Project 6: Randomization and Matching

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.

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

```
## Warning: package 'ggplot2' was built under R version 4.3.1

## Warning: package 'tidyr' was built under R version 4.3.1

## Warning: package 'readr' was built under R version 4.3.1

## Warning: package 'dplyr' was built under R version 4.3.1

## Warning: package 'lubridate' was built under R version 4.3.1

## -- 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   1.5.0
## 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 error
```

```r
library(MatchIt)
```

```
## Warning: package 'MatchIt' was built under R version 4.3.1
```

```r
# Load ypsps data
ypsps <- read_csv('/Users/marisatsai/Downloads/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.
```

```r
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>, ...
```

```r
#all variables with 1975 in them are "post-treatment"
names(ypsps)
```

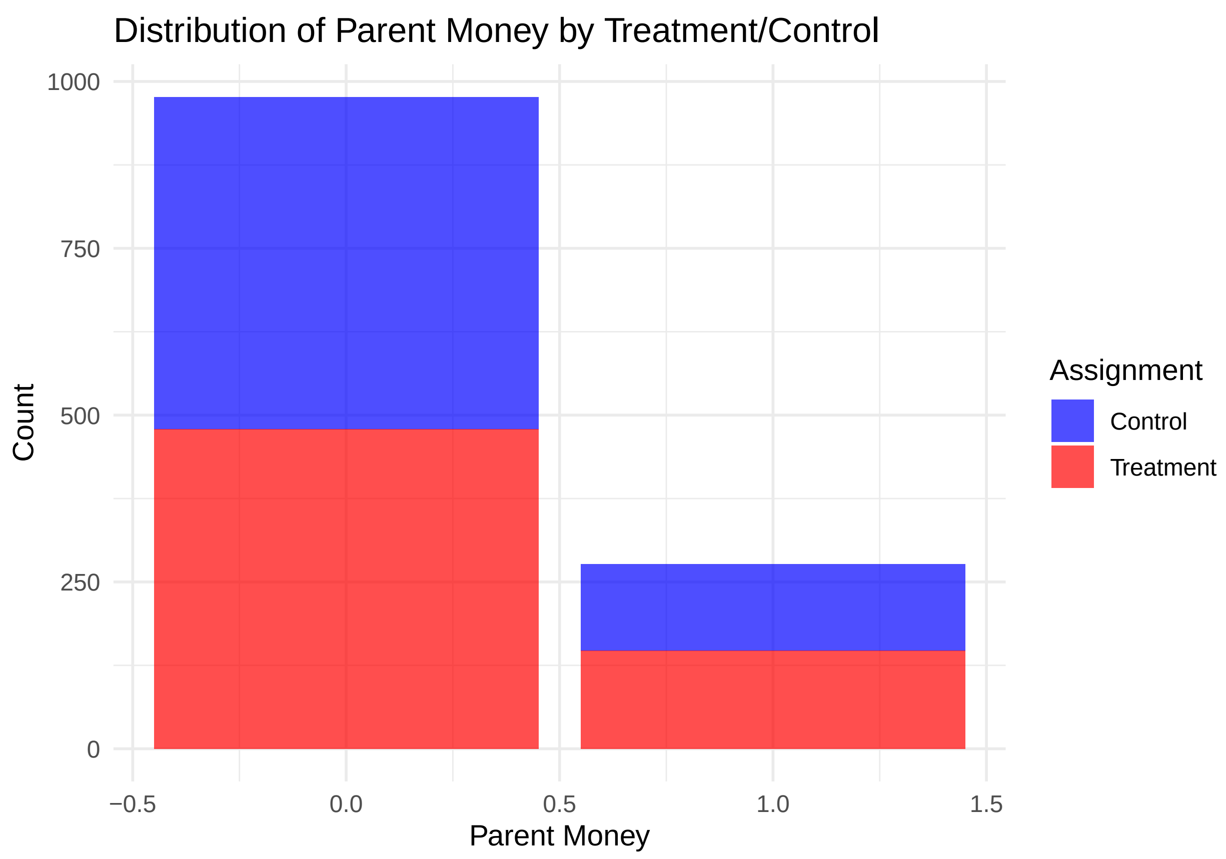
```
##   [1] "interviewid"                   "college"
##   [3] "student_vote"                  "student_meeting"
##   [5] "student_other"                 "student_button"
##   [7] "student_money"                 "student_communicate"
##   [9] "student_demonstrate"           "student_community"
##  [11] "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"
```

