Computational Social Science: Matching Methods

Your Name Here

MM/DD/YYYY

```
set.seed(13)
library(dplyr)
library(ggplot2)
library(purrr)
library(MatchIt)
```

As we saw in last week's lab, an important advantage of randomized experiments are that they allow researchers to ensure independence between the exposure variable and other covariates, or rather that treatment and control groups have similar covariate distributions and differ only randomly.

The same cannot be said of observational studies, *no matter how large the sample size*. Thus, researchers often use a variety of matching methods to try to replicate this matching of covariate distributions between exposure groups.

In this lab we will consider some of these matching methods. Note that these methods are all implemented in the analysis stage (i.e. after the study has already been completed), and are distinct from (though may be similar to) methods of conducting studies which are matched from the outset.

Simulation

We will again use the simulated example from last week assessing the effectiveness of AspiTyleCedrin at treating migraines. As a reminder, this dataset contained the following variables:

- `A`: Treatment variable indicating whether the individual i took AspiTyleCedrin ($A_i = 1$) or not ($A_i = 0$).
- `Y_obs`: Outcome variable indicating whether the individual experienced a migraine ($Y_{i_{obs}} = 1$) or not ($Y_{i_{obs}} = 0$).
- `W1`: Variable representing sex assigned at birth, with $W1 = 0$ indicating AMAB (assigned male at birth), $W1 = 1$ indicating AFAB (assigned female at birth), and $W1 = 2$ indicating an X on the birth certificate, possibly representing an intersex individual or left blank.
- `W2`: Variable representing simplified racial category, with $W2 = 0$ indicating White, $W2 = 1$ indicating Black or African American, $W2 = 2$ indicating Non-White Hispanic or Latinx, $W2 = 3$ indicating American Indian or Alaska Native, $W2 = 4$ indicating Asian, and $W2 = 5$ indicating Native Hawaiian or Other Pacific Islander.

Say that the creators of AspiTyleCedrin are also concerned that it may be less effective among individuals with a higher Body Mass Index (BMI). To simulate this, we will modify the code we used to create the original AspiTyleCedrin dataset to also include the variable `W3` representing an individual's BMI. (We'll also modify the observed outcomes to be confounded by this variable.)

```r
n = 1e6 # Number of individuals

# NOTE: Again, don't worry too much about how we're creating this dataset,
# this is just an example.

# W3 scaled to have mu=24 and sigma=4 a la
# https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4789291/
# where k = mu^2/sigma^2 and theta = sigma^2/mu

# Also make treatment less likely so that there are more controls,
# and add ID column
df <- data.frame(ID = seq.int(n),
                 W1 = sample(0:2, size = n, replace = TRUE,
                             prob = c(0.49,0.50,0.01)),
                 W2 = sample(0:5, size = n, replace = TRUE,
                             prob = c(0.60,0.13,0.19,0.06, 0.015, 0.005)),
                 W3 = rgamma(n,
                    shape = 36,
                    scale = (2/3)))
df <- df %>%
  mutate(W3 = W3 + 3*(W1 == 1)+ 4*(W2==2) + 3*(W2==3) + 1*(W2==4) + (-1)*(W2 == 5),
         A = as.numeric(rbernoulli(n,
                                   p = (0.16 + 0.07*(W1 > 0) + 0.21*(W2 == 0)))),
         Y_0 = as.numeric(rbernoulli(n,
                                     p = (0.87 + 0.035*(W1 > 0) + 0.05*(W2 > 0)) +
                                       abs((W3 - 22)/100))),
         Y_1 = as.numeric(rbernoulli(n,
                                     p = (0.34 + 0.035*(W1 > 0) + 0.3*(W2 > 0)) +
                                       abs((W3 - 22)/100) + 0.2*(W3 > 30))),
         ITE = Y_1 - Y_0,
         Y_obs = as.numeric((A & Y_1) | (!A & Y_0)))

ATE.true <- mean(df$ITE)
df.a1 <- df %>% filter(A == 1)
ATT.true <- mean(df.a1$ITE)

df <- df %>% select(-Y_0, -Y_1, -ITE)
df.a1 <- df.a1 %>% select(-Y_0, -Y_1, -ITE)
df.a0 <- df %>% filter(A == 0)

head(df)
```

```
##   ID W1 W2       W3 A Y_obs
## 1  1  0  2 30.24495 0     1
## 2  2  1  0 28.31968 1     0
## 3  3  1  1 29.55123 1     1
## 4  4  1  0 26.69831 0     1
## 5  5  0  2 22.87836 0     0
## 6  6  1  0 29.31223 0     1
```

```r
summary(df)
```

```
##        ID                W1               W2               W3
```

```
##  Min.   :      1   Min.   :0.0000   Min.   :0.0000   Min.   :10.49
##  1st Qu.: 250001   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:23.25
##  Median : 500000   Median :1.0000   Median :0.0000   Median :26.24
##  Mean   : 500000   Mean   :0.5192   Mean   :0.7736   Mean   :26.45
##  3rd Qu.: 750000   3rd Qu.:1.0000   3rd Qu.:2.0000   3rd Qu.:29.42
##  Max.   :1000000   Max.   :2.0000   Max.   :5.0000   Max.   :53.65
##        A              Y_obs
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:1.0000
##  Median :0.0000   Median :1.0000
##  Mean   :0.3223   Mean   :0.8133
##  3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000
```

