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)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.1     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- 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)
library(cobalt)
```

```
##  cobalt (Version 4.5.0, Build Date: 2023-03-21)
##
## Attaching package: 'cobalt'
##
## The following object is masked from 'package:MatchIt':
##
##     lalonde
```

```r
# 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.
```

```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>, ...
```

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.

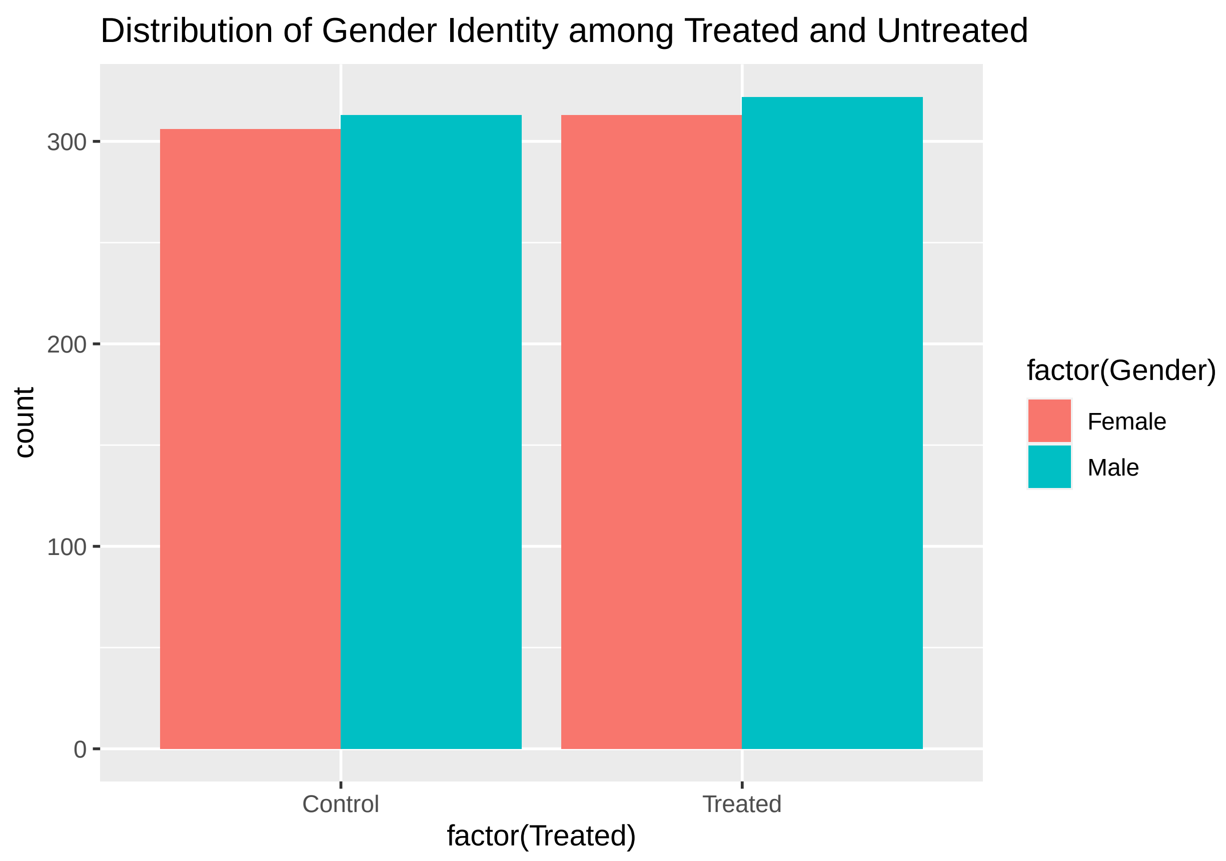
```
set.seed(1234)
n <- dim(ypsps)[1]

# Generate a vector that randomly assigns each unit to treatment/control
ypsps <- ypsps %>% mutate(Treatment = as.numeric(rbernoulli(n)))
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `Treatment = as.numeric(rbernoulli(n))`.
## Caused by warning:
## ! `rbernoulli()` was deprecated in purrr 1.0.0.
```

```
# Choose a baseline covariate (use dplyr for this)
ypsps %>% select(student_Gen, Treatment)
```

