Project 6: Randomization and Matching

Sofia Guo, working with Stacy Chen

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)
library(MatchIt)
library(ggplot2)
library(cobalt)
library(gridExtra)
library(optmatch)

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

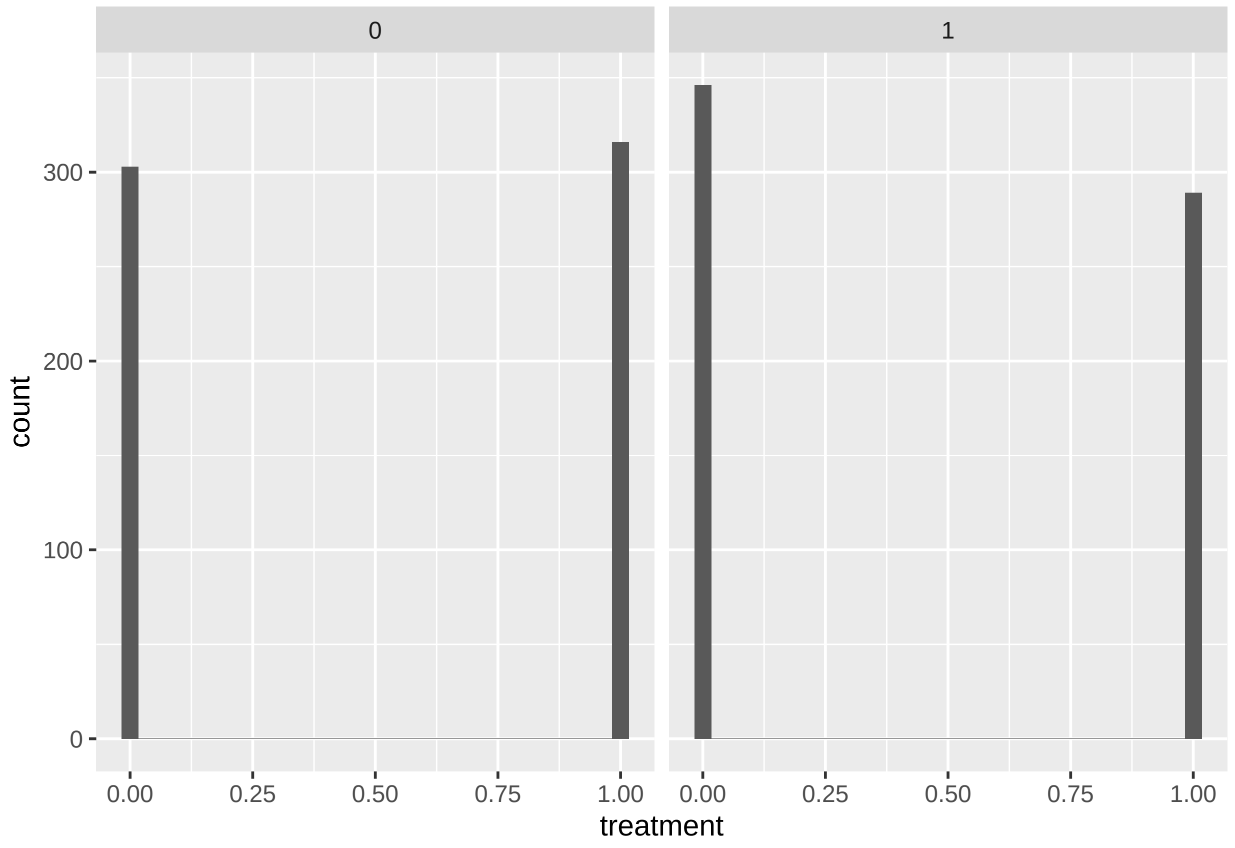
Randomization

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:

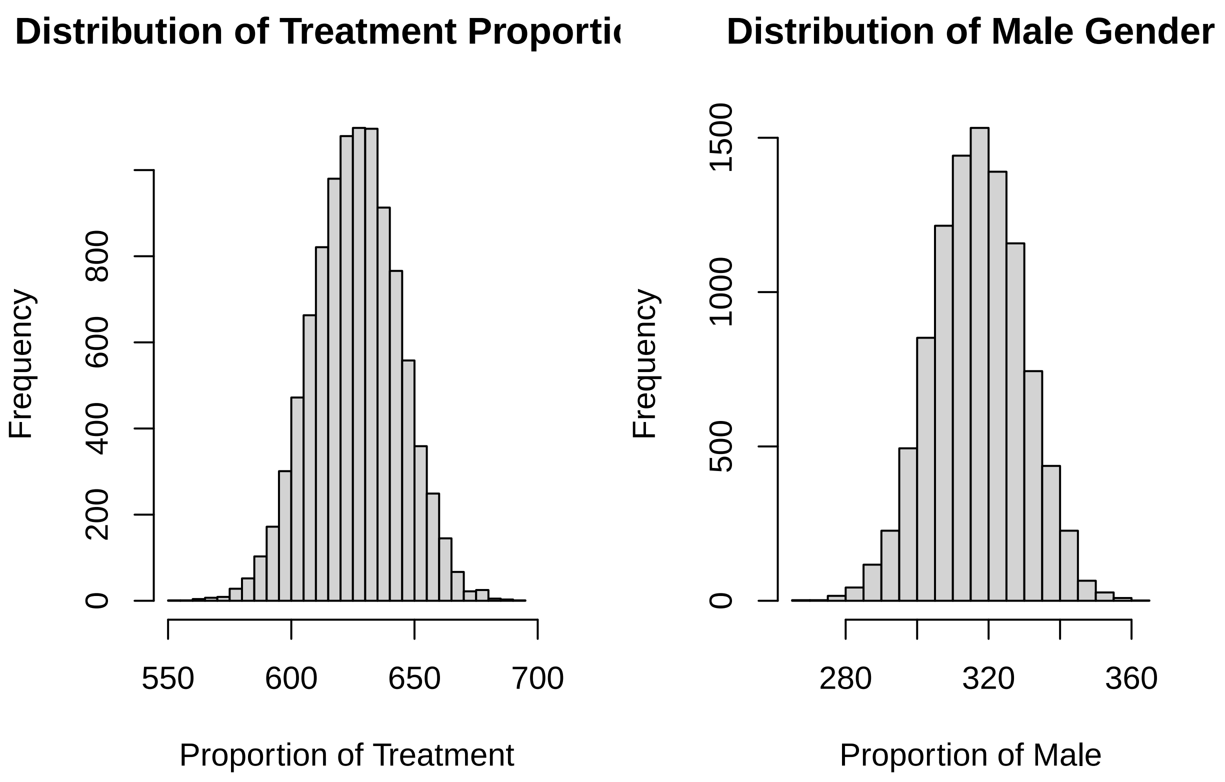
1. Generate a vector that randomly assigns each unit to either treatment or control
2. Choose a baseline covariate (for either the student or parent). A binary covariate is probably best for this exercise.
3. Visualize the distribution of the covariate by treatment/control condition. Are treatment and 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.

```r
# Generate a vector that randomly assigns each unit to treatment/control
# completely randomize treatment
# ----------
# set seed - unfortunately seed needs to be set within cell to be reproducible in .Rmd but just once in
set.seed(14)
```

```
df <-                                                               # save object
  ypsps %>%                                                         # pass data
  mutate(treatment = as.numeric(rbernoulli(length(unique(interviewid)), p=0.5)))             # create c
      # Y_comp = as.numeric((A_comp & student_ppnscal) | (!A_comp & student_ppnscal))) # create comple

# Choose a baseline covariate (use dplyr for this)
baselinecov <- df %>%
  select(student_Gen,treatment)

# Visualize the distribution by treatment/control (ggplot)
ggplot(baselinecov) +
  geom_histogram(aes(treatment)) +
  facet_wrap(.~student_Gen)
```



```
# Simulate this 10,000 times (monte carlo simulation - see R Refresher for a hint)
n_simulations <- 10000

# Perform Monte Carlo simulation
# Empty matrix to store simulation results
sim_results <- matrix(nrow = n_simulations, ncol = 2)

# Perform Monte Carlo simulation
for (i in 1:n_simulations) {
  # Generate treatment assignment vector
  df <- ypsps %>%
    select(interviewid,student_Gen) %>%
    mutate(treatment = as.numeric(rbernoulli(length(unique(interviewid)), p=0.5)))
  # Calculate the proportion of treatment units
  proportion_treatment <- sum(df$treatment)

  # Calculate the proportion of Male gender
  proportion_male <- sum(df$student_Gen[df$treatment == 1])

  # Store the results
  sim_results[i, 1] <- proportion_treatment
  sim_results[i, 2] <- proportion_male
}

# Visualize the distribution of treatment proportions
par(mfrow = c(1, 2)) # Arrange plots in one row and two columns
hist(sim_results[, 1], breaks = 30, main = "Distribution of Treatment Proportions",
     xlab = "Proportion of Treatment", ylab = "Frequency")
hist(sim_results[, 2], breaks = 30, main = "Distribution of Male Gender",
     xlab = "Proportion of Male", ylab = "Frequency")
```