Let's take a look at the covariate distributions, comparing those that did and did not take AspiTyleCedrin:

Sex Assigned at Birth (SAAB)

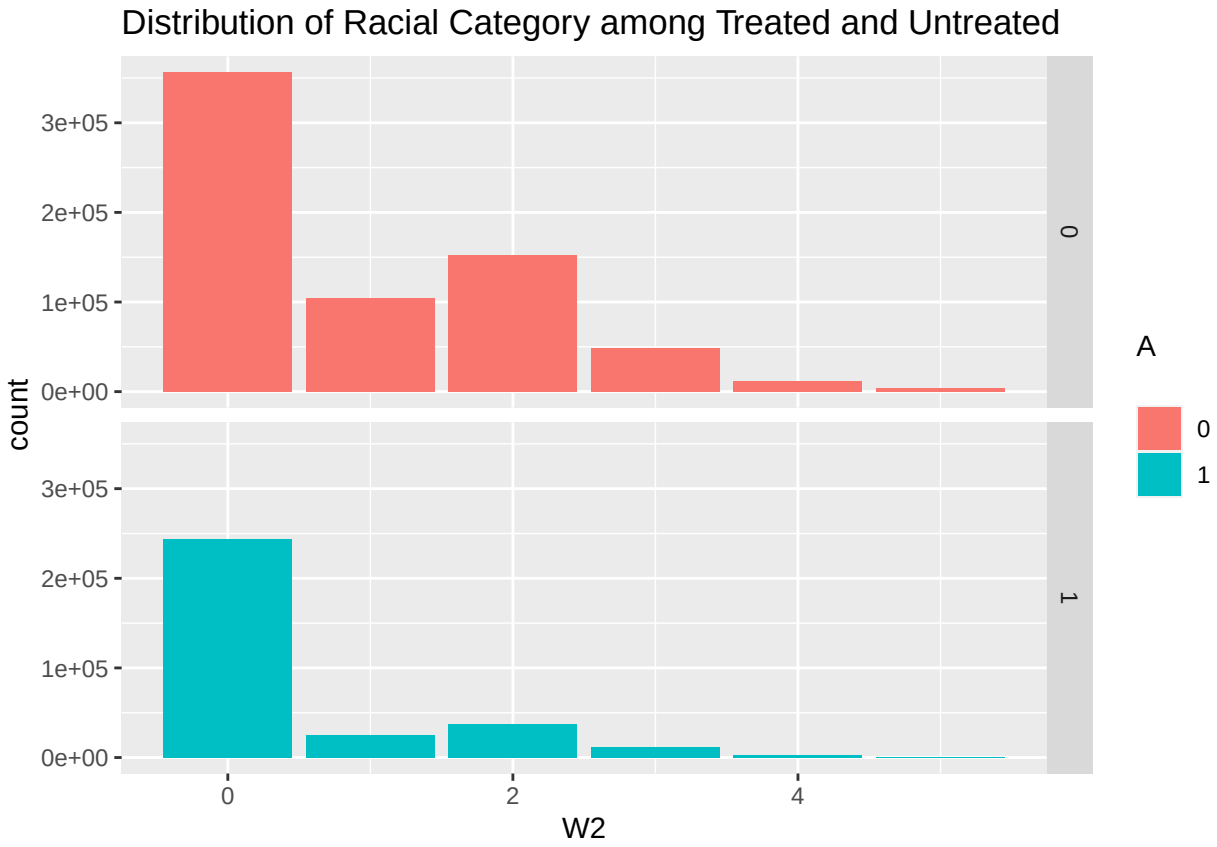
```
ggplot(df, aes(x = W1, fill = factor(A))) +
  geom_bar() +
  facet_grid(A~.) +
  labs(title = "Distribution of Sex Assigned at Birth among Treated and Untreated", fill = "A\n")
```

```
chisq.test(table(df$A, df$W1))
```

```
##
##  Pearson's Chi-squared test
##
## data:  table(df$A, df$W1)
## X-squared = 5604.5, df = 2, p-value < 2.2e-16
```

