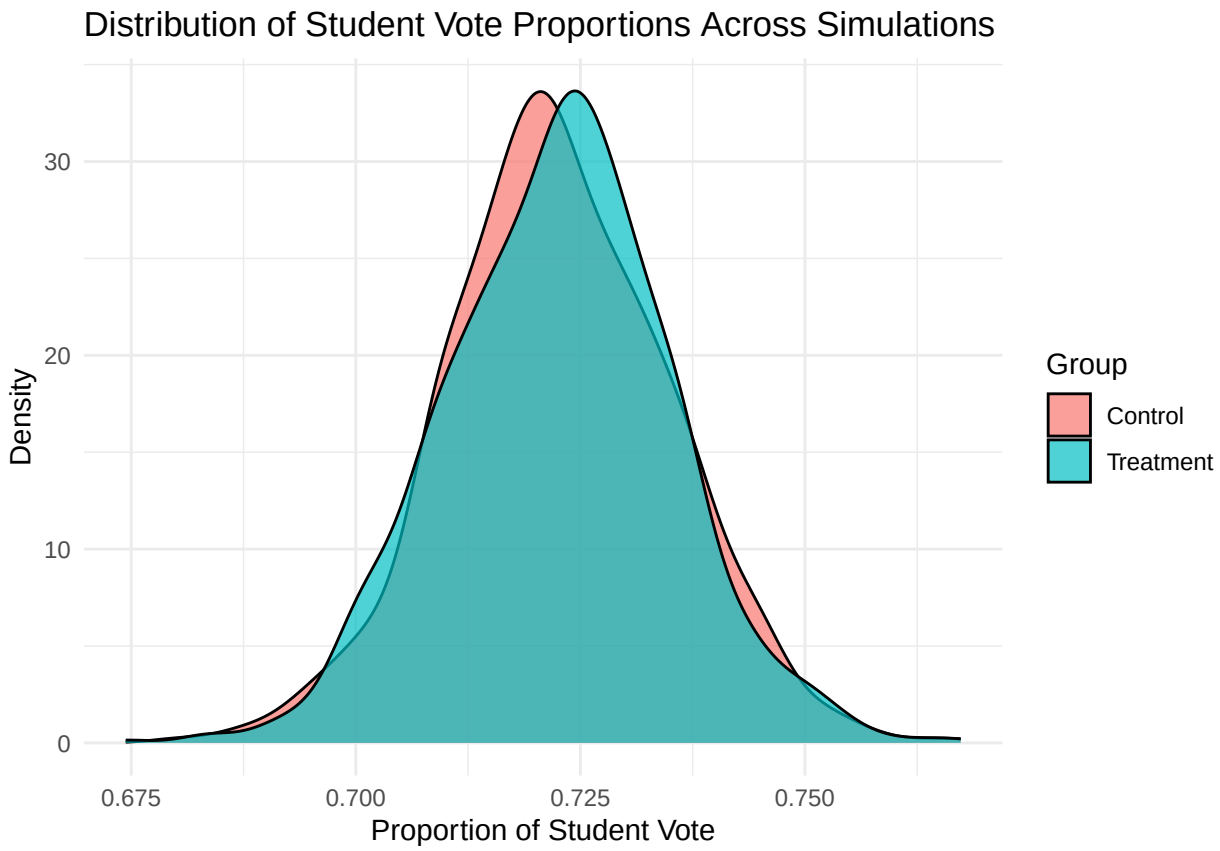
Interpretation: yes, generally looks correct. the control and treatment values are pretty close.

```r
# visualize in a graph

library(ggplot2)

results_long <- pivot_longer(results, -simulation_id, names_to = "group", values_to = "proportion_vote")

ggplot(results_long, aes(x = proportion_vote, fill = group)) +
  geom_density(alpha = 0.7) + # making it looks prettier
  labs(title = "Distribution of Student Vote Proportions Across Simulations",
       x = "Proportion of Student Vote",
       y = "Density",
       fill = "Group") +
  theme_minimal() # theme looks better
```



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?**

Your Answer: For the student_vote variable, the proportions are pretty similar, but not perfectly overlapping. Independence of treatment assignment and baseline covariates does not guarantee balance of treatment assignment and baseline covariates usually because of random variation or chance. Also when you add more covariates that you are simultaneously trying to balance, it becomes increasingly difficult to find balance.

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.