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?**

Across our simulations, we see that although the proportion of treatment is almost exactly balanced (half of 1254 is 627, which according to the histogram on the left is right where the mean sits), slightly more than half of the student genders are equal to 1 (since the documentation is unclear on whether student_gender ==1 is male or female, I just assume that it means male for the purpose of this exercise). I make this observation because half of 627 is 314, but the mean of the right side histogram of the gender distribution appears to be closer to 320 than 314. This imbalance can occur because of random chance and when our simulation/sampling size is not large enough towards infinity.

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 ≤ .1, report the number of covariates that meet that balance threshold.

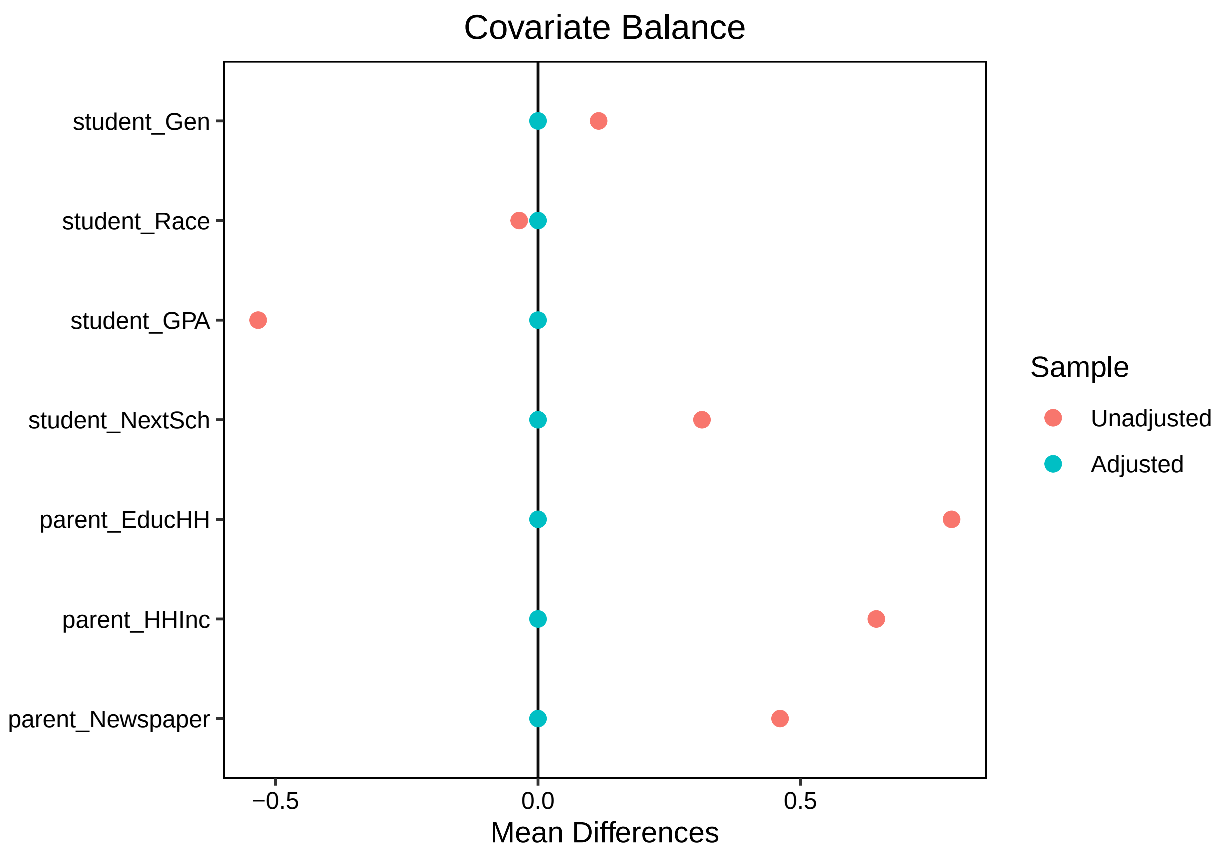
```
# Select covariates that represent the "true" model for selection, fit model
df <- ypsps %>%
  select(interviewid, college, student_ppnscal, student_Gen, student_Race, student_GPA, student_NextSch

# fit model
M1_student_college <- glm(college ~ student_Gen + student_Race + student_GPA + student_NextSch + parent_

# Step 2: Predict propensity scores
df$propensity_score <- predict(M1_student_college, type = "response")

# Step 3: Calculate ATT
# Assuming 'df' is your dataset containing treatment, covariates, and outcome
# Match treated and control units
match_exact_att <- matchit(college ~ student_Gen + student_Race + student_GPA + student_NextSch + paren

# Report the overall balance and the proportion of covariates that meet the balance threshold
match_summ <- summary(match_exact_att, un=F)
match_summ$sum.matched[ , "Std. Mean Diff."]
```

```
##      student_Gen     student_Race      student_GPA  student_NextSch
##    -2.220446e-16    -2.220446e-16    -1.776357e-15    -2.220446e-16
##    parent_EducHH     parent_HHInc parent_Newspaper
##    -2.220446e-15    -4.440892e-15    -2.220446e-15
```

```
match_summ$nn
```

```
##                 Control Treated
## All (ESS)     451.00000     803
## All           451.00000     803
## Matched (ESS)  76.64557     332
## Matched       178.00000     332
## Unmatched     273.00000     471
## Discarded       0.00000       0
```

```
#make covariate plot
love.plot(match_exact_att)
```



```
  # Calculate SMD before matching
  #define covariates
  covariates <- c("student_Gen", "student_Race", "student_GPA","student_NextSch", "parent_EducHH", "pare
  matched_df <- match.data(match_exact_att)

  smd_before <- sapply(df[, covariates], function(x) {
    (mean(x[df[["college"]] == 1]) - mean(x[df[["college"]] == 0])) /
    sqrt((var(x[df[["college"]] == 1]) + var(x[df[["college"]] == 0])) / 2)
  })

  # Calculate SMD after matching
  smd_after <- sapply(df[, covariates], function(x) {
    (mean(x[matched_df[["college"]] == 1]) - mean(x[matched_df[["college"]] == 0])) /
    sqrt((var(x[matched_df[["college"]] == 1]) + var(x[matched_df[["college"]] == 0])) / 2)
  })

  # Calculate mean percent improvement
  mean_percent_improvement <- mean((smd_before - smd_after) / smd_before * 100, na.rm = TRUE)

print(mean_percent_improvement)
```

```
## [1] 133.8127
```