The bar plot above clearly shows a difference in the distribution of SAAB among the two groups, and this is confirmed by the very small p-value from the χ^2 test.

```
ggplot(df, aes(x = W2, fill = factor(A))) +
  geom_bar() +
  facet_grid(A~.) +
  labs(title = "Distribution of Racial Category among Treated and Untreated", fill = "A\n")
```

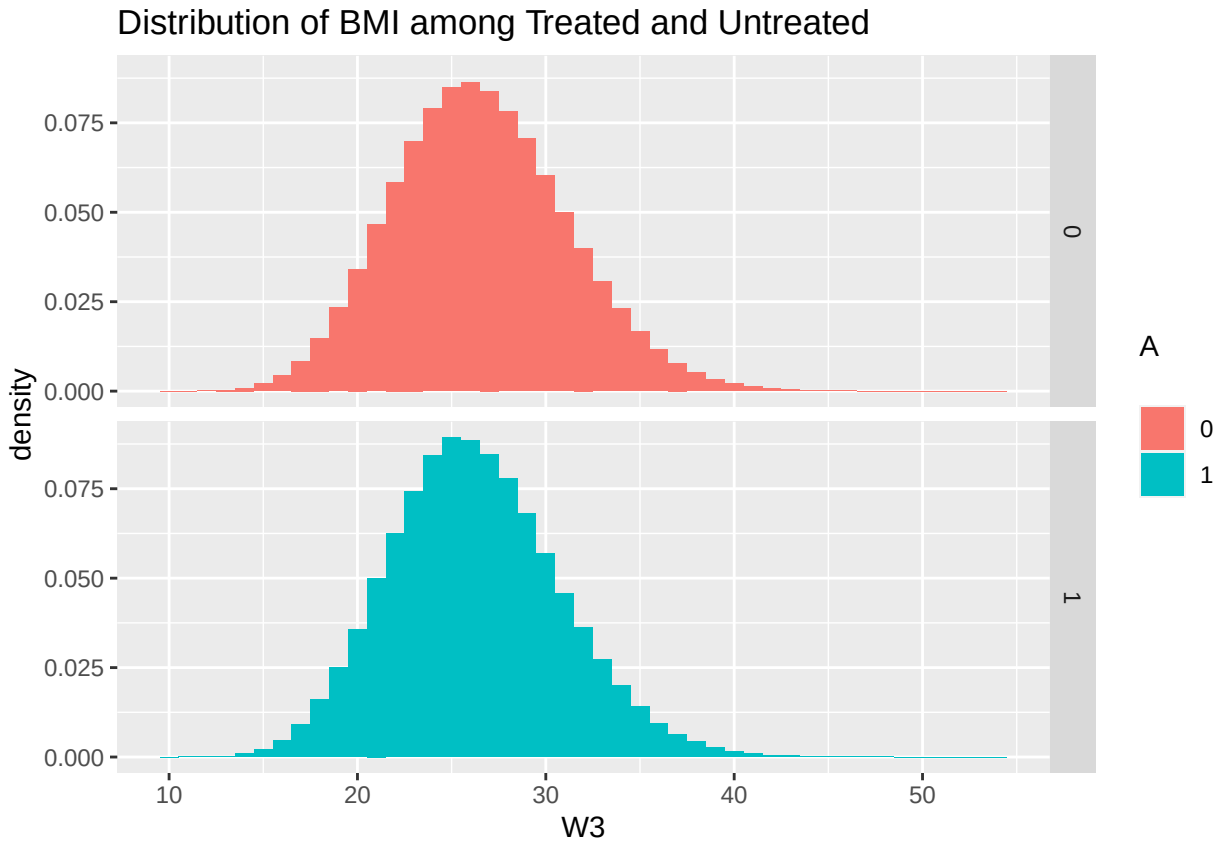



```
chisq.test(table(df$A, df$W2))
```

```
##
##  Pearson's Chi-squared test
##
## data:  table(df$A, df$W2)
## X-squared = 48752, df = 5, p-value < 2.2e-16
```