```
## # A tibble: 1,254 x 2
##    student_Gen Treatment
##          <dbl>     <dbl>
##  1           0         0
##  2           0         1
##  3           1         1
##  4           0         1
##  5           0         1
##  6           1         1
##  7           0         0
##  8           1         0
##  9           1         1
## 10           1         1
## # i 1,244 more rows
```

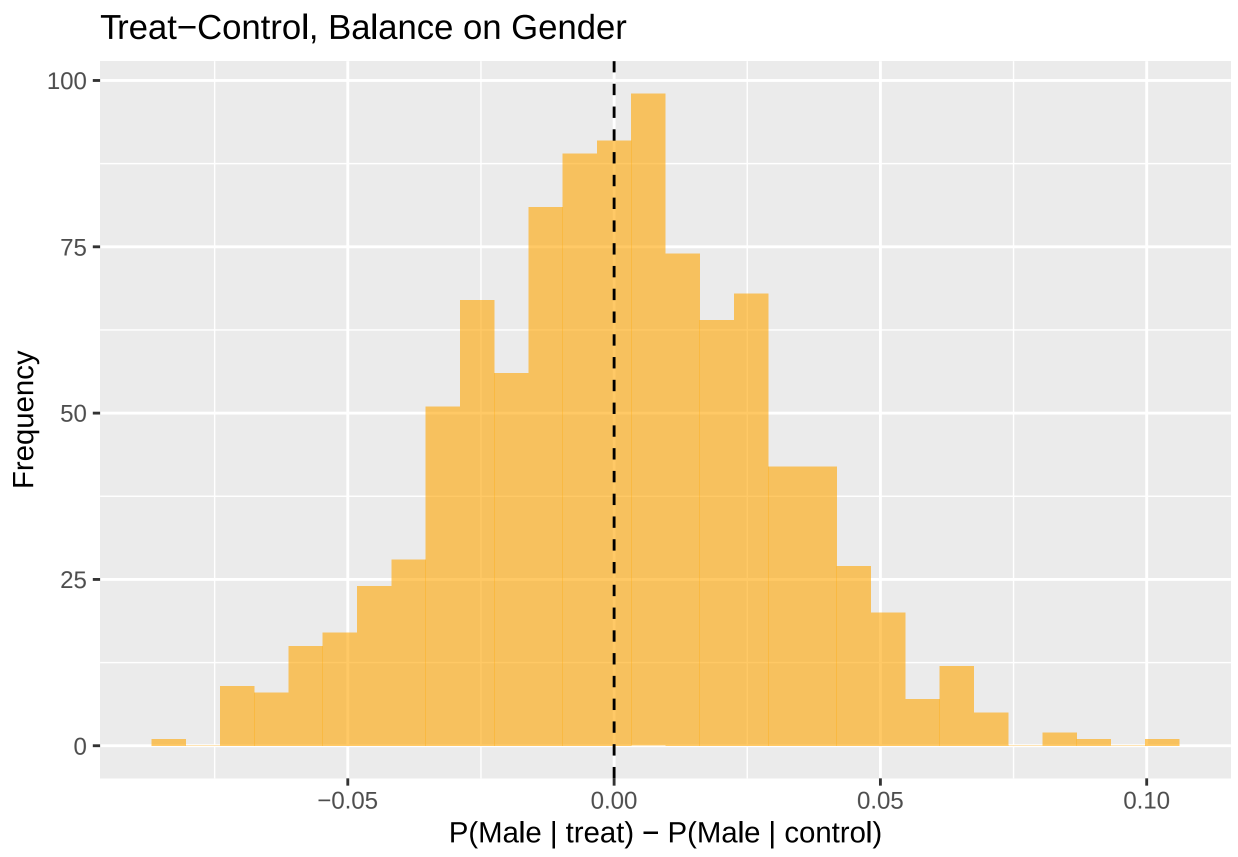
```
# Visualize the distribution by treatment/control (ggplot)
ggplot(ypsps %>% mutate(
  Gender = ifelse(student_Gen == 1, "Male", "Female"),
  Treated = ifelse(Treatment == 1, "Treated", "Control")
)
,
aes(x = factor(Treated), fill = factor(Gender))) +
  geom_bar(position = "dodge") +
  #facet_grid(Treatment~.) +
  labs(title = "Distribution of Gender Identity among Treated and Untreated")
```