The calculated mean percent improvement in the standardized mean difference is approximately 133. It looks like all the covariates I chose met the threshold, so I went ahead and estimated the ATT:

```
#estimate the ATT using linear regression
match_exact_att_data <- match.data(match_exact_att)
```

```r
#specify model
lm_full_att <- lm(student_ppnscal ~ college + student_Gen + student_Race + student_GPA + student_NextSc

#summarize results
lm_full_att_summ <- summary(lm_full_att)

#calculate ATT
ATT_full <- lm_full_att_summ$coefficients["college","Estimate"]
ATT_full
```

```
## [1] 0.9228414
```

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:

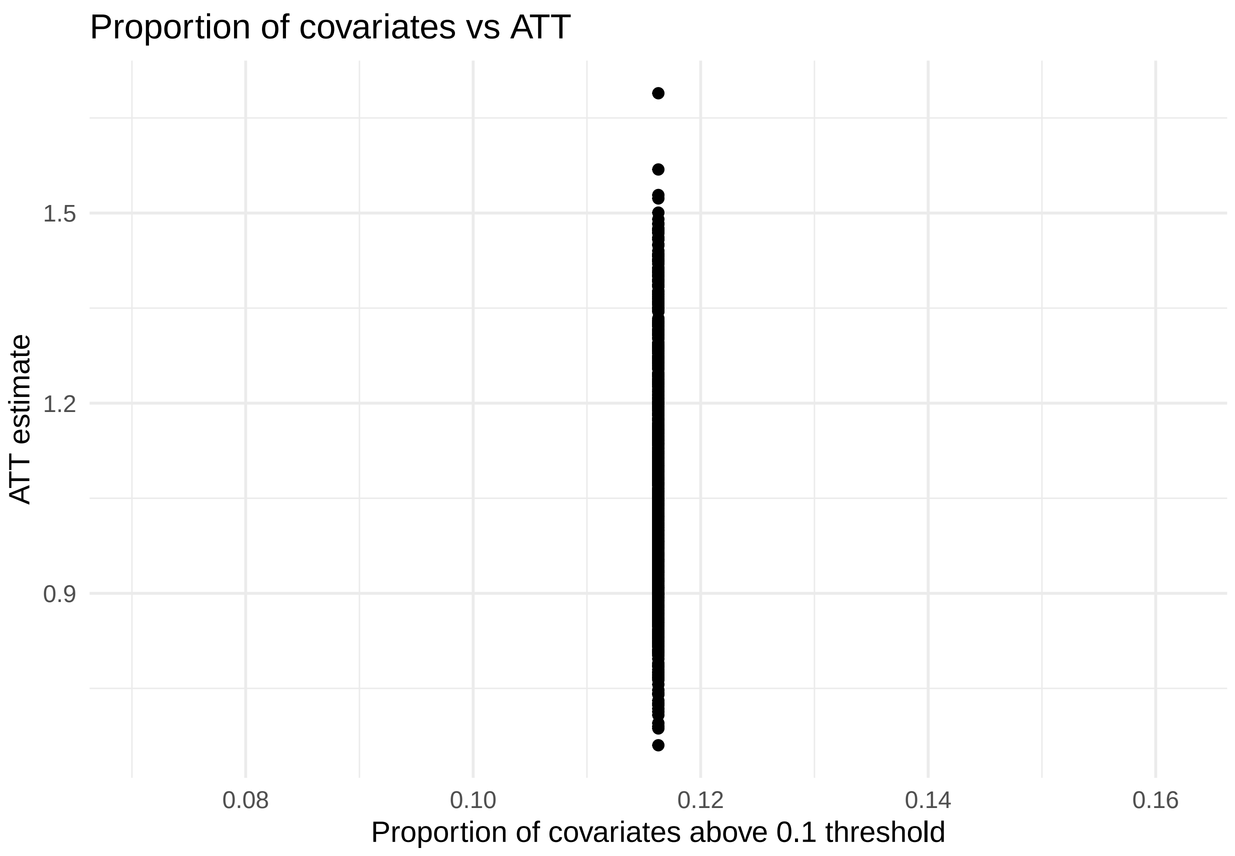
- Using as many simulations as is feasible (at least 10,000 should be ok, more is better!), randomly select the number of and the choice of covariates for the propensity score model.
- For each run, store the ATT, the proportion of covariates that meet the standardized mean difference $\leq .1$ threshold, 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
set.seed(14)
# Remove post-treatment covariates
post_vars <- c(colnames(ypsps)[1:11],colnames(ypsps)[123:174])

# Step 2: Drop post-treatment variables using dplyr
prevars_df <- ypsps %>%
  select(-(c(post_vars, parent_GPHighSchoolPlacebo, parent_HHCollegePlacebo))) %>%
  filter(complete.cases(.))

# define the prevars column names
pre_vars <- colnames(prevars_df)

# Initialize an empty matrix to store results
result_matrix <- matrix(nrow = 10000, ncol = 3)
colnames(result_matrix) <- c("ATT", "Proportion", "PercentImproved")
```

```r
# Set up loop to iterate 10,000 times
for (i in 1:10000) {
  # 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))

  # Fit the propensity score model (assuming glm for simplicity)
  model <- glm(as.formula(paste("college ~", paste(random_covariates, collapse = "+"))), data = df, fam

  # Step 3: Calculate ATT
  # Match treated and control units
  match_att <- matchit(as.formula(paste("college ~", paste(random_covariates, collapse = "+"))), data =

  # Report the overall balance and the proportion of covariates that meet the balance threshold
  att_summ <- summary(match_att, un=F)
  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)

  # Estimate the ATT using linear regression
  match_att_data <- match.data(match_att)

  # Calculate SMD before matching
  #define covariates
  covariates <- random_covariates
  matched_df <- match_att_data

  smd_before <- sapply(df[, covariates], function(x) {
  (mean(x[df[["college"]] == 1],na.rm=T) - mean(x[df[["college"]] == 0],na.rm=T)) /
  sqrt((var(x[df[["college"]] == 1]) + var(x[df[["college"]] == 0])) / 2)
  })

  # Calculate SMD after matching
  smd_after <- sapply(df[, covariates], function(x) {
    (mean(x[matched_df[["college"]] == 1],na.rm=T) - mean(x[matched_df[["college"]] == 0],na.rm=T)) /
    sqrt((var(x[matched_df[["college"]] == 1]) + var(x[matched_df[["college"]] == 0])) / 2)
  })

  # Calculate mean percent improvement
  mean_percent_improvement <- mean((smd_before - smd_after) / smd_before * 100, na.rm = TRUE)

