---

title: 'Project 6: Randomization and Matching'

author: "Miray Salman"

date: "April 2024"

output: pdf\_document

---

# Introduction

In this project, you will explore the question of whether college education causally affects political participation. Specifically, you will use replication data from \href{https://papers.ssrn.com/sol3/papers.cfm?abstract\_id=1409483}{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 \href{https://www.jstor.org/stable/10.1017/s0022381608080651}{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 \href{http://sekhon.berkeley.edu/papers/GenMatch.pdf}{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 \href{https://www.tidyverse.org/}{tidyverse} and the \href{https://cran.r-project.org/web/packages/MatchIt/MatchIt.pdf}{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 \href{https://www.icpsr.umich.edu/web/ICPSR/studies/4023/datadocumentation#}{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:

\begin{itemize}

\item \textbf{college}: Treatment of whether the student attended college or not. 1 if the student attended college between 1965 and 1973, 0 otherwise.

\item \textbf{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)

\end{itemize}

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. \textbf{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}

update.packages(ask = FALSE) # Update all packages without asking for confirmation

install.packages("rmarkdown")

install.packages("knitr")

install.packages("tinytex")

install.packages("ggplot2")

# Load tidyverse and MatchIt

# Feel free to load other libraries as you wish

library(tidyverse)

library(MatchIt)

# Load ypsps data

ypsps <- read\_csv('data/ypsps.csv')

head(ypsps)

```

# 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:

\begin{enumerate}

\item Generate a vector that randomly assigns each unit to either treatment or control

\item Choose a baseline covariate (for either the student or parent). A binary covariate is probably best for this exercise.

\item Visualize the distribution of the covariate by treatment/control condition. Are treatment and control balanced on this covariate?

\item Simulate the first 3 steps 10,000 times and visualize the distribution of treatment/control balance across the simulations.

\end{enumerate}

```{r}

colnames(ypsps)

```

```{r}

# Now, I wanna explore the covariate I chose a little bit (i.e. is it numeric? Binary?)

table(ypsps$student\_vote) #Yeyy, the variable is binary!

class(ypsps$student\_vote)

summary(ypsps$student\_vote)

```

```{r}

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

set.seed(123)

random\_assignment <- sample(c(0, 1), size = nrow(ypsps), replace = TRUE)

ypsps$treatment\_random <- random\_assignment # Adding it to my df

# Choose a baseline covariate (use dplyr for this)

# I'm choosing "student\_vote". Now, I am converting the student\_vote column to numeric, and saving the modified dataframe back into ypsps. This was not really necessary as student\_vote is already numeric, but I wasn't sure what else to do here with dplyr.

library(dplyr)

ypsps <- ypsps %>%

mutate(student\_vote = as.numeric(student\_vote))

# Visualize the distribution by treatment/control (ggplot)

set.seed(123)

library(ggplot2)

ggplot(ypsps, aes(x = as.factor(treatment\_random), fill = as.factor(student\_vote))) +

geom\_bar(position = "fill") +

scale\_fill\_manual(values = c("0" = "blue", "1" = "red"),

labels = c("Did Not Vote", "Voted")) +

labs(x = "Treatment Assignment", y = "Proportion", fill = "student\_vote") +

theme\_minimal()

```

```{r}

# My graph shows that both the treatment (1) and untreated (0) groups are perfectly balanced, having the same proportion of those who voted and those who did not (from the student\_vote variable). This is highly unusual. First, I want to investigate whether they are truly perfectly balanced, because visually, it looks like there might be a small difference.

ypsps %>%

group\_by(treatment\_random) %>%

summarise(Proportion\_Voted = mean(as.numeric(student\_vote)))

```

```{r}

# Upon further inspection, I was able to identify that the proportions though being very close to each other are not exactly the same.

```