Randomization

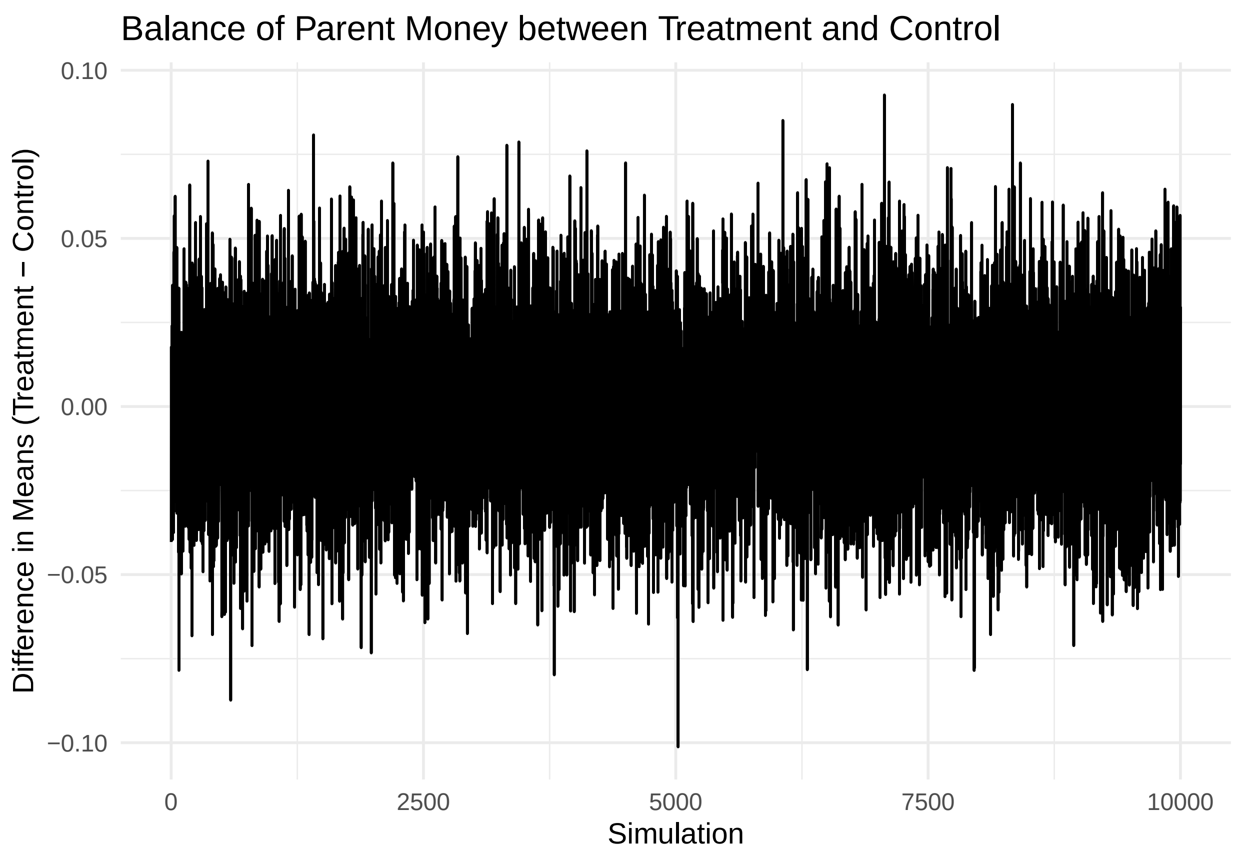
Matching is usually used in observational studies to to approximate random assignment to treatment. But could it be useful even in randomized studies? To explore the question do the following:

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.

```r
# Generate a vector that randomly assigns each unit to treatment/control
n <- nrow(ypsps)
assignment <- sample(c(0, 1), size = n, replace = TRUE)

data <- cbind(ypsps, assignment)

# Choose a baseline covariate (use dplyr for this)
#choose parent_Money (parent donation to parties or campaigns)

# Visualize the distribution by treatment/control (ggplot)

library(ggplot2)

# Convert 'parent_money' to a factor since it is categorical
#data$parent_Money <- factor(data$parent_Money)

ggplot(data, aes(x = parent_Money, fill = factor(assignment))) +
  geom_bar(position = "stack", alpha = 0.7) +
  labs(x = "Parent Money", y = "Count", fill = "Assignment", title = "Distribution of Parent Money by T
  scale_fill_manual(values = c("blue", "red"), labels = c("Control", "Treatment")) +
  theme_minimal()
```



```r
#########
# Simulate this 10,000 times (monte carlo simulation - see R Refresher for a hint)
#Simulate the first 3 steps 10,000 times and visualize the distribution of treatment/control balance ac

# creating list to store the results of each simulation
ypsps_simulation <- list()

#10000 simulations
for (i in 1:10000) {
 # Generate a vector that randomly assigns each unit to treatment/control for each simulation
  assignment <- sample(c(0, 1), size = nrow(data), replace = TRUE)

  # adding the assignment vector to original dataset
  data$assignment <- assignment

  # Store the datasets
  ypsps_simulation[[i]] <- data
}


# Initialize an empty data frame to store summary statistics
summary_data <- data.frame(Simulation = integer(), Treatment_Mean = numeric(), Control_Mean = numeric()

# Calculate summary statistics for each simulation
for (i in 1:10000) {
  # Subset the data for this simulation
  sim_data <- ypsps_simulation[[i]]

  # Calculate means for treatment and control groups
  treatment_mean <- mean(sim_data$parent_Money[sim_data$assignment == 1])
  control_mean <- mean(sim_data$parent_Money[sim_data$assignment == 0])

  # Append summary statistics to the data frame
  summary_data <- rbind(summary_data, data.frame(Simulation = i, Treatment_Mean = treatment_mean, Contr
}

# Create a ggplot object to visualize the balance of parent_money between treatment and control
ggplot(summary_data, aes(x = Simulation, y = Treatment_Mean - Control_Mean)) +
  geom_line() +
  labs(x = "Simulation", y = "Difference in Means (Treatment - Control)", title = "Balance of Parent Mo
  theme_minimal()
```