  # Specify model
  lm_full_att <- lm(as.formula(paste("student_ppnscal ~", paste("college", paste(random_covariates, col

  # Summarize results
  lm_att_summ <- summary(lm_full_att)

  # Calculate ATT
  ATT <- lm_att_summ$coefficients["college", "Estimate"]

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

```
set.seed(14)
# plot the ATT's against proportion of covariates
# Convert matrix to long-format data frame
result_df <- as.data.frame(result_matrix)

#take subsample for better plotting
# Randomly subsample the data frame
subsample_df <- result_df[sample(nrow(result_df), 1000), ]

# scatterplot
ggplot(subsample_df, aes(proportion_true, ATT)) +
  geom_point()+
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(title = "Proportion of covariates vs ATT", x = "Proportion of covariates above 0.1 threshold", y
  theme_minimal()
```

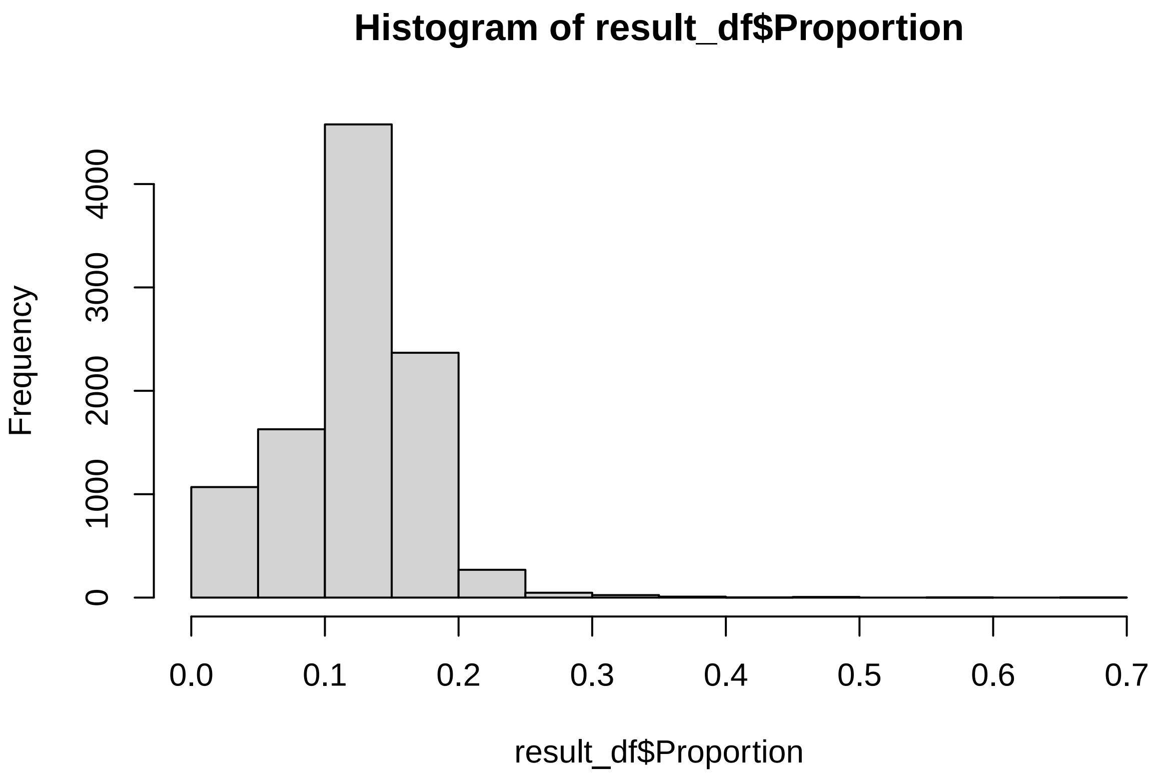


```
#count number of simulations where proportion was higher

#find mean proportion
meanprop <- mean(result_df$Proportion)
```

```r
#filter for higher than mean proportion
higherprop <- result_df %>%
  filter(Proportion > meanprop)

#count number of simulations
nrow(higherprop)
```

```
## [1] 5790
```

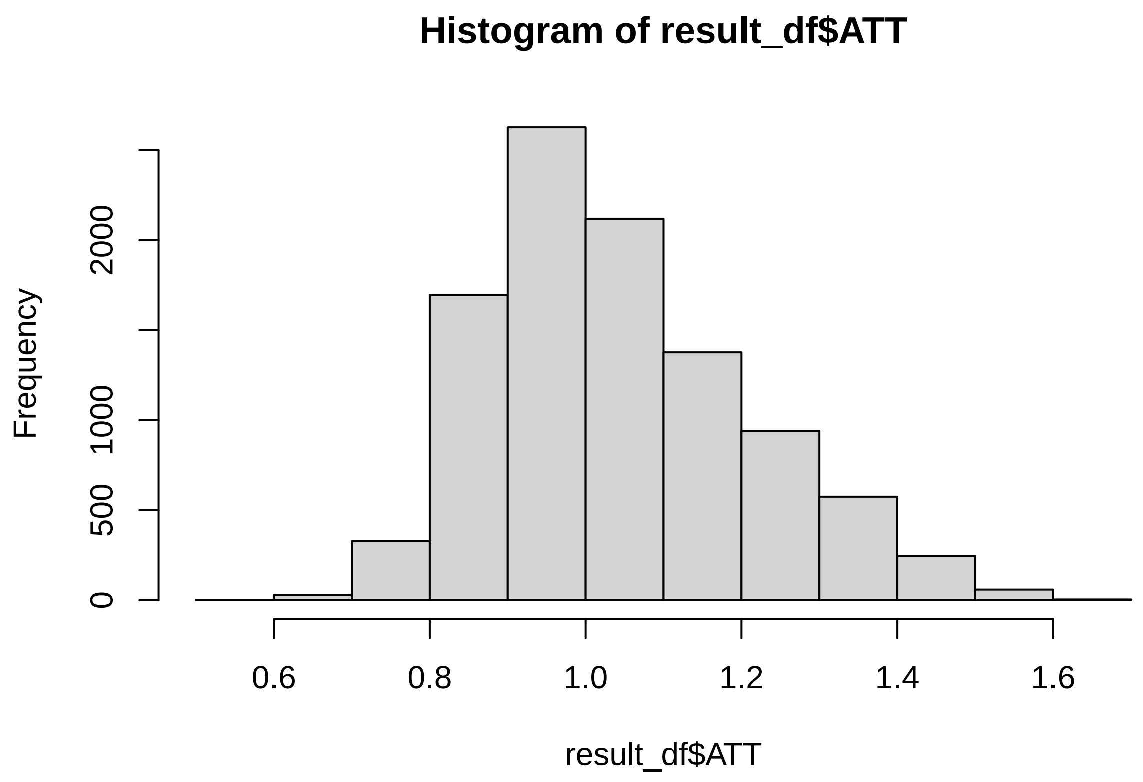
```r
#histogram of proportion true
hist(result_df$Proportion)
```



```r
#examine the higher proportion subset
summary(higherprop)
```

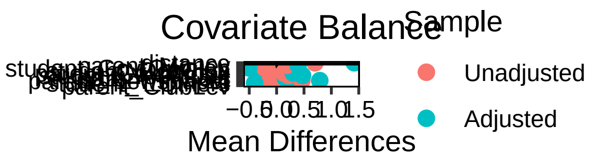
```
##       ATT           Proportion     PercentImproved
##  Min.   :0.5761   Min.   :0.1220   Min.   :-290.51
##  1st Qu.:0.9015   1st Qu.:0.1364   1st Qu.:  76.38
##  Median :0.9806   Median :0.1489   Median :  88.25
##  Mean   :1.0076   Mean   :0.1564   Mean   :  88.06
##  3rd Qu.:1.0917   3rd Qu.:0.1667   3rd Qu.: 100.85
##  Max.   :1.6439   Max.   :0.6667   Max.   : 930.48
```