The bar plot above again shows a difference in the distribution of simplified racial category among the two groups, and this is again confirmed by the very small p-value from the χ^2 test.

```
ggplot(df, aes(x = W3, fill = factor(A))) +
  geom_histogram(binwidth = 1, aes(y = ..density..)) +
  facet_grid(A~.) +
  labs(title = "Distribution of BMI among Treated and Untreated", fill = "A\n")
```




```
t.test(W3 ~ A, data = df)
```

```
##
##  Welch Two Sample t-test
##
## data:  W3 by A
## t = 31.935, df = 648377, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.2909256 0.3289706
## sample estimates:
## mean in group 0 mean in group 1
##        26.55384        26.24389
```

While it may be difficult to determine from the histogram above how the distribution of BMI differs among the two groups, the very small p-value from the t-test shows evidence of a clear difference.

Thus we can see the need to improve the matching of these covariate distributions.

Matching Considerations

There are a number of factors to consider when choosing a matching method, including the following:

- Distance Metric
- Greediness
- Control:Treatment Ratio
- Caliper Width
- Replacement

Distance Metric

The goal of matching is to match together each treatment unit (in our case, each individual who took AspiTyleCedrin, `A == 1`) to one or more "control" unit (in our case, individuals who did not take AspiTyleCedrin, `A == 0`) based on baseline covariates (in our case, `W1, W2, W3`). Note that conceptually, this means we are trying to find the control unit(s) that most closely resemble the counterfactual for each treatment unit.

Exact Matching

Ideally, we would like to find the control unit(s) which have all identical covariate values. This is called "exact matching".

For our dataset, this would mean each individual who took AspiTyleCedrin (`A == 1`) would be matched with individual(s) who did not take AspiTyleCedrin (`A == 0`) with the *exact* same SAAB (`W1`), racial category (`W2`), and BMI (`W3`).

In other words, the exact distance between two points X_i, X_j, where $X_i = \{W1_i, W2_i, W3_i\}$ and $X_j = \{W1_j, W2_j, W3_j\}$ is defined as:

$$\text{Distance}(X_i, X_j) = \begin{cases} 0, & \text{if } X_i = X_j \\ \infty, & \text{if } X_i \neq X_j \end{cases}$$

Question 1: The data frame `df.a0` contains all the individuals that did not take AspiTyleCedrin, and the data frame `df.a1` contains all those who did. In the `R` code chunk below, use the first ten rows of `df.a0` and the first five rows of `df.a1` to find the exact distance of the first ten individuals who did not take AspiTyleCedrin from *each* of the first five individuals who did. (Hint: How many comparisons should you be making?)

```
df.a0.small <- df.a0[1:10,]
df.a1.small <- df.a1[1:5,]
cols <- c("W1", "W2", "W3")

dist.exact <- function(x,y) {
  ifelse(all(x == y), 0, Inf)
}

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.ex <- calculate.dist(df.a1.small[, cols], df.a0.small[, cols], dist.exact)
dists.ex
```

```
##      1   4   5   6   7   8   9  11  12  13
## 2  Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## 3  Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## 10 Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## 18 Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## 22 Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
```

While exact matching is ideal, it is not always possible, such as in the case of continuous variables, such as our BMI variable, `W3`.

Question 2: Explain why matching on a continuous variable would likely be impossible.

The probability of any exact value of a continuous variable is by definition zero, so even taking rounding into account, the probability of finding exact matches on a continuous variable is very low.

Question 3: Modify your code above to only check the distance for `W1` and `W2` values.

```
dists.ex.lim <- calculate.dist(df.a1.small[, cols[1:2]], df.a0.small[, cols[1:2]], dist.exact)
dists.ex.lim
```

```
##      1   4   5   6   7   8   9  11  12  13
## 2  Inf   0 Inf   0 Inf Inf Inf Inf Inf Inf
## 3  Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## 10 Inf   0 Inf   0 Inf Inf Inf Inf Inf Inf
## 18 Inf Inf Inf Inf   0 Inf Inf   0 Inf   0
## 22 Inf Inf Inf Inf   0 Inf Inf   0 Inf   0
```

Since exact matching is not always possible, there are a variety of alternative distance metrics which may be used to determine how similar a potential match is. A few of these methods are discussed below.

