Computational Social Science: Matching Methods

Your Name Here

MM/DD/YYYY

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

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.

Furthermore, matching should not be seen as an alternative to modeling adjustments such as regression, but instead are often used together.

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 there is concern among providers that AspiTyleCedrin 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 treatment and observed outcomes to be confounded by this variable.)

```r
n = 1e4 # Number of individuals (smaller than last time)

# 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 + 8*(W1 == 1)+ 12*(W2==2) +
           8*(W2==3) + 4*(W2==4) + (-4)*(W2 == 5),
         A = as.numeric(rbernoulli(n,
                                    p = (0.16 + 0.07*(W1 > 0) + 0.21*(W2 == 0) -
                                           0.1*(W3 > 25) ))),
         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  1 18.36129 0     1
## 2  2  1  2 40.50231 1     1
## 3  3  1  2 45.38797 0     1
## 4  4  1  0 27.88791 0     1
## 5  5  0  1 26.55001 0     1
## 6  6  1  0 32.22700 0     1
```

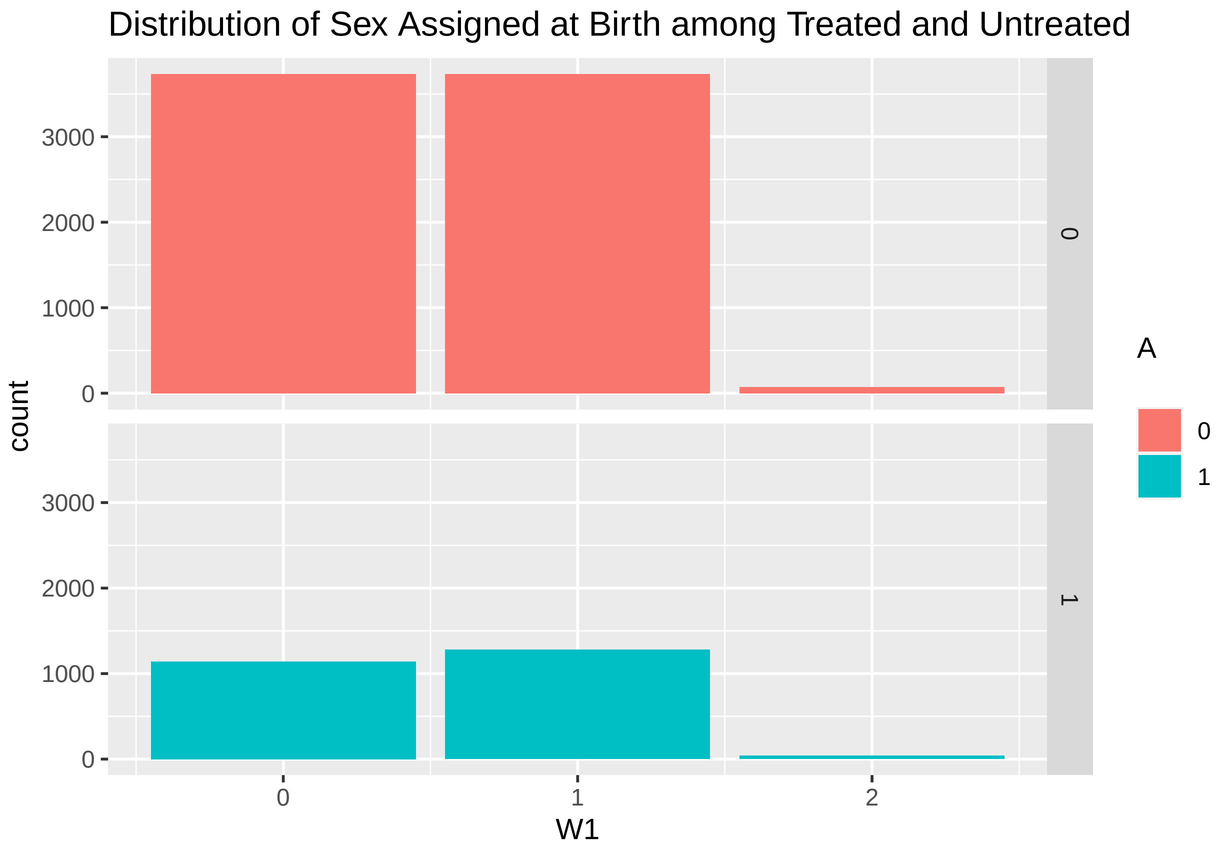
```r
summary(df)
```

```
##        ID              W1               W2               W3
##  Min.   :    1   Min.   :0.0000   Min.   :0.0000   Min.   :12.97
##  1st Qu.: 2501   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:25.29
##  Median : 5000   Median :1.0000   Median :0.0000   Median :30.40
##  Mean   : 5000   Mean   :0.5235   Mean   :0.7917   Mean   :30.90
##  3rd Qu.: 7500   3rd Qu.:1.0000   3rd Qu.:2.0000   3rd Qu.:35.66
##  Max.   :10000   Max.   :2.0000   Max.   :5.0000   Max.   :61.84
##        A              Y_obs
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:1.0000
##  Median :0.0000   Median :1.0000
##  Mean   :0.2461   Mean   :0.8702
##  3rd Qu.:0.0000   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)