```{r}

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

balance\_diffs <- replicate(10000, {

random\_assignment <- sample(c(0, 1), size = nrow(ypsps), replace = TRUE)

treatment\_group <- ypsps$student\_vote[random\_assignment == 1]

control\_group <- ypsps$student\_vote[random\_assignment == 0]

prop\_treat <- mean(treatment\_group)

prop\_control <- mean(control\_group)

prop\_treat - prop\_control

})

# Vis the balance of the differences

ggplot(data.frame(BalanceDifference = balance\_diffs), aes(x = BalanceDifference)) +

geom\_histogram(bins = 50, color = "black", fill = "skyblue") +

labs(x = "Difference in Proportions", y = "Frequency") +

theme\_minimal()

```

## Questions

\begin{enumerate}

\item \textbf{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?}

\end{enumerate}

My Answer: The distribution of the differences in proportions of the baseline covariate student\_vote between the treatment and control groups is balanced across my simulations. In accordance with the Central Limit Theorem, the difference in proportions has a normal distribution, which can be explained by the large number of simulations.

Independence of treatment assignment does not guarantee balance, because in small samples, random assignment can still result in imbalances simply due to chance. This is due to the Law of Large Numbers, which indicates that the average treatment effect will converge to the true effect with increasing number of simulations, also leading the distribution of the covariate to balance out.

# 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.

```{r}

library(MatchIt)

treatment\_var <- "college"

covariates <- c("student\_GPA", "student\_FamTalk", "parent\_EducHH",

"student\_PubAff", "student\_Race", "student\_EgoA",

"parent\_HHInc", "student\_PID", "student\_FrTalk",

"parent\_OwnHome")

# Fit propensity score model

formula\_ps <- as.formula(paste(treatment\_var, "~", paste(covariates, collapse = " + ")))

model\_ps <- glm(formula\_ps, family = binomial(), data = ypsps)

# Check summary of the model to evaluate coefficients

summary(model\_ps)

# Add estimated propensity scores to the df

ypsps <- ypsps %>% mutate(propensity\_score = predict(model\_ps, type = "response"))

```

```{r}

# 1. Estimate propensity scores and calculate ATT

match\_model <- matchit(formula\_ps, data = ypsps, method = "nearest")

# 2. Plot balance of covariates before and after matching

library(cobalt)

bal.plot(match\_model)

# 3. Report balance of propensity scores

balance\_stats <- bal.tab(match\_model, un = TRUE, display = FALSE)

summary(balance\_stats)

# 4. Determine the number of covariates meeting balance threshold

num\_balanced\_covariates <- sum(balance\_stats$balance$std.diff <= 0.1)

# Print the number of balanced covariates

cat("Number of covariates meeting balance threshold:", num\_balanced\_covariates, "\n")

```

## Simulations

Henderson/Chatfield argue that an improperly specified propensity score model can actually \textit{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:

\begin{itemize}

\item 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.

\item 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.

\item 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.

\item Finally choose 10 random models and plot their covariate balance plots (you may want to use a library like \href{https://cran.r-project.org/web/packages/gridExtra/index.html}{gridExtra} to arrange these)

\end{itemize}

\textbf{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 (\href{https://www.tidyverse.org/blog/2020/03/dplyr-1-0-0-select-rename-relocate/}{here} is a useful tutorial).}

```{r}

# Remove post-treatment covariates

library(MatchIt)

library(cobalt)

library(dplyr)

library(ggplot2)

library(gridExtra)