Mahalanobis Distance Matching

The Mahalanobis distance in general is a "multi-dimensional generalization of the idea of measuring how many standard deviations away [some point] P is from the mean of [some distribution] D." However, in the context of matching, the Mahalanobis distance measures this distance between the two points X_i, X_j rather than that between one point and a distribution.

Mathematically, this version of the Mahalanobis distance is defined as follows:

$$\text{Distance}(X_i, X_j) = \sqrt{(X_i - X_j)^T S^{-1} (X_i - X_j)}$$

where S^{-1} is the covariance matrix of X_i and X_j.

Question 4: Using the `cov()` function to find the covariance matrix of $\{W1, W2, W3\}$ from the *whole dataset*, modify your code from **Question 1** to instead find the Mahalanobis distance of the first ten individuals who did not take AspiTyleCedrin from *each* of the first five individuals who did. (Hint: The `t()` function will transpose a vector or matrix, and matrix multiplication uses the `%*%` character, not `*`)

```r
cov.df <- cov(df[,cols])

dist.mahalanobis <- function(x,y) {
  diff <- (x - y)
  sqrt( t(diff) %*% cov.df %*% (diff) )
}

dists.ma <- calculate.dist(df.a1.small[, cols], df.a0.small[, cols], dist.mahalanobis)
dists.ma
```

```
##            1         4         5         6         7         8         9
## 2   9.508597  7.424691 24.582359  4.545178 18.907663  6.586775  3.937466
## 3   3.525620 13.401316 30.442592  1.742678 24.861811  1.231547  2.928815
## 10 45.487268 28.759145 11.905722 40.729014 17.289715 42.448770 39.509381
## 18 44.765849 28.046054 11.182146 40.014620 16.565191 41.727128 38.787500
## 22 24.519680  7.770400  9.538621 19.730716  3.721855 21.490140 18.562836
##           11        12        13
## 2  17.784291 29.861838 27.305899
## 3  23.739227 35.776452 33.255658
## 10 18.413063  6.458813  8.896171
## 18 17.688971  5.725968  8.164965
## 22  2.598075 14.695956 12.122081
```

Propensity Score Matching

The propensity score of an individual is a measure of the probabilty of that individual receiving the treatment based upon the baseline covariates. That is, given a set of covariate values ($\{W1_i, W2_i, W3_i\}$ in our case), the propensity score represents the estimated probability of treatment ($A_i = 1$). The propensity score is often estimated using a logit model and is therefore defined as follows:

$\pi_i = P(A_i = 1|X_i) = \frac{1}{1+e^{-X_i\beta}}$

We can estimate these propensity scores using logistic regression, by regressing the treatment A on the baseline covariates X, like so:

```r
model.ps <- glm(A ~ W1 + W2 + W3, family = binomial(), data = df)
summary(model.ps)
```

```
##
## Call:
## glm(formula = A ~ W1 + W2 + W3, family = binomial(), data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1889  -0.9304  -0.7309   1.3068   2.4103
##
## Coefficients:
##               Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -0.6411946  0.0131967  -48.587   <2e-16 ***
## W1           0.3117086  0.0044391   70.219   <2e-16 ***
## W2          -0.4445095  0.0024568 -180.933   <2e-16 ***
## W3           0.0011424  0.0005288    2.161   0.0307 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1257247  on 999999  degrees of freedom
## Residual deviance: 1210954  on 999996  degrees of freedom
## AIC: 1210962
##
## Number of Fisher Scoring iterations: 4
```

We can then use this model and the `predict()` function to add all of the estimated propensity scores for each data point in `df`:

```
df <- df %>% mutate(prop_score = predict(model.ps))

# Also update the subsetted datasets
df.a0 <- df %>% filter(A == 0)
df.a1 <- df %>% filter(A == 1)
df.a0.small <- df.a0[1:10,]
df.a1.small <- df.a1[1:5,]
```