```r
subset <- ypsps %>% select(student_Gen)

subset %>%
  mutate(Treatment = as.numeric(rbernoulli(n))) %>%
  group_by(Treatment) %>%
  summarize(mean = mean(student_Gen))
```

```
## # A tibble: 2 x 2
##   Treatment  mean
##       <dbl> <dbl>
## 1         0 0.479
## 2         1 0.534
```

```r
set.seed(1234)
# Simulate this 10,000 times (monte carlo simulation - see R Refresher for a hint)
differences <- c()
subset <- ypsps %>% select(student_Gen)

for (i in 1:1000){
  ypsps_mc <- subset %>% mutate(Treatment = as.numeric(rbernoulli(n)))
  treat_proportion <- ypsps_mc %>%
    group_by(Treatment) %>%
    summarise(mean_out = mean(student_Gen)) %>%
    subset(Treatment == 1) %>% select(mean_out) %>%
    max()
  control_proportion <- ypsps_mc %>%
    group_by(Treatment) %>%
    summarise(mean_out = mean(student_Gen)) %>%
    subset(Treatment == 0) %>% select(mean_out) %>%
    max()
  differences[i] <- treat_proportion - control_proportion
}

ggplot(data = data.frame(x = differences), aes(x = x)) +
  geom_histogram(bins=30, fill="orange", alpha=.6) +
  geom_vline(xintercept=0, linetype = "dashed") +
  labs(title = "Treat-Control, Balance on Gender", x = "P(Male | treat) - P(Male | control)", y = "Frequ
```



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: I observe that, while the most likely outcome is that treatment and control are balanced on gender, gender was still very unbalanced in a pretty large number of my simulations. I speculate that the randomization process may result in chance imbalances between the treatment and control groups, even when the sample size is large. These chance imbalances would lead to bias in the estimated treatment effect. # 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.

```
# Select covariates that represent the "true" model for selection, fit model
model_data <- ypsps %>%
  select(college,
         student_ppnscal,
         student_Gen,
         student_Race,
         student_GPA,
         parent_HHInc,
         parent_EducW,
         parent_EducHH)

model_ps <-
  matchit(
    formula = college ~ student_Gen + student_Race + student_GPA +
      parent_HHInc + parent_EducW + parent_EducHH,
    data = model_data,
    method = "nearest",
    distance = "glm",
    link="logit",
    estimand = "ATT"
  )
```

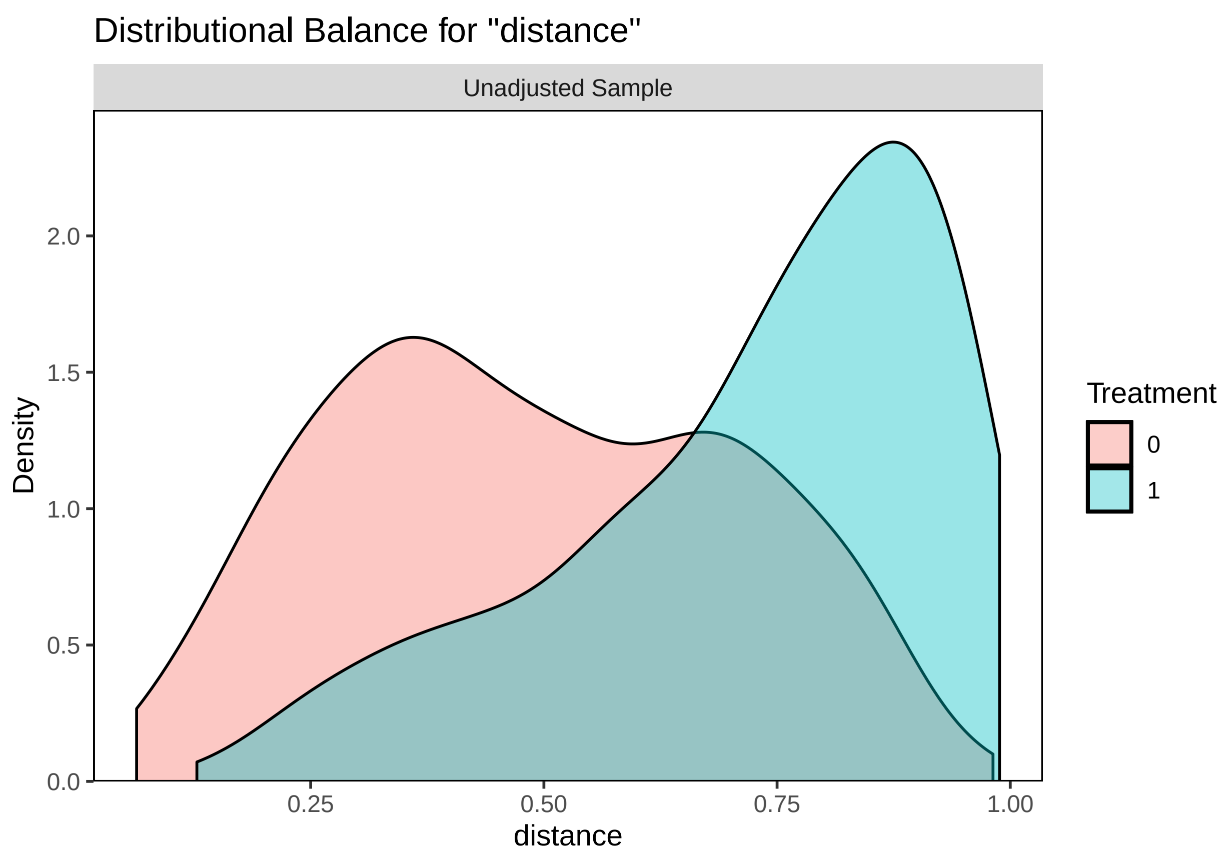
```
## Warning: Fewer control units than treated units; not all treated units will get
## a match.
```