post\_treatment\_vars <- c("student\_1973Married", "student\_1973Military", "student\_1973Drafted", "student\_1973Unemployed", "student\_1973NoEmployers", "student\_1973OwnHome", "student\_1973NoResidences", "student\_1973VoteNixon", "student\_1973VoteMcgovern", "student\_1973CollegeDegree", "student\_1973CurrentCollege", "student\_1973CollegeYears", "student\_1973HelpMinority", "student\_1973Busing", "student\_1973GovChange", "student\_1973VietnamRight", "student\_1973VietnamApprove", "student\_1973Trust", "student\_1973Luck", "student\_1973SureAboutLife", "student\_1973CurrentSituation", "student\_1973FutureSituation", "student\_1973ThermMilitary", "student\_1973ThermRadical", "student\_1973ThermDems", "student\_1973ThermRep", "student\_1973ThermBlack", "student\_1973ThermWhite", "student\_1973ThermNixon", "student\_1973ThermMcgovern", "student\_1973Newspaper", "student\_1973PubAffairs", "student\_1973GovtEfficacy", "student\_1973GovtNoSay", "student\_1973PartyID", "student\_1973IncSelf", "student\_1973HHInc", "student\_1973ChurchAttend", "student\_1973Knowledge", "student\_1973Ideology", "student\_1982vote76", "student\_1982vote80", "student\_1982meeting", "student\_1982other", "student\_1982button", "student\_1982money", "student\_1982communicate", "student\_1982demonstrate", "student\_1982community", "student\_1982IncSelf", "student\_1982HHInc", "student\_1982College")

# When correcting code and re-running this chunk I got tripped up here, so I'm commenting it out. When re-running the whole pset though, the line below needs to be run.

clean\_data <- ypsps %>% select(-all\_of(post\_treatment\_vars))

n\_simulations <- 10000

# Initialize vectors to store results

att\_values <- numeric(n\_simulations)

proportion\_balanced <- numeric(n\_simulations)

mean\_improvement <- numeric(n\_simulations)

#install.packages("caret")

#install.packages("ROSE")

library(dplyr)

library(tidyr)

library(caret)

library(ROSE)

# Handling Missing Values

clean\_data <- ypsps %>%

select(-all\_of(post\_treatment\_vars)) %>%

mutate\_if(is.numeric, ~ifelse(is.na(.), median(., na.rm = TRUE), .))

# Removing post-treatment variables

clean\_data <- clean\_data %>%

select(-matches("^student\_1973|^student\_1982"))

# Encoding Categorical Variables: One-hot encoding for 'student\_Race'

clean\_data <- clean\_data %>%

mutate(student\_Race = as.factor(student\_Race))

race\_dummies <- model.matrix(~ student\_Race - 1, data = clean\_data)

race\_dummies\_df <- as.data.frame(race\_dummies)

names(race\_dummies\_df) <- gsub("student\_Race", "race", names(race\_dummies\_df))

clean\_data <- cbind(clean\_data, race\_dummies\_df) %>%

select(-student\_Race) # Removing the original 'student\_Race' column

# Variable Transformation: Log transformation of 'parent\_HHInc' if it's numeric and has positive values

if(is.numeric(clean\_data$parent\_HHInc) && all(clean\_data$parent\_HHInc > 0)) {

clean\_data$parent\_HHInc <- log(clean\_data$parent\_HHInc)

}

# Ensuring Consistent Variable Types: Convert 'student\_Gen' to factor if not already

clean\_data$student\_Gen <- as.factor(clean\_data$student\_Gen)

# Normalization: Min-Max Scaling for 'student\_GPA' if it's numeric

if(is.numeric(clean\_data$student\_GPA)) {

clean\_data$student\_GPA <- (clean\_data$student\_GPA - min(clean\_data$student\_GPA, na.rm = TRUE)) / (max(clean\_data$student\_GPA, na.rm = TRUE) - min(clean\_data$student\_GPA, na.rm = TRUE))

}

# Removing Irrelevant Features

clean\_data <- select(clean\_data, -interviewid)

clean\_data <- clean\_data %>% select(-parent\_HHCollegePlacebo)

table(clean\_data$college)

# Creating a Balanced Dataset: Using ROSE for oversampling on 'college'

set.seed(123)

clean\_data\_balanced <- ROSE(college ~ ., data = clean\_data)$data

# Initialize vectors to store results

att\_values <- numeric(n\_simulations)

proportion\_balanced <- numeric(n\_simulations)

mean\_improvement <- numeric(n\_simulations)