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 see that independence of treatment assignment and covariates do not guarantee balance of treatment assignment and baseline covariates– if it did, then the difference in means of the parent_Money variable should be zero. Instead we see many simuluated datasets with differences between -0.05 and 0.05.

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.

```
# ATT using matchit for exact
# --------------------------------------------------
match_exact_att <- matchit(formula = college ~ student_LifeWish + student_Govern+ student_GPA +student_
                           method = "exact",                    # method
                           estimand = "ATT")                    # estimand

# summary
summary(match_exact_att, un = FALSE)
```

```
##
## Call:
## matchit(formula = college ~ student_LifeWish + student_Govern +
##     student_GPA + student_NextSch + student_Race + parent_EducHH +
##     parent_LifeWish + parent_Vote + parent_FPlans, data = ypsps,
##     method = "exact", estimand = "ATT")
##
## Summary of Balance for Matched Data:
##                  Means Treated Means Control Std. Mean Diff. Var. Ratio
## student_LifeWish        2.1876        2.1876               0     0.9916
## student_Govern          0.9857        0.9857               0          .
## student_GPA             2.3705        2.3705               0     0.9916
## student_NextSch         0.9596        0.9596              -0          .
## student_Race            1.0190        1.0190               0     0.9916
## parent_EducHH           2.9074        2.9074               0     0.9916
## parent_LifeWish         2.1045        2.1045               0     0.9916
## parent_Vote             0.9311        0.9311               0          .
## parent_FPlans           2.4109        2.4109               0     0.9916
##                  eCDF Mean eCDF Max Std. Pair Dist.
## student_LifeWish         0        0               0
## student_Govern           0        0               0
## student_GPA              0        0               0
## student_NextSch          0        0               0
## student_Race             0        0               0
## parent_EducHH            0        0               0
## parent_LifeWish          0        0               0
## parent_Vote              0        0               0
## parent_FPlans            0        0               0
##
## Sample Sizes:
##               Control Treated
## All            451.       803
## Matched (ESS)   93.17     421
## Matched        219.       421
## Unmatched      232.       382
## Discarded        0.         0
```

```
#
# estimate the ATT using linear regression
# ---------

# construct a matched dataset from the matchit object
match_exact_att_data <- match.data(match_exact_att)


# specify a linear model
lm_exact_att <- lm(student_ppnscal ~ college+ student_LifeWish + student_Govern+ student_GPA +student_Ne
                   data = match_exact_att_data, # data
                   weights = weights)           # weights

# view summary of results
lm_exact_att_summ <- summary(lm_exact_att)
lm_exact_att_summ
```

```
##
## Call:
## lm(formula = student_ppnscal ~ college + student_LifeWish + student_Govern +
##     student_GPA + student_NextSch + student_Race + parent_EducHH +
##     parent_LifeWish + parent_Vote + parent_FPlans, data = match_exact_att_data,
##     weights = weights)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9944 -1.0811 -0.3327  0.8070  6.5292
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.24563    0.94431   1.319  0.18762
## college            0.91231    0.14342   6.361 3.86e-10 ***
## student_LifeWish   0.04435    0.07158   0.620  0.53574
## student_Govern     0.02822    0.58400   0.048  0.96148
## student_GPA       -0.11140    0.13036  -0.855  0.39313
## student_NextSch   -0.20209    0.35825  -0.564  0.57289
## student_Race       0.21100    0.50972   0.414  0.67904
## parent_EducHH      0.18985    0.06097   3.114  0.00193 **
## parent_LifeWish   -0.06027    0.07143  -0.844  0.39914
## parent_Vote        0.35746    0.27919   1.280  0.20089
## parent_FPlans     -0.05701    0.08012  -0.712  0.47703
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.721 on 629 degrees of freedom
## Multiple R-squared:  0.07969,    Adjusted R-squared:  0.06506
## F-statistic: 5.446 on 10 and 629 DF,  p-value: 8.978e-08
```

```
#
# pull out ATT
```

```r
# ---------
ATT_exact <- lm_exact_att_summ$coefficients["college", "Estimate"]
ATT_exact
```