In reading through the variables and documentation, my guess is that the following 7 covariates best represent the "true" model predicting whether a student chooses to attend college: "student_Gen" "student_Race" "student_GPA" "student_SchOfficer" "student_FPlans" "student_EgoA" "student_WinArg"

```
# estimating a propensity score model to calculate the Average Treatment Effect on the Treated (ATT)
m.out <- matchit(college ~ student_Gen + student_Race + student_GPA +
                     student_SchOfficer + student_FPlans + student_EgoA +
                     student_WinArg,
                 data = ypsps,
                 method = "nearest", # Nearest neighbor matching
                 ratio = 1) # 1:1 matching
```

```
## Warning: Fewer control units than treated units; not all treated units will get
## a match.
```

```
# Extract matched data
matched_data <- match.data(m.out)

# Calculate ATT using the matched data
# Assume that the 'treatment' variable in 'matched_data' is named the same as in 'ypsps'
att <- with(matched_data, {
  treated_outcomes <- student_ppnscal[college == 1]
  control_outcomes <- student_ppnscal[college == 0 & !is.na(subclass)]

  mean(treated_outcomes) - mean(control_outcomes)
})

# Print the ATT result
print(att)
```

```
## [1] 1.45898
```

^^ I looked online about this warning message and it doesn't seem to be an issue, so hopefully it doesn't present problems further down

```
# Plot the balance for these covariates

# I was reading online and found that the 'cobalt" package can make this step a lot easier, so installi
#install.packages("cobalt")
library(cobalt)
```

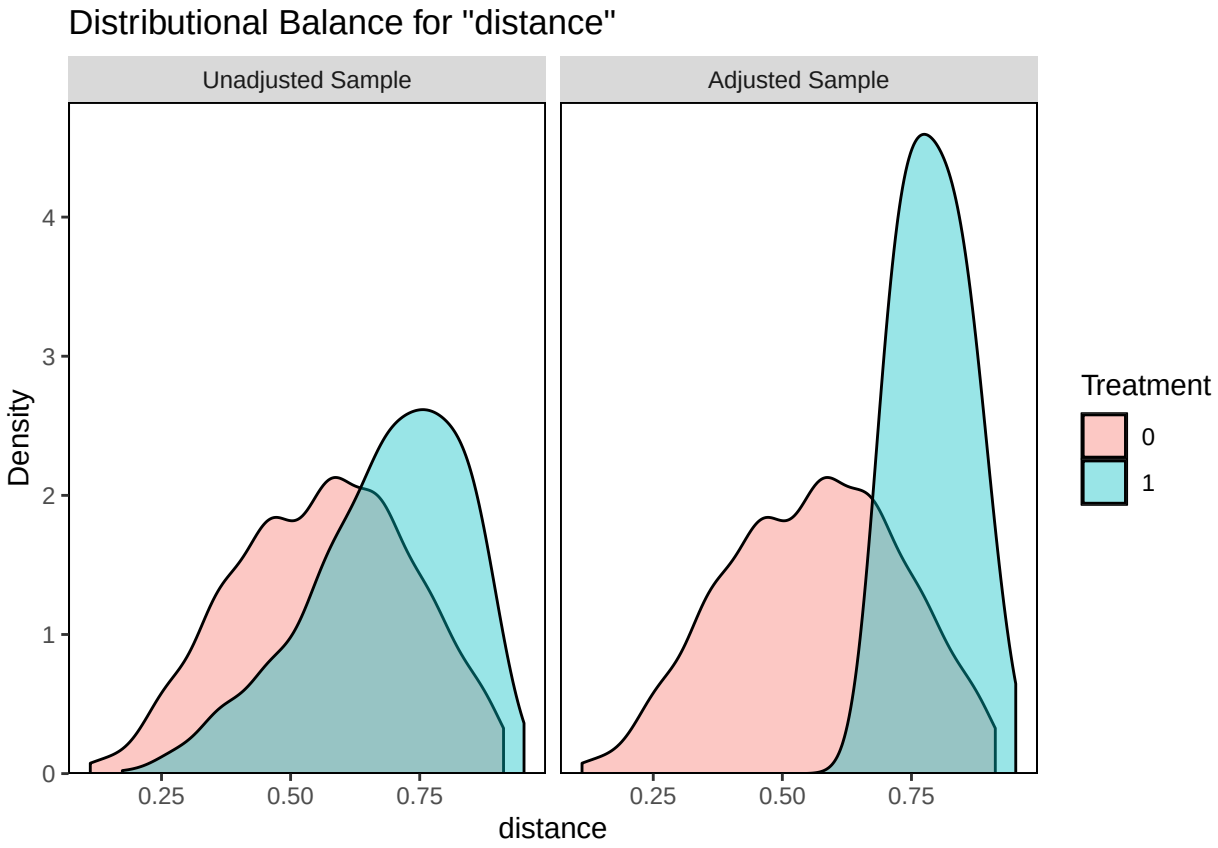
```
##  cobalt (Version 4.5.5, Build Date: 2024-04-02)
```

```
##
## Attaching package: 'cobalt'
```

```
## The following object is masked from 'package:MatchIt':
##
##     lalonde
```

```
bal.plot(m.out, which = "both")
```

```
## No `var.name` was provided. Displaying balance for distance.
```