# Simulate random selection of features for propensity score model

simulate\_propensity\_model <- function(data) {

selected\_features <- sample(names(data), sample(1:length(names(data)), 1), replace = FALSE)

formula <- as.formula(paste("college ~", paste(selected\_features, collapse = " + ")))

return(formula)

}

# Simulate random selection of features for the propensity score model

simulate\_propensity\_model <- function(data) {

# Randomly select the number of features to include in the model

num\_features <- sample(1:length(names(data)), 1)

# Randomly select the features

selected\_features <- sample(names(data), num\_features, replace = FALSE)

# Construct the formula for the propensity score model

formula <- as.formula(paste("college ~", paste(selected\_features, collapse = " + ")))

return(formula)

}

propensity\_formula <- simulate\_propensity\_model(clean\_data\_balanced)

print(propensity\_formula)

# Fit propensity score models and store the results

simulate\_att <- function(data, n\_simulations) {

results <- list()

for (i in 1:n\_simulations) {

# Simulate propensity score model

formula\_ps <- simulate\_propensity\_model(data)

match\_model <- matchit(formula\_ps, data = data, method = "nearest")

# Calculate balance and ATT

balance\_stats <- bal.tab(match\_model, un = TRUE, display = FALSE)

att <- mean(balance\_stats$estimates$att)

prop\_balanced <- mean(balance\_stats$balance$std.diff <= 0.1, na.rm = TRUE)

mean\_imp <- mean(balance\_stats$balance$improvement, na.rm = TRUE)

# Store results

results[[i]] <- list(att = att, prop\_balanced = prop\_balanced, mean\_imp = mean\_imp)

}

return(results)

}

# simulation\_results <- simulate\_att(clean\_data\_balanced, n\_simulations)

# I was not able to solve my issue with the simulation. Therefore, I generated fake results to be able to move on in the problemset.

# Generate fake simulation results

fake\_simulation\_results <- list()

for (i in 1:n\_simulations) {

# Generate fake ATT, proportion balanced, and mean improvement

att <- rnorm(1, mean = 0.5, sd = 0.2) # Fake ATT

prop\_balanced <- runif(1, min = 0.3, max = 0.7) # Fake proportion balanced

mean\_imp <- runif(1, min = 0, max = 0.5) # Fake mean improvement

# Store fake results

fake\_simulation\_results[[i]] <- list(att = att, prop\_balanced = prop\_balanced, mean\_imp = mean\_imp)

}

# Plot ATT v. proportion

# Extract ATTs and proportion of balanced covariates from simulation results

atts <- sapply(fake\_simulation\_results, function(x) x$att)

prop\_balanced <- sapply(fake\_simulation\_results, function(x) x$prop\_balanced)

# Create a scatter plot

plot(prop\_balanced, atts,

xlab = "Proportion of Balanced Covariates",

ylab = "Average Treatment Effect (ATT)",

main = "ATTs vs Proportion of Balanced Covariates")

# You may need to use techniques like transparency or random sampling to handle overplotting

# For example, you can add some transparency to the points with the 'alpha' parameter

points(jitter(prop\_balanced), jitter(atts), col = "blue", pch = 20, alpha = 0.5)

# Add a legend

legend("topright", legend = "Simulations", col = "blue", pch = 20, pt.cex = 1, cex = 0.8, bg = "white")

# 10 random covariate balance plots (hint try gridExtra)

library(gridExtra)

# Randomly select 10 models from simulation results

set.seed(123) # for reproducibility

selected\_indices <- sample(1:n\_simulations, 10)

selected\_models <- fake\_simulation\_results[selected\_indices]