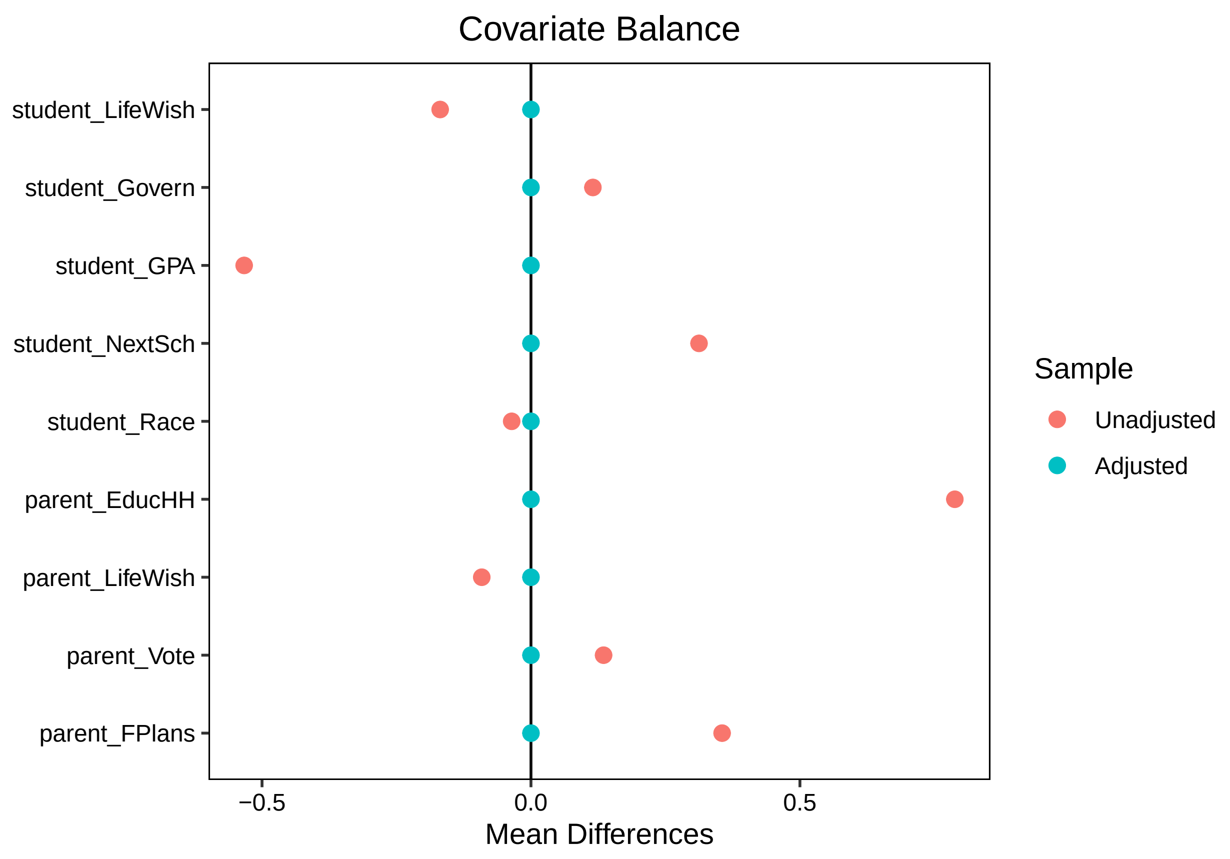
```
## [1] 0.912312
```

```r
#install.packages("cobalt")
library(cobalt)
```

```
## Warning: package 'cobalt' was built under R version 4.3.1

##  cobalt (Version 4.5.4, Build Date: 2024-02-26)

##
## Attaching package: 'cobalt'

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