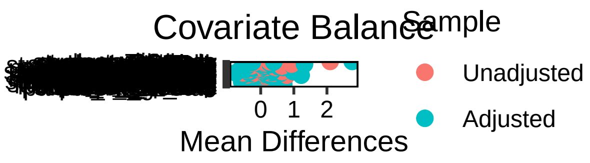
```r
#histogram of ATT
hist(result_df$ATT)
```

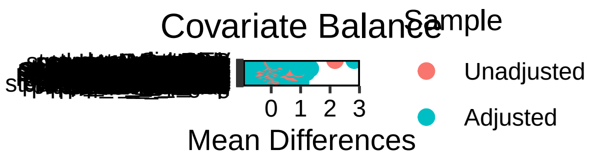


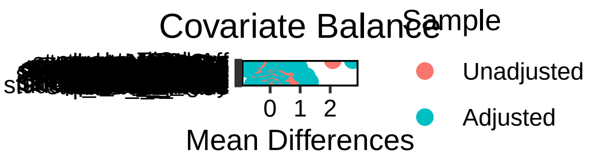
```r
#randomly choose 10 balance plots
set.seed(14)

#empty list of love plots
match_list <- list()

# Set up loop to iterate 10 times
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))

  # Step 3: Calculate ATT
  # Match treated and control units
  match_att <- matchit(as.formula(paste("college ~", paste(random_covariates, collapse = "+"))), data =

  # Store the results in the result matrix
  match_list[[i]] <- love.plot(match_att)
}
```

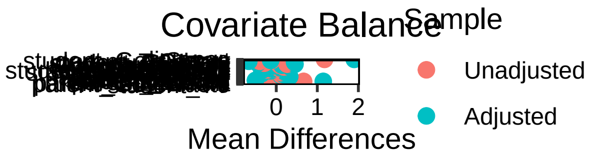
```r
# Arrange the grobs using grid.arrange
grid.arrange(grobs = match_list, ncol = 2)
```

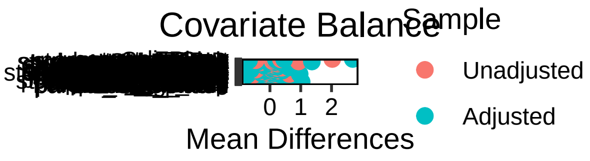


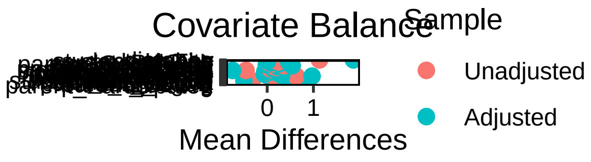


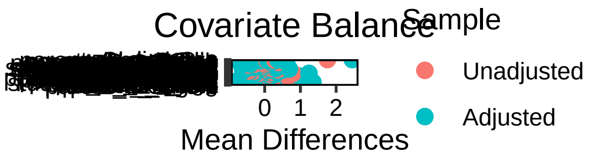


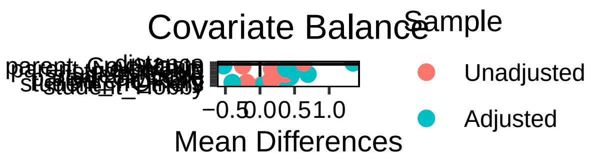


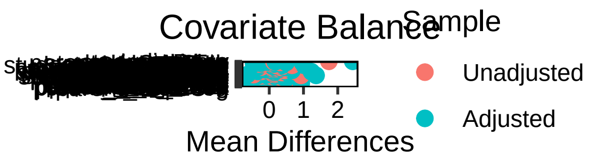












## Questions

1. **How many simulations resulted in models with a higher proportion of balanced covariates? Do you have any concerns about this?** Assuming that "higher proportion" means greater than the average proportion of balanced covariates, I found 5,790 out of the 10,000 simulations to have higher proportions. This is approximately 58 percent of the simulations, which is slightly higher than what you would expect if working with a normal distribution (50 percent). Thus, I would be concerned that my propensity scoring method is upwardly biased in making my covariates seem stronger than they would be if they were coming from a normal distribution. Based on the histogram of the proportions above, you can see that the distribution indeed appears left-skewed and not normal.

2. **Analyze the distribution of the ATTs. Do you have any concerns about this distribution?** The ATTs seem to be non-normally distributed with more mass below the mean (right skew). Therefore, I would be concerned that the treatment effects are being underestimated depending on the specification used.

3. **Do your 10 randomly chosen covariate balance plots produce similar numbers on the same covariates? Is it a concern if they do not?** It is hard to tell from the covariate balance plots which covariates are the same because some of them have a lot of covariates (due to the randomness of the number and type of covariate selected). Overall, it looks like the balance is very inconsistent regardless if there are many or few covariates, which is concerning because it means that my estimated treatment effects are very sensitive to the model specification. Unless I know for certain that I have specified the exact model for propensity score matching, I have no way to really know whether my results are being pulled in one direction or the other by misspecification.