```
match_exact_ate_data <- match.data(model_ps)
lm_att <-
  lm(
    student_ppnscal ~ college + student_Gen + student_Race + student_GPA +
      parent_HHInc + parent_EducW + parent_EducHH,
    data = match_exact_ate_data,
    weights = weights
  )
att_ps <- lm_att$coefficients['college1']
att_ps
```

```
## <NA>
##   NA
```

```
bal.plot(model_ps, which = 'unadjusted')
```

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



```r
out <- bal.tab(model_ps, binary = "std", threshold = 0.1)
out
```

```
## Balance Measures
##                    Type Diff.Adj        M.Threshold
## distance       Distance   1.9101
## student_Gen      Binary   0.3341 Not Balanced, >0.1
## student_Race    Contin.  -0.0727     Balanced, <0.1
## student_GPA     Contin.  -0.8298 Not Balanced, >0.1
## parent_HHInc    Contin.   1.0642 Not Balanced, >0.1
## parent_EducW    Contin.   1.1467 Not Balanced, >0.1
## parent_EducHH   Contin.   1.3672 Not Balanced, >0.1
##
## Balance tally for mean differences
##                    count
## Balanced, <0.1         1
## Not Balanced, >0.1     5
##
## Variable with the greatest mean difference
##       Variable Diff.Adj        M.Threshold
##  parent_EducHH   1.3672 Not Balanced, >0.1
##
## Sample sizes
##           Control Treated
## All           451     803
## Matched       451     451
## Unmatched       0     352
```

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)

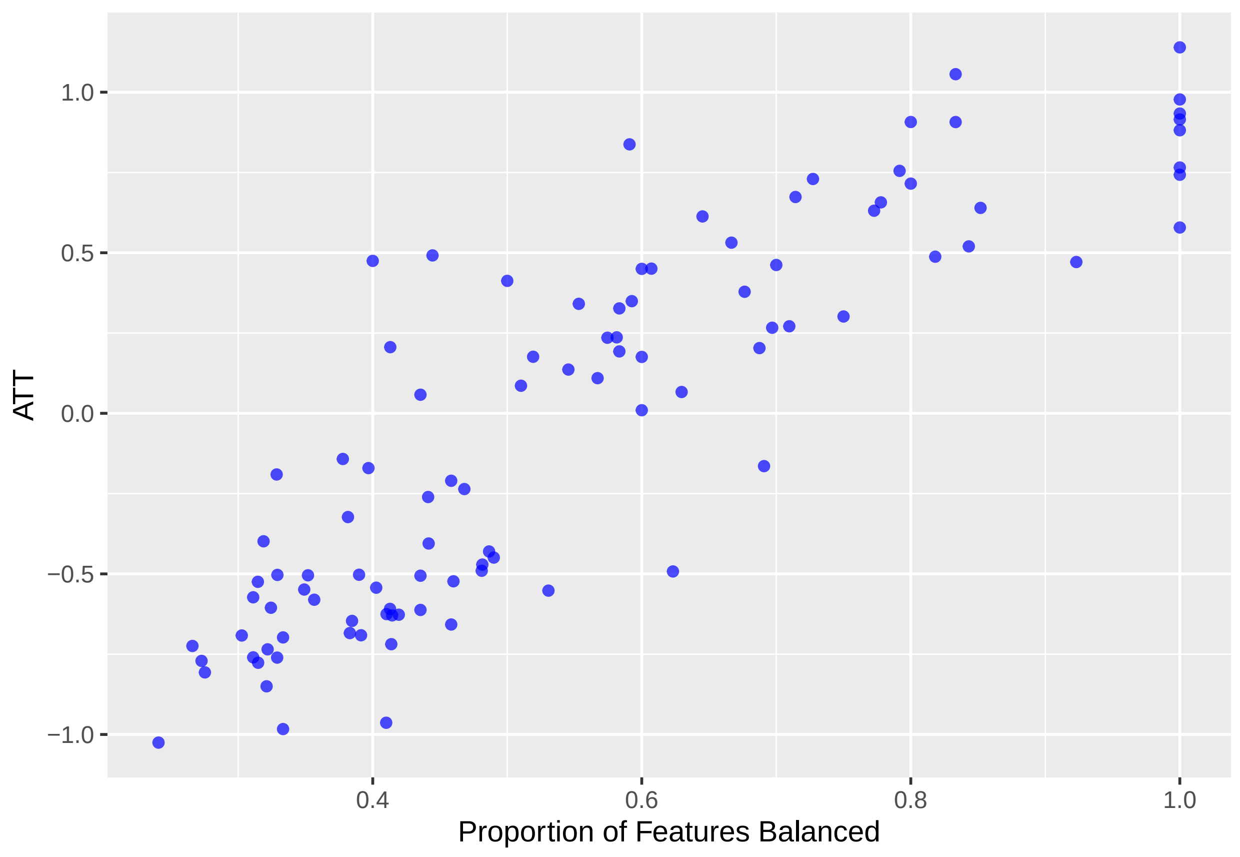
Note: There are lots of post-treatment covariates in this dataset (about 50!)! You need to be careful not to include these in the pre-treatment balancing. Many of you are probably used to selecting or dropping columns manually, or positionally. However, you may not always have a convenient arrangement 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
# Remove post-treatment covariates
ypsps_clean <- ypsps %>%
  select(-c(