```r
love.plot(match_exact_att)
```

```
## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the `stars` argument to distinguish between them and appropriately label the x-axis.
```



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).

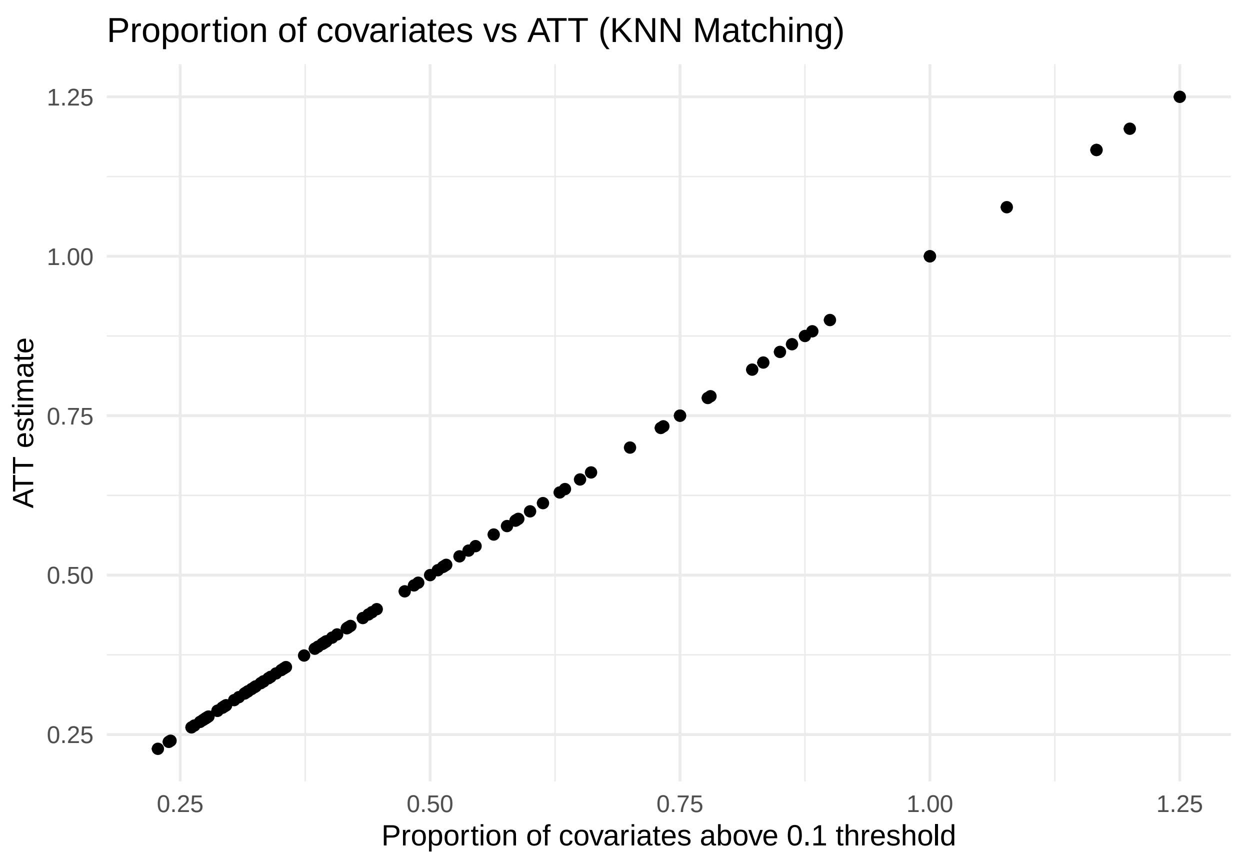
```
# Remove post-treatment covariates
# Data manipulation: rename post treatment covariates with "post_", create list of post_vars names, and
df_renamed <- ypsps %>% rename_with(~paste0("post_", .), contains("1973") | contains("1983"))

# Get list of post-treatment variable names
post_vars <- names(df_renamed) %>% keep(~str_starts(., "post_"))

# Create prevars_df excluding post-treatment variables and specific placebo variables
prevars_df <- df_renamed %>% select(-any_of(c(post_vars, "college", "interviewid", "treatment"))) %>% f

#Filter out rows with any NA values
# Get names of pre-treatment variables
pre_vars <- colnames(prevars_df)

#dropping NAs bc "Missing and non-finite values are not allowed in the covariates" for matchit
prevars_df_clean <- na.omit(prevars_df)
df<-ypsps
df[is.na(df)] <- 0
print(ypsps$college)
```

```
##    [1] 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 1 0 1 0
##   [38] 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 0 1 1 0 0 0 1 0 1
##   [75] 0 0 0 0 0 0 0 0 1 1 0 1 1 0 1 0 0 0 1 0 0 1 0 0 0 0 1 1 0 1 1 1 1 0 1 1 1
##  [112] 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0
##  [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 0 0 1 0 1 1 1 0 0 0 0 0 1
##  [186] 1 1 1 1 0 0 0 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0
##  [223] 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0 1 1 1 1 0 1
##  [260] 1 1 1 1 0 1 0 1 1 0 1 0 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0
##  [297] 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 1 0 1 1 0 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1
##  [334] 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1 1 1
##  [371] 0 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0
##  [408] 1 1 1 1 1 0 0 1 1 0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 0 0 1 0 1 0 1 0 1 1 1 1 1
##  [445] 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0 0 1 0 0
##  [482] 0 0 1 1 1 0 1 0 0 1 0 1 0 0 1 1 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0
##  [519] 1 0 1 1 1 0 1 1 1 0 1 0 0 0 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1 0 0 0 1 1 1 0 0
##  [556] 1 0 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0
##  [593] 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1
##  [630] 1 0 1 0 0 1 1 1 1 1 1 1 0 0 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1
##  [667] 1 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 1
##  [704] 1 1 0 0 1 1 1 1 0 1 1 1 0 1 0 0 1 1 1 0 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1
##  [741] 1 0 0 0 1 0 0 1 1 1 0 1 1 1 1 0 0 0 1 1 1 0 0 1 0 1 1 1 1 0 0 1 1 1 0 1 1
##  [778] 0 0 0 0 1 1 1 1 1 0 1 0 1 0 0 0 1 1 0 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1
##  [815] 0 1 1 1 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 1 1 1 1 0 1 0 0 1 1 0 1 1 1 0 1 1 1
##  [852] 0 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 0 1 0 0 0 1 0 0
##  [889] 1 0 0 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 0 0 1 1 1 0 1 1
##  [926] 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 0 1 0 1
##  [963] 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 1
## [1000] 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 0 1
## [1037] 1 0 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1 0 0 1 1
## [1074] 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
## [1111] 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 0 1
## [1148] 0 1 1 1 1 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 0 0 0
## [1185] 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 0
## [1222] 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1
```