Finding: covariates do not look very balanced. it's interesting that in the adjusted sample they're even more unbalanced

```
# Report the overall balance and the proportion of covariates that meet the balance threshold

# balance statistics
balance <- bal.tab(m.out, display = "all")
print(balance)
```

```
## Balance Measures
##                         Type Diff.Adj
```

```
## distance           Distance   1.5401
## student_Gen          Binary   0.2217
## student_Race        Contin.  -0.0436
## student_GPA         Contin.  -0.9788
## student_SchOfficer  Contin.  -0.6021
## student_FPlans      Contin.  -0.5211
## student_EgoA        Contin.   0.6416
## student_WinArg      Contin.  -0.6389
##
## Sample sizes
##           Control Treated
## All           451     803
## Matched       451     451
## Unmatched       0     352
```

```
# threshold of SMD  .1
balanced_covariates <- sum(balance$sumstats[, "SMD"] <= .1)
print(paste("Number of covariates that meet the balance threshold of SMD  .1:", balanced_covariates))
```

```
## [1] "Number of covariates that meet the balance threshold of SMD  .1: 0"
```

This output also shows that the covariates are not very balanced. 0 meet the balance threshold. That seems unlikely, so I wonder if I'm doing something wrong in the code above. Or maybe the 7 covariates I chose are just particularly unbalanced.

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).

```
library(dplyr)

# Filter out the post-treatment columns by detecting "1973" or "1982" in their names
pre_treatment_ypsps <- ypsps %>%
  select(-matches("1973|1982"))
```

```r
# View the column names to confirm
colnames(pre_treatment_ypsps)
```

```
##   [1] "interviewid"                "college"
##   [3] "student_vote"               "student_meeting"
##   [5] "student_other"              "student_button"
##   [7] "student_money"              "student_communicate"
##   [9] "student_demonstrate"        "student_community"
##  [11] "student_ppnscal"            "student_PubAff"
##  [13] "student_Newspaper"          "student_Radio"
##  [15] "student_TV"                 "student_Magazine"
##  [17] "student_FamTalk"            "student_FrTalk"
##  [19] "student_AdultTalk"          "student_PID"
##  [21] "student_SPID"               "student_GovtOpinion"
##  [23] "student_GovtCrook"          "student_GovtWaste"
##  [25] "student_TrGovt"             "student_GovtSmart"
##  [27] "student_Govt4All"           "student_Cynic"
##  [29] "student_LifeWish"           "student_GLuck"
##  [31] "student_FPlans"             "student_EgoA"
##  [33] "student_WinArg"             "student_StrOpinion"
##  [35] "student_MChange"            "student_EgoB"
##  [37] "student_TrOthers"           "student_OthHelp"
##  [39] "student_OthFair"            "student_Trust"
##  [41] "student_Senate"             "student_Tito"
##  [43] "student_Court"              "student_Govern"
##  [45] "student_CCamp"              "student_FDR"
##  [47] "student_Knowledge"          "student_NextSch"
##  [49] "student_GPA"                "student_SchOfficer"
##  [51] "student_SchPublish"         "student_Hobby"
##  [53] "student_SchClub"            "student_OccClub"
##  [55] "student_NeighClub"          "student_RelClub"
##  [57] "student_YouthOrg"           "student_MiscClub"
##  [59] "student_ClubLev"            "student_Phone"
##  [61] "student_Gen"                "student_Race"
##  [63] "parent_Newspaper"           "parent_Radio"
##  [65] "parent_TV"                  "parent_Magazine"
##  [67] "parent_LifeWish"            "parent_GLuck"
##  [69] "parent_FPlans"              "parent_WinArg"
##  [71] "parent_StrOpinion"          "parent_MChange"
##  [73] "parent_TrOthers"            "parent_OthHelp"
##  [75] "parent_OthFair"             "parent_PID"
##  [77] "parent_SPID"                "parent_Vote"
##  [79] "parent_Persuade"            "parent_Rally"
##  [81] "parent_OthAct"              "parent_PolClub"
##  [83] "parent_Button"              "parent_Money"
##  [85] "parent_Participate1"        "parent_Participate2"
##  [87] "parent_ActFrq"              "parent_GovtOpinion"
##  [89] "parent_GovtCrook"           "parent_GovtWaste"
##  [91] "parent_TrGovt"              "parent_GovtSmart"
##  [93] "parent_Govt4All"            "parent_Employ"
##  [95] "parent_EducHH"              "parent_EducW"
##  [97] "parent_ChurchOrg"           "parent_FratOrg"
##  [99] "parent_ProOrg"              "parent_CivicOrg"
```

```
## [101] "parent_CLOrg"                "parent_NeighClub"
## [103] "parent_SportClub"            "parent_InfClub"
## [105] "parent_FarmGr"               "parent_WomenClub"
## [107] "parent_MiscClub"             "parent_ClubLev"
## [109] "parent_FInc"                 "parent_HHInc"
## [111] "parent_OwnHome"              "parent_Senate"
## [113] "parent_Tito"                 "parent_Court"
## [115] "parent_Govern"               "parent_CCamp"
## [117] "parent_FDR"                  "parent_Knowledge"
## [119] "parent_Gen"                  "parent_Race"
## [121] "parent_GPHighSchoolPlacebo" "parent_HHCollegePlacebo"
## [123] "treatment_random"
```