    matches("_19\\d\\d"),
    'interviewid',
    'Treatment',
    matches('\\wPlacebo')
  ))

# Simulate random selection of features 10k+ times
atts <- c()
prop_balanced <- c()
percent_imp <- c()

for (i in 1:100) {
  # Randomly select features
  colnames <- sample(names(ypsps_clean %>% select(-c(
    student_ppnscal, college
  ))),
  sample(2:ncol(ypsps_clean), 1) - 1)
  independent <- paste(colnames, collapse = " + ")
  ps_formula <- as.formula(paste("college ~", independent))
  # Fit p-score models and save ATTs, proportion of balanced covariates, and mean percent balance impro
  model_ps <-
    matchit(
      replace = TRUE,
      formula = ps_formula,
      data = ypsps_clean,
      method = "nearest",
      ratio = 1,
      distance = "glm",
      link = "logit",
      estimand = "ATT",
    )
  lm_att <-
    lm(
      student_ppnscal ~ college + student_Gen + student_Race + student_GPA +
        parent_HHInc + parent_EducW + parent_EducHH,
      data = match.data(model_ps),
      weights = weights
    )
  out <- bal.tab(model_ps, threshold = .1)
  att_ps <- lm_att$coefficients['college']
  atts[i] <- att_ps
  prop_balanced[i] <- out$Balanced.mean.diffs['Balanced, <0.1', ] /
    out$Balanced.mean.diffs %>% sum()
  pre <- data.frame(summary(model_ps)$sum.all) %>%
    pull(Std..Mean.Diff.) %>%
    mean()
  post <- data.frame(summary(model_ps)$sum.matched) %>%
    pull(Std..Mean.Diff.) %>%
    mean()
  percent_imp[i] <- (post - pre) / pre
}
```

```
sim_results <- data.frame(atts, prop_balanced, percent_imp)

sim_results %>% filter(prop_balanced > 2/29) %>% nrow()
```

```
## [1] 100
```

```
# Plot ATT v. proportion
# Customize the scatterplot
ggplot(sim_results, aes(x = prop_balanced, y = atts)) +
  geom_point(color = "blue",
             fill = "blue",
             alpha = .7) +
  labs(x = "Proportion of Features Balanced", y = "ATT")
```