```r
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")
```

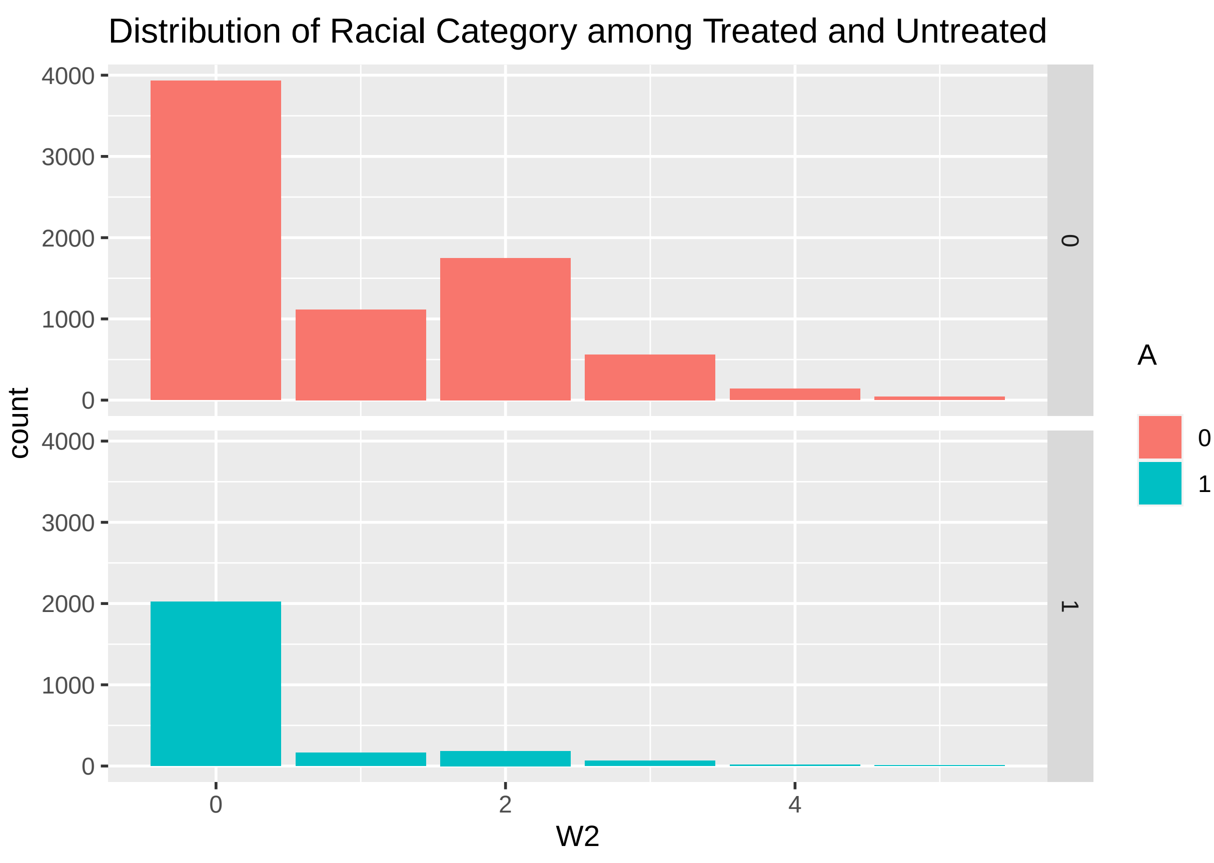