```r
tabulate(ypsps$college)
```

```
## [1] 803
```

```r
# Randomly select features
result_matrix <- matrix(NA, nrow = 100, ncol = 2)
colnames(result_matrix) <- c("ATT", "proportion_true")

for (i in 1:100) {
  # Randomly select the number of covariates
  num_covariates <- sample(1:length(pre_vars), 1)

  # Randomly choose covariates
  random_covariates <- sample(pre_vars, num_covariates)

  # Select the random columns
  df_1 <- ypsps %>%
    select(interviewid, college, student_ppnscal, all_of(random_covariates))

  # Fit the propensity score model
  match_knn_att <- matchit(as.formula(paste("college ~", paste(random_covariates, collapse = "+"))),
                           data = df,
                           method = "nearest",
                           distance = "glm",
                           link = "logit",
                           discard = "control",
                           replace = TRUE,
                           ratio = 2)


  ######## Calculate ATT using KNN matching
 # ATT <- summary(match_knn_att)$estimates$ATT
  ATT <- summary(match_knn_att)$estimates$ATT

  # Calculate the proportion of covariates that meet the balance threshold
  att_summ <- summary(match_knn_att)
  st_diffs_true_index <- as.numeric(which(abs(att_summ$sum.matched[, "Std. Mean Diff."]) <= 0.1))
  proportion_true <- length(st_diffs_true_index) / length(random_covariates)

  # Store the results in the result matrix
  result_matrix[i, ] <- c(ATT, proportion_true)
}

# Plot ATT v. proportion
result_df <- as.data.frame(result_matrix)
subsample_df <- result_df[sample(nrow(result_df), 100), ]
ggplot(subsample_df, aes(proportion_true, ATT)) +
  geom_point() +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(title = "Proportion of covariates vs ATT (KNN Matching)",
       x = "Proportion of covariates above 0.1 threshold",
       y = "ATT estimate") +
  theme_minimal()
```



```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
# Simulate random selection of features 10k+ times
balance_plots <- list()

for (i in 1:10) {
  # Randomly select the number of covariates
  num_covariates <- sample(1:length(pre_vars), 1)

  # Randomly choose covariates
  random_covariates <- sample(pre_vars, num_covariates)

  # Select the random columns
  df <- ypsps %>%
    select(interviewid, college, student_ppnscal, all_of(random_covariates))
df_clean <- na.omit(df)

  # Fit the propensity score model using KNN matching
  match_knn_att <- matchit(as.formula(paste("college ~", paste(random_covariates, collapse = "+"))),
                          data = df_clean,
                          method = "nearest",
                          distance = "glm",
                          link = "logit",
                          discard = "control",
                          replace = FALSE,
                          ratio = 2)



  # Save matched data
  matched_data <- match.data(match_knn_att)

  # Create balance plots for matched covariates
  balance_plot <- bal.plot(match_knn_att)

  # Store balance plot in the list
  balance_plots[[i]] <- balance_plot
}
```

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

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

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

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

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

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

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

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

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

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

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

## No `var.name` was provided. Displaying balance for distance.
## Warning: Fewer control units than treated units; not all treated units will get
## a match.
```