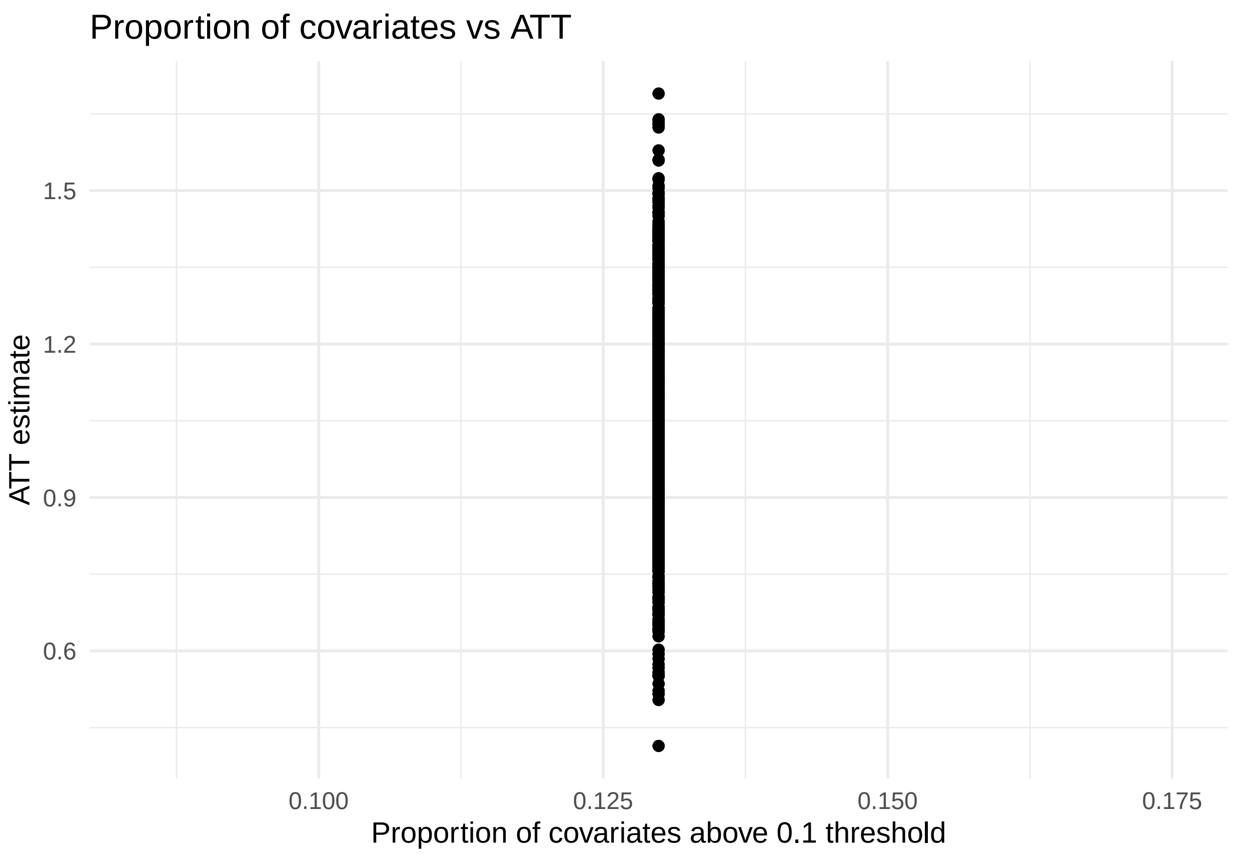
Matching Algorithm of Your Choice

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
#k-nearest neighbor matching
set.seed(14)
# Remove post-treatment covariates
post_vars <- c(colnames(ypsps)[1:11],colnames(ypsps)[123:174])

# Step 2: Drop post-treatment variables using dplyr
prevars_df <- ypsps %>%
  select(-(c(post_vars, parent_GPHighSchoolPlacebo, parent_HHCollegePlacebo))) %>%
  filter(complete.cases(.))

# define the prevars column names
pre_vars <- colnames(prevars_df)

# Initialize an empty matrix to store results
result_matrix1 <- matrix(nrow = 10000, ncol = 3)
colnames(result_matrix1) <- c("ATT", "Proportion", "PercentImproved")

# Set up loop to iterate 10,000 times
for (i in 1:10000) {
  # 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)) %>%
    slice_sample(n = 1000)

  # Step 3: Calculate ATT
  # Match treated and control units
  match_att <- matchit(as.formula(paste("college ~", paste(random_covariates, collapse = "+"))),
                     data = df,
                     method = "nearest",
                     distance = "glm",
                     link = "logit",
                     discard = "control",
                     replace = FALSE,
                     ratio = 2)

  # Report the overall balance and the proportion of covariates that meet the balance threshold
  att_summ <- summary(match_att, un=F)
  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)
  # Estimate the ATT using linear regression
  match_att_data <- match.data(match_att)

  # Calculate SMD before matching
  #define covariates
  covariates <- random_covariates
  matched_df <- match_att_data

  smd_before <- sapply(df[, covariates], function(x) {
    (mean(x[df[["college"]] == 1],na.rm=T) - mean(x[df[["college"]] == 0],na.rm=T)) /
    sqrt((var(x[df[["college"]] == 1]) + var(x[df[["college"]] == 0])) / 2)
  })

  # Calculate SMD after matching
  smd_after <- sapply(df[, covariates], function(x) {
    (mean(x[matched_df[["college"]] == 1],na.rm=T) - mean(x[matched_df[["college"]] == 0],na.rm=T)) /
    sqrt((var(x[matched_df[["college"]] == 1]) + var(x[matched_df[["college"]] == 0])) / 2)
  })

  # Calculate mean percent improvement
  mean_percent_improvement <- mean((smd_before - smd_after) / smd_before * 100, na.rm = TRUE)

  # Specify model
  lm_full_att <- lm(as.formula(paste("student_ppnscal ~", paste("college", paste(random_covariates, col

  # Summarize results
  lm_att_summ <- summary(lm_full_att)

  # Calculate ATT
  ATT <- lm_att_summ$coefficients["college", "Estimate"]

  # Store the results in the result matrix
  result_matrix1[i, ] <- c(ATT, proportion_true, mean_percent_improvement)
}
```

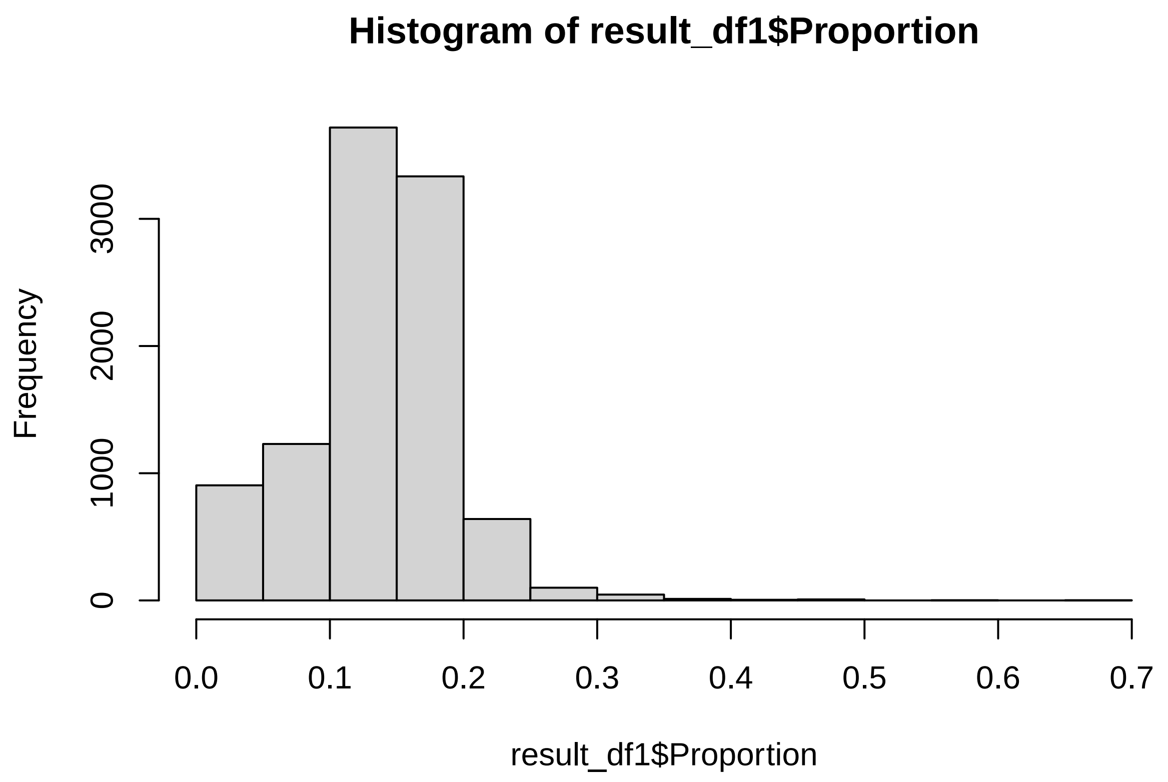
```r
set.seed(14)
# plot the ATT's against proportion of covariates
# Convert matrix to long-format data frame
result_df1 <- as.data.frame(result_matrix1)

#take subsample for better plotting
# Randomly subsample the data frame
subsample_df1 <- result_df1[sample(nrow(result_df1), 1000), ]

# scatterplot
ggplot(subsample_df1, aes(proportion_true, ATT)) +
  geom_point()+
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(title = "Proportion of covariates vs ATT", x = "Proportion of covariates above 0.1 threshold", y
  theme_minimal()
```



```r
#count number of simulations where proportion was higher

#find mean proportion
meanprop1 <- mean(result_df1$Proportion)

#filter for higher than mean proportion
higherprop1 <- result_df1 %>%
  filter(Proportion > meanprop)