Looks good. No variables with 1972 or 1983 in the variable name

```
# Randomly select features (only doing 5 like suggested in lab)

# Simulate random selection of features 10k+ times

# Fit p-score models and save ATTs, proportion of balanced covariates, and mean percent balance improve

library(MatchIt)
library(cobalt)

# Define the simulation function
simulate_ATT_and_balance <- function(data, covariates) {
  selected_covariates <- sample(covariates, 5)
  # formula for the propensity score model indicating the treatment variable
  formula <- as.formula(paste("college ~", paste(selected_covariates, collapse = " + ")))
  # use matchit to estimate propensity scores. using nearest neighbor algorithm
  m.out <- matchit(formula, data = data, method = "nearest", replace = TRUE)

  # Extract matched data
  matched_data <- match.data(m.out)

  # Calculate ATT by comparing outcomes between matched treated and control units
  treated_outcomes <- matched_data$student_ppnscal[matched_data$college == 1]
  control_outcomes <- matched_data$student_ppnscal[matched_data$college == 0]
  att <- mean(treated_outcomes) - mean(control_outcomes)

  # now need to calculate balance statistics using pre defined formulate
  balance <- bal.tab(m.out, display = "none")

  # calc covariates with SMD   .1
  prop_smd_lt_01 <- mean(balance$sumstats[, "SMD"] <= .1)

  # find mean percent improvement in SMD
  mean_imp_smd <- mean((balance$sumstats[, "SMD.un"] - balance$sumstats[, "SMD"]) / balance$sumstats[,

  return(list(att = att, prop_smd_lt_01 = prop_smd_lt_01, mean_imp_smd = mean_imp_smd))
}

# when i first ran this, I got errors so i need to define the covariates to be potentially excluded in

# excluding treatment, outcome, and id variables
```

```r
potential_covariates <- setdiff(colnames(pre_treatment_ypsps), c("interviewid", "college", "student_ppns

# Exclude covariates with missing or non-finite values
covariates_to_exclude <- c("parent_GPHighSchoolPlacebo", "parent_HHCollegePlacebo")

# Update the list of potential covariates
potential_covariates <- setdiff(
  potential_covariates,
  covariates_to_exclude
)

# preparing to run your simulations. initialize a container for the results. again have to do 1000 beca
results <- vector("list", 1000)

# Set seed
set.seed(123)

# Run simulations (my computer took superrr long to do 10K and even 5K, so just doing 1K here for learn
for (i in 1:1000) {
  results[[i]] <- simulate_ATT_and_balance(pre_treatment_ypsps, potential_covariates)
}

# Convert to a data frame
results_df <- do.call(rbind.data.frame, results)

# Rename as needed
colnames(results_df) <- c("ATT", "Proportion_SMD_lt_01", "Mean_Improvement_SMD")

head(results_df)
```

```
##        ATT Proportion_SMD_lt_01 Mean_Improvement_SMD
## 1 1.366268                  NaN                  NaN
## 2 1.305275                  NaN                  NaN
## 3 1.306483                  NaN                  NaN
## 4 1.336753                  NaN                  NaN
## 5 1.282546                  NaN                  NaN
## 6 1.297243                  NaN                  NaN
```