```r
chisq.test(table(df$A, df$W1))
```

```
##
##  Pearson's Chi-squared test
##
## data:  table(df$A, df$W1)
## X-squared = 14.594, df = 2, p-value = 0.0006776
```

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.

```r
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")
```

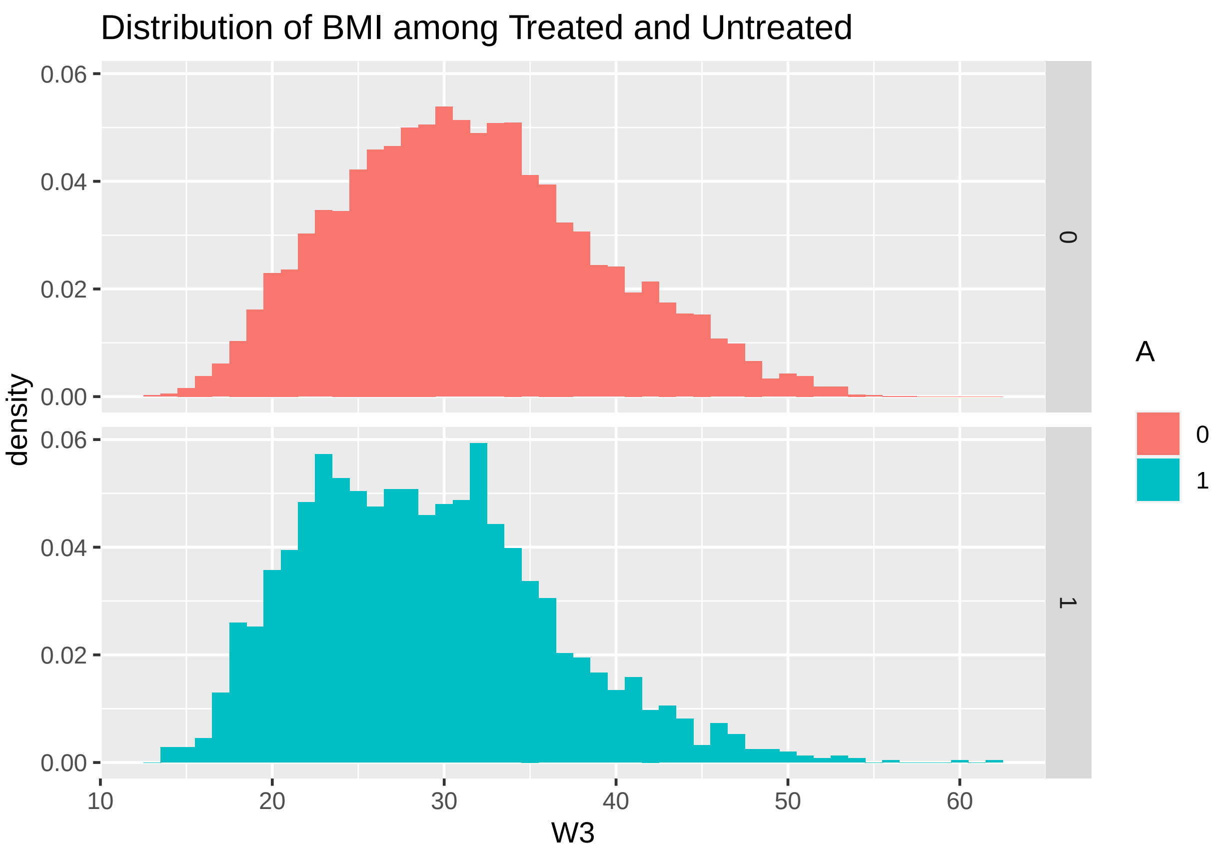