Propensity score *matching* uses the absolute difference between two propensity scores as its distance metric, or rather:

$$\text{Distance}(X_i, X_j) = |\pi_i - \pi_j|$$

Question 5: Again modify your previous code to find the propensity score distance of the first ten individuals who did not take AspiTyleCedrin from *each* of the first five individuals who did.

```
dist.prop.score <- function(x,y) {
  abs(x-y)
}

dists.ps <- calculate.dist(as.matrix(df.a1.small[, "prop_score"]),
                           as.matrix(df.a0.small[, "prop_score"]),
                           dist.prop.score)
dists.ps
```

```
##            1           4         5           6           7         8         9
## 2  1.1985282 0.001852328 1.2069441 0.001133941 0.3163847999 1.1992872 1.2000215
## 3  0.7554256 0.441250207 0.7638416 0.444236475 0.1267177346 0.7561846 0.7569190
## 10 1.1895009 0.007174892 1.1979169 0.010161161 0.3073575798 1.1902599 1.1909943
## 18 0.8780106 0.318665191 0.8864266 0.321651459 0.0041327189 0.8787696 0.8795040
## 22 0.8830719 0.313603936 0.8914878 0.316590204 0.0009285364 0.8838309 0.8845653
##              11        12          13
## 2  0.3161044369 0.7636966 0.318480506
## 3  0.1269980976 0.3205940 0.124622028
## 10 0.3070772167 0.7546693 0.309453286
## 18 0.0044130820 0.4431790 0.002037013
## 22 0.0006481733 0.4482403 0.003024242
```

Greediness

Once deciding upon a distance metric, we must also choose a matching algorithm. That is, how shall the computed distances be used to determine a match? The various matching algorithms fall into two general categories: "greedy" and optimal.

"Greedy" Matching

Greedy algorithms in general are used to reduce larger problems to smaller ones by taking the best option at the time and repeating, while never returning to earlier choices to make changes. In the context of matching, this means that a greedy matching algorithm chooses the best single match first and removes that chosen match. It then repeats this process by choosing the best single match still remaining and removing that match, and so on.

There are a number of different ways to decide which match to deem "best", including but not limited to:

- Choose the treatment participant with the highest propensity score first, and match it to the "control" participant with the closest propensity score (shortest propensity score distance).
- Same as above but start with lowest rather than highest propensity score.
- The best overall match (minimum of all match distances) in the entire dataset.
- Random selection.

Some of these will be discussed in greater detail below.

Question 6: Using the propensity score distances you made in Question 5, find the greedy matching of this subset using highest to lowest propensity score. Report the IDs of both elements of each matched pair. (Hint: You may find the `which.min()` and `which.max()` functions helpful)

```
treat <- c()
control <- c()
df.a1.small.copy <- as.data.frame(df.a1.small)
dists.ps.copy <- as.data.frame(dists.ps)

for(i in 1:nrow(df.a1.small)) {
  max.treat <- which.max(df.a1.small.copy$prop_score)
  treat[i] <- names(max.treat)
  df.a1.small.copy <- df.a1.small.copy %>% slice(-max.treat)

  match.control <- which.min(dists.ps.copy[max.treat,])
  control[i] <- names(match.control)
  dists.ps.copy <- dists.ps.copy %>%
      select(-match.control) %>%
      slice(-max.treat)
}
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(match.control)` instead of `match.control` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
treat
```

```
## [1] "2"  "10" "22" "18" "3"
```

```
control
```

```
## [1] "6"  "4"  "11" "13" "7"
```

Question 7: Same as Question 6, but now find the greedy matching of this subset using lowest to highest propensity score.

```
treat <- c()
control <- c()
df.a1.small.copy <- as.data.frame(df.a1.small)
dists.ps.copy <- as.data.frame(dists.ps)

for(i in 1:nrow(df.a1.small)) {
  min.treat <- which.min(df.a1.small.copy$prop_score)
  treat[i] <- names(min.treat)
  df.a1.small.copy <- df.a1.small.copy %>% slice(-min.treat)

  match.control <- which.min(dists.ps.copy[min.treat,])
  control[i] <- names(match.control)
  dists.ps.copy <- dists.ps.copy %>%
      select(-match.control) %>%
      slice(-min.treat)
}

treat
```

```
## [1] "3"  "18" "22" "10" "2"
```

```
control
```

```
## [1] "13" "7"  "11" "4"  "6"
```

Question 8: Same as in the previous two problems, but now find the greedy matching of this subset using best overall match.

```
treat <- c()
control <- c()
dists.ps.copy <- as.data.frame(dists.ps)

for(i in 1:nrow(df.a1.small)) {
  best <- which(dists.ps.copy == min(dists.ps.copy), arr.ind = TRUE)
  treat[i] <- rownames(dists.ps.copy)[best[1]]
  control[i] <- colnames(dists.ps.copy)[best[2]]

  dists.ps.copy <- dists.ps.copy %>%
    slice(-(best[1])) %>%
    select(-(best[2]))
}

treat
```

```
## [1] "22" "2"  "18" "10" "3"
```

```
control
```

```
## [1] "11" "6"  "13" "4"  "7"
```

Question 9: Were there any differences in the matchings you found in the previous three problems?