Questions

1. **How many simulations resulted in models with a higher proportion of balanced covariates? Do you have any concerns about this?**
2. **Your Answer**: It seems that all of the simulations resulted in having a higher propostion of balanced covariates, which does cause me concern, though moreso because this is probably some sort of coding error.
3. **Analyze the distribution of the ATTs. Do you have any concerns about this distribution?**
4. **Your Answer:** The ATTs range from -1 to above 1 – they are variable and the simulations show that under a misspecified model it is possible to get the totally opposite sign for the ATT.
5. **Do your 10 randomly chosen covariate balance plots produce similar numbers on the same covariates? Is it a concern if they do not?**
6. **Your Answer:**

Matching Algorithm of Your Choice

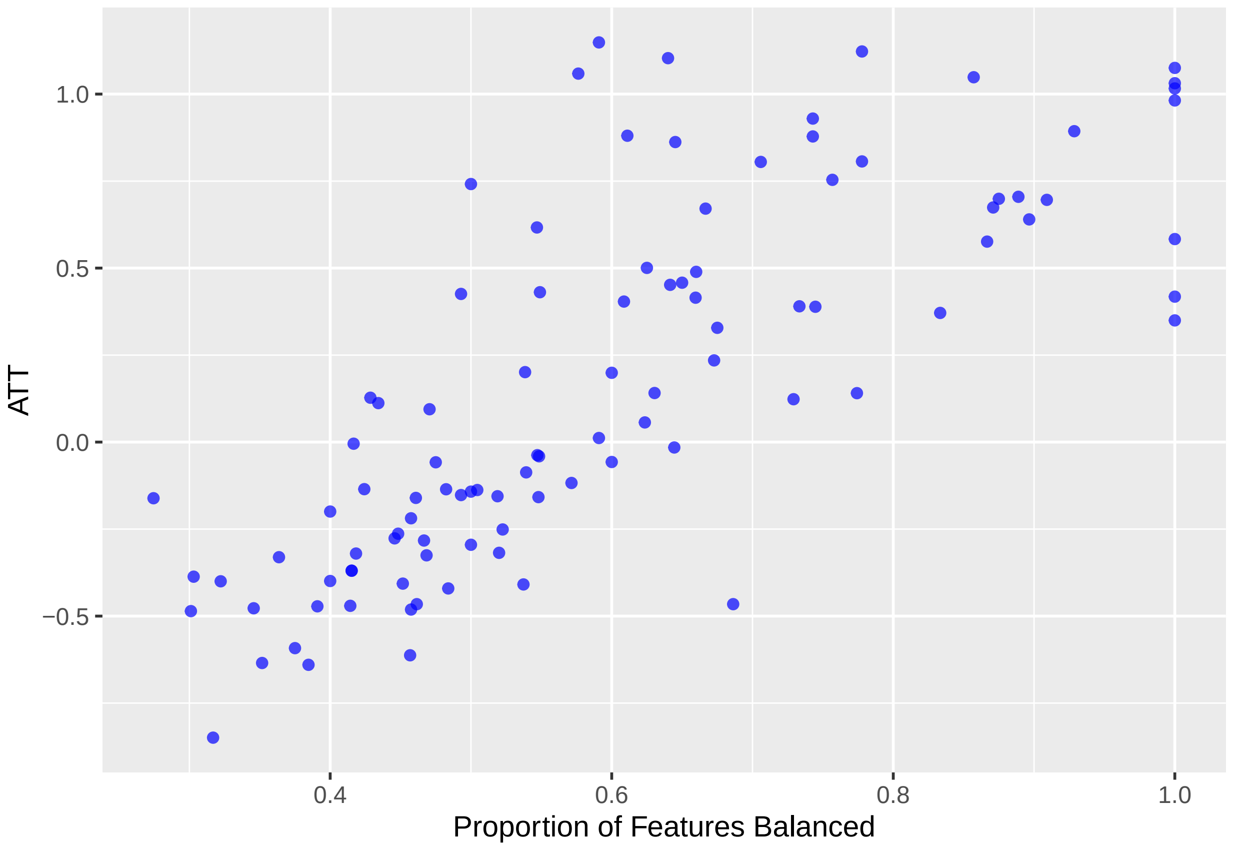
Simulate Alternative Model

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:

```r
# Simulate random selection of features 10k+ times
atts <- c()
prop_balanced <- c()
percent_imp <- c()

for (i in 1:100) {
  # Randomly select features
  colnames <- sample(names(ypsps_clean %>% select(-c(
    student_ppnscal, college
  ))),
  sample(2:ncol(ypsps_clean), 1) - 1)
  independent <- paste(colnames, collapse = " + ")
  ps_formula <- as.formula(paste("college ~", independent))
  # Fit p-score models and save ATTs, proportion of balanced covariates, and mean percent balance impro
  model_ps <-
    matchit(
      formula = ps_formula,
      data = ypsps_clean,
      method = "quick",
      ratio = 1,
      distance = "glm",
      link = "logit",
      estimand = "ATT",
    )
  lm_att <-
    lm(
      student_ppnscal ~ college + student_Gen + student_Race + student_GPA +
        parent_HHInc + parent_EducW + parent_EducHH,
      data = match.data(model_ps),
      weights = weights
    )
  out <- bal.tab(model_ps, threshold = .1)
  att_ps <- lm_att$coefficients['college']
  atts[i] <- att_ps
  prop_balanced[i] <- out$Balanced.mean.diffs['Balanced, <0.1', ] /
    out$Balanced.mean.diffs %>% sum()
  pre <- data.frame(summary(model_ps)$sum.all) %>%
    pull(Std..Mean.Diff.) %>%
    mean()
  post <- data.frame(summary(model_ps)$sum.matched) %>%
    pull(Std..Mean.Diff.) %>%
    mean()
  percent_imp[i] <- (post - pre) / pre
}
```

```