```r
chisq.test(table(df$A, df$W2))
```

```
##
##  Pearson's Chi-squared test
##
## data:  table(df$A, df$W2)
## X-squared = 702.62, 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 = 15.107, df = 4339.1, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  2.224104 2.887448
## sample estimates:
## mean in group 0 mean in group 1
##        31.52759        28.97181
```

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
- Estimand

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 AspiTyle-Cedrin, `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
```

```r
  colnames(dists) <- ynames
  return(dists)
}

dists_ex <- calculate.dist(df_a1_small[, cols], df_a0_small[, cols], dist.exact)
dists_ex
```

```
##      1   3   4   5   6   7   8   9  11  12
## 2  Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## 10 Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## 14 Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## 16 Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## 20 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.

```r
dists_ex_lim <- calculate.dist(df_a1_small[, cols[1:2]], df_a0_small[, cols[1:2]], dist.exact)
dists_ex_lim
```

```
##      1   3   4   5   6   7   8   9  11  12
## 2  Inf   0 Inf Inf Inf Inf Inf Inf Inf Inf
## 10 Inf Inf   0 Inf   0 Inf Inf Inf Inf Inf
## 14 Inf Inf Inf Inf Inf Inf Inf   0   0   0
## 16 Inf Inf   0 Inf   0 Inf Inf Inf Inf Inf
## 20 Inf Inf Inf Inf Inf Inf Inf   0   0   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         3        4          5         6         7          8
## 2  167.40895  36.76637 96.01069 105.787309 63.367522 57.721573 106.425165
## 10 112.58080  91.61877 41.15923  50.965281  8.505973  3.585925  51.602988
## 14  40.93774 163.25693 30.49238  20.719605 63.143892 68.787655  20.082446
## 16  58.41317 145.78463 13.01426   3.435836 45.667517 51.329614   2.846252
## 20  28.53868 175.66024 42.89545  33.115147 75.547667 81.187522  32.477546
##             9        11        12
## 2  113.176767 102.473144 160.53788
## 10  58.331578  47.626777 105.69635
## 14  13.318421  24.023621  34.04714
## 16   4.181099   6.564638  51.52389
## 20  25.722465  36.427665  21.64310
```

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
## -1.2177  -0.8652  -0.6325  -0.2706   2.6683
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.110248   0.123304  -0.894    0.371
## W1           0.348073   0.056779   6.130 8.77e-10 ***
## W2          -0.546561   0.034490 -15.847  < 2e-16 ***
## W3          -0.028853   0.004888  -5.903 3.57e-09 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11160  on 9999  degrees of freedom
## Residual deviance: 10474  on 9996  degrees of freedom
## AIC: 10482
##
## Number of Fisher Scoring iterations: 5
```

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

```r
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.