> Your answer here.

Optimal Matching

Optimal matching, as the name implies, seeks to find an optimal matching scheme in which the overall match difference is minimized. For example, if we were to add the distances of all match pairs chosen, an optimal matching would seek the set of match pairs which produces the smallest sum. A disadvantage of optimal matching is that it can be computationally intensive without providing sufficient improvements over greedy matching.

Control:Treatment Ratio

You may have noticed that in the previous examples we only selected one "control" individual for each treatment individual, often called $1:1$ matching. However, in some cases we may prefer to match more than one control to each treatment, often called $k:1$ matching, where k is the number of control individuals desired per treatment individual.

Question 10: Modify your code from Question 6 to perform a 2:1 matching rather than 1:1. That is, find the two best "control" matches for each treatment individual, using highest to lowest propensity score.

```
treat <- c()
control.1 <- c()
control.2 <- c()
df.a1.small.copy <- as.data.frame(df.a1.small)
dists.ps.copy <- as.data.frame(dists.ps)

for(i in 1:nrow(df.a1.small)) {
  max.treat <- which.max(df.a1.small.copy$prop_score)
  treat[i] <- names(max.treat)
  df.a1.small.copy <- df.a1.small.copy %>% slice(-max.treat)

  match.control.1 <- which.min(dists.ps.copy[max.treat,])
  control.1[i] <- names(match.control.1)
  dists.ps.copy <- dists.ps.copy %>% select(-match.control.1)

  if(ncol(dists.ps.copy) > 1) {
    match.control.2 <- which.min(dists.ps.copy[max.treat,])
    control.2[i] <- names(match.control.2)
    dists.ps.copy <- dists.ps.copy %>%
      select(-match.control.2) %>%
      slice(-max.treat)
  } else {
    control.2[i] <- names(dists.ps.copy)
  }

}
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use ‘all_of(match.control.1)‘ instead of ‘match.control.1‘ to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use ‘all_of(match.control.2)‘ instead of ‘match.control.2‘ to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
treat
```

```
## [1] "2"  "10" "22" "18" "3"
```

```
control.1
```

```
## [1] "6"  "11" "13" "1"  "9"
```

```
control.2
```

```
## [1] "4"  "7"  "12" "8"  "5"
```

Question 11: Did any of the matches you made in Question 6 change in Question 10?

> Your answer here.

It is also possible to have a variable number of control individuals per treatment individual in "full" matching. Full matching assures that every individual in the dataset is paired. Full matching can only by achieved using an optimal matching algorithm.

Caliper Width

As seen in $1:1$ and $k:1$ matching, some data may be pruned in favor of other priorities. We may also choose to prune data for which a sufficiently close match can be found. For this method we choose a threshold, or "caliper", and only consider matches whose distance is within this caliper width, discarding any individuals left unmatched.

Replacement

Another consideration when deciding upon a matching algorithm is whether matches are made with or without replacement. That is, can the same control individual be matched to more than one treatment individual. You may notice that so far we have only considered matching without replacement.

Question 12: Write code to perform the same greedy matching as in Question 6 but **with** replacement. (Hint: This code will likely be much simpler!)

```
row.mins <- apply(dists.ps, 1, which.min)
treat <- names(row.mins)
control <- colnames(dists.ps)[row.mins]
treat
```

```
## [1] "2"  "3"  "10" "18" "22"
```

```
control
```

```
## [1] "6"  "13" "4"  "13" "11"
```

Question 13: Compare these matches to those you found in Question 6.

> Your answer here.

Matching Algorithm Examples

References

http://www.stephenpettigrew.com/teaching/gov2001/section11_2015.pdf

https://en.wikipedia.org/wiki/Mahalanobis_distance

https://www.statisticshowto.com/greedy-algorithm-matching/

https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Data_Matching-Optimal_and_Greedy.pdf