#count number of simulations
nrow(higherprop1)
```

```
## [1] 6674
```

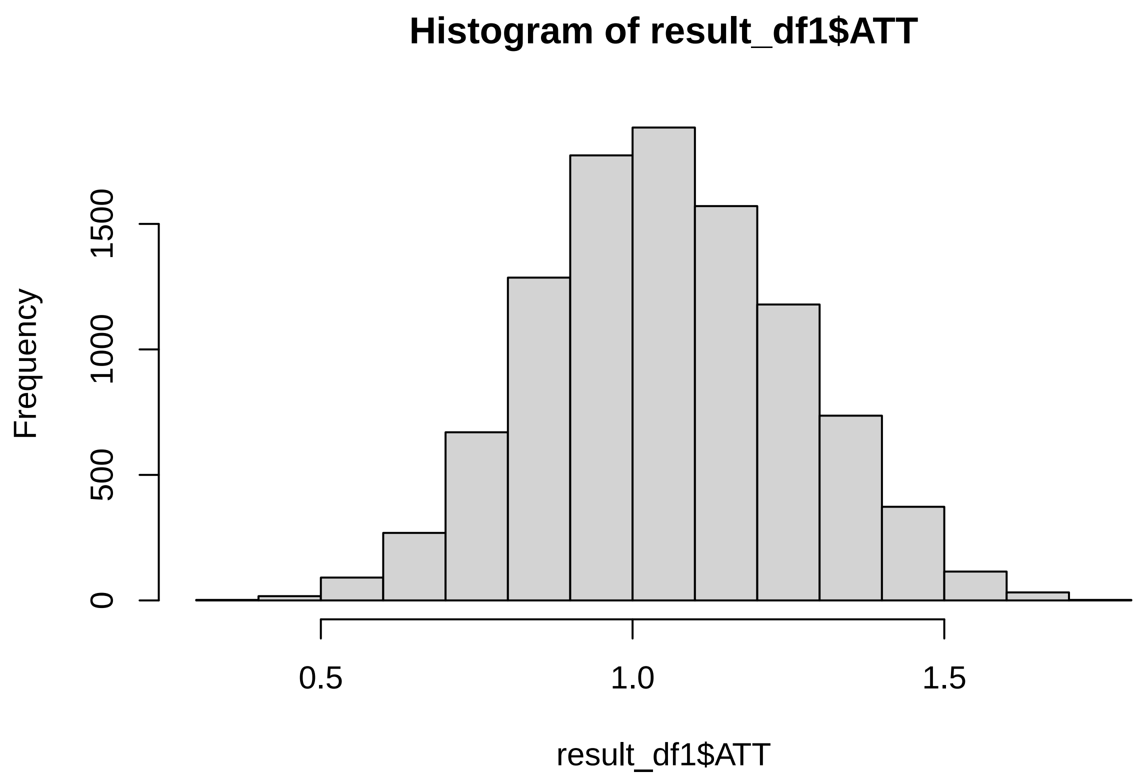
```r
#histogram of proportion true
hist(result_df1$Proportion)
```



```
#examine the higher proportion subset
summary(higherprop1)
```

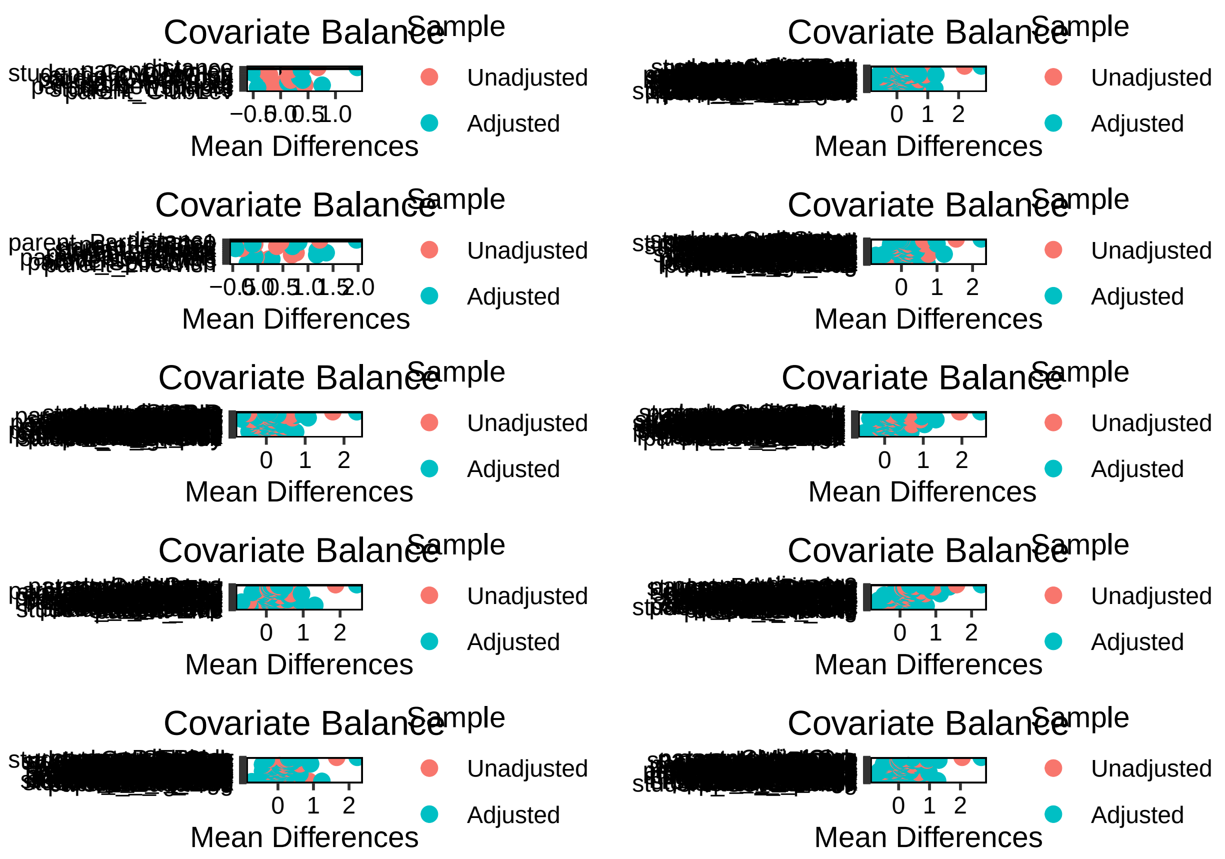
```
##       ATT            Proportion      PercentImproved
##  Min.   :0.3941   Min.   :0.1220   Min.   :  -Inf
##  1st Qu.:0.8914   1st Qu.:0.1429   1st Qu.: 80.38
##  Median :1.0211   Median :0.1596   Median : 99.78
##  Mean   :1.0263   Mean   :0.1669   Mean   :   NaN
##  3rd Qu.:1.1587   3rd Qu.:0.1818   3rd Qu.:119.57
##  Max.   :1.7460   Max.   :0.6667   Max.   :   Inf
##                                    NA's   :1
```