```r
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         3          4         5          6         7         8
## 2  0.8373168 0.1409645 1.45708108 0.6010499 1.33188650 0.1277508 0.6034956
## 10 0.4619573 1.4402386 0.15780700 0.6982242 0.03261241 1.4270249 0.6957785
## 14 0.3876089 1.3658903 0.23215533 0.6238759 0.10696074 1.3526766 0.6214302
## 16 0.6696617 1.6479430 0.04989745 0.9059286 0.17509203 1.6347294 0.9034830
## 20 0.4351668 1.4134481 0.18459747 0.6714337 0.05940289 1.4002344 0.6689881
##             9         11         12
## 2  1.17386212 1.13281774 1.35546457
## 10 0.12541196 0.16645634 0.05619048
## 14 0.05106363 0.09210802 0.13053881
## 16 0.33311640 0.37416079 0.15151396
## 20 0.09862149 0.13966587 0.08298095
```

Double Robustness A key advantage of propensity score matching is that, when used in conjunction with outcome regression, provides a "doubly robust" estimator. That is,

> "When used individually to estimate a causal effect, both outcome regression and propensity score methods are unbiased only if the statistical model is correctly specified. The doubly robust estimator combines these 2 approaches such that only 1 of the 2 models need be correctly specified to obtain an unbiased effect estimator."

"Correctly specified" means that a model accurately represents the relationship between the variables. E.g. a linear model between x and y is correctly specified if and only if x and y truly do have a linear relationship to each other.

This means that only one of the two models (the model of treatment to covariates or the model of outcome to treatment and covariates) needs to accurately represent the relationships among the respective variables in order for the estimate to be unbiased.

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.

Most greedy matching algorithms in common use (including those listed above) are "nearest neighbor" algorithms, which choose a treatment individual first and match to a control individual rather than the reverse.

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] "16" "10" "20" "14" "2"
```

```
control
```

```
## [1] "4"  "6"  "12" "9"  "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] "2"  "14" "20" "10" "16"
```

```
control
```

```
## [1] "7"  "9"  "6"  "12" "4"
```

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] "10" "16" "14" "20" "2"
```

```
control
```

```
## [1] "6"  "4"  "9"  "12" "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. (Note: while we are not considering them here, there are matching algorithms which discard treatment individuals rather than control individuals)

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()
```

```r
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.
```

```r
treat
```

```
## [1] "16" "10" "20" "14" "2"
```

```r
control_1
```

```
## [1] "4"  "6"  "11" "8"  "7"
```

```r
control_2
```

```
## [1] "12" "9"  "1"  "5"  "3"
```

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"  "10" "14" "16" "20"
```

```
control
```

```
## [1] "7" "6" "9" "4" "6"
```

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

> Your answer here.

Estimand

Depending on the matching algorithm used, you may be limited in whether it is possible to estimate the Average Treatment Effect (ATE) or the Average Treatment Effect on the Treated (ATT) only. For example, 1:1 nearest neighbor matching almost always estimates the ATE and cannot estimate the ATE.

Question 14: Briefly explain why 1:1 nearest neighbor matching may not be able to estimate the ATE.

> Your answer here.

Matching Algorithm Examples

As we've seen using our small subset of the data, implementing matching algorithms from scratch can be rather complex. Thankfully, we can use the `MatchIt` package which can implement many different matching algorithm variations for us.

The main `matchit()` function of this package includes the following arguments:

- `formula` : A formula object specifying the the treatment variable `A` and the covariates to be matched upon `X`, X2,`... in the following format:`A ~ X1 + X2 + ...‘.

- `data` : The data frame.
- `method`: Matching method to be used. Options include (but are not limited to): "nearest" (i.e. Nearest Neighbor), "optimal", "full", "exact".
- `distance`: Distance metric to be used. Options include (but are not limited to): "glm" (e.g.Propensity score matching using a generalized linear model such as regression), "mahalanobis", a numeric vector containing already calculated distances.
- `link`: The link function used with the option chosen in `distance`. (e.g. "logit" if using logistic regression for propensity score matching)
- `estimand`: The value to be estimated. Options include (but are not limited to): "ATE", "ATT". Note that "ATE" is not available for all matching methods.
- `discard`: Which type of units may be discardes. Options are: "control" (i.e. most of the examples we have considered so far), "treatment", "none", "both".
- `replace`: Whether matching should be done with (`TRUE`) or without (`FALSE`) replacement.
- `caliper`: The caliper widths to use for each variable (if any) while matching.
- `ratio`: How many control units should be matched to each treatment unit.