## Warning: (from quickmatch) Most seeds are at zero distance to their neighbors
## in the clustering NNG. This typically happens with discrete low-dimensional
## covariates. Consider using exact matching with such data. Running with
## `secondary_unassigned_method == "ignore"`.-2

## Warning: (from quickmatch) Most seeds are at zero distance to their neighbors
## in the clustering NNG. This typically happens with discrete low-dimensional
## covariates. Consider using exact matching with such data. Running with
## `secondary_unassigned_method == "ignore"`.-2
## Warning: (from quickmatch) Most seeds are at zero distance to their neighbors
## in the clustering NNG. This typically happens with discrete low-dimensional
## covariates. Consider using exact matching with such data. Running with
## `secondary_unassigned_method == "ignore"`.-2
```

```
# Visualization for distributions of percent improvement
sim_results <- data.frame(atts, prop_balanced, percent_imp)

ggplot(sim_results, aes(x = prop_balanced, y = atts)) +
  geom_point(color = "blue",
             fill = "blue",
             alpha = .7) +
  labs(x = "Proportion of Features Balanced", y = "ATT")
```



Questions

1. **Does your alternative matching method have more runs with higher proportions of balanced covariates?**
2. **Your Answer:** I do see that the quick-match algorithm tightened the distribution of the ATT estimates, which I would say is a better sign.
3. **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.**
4. **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?
2. **Your Answer:** Random selection can still lead to clas imbalance, if only for the random chance that classes are unbalanced. Matching will remove the bias that this imbalance causes in our estimates.
3. 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?
4. **Your Answer:** I think that using Logistic Regression for all of the variables in this dataset and matching (or even a very large number of them) would be a bad idea given multicollinearity. At the least, I think that the addition of L1 regularization (lasso) would be beneficial in this case to produce a parsimonious model. I think that using more complex methods, like ensembles, could surely create more accurate measures of propensity to receive treatment (or at the least adding cross validation when we model propensity scores), though I wonder if having an unintelligible model (e.g. gradient boosted trees), might be problematic.