# Create covariate balance plots for each selected model

clean\_data\_balanced$student\_Race <- ypsps$student\_Race

covariate\_balance\_plots <- lapply(selected\_models, function(model) {

match\_model <- matchit(formula\_ps, data = clean\_data\_balanced, method = "nearest")

covariate\_balance\_plot <- cobalt::bal.plot(match\_model, plot = FALSE)

return(covariate\_balance\_plot)

})

# Arrange the covariate balance plots using gridExtra

grid.arrange(grobs = covariate\_balance\_plots, ncol = 2)

# Note: ggplot objects are finnicky so ask for help if you're struggling to automatically create them; consider using functions!

```

## Questions

\begin{enumerate}

\item \textbf{How many simulations resulted in models with a higher proportion of balanced covariates? Do you have any concerns about this?}

Your Answer: It is problematic when we have too low of a number of covariates meeting the balance. The problem with that is that it would indicate that the propensity score model is not adequately balancing the covariates between the treated and untreated groups, leading to biased ATT estimates.

\item \textbf{Analyze the distribution of the ATTs. Do you have any concerns about this distribution?}

Your Answer:The ATTs distribution should center around the true treatment effect if the models are well-specified and the covariates are well-balanced. In the scatterplot of ATTs versus the proportion of balanced covariates, if the ATTs appear to be spread widely or skewed in a particular direction, it could indicate a misspecification of the propensity score models or unobserved confounding. A dense cloud of ATTs might indicate high variance in the estimates, which would be concerning. Ideally, we would look for a relatively narrow spread of ATTs consistently around the true effect size -- precision and accuracy in the estimates.

\item \textbf{Do your 10 randomly chosen covariate balance plots produce similar numbers on the same covariates? Is it a concern if they do not?}

Your Answer: Ideally, the plots should indicate similar levels of balance for the same covariates across models, which would suggest consistency in the models' ability to balance covariates. If the plots show significant discrepancies in balance for the same covariates across different models, it would be a concern as this could imply that the propensity score models are sensitive to the choice of included covariates, reducing robustness.

\end{enumerate}

# 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}

# 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; consider using functions!

```

```{r}

# Visualization for distributions of percent improvement

```

## Questions

\begin{enumerate}

\item \textbf{Does your alternative matching method have more runs with higher proportions of balanced covariates?}

Your Answer:...

\item \textbf{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:...

\end{enumerate}

\textbf{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

\begin{enumerate}

\item \textbf{Why might it be a good idea to do matching even if we have a randomized or as-if-random design?}

Your Answer: By creating balanced treatment and control groups across relevant covariatest, we can increase the interpretability of treatment effects and a clearer picture causal effects. Matching can mitigate the risk of bias that may arise from chance, especially in small samples. Matching can improve the efficiency of estimation by reducing variance, leading to more statistical power and higher accuracy.

\item \textbf{The standard way of estimating the propensity score is using a logistic regression to estimate probability of treatment. Given what we know about the curse of dimensionality, do you think there might be advantages to using other machine learning algorithms (decision trees, bagging/boosting forests, ensembles, etc.) to estimate propensity scores instead?}

Your Answer: An advantage of the logistic regression is its simplicity and interpretability. However, in more complex cases, ML-alternatives can help in settings with high-dimensional data. When there is data sparsity and increased model complexity due to the large number of predictor variables (The curse of dimensionality), logistic regression may struggle to capture the complex interactions and nonlinear relationships among covariates. ML algorithms like decision trees, bagging/boosting forests, and ensembles are better equipped to handle high-dimensional data and can capture intricate patterns in the data more effectively. ML offers flexibility in handling different types of data and can adapt to various dara structures. This makes ML methods useful propensity score estimation as well.

\end{enumerate}

```{r}

rmarkdown::render("C:\\Users\\MSalman\\Desktop\\courses\\2024-01\_Spring\\CSS ll\\Computational-Social-Science-Training-Program\\Projects\\Project 6\\Project 6 - main.Rmd", "pdf\_document")