```
#histogram of ATT
hist(result_df1$ATT)
```

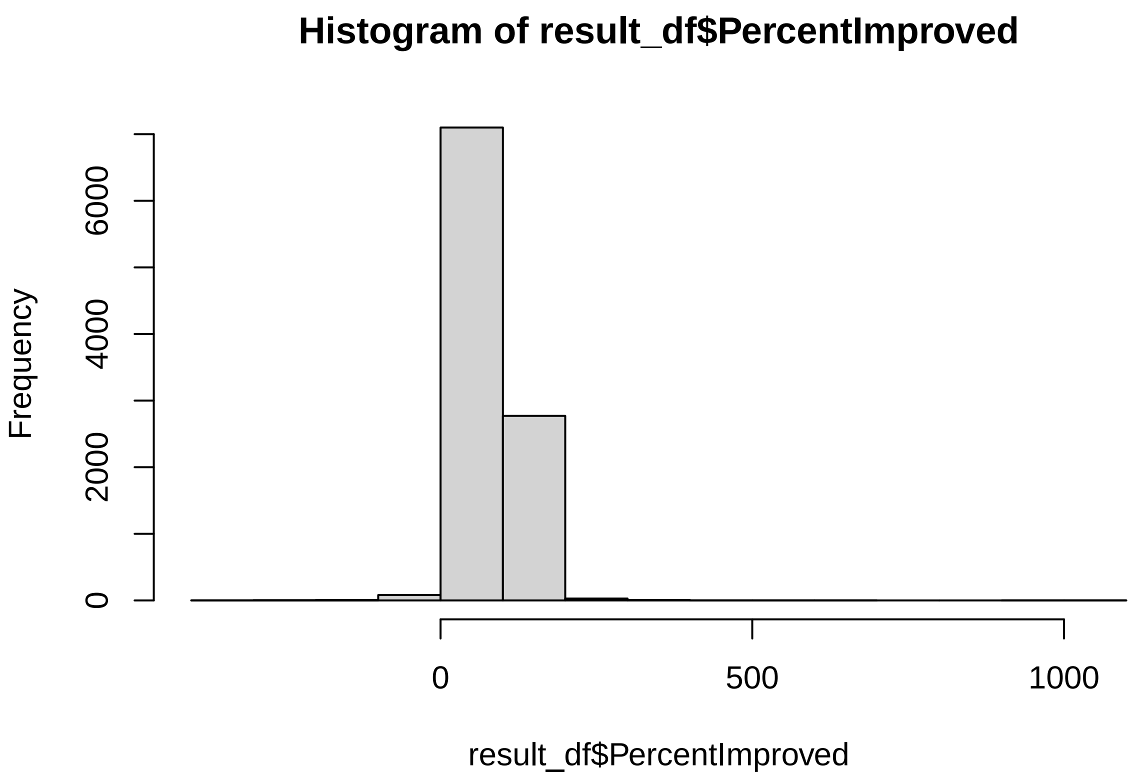


```r
#randomly choose 10 balance plots
set.seed(14)

#empty list of love plots
match_list1 <- list()

# Set up loop to iterate 10 times
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)) %>%
    slice_sample(n=1000)

  # Step 3: Calculate ATT
  match_att <- matchit(as.formula(paste("college ~", paste(random_covariates, collapse = "+"))),
                       data = df,
                       method = "nearest",
                       distance = "glm",
                       link = "logit",
                       discard = "control",
                       replace = FALSE,
                       ratio = 2)
  # Store the results in the result matrix
  match_list1[[i]] <- love.plot(match_att)
}
```

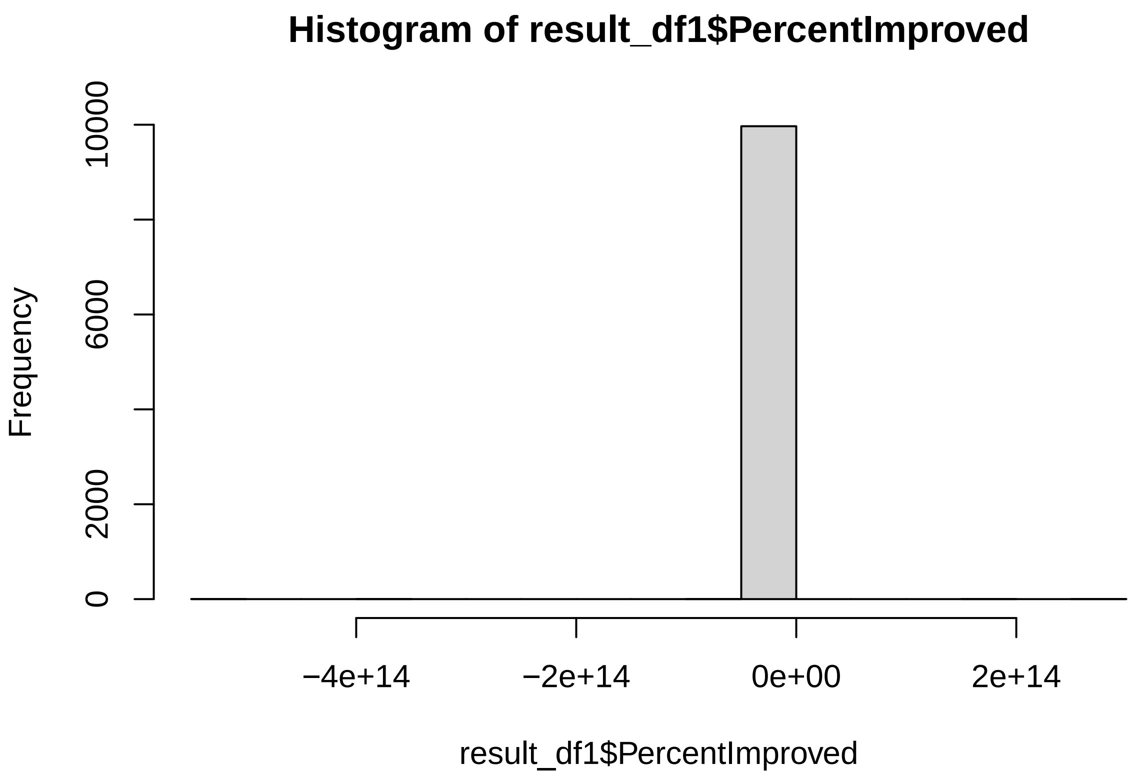
```
# Arrange the grobs using grid.arrange
grid.arrange(grobs = match_list1, ncol = 2)
```



```
# Visualization for distributions of percent improvement
# old distribution of percent improvement
hist(result_df$PercentImproved)
```



```
# new distribution of percent improvement
hist(result_df1$PercentImproved)
```



Questions

1. **Does your alternative matching method have more runs with higher proportions of balanced covariates?** My alternative k-nearest neighbors matching method has less runs with higher proportions of balanced covariates; the initial matching method had 5790 runs versus the KNN method had 5548.
2. **Use a visualization to examine the change in the distribution of the percent improvement in balance in propensity score matching vs. the distribution of the percent improvement in balance in your new method. Which did better? Analyze the results in 1-2 sentences.** The distribution of the KNN method is centered at a higher mean than the original matching method improvement percents, suggesting that KNN offers better mean balancing improvements than propensity score matching.

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

Discussion Questions

1. **Why might it be a good idea to do matching even if we have a randomized or as-if-random design?** Matching reduces the bias caused by imbalanced covariates across treatment and control groups - there might still be enough of differences in means in unmatched samples to bias the estimates of the ITE/ATT/ATE, etc. It also provides a way to conduct sensitivity analyses so unobserved confounding can be somewhat quantified by comparing matched and unmatched runs. Random or as-if random designs don't guarantee that individual covariates are also exactly randomly distributed.
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, given that overfitting is a big downside to logistic regression, it would be potentially helpful to use other ML algorithms to estimate propensity scores so that individuals who are matched on propensity scores aren't being mistakenly matched based on noise or idiosyncratic aspects of the covariates.