I spent a long time trying to figure out why I'm getting NaNs for the last 2 columns, but can't seem to solve the issue… - the first times I was getting an error, it was due to having covariates with missing or non-finite values in teh dataset, but i corrected that (ln 286) - sometimes when I ran it, i got an error about fewer control units than treated units? online it said that shouldn't be an issue, but maybe it is - other thought, unlikely, but maybe it's related to only being able to run 1K simulations

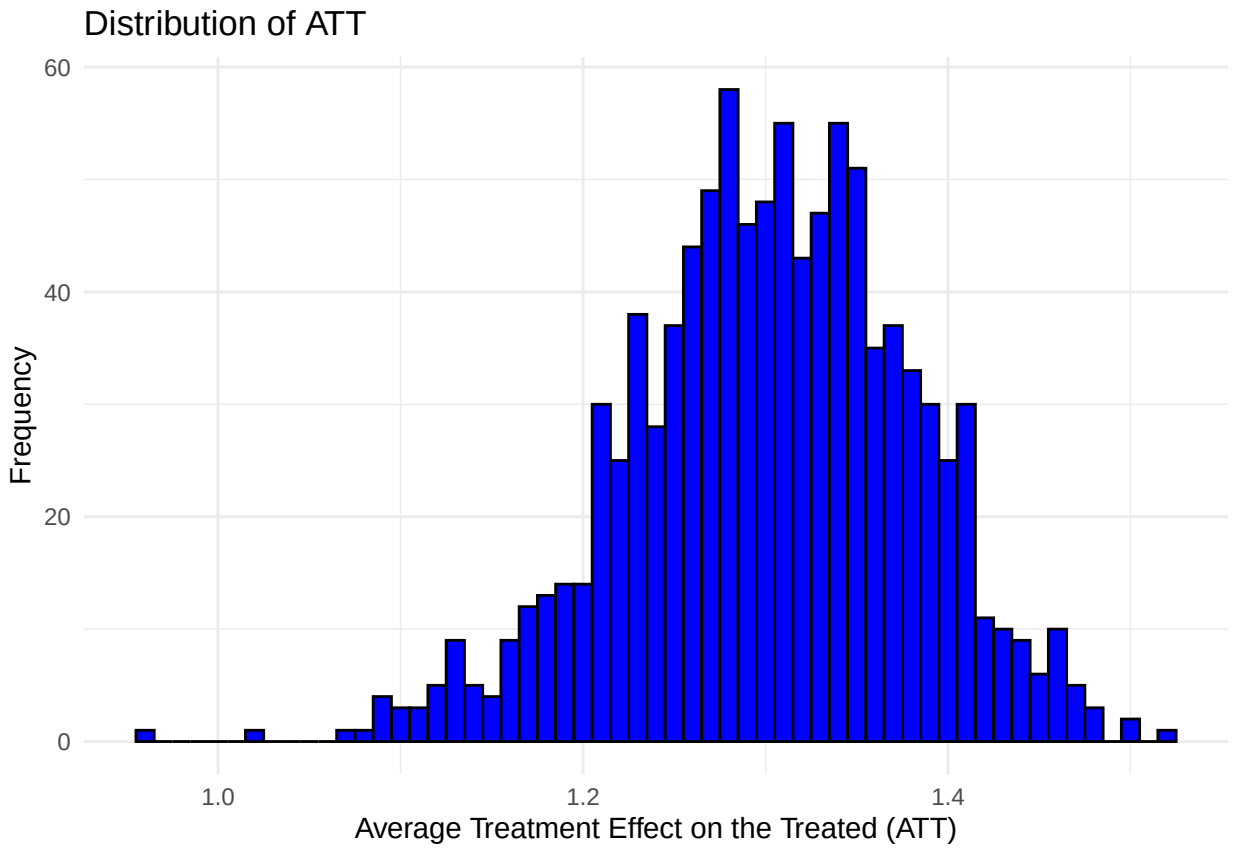
I'm just going to move forward for the sake of time – my results for the last part of this question won't make sense, but the logic behind my code should still make sense

Making this histogram for the discussion question below.

```r
library(ggplot2)

# Create a histogram of the ATT variable
ggplot(results_df, aes(x = ATT)) +
  geom_histogram(binwidth = .01, fill = "blue", color = "black") +
  theme_minimal() +
  labs(title = "Distribution of ATT",
```

```
       x = "Average Treatment Effect on the Treated (ATT)",
       y = "Frequency")
```




```
library(ggplot2)

# Plotting ATTs against the proportion of balanced covariates
ggplot(results_df, aes(x = Proportion_SMD_lt_01, y = ATT)) +
  geom_point(alpha = 0.1) +  # Alpha for transparency to mitigate overplotting
  theme_minimal() +
  labs(title = "ATT vs. Proportion of Balanced Covariates",
       x = "Proportion of Covariates with SMD  0.1",
       y = "Average Treatment Effect on the Treated (ATT)")
```

```
## Warning: Removed 1000 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Proportion of Covariates with SMD  0.1' in
## 'mbcsToSbcs': dot substituted for <e2>
```

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Proportion of Covariates with SMD  0.1' in
## 'mbcsToSbcs': dot substituted for <89>
```

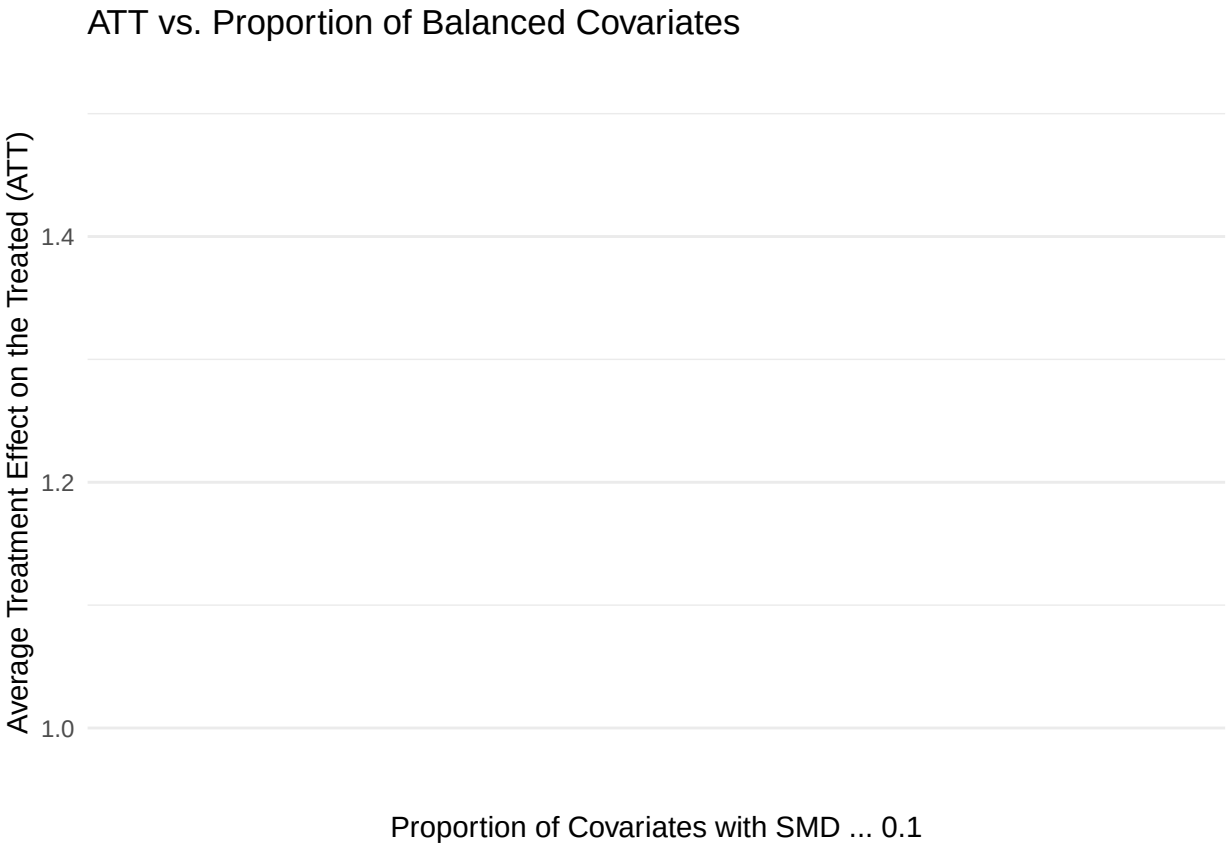
```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Proportion of Covariates with SMD  0.1' in
## 'mbcsToSbcs': dot substituted for <a4>
```

```
## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
```

```
## conversion failure on 'Proportion of Covariates with SMD  0.1' in
## 'mbcsToSbcs': dot substituted for <e2>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Proportion of Covariates with SMD  0.1' in
## 'mbcsToSbcs': dot substituted for <89>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Proportion of Covariates with SMD  0.1' in
## 'mbcsToSbcs': dot substituted for <a4>
```



Blank because I have NaNs above

Questions

Note: I'm going to answer these more theoretically (to show my understanding of the concepts) rather than specifically about my results becuase I was having trouble with the code in this section

1. **How many simulations resulted in models with a higher proportion of balanced covariates? Do you have any concerns about this?** Your Answer: None of my simulations results in models with balanced covariates. Yes, I'm obviously concerned about this. Even if my code did produce more plausible results, there is still a good chance that many of the simulations would not have resulted in models with a higher proportion of balanced covariates (eg kind of like we talked about in lab). This is overall worrisome and a limitation/criticism of PSM models. When selecting covariates, it is difficult to find combinations that are simultaneously balanced - and even if there are, there's definitely room for researcher subjectivity in choosing the covariates.
2. **Analyze the distribution of the ATTs. Do you have any concerns about this distribution?** Your Answer: See histogram above. I'm slightly concerned because it's not a very narrow spread, indicating a fair about of variability in the estimates. I wonder if this would be tighter if I could have

done more simulations. It's pretty symmetric, but there is a longer tail on the left side, which is also a bit concerning. Last, there are some outliers, even a value < 1, which is cause for concern.

3. **Do your 10 randomly chosen covariate balance plots produce similar numbers on the same covariates? Is it a concern if they do not?** Your Answer: No they don't. It is a concern because it's a sign that the model is sensitivity to which covariates are selected and overall highlights the inconsistency of the model results. If this is the case in a research project, the researchers should be very careful any "causal inference" phrasing used.

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:

I'm going to use greedy algorithms because I was having issues with computational intensity above to run the models and my understanding is greedy is less computational intensive, so i wonder if it will make a difference.

```
library(MatchIt)
library(cobalt)
library(dplyr)

# makin a  simulate_greedy_matching function
simulate_greedy_matching <- function(data) {
  # Randomly selecting 5 'student_' features. i wanted to see what would happen if i just used student,
  selected_covariates <- sample(grep("^student_", names(data), value = TRUE), 5)
  # Create a formula
  formula <- as.formula(paste("college ~", paste(selected_covariates, collapse = " + ")))
  # Estimate propensity scores and perform nearest neighbor matching
  m.out <- matchit(formula, data = data, method = "nearest", replace = FALSE)

  # important, proceed only if matching was successful
  if (!is.null(m.out)) {

    # Calculate balance statistics with the correct parameter to suppress output. I first had FALSE (ra
    balance <- bal.tab(m.out, display = "none")

    # get the matched dataset
    matched_data <- match.data(m.out)

    # Calculate ATT only for matched units by using the subset that was matched
    att <- with(matched_data, mean(student_ppnscal[college == 1], na.rm = TRUE) - mean(student_ppnscal[

    # covariates with SMD   .1
    prop_balanced <- mean(balance$sumstats[, "SMD"] <= .1, na.rm = TRUE)

    # Mean percent improvement in SMD. I was also having issues with this line of code, so i had to loo
    mean_imp_balance <- mean((balance$sumstats[, "SMD.un"] - balance$sumstats[, "SMD"]) / balance$sumsta

    return(c(ATT = att, Proportion_Balanced = prop_balanced, Mean_Improvement_Balance = mean_imp_balanc
  }
```

```
}

# see screenshot below for why i'm doing 10
test_results <- replicate(10, simulate_greedy_matching(pre_treatment_ypsps), simplify = "data.frame")
```

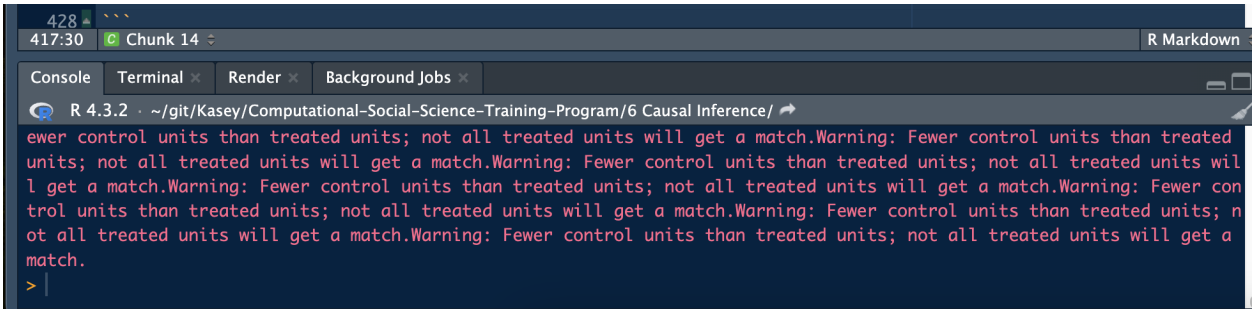
```
## Warning: Fewer control units than treated units; not all treated units will get a match.
## Fewer control units than treated units; not all treated units will get a match.
## Fewer control units than treated units; not all treated units will get a match.
## Fewer control units than treated units; not all treated units will get a match.
## Fewer control units than treated units; not all treated units will get a match.
## Fewer control units than treated units; not all treated units will get a match.
## Fewer control units than treated units; not all treated units will get a match.
## Fewer control units than treated units; not all treated units will get a match.
## Fewer control units than treated units; not all treated units will get a match.
## Fewer control units than treated units; not all treated units will get a match.
```

```
# Convert the results to a data frame
test_results_df <- as.data.frame(t(test_results))
names(test_results_df) <- c("ATT", "Proportion_Balanced", "Mean_Improvement_Balance")

head(test_results_df)
```

```
##        ATT Proportion_Balanced Mean_Improvement_Balance
## 1 1.450111                NaN                      NaN
## 2 1.534368                NaN                      NaN
## 3 1.492239                NaN                      NaN
## 4 1.793792                NaN                      NaN
## 5 1.949002                NaN                      NaN
## 6 1.733925                NaN                      NaN
```

whenever I increase the simulations above 10, I get the repeated message below as it's running. in order to not make the knitted PDF super long, I'm just going to do 10 simulations. I'm not sure where the issue stems from.



I'm not going to run code for the ATT v. proportion plots or the 10 random covariate balance plots since I'm still having the NaN issue and you can see my code in the section above. Again, I'll answer the questions below more theoretically to show my understading rather than specifically in reference to my results

Questions

1. **Does your alternative matching method have more runs with higher proportions of balanced covariates?** Your Answer: It does not, but assuming both the propensity matching algorithm and greedy algorithm ran correctly for me, I would expect the greedy algorithm to have fewer runs with high proportions of balanced covariates.

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.** Your Answer: I would assume that the greedy algorithm would do worse. Although I missed lab because of a trip, my understanding from the material is that greedy algorithms, such as nearest neighbor, are relatively fast (but also simple) and they just match based on shortest distance at a particular steps without taking into account future steps. Matching algorithms that consider the problem more holistically often do better on balance.

Discussion Questions

1. **Why might it be a good idea to do matching even if we have a randomized or as-if-random design?** Your Answer:There's a few reasons. One is robustness checks – if you get the same (or similar) with matching and your randomized design, it gives readers more confidence in your results. Another reason is to address imbalances in the treatment and control group in a randomized experiment. Especially with smaller sample sizes, imbalances can occur in RCTs and matching can help address this. Last, for quasi-experimental designs (as-if-random designs), where the researcher does not control the assignment to treatment, matching can help to mimic a randomized experiment by creating comparable treatment and control groups. This is really useful in situations where treatment assignment may not be exogenous.

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?** Your Answer: Yes of course. ML models are more flexible than logistic regression in terms of functional form. For example, with high-dimensional data, ML can better capture complex interactions and non-linear relationships between covariates - without explicitly specifying interaction terms, as would be necessary in logistic regression. However, there's obviously potential tradeoffs with ML, such as computational intensity, risk of overfitting, and reduced interpretability.