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

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

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

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

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

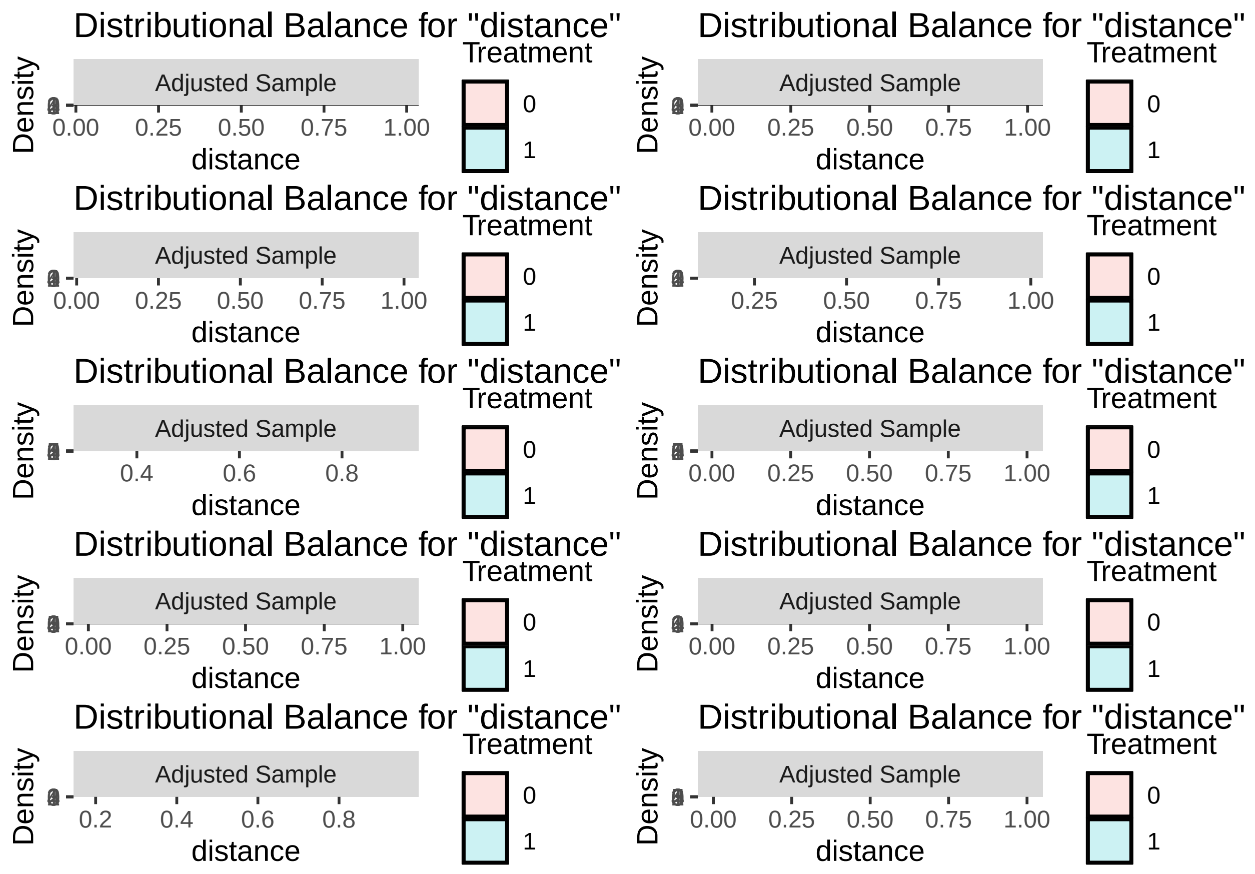
```
## Warning: Fewer control units than treated units; not all treated units will get
## a match.
```

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

```
# Arrange balance plots using gridExtra
grid.arrange(grobs = balance_plots, ncol = 2)
```

```
## Warning: No shared levels found between ‘names(values)‘ of the manual scale and the
## data’s colour values.
```

```
## Warning: No shared levels found between ‘names(values)‘ of the manual scale and the
## data’s colour values.
## No shared levels found between ‘names(values)‘ of the manual scale and the
## data’s colour values.
## No shared levels found between ‘names(values)‘ of the manual scale and the
## data’s colour values.
## No shared levels found between ‘names(values)‘ of the manual scale and the
## data’s colour values.
## No shared levels found between ‘names(values)‘ of the manual scale and the
## data’s colour values.
## No shared levels found between ‘names(values)‘ of the manual scale and the
## data’s colour values.
## No shared levels found between ‘names(values)‘ of the manual scale and the
## data’s colour values.
## No shared levels found between ‘names(values)‘ of the manual scale and the
## data’s colour values.
## No shared levels found between ‘names(values)‘ of the manual scale and the
## data’s colour values.
```



## Questions

1. **How many simulations resulted in models with a higher proportion of balanced covariates? Do you have any concerns about this?** Not sure how we would define "higher proportion of".

2. **Analyze the distribution of the ATTs. Do you have any concerns about this distribution?** As ATT increased, the proportion of balanced covariates also increased. I'm not sure why this would be the case.

3. **Do your 10 randomly chosen covariate balance plots produce similar numbers on the same covariates? Is it a concern if they do not?**