Exact Matching Example

For example, for an exact matching on our dataset ignoring BMI we would do the following to estimate ATE:

```
match_exact_ate <- matchit(formula = A ~ W1 + W2, data = df, method = "exact", estimand = "ATE")
summary(match_exact_ate)
```

```
##
## Call:
## matchit(formula = A ~ W1 + W2, data = df, method = "exact", estimand = "ATE")
##
## Summary of Balance for All Data:
##    Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
## W1        0.5526        0.5140          0.0737     1.0446    0.0129   0.0314
## W2        0.3430        0.9382         -0.5944     0.5354    0.0992   0.3006
##
##
## Summary of Balance for Matched Data:
##    Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
## W1        0.5220        0.5220               0     1.0005         0        0
## W2        0.7887        0.7887               0     1.0005         0        0
##    Std. Pair Dist.
## W1               0
## W2               0
##
## Percent Balance Improvement:
##    Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
## W1             100       98.9       100      100
## W2             100       99.9       100      100
##
## Sample Sizes:
##               Control Treated
## All           7539.   2461.
## Matched (ESS) 7359.06 1581.65
## Matched       7529.   2461.
## Unmatched       10.      0.
```

```
## Discarded        0.      0.
```

We can see from the summary how much the balance has improved after matching, but remember that this is only the balance on `W1` and `W2`.

To use this matching to estimate the ATE we first get the matched data using the `match.data()` function. We can then use logistic regression to estmate the ATE.

```r
match_exact_ate_data <- match.data(match_exact_ate)
lm_exact_ate <- lm(Y_obs ~ A + W1 + W2 + W3, data = match_exact_ate_data, weights = weights)
lm_exact_ate_summ <- summary(lm_exact_ate)
lm_exact_ate_summ
```

```
##
## Call:
## lm(formula = Y_obs ~ A + W1 + W2 + W3, data = match_exact_ate_data,
##     weights = weights)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -1.61736 -0.06524  0.03518  0.11897  0.76860
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.7029784  0.0131460  53.475  < 2e-16 ***
## A           -0.2994747  0.0063098 -47.462  < 2e-16 ***
## W1           0.0223401  0.0063506   3.518 0.000437 ***
## W2           0.0302368  0.0029684  10.186  < 2e-16 ***
## W3           0.0073108  0.0004996  14.632  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2715 on 9985 degrees of freedom
## Multiple R-squared:  0.2432, Adjusted R-squared:  0.2429
## F-statistic: 802.1 on 4 and 9985 DF,  p-value: < 2.2e-16
```

The ATE estimate is the coefficient estimate on the treatment variable `A`:

```r
ATE_exact <- lm_exact_ate_summ$coefficients["A", "Estimate"]
ATE_exact
```

```
## [1] -0.2994747
```

We could also have estimated the ATT using this method:

```r
match_exact_att <- matchit(formula = A ~ W1 + W2, data = df, method = "exact", estimand = "ATT")
summary(match_exact_att, un = FALSE)
```

```
##
## Call:
## matchit(formula = A ~ W1 + W2, data = df, method = "exact", estimand = "ATT")
##
```

```
## Summary of Balance for Matched Data:
##    Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
## W1        0.5526        0.5526               0     1.0002         0        0
## W2        0.3430        0.3430               0     1.0002         0        0
##    Std. Pair Dist.
## W1               0
## W2               0
##
## Sample Sizes:
##               Control Treated
## All           7539.      2461
## Matched (ESS) 5453.76    2461
## Matched       7529.      2461
## Unmatched       10.         0
## Discarded        0.         0
```

```r
match_exact_att_data <- match.data(match_exact_att)
lm_exact_att <- lm(Y_obs ~