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:

```
# libraries
xfun::pkg_attach2(c("tidyverse", # load all tidyverse packages
                    "here",      # set file path
                    "MatchIt",   # for matching
```

```
                     "optmatch",  # for matching
                     "cobalt"))   # for matching assessment
```

```
## here() starts at /Users/marisatsai/Downloads
```

```
#install.packages("updog")
library(updog)

# fit a logit model
# --------------------------------------------------
model_ps <-                 # save logit model as an object
  glm(college ~ student_LifeWish + student_Govern+ student_GPA +student_NextSch + student_Race + parent_
      family = binomial(),  # specifying binomial calls a logit model
      data = ypsps)            # specify data for regression

# print summary
summary(model_ps)
```

```
##
## Call:
## glm(formula = college ~ student_LifeWish + student_Govern + student_GPA +
##     student_NextSch + student_Race + parent_EducHH + parent_LifeWish +
##     parent_Vote + parent_FPlans + parent_GovtOpinion, family = binomial(),
##     data = ypsps)
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -2.895333   0.639673  -4.526 6.00e-06 ***
## student_LifeWish   -0.007527   0.074261  -0.101   0.9193
## student_Govern      0.709826   0.226153   3.139   0.0017 **
## student_GPA        -0.528255   0.110248  -4.792 1.66e-06 ***
## student_NextSch     2.058158   0.213816   9.626  < 2e-16 ***
## student_Race        0.328542   0.227751   1.443   0.1491
## parent_EducHH       0.499098   0.056620   8.815  < 2e-16 ***
## parent_LifeWish     0.018136   0.075089   0.242   0.8091
## parent_Vote         0.480083   0.190369   2.522   0.0117 *
## parent_FPlans       0.194762   0.077365   2.517   0.0118 *
## parent_GovtOpinion -0.098524   0.102526  -0.961   0.3366
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1638.3  on 1253  degrees of freedom
## Residual deviance: 1237.2  on 1243  degrees of freedom
## AIC: 1259.2
##
## Number of Fisher Scoring iterations: 4
```

```
# predict
# ---------
df <-                                      # save over df dataframe object
```

```
  ypsps %>%                                       # pass data
  mutate(prop_score = predict(model_ps)) # create a new variable that predicts propensity score based o

df_a0 <- df %>% filter(college == 0) # save anything under control as a dataframe
df_a1 <- df %>% filter(college==1) # save anything under treatment as a dataframe
df_a0_small <- df_a0[1:10,]     # further subsetting
df_a1_small <- df_a1[1:5,]      # further subsetting

# calculate distances based on propensity scores
# ---------
dist.prop.score <- function(x,y) {
  abs(x-y)  # distance based on absolute value
}

# apply function

# function to calculate distances
# ---------
calculate.dist <- function(x, y, dist.method, xnames = df_a1_small$ID, ynames = df_a0_small$ID) {
  dists <- apply(y, 1, function(j) {apply(x, 1, function(i) {dist.method(i,j)})})
  rownames(dists) <- xnames
  colnames(dists) <- ynames
  return(dists)
}
# ---------
dists_ps <- calculate.dist(as.matrix(df_a1_small[, "prop_score"]), # x
                           as.matrix(df_a0_small[, "prop_score"]), # y
                           dist.prop.score)                        # method
# view
dists_ps
```

```
##          [,1]     [,2]      [,3]     [,4]      [,5]     [,6]     [,7]     [,8]
## [1,] 1.006133 1.340580 0.7792304 1.682905 0.7151338 2.269373 2.692512 2.310953
## [2,] 1.362562 1.697008 1.1356587 2.039334 1.0715621 2.625801 3.048941 2.667381
## [3,] 1.138644 1.473091 0.9117409 1.815416 0.8476444 2.401883 2.825023 2.443463
## [4,] 1.413793 1.748240 1.1868901 2.090565 1.1227935 2.677032 3.100172 2.718613
## [5,] 1.395657 1.730103 1.1687538 2.072429 1.1046572 2.658896 3.082036 2.700476
##          [,9]    [,10]
## [1,] 1.731467 1.566663
## [2,] 2.087895 1.923091
## [3,] 1.863978 1.699174
## [4,] 2.139127 1.974323
## [5,] 2.120990 1.956186
```

```
# use greedy matching - subset on highest to lowest propensity
# ---------------------------------------------------

# create new datasets
# ---------
treat <- c()   # create empty treatment vector
control <- c() # create empty control vector
df_a1_small_copy <- as.data.frame(df_a1_small) # create a copy to prevent overwrite within cell
dists_ps_copy <- as.data.frame(dists_ps)       # create a copy to prevent overwrite within cell
```

```
# loop through to grab matches based on propensity scores
# ---------
for(i in 1:nrow(df_a1_small)) {
  max_treat <- which.max(df_a1_small_copy$prop_score)# %>% select(-ID)) # save max propensity score
  treat[i] <- names(max_treat)                                         # add max_treat names
  df_a1_small_copy <- df_a1_small_copy %>% slice(-max_treat)           # remove it from the dataframe

  match_control <- which.min(dists_ps_copy[max_treat,])                # find it's match in control
  control[i] <- names(all_of(match_control))                           # store names as control
  dists_ps_copy <- dists_ps_copy %>%                                   # drop what we have just select
      select(-match_control) %>%
      slice(-max_treat)
}

# print
# ---------
treat
```

```
## NULL
```

```
control
```

```
## [1] "V5"  "V3"  "V1"  "V2"  "V10"
```

```
# Remove post-treatment covariates

# Randomly select features

# Simulate random selection of features 10k+ times

# Fit  models and save ATTs, proportion of balanced covariates, and mean percent balance improvement

# Plot ATT v. proportion

# 10 random covariate balance plots (hint try gridExtra)
# Note: ggplot objects are finnicky so ask for help if you're struggling to automatically create them;
```

```
# Visualization for distributions of percent improvement
```

Questions

1. **Does your alternative matching method have more runs with higher proportions of balanced covariates?** Your Answer:...
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:...

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?**

   It might be a good idea to do matching because it's possible that randomization does not result in balanced sets of covariates across treatment and control. Also, it's possible that treatment take-up is imperfect– those that were assigned to treament may not have taken it, and vice versa– so matching might give us a more precise and valid estimate.

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, probably. Several of these methods can capture interactions and nonlinear relationships between covariates that a logistic regression would not automatically include. The ML methods can also prevent overfitting (overemphasizing bias over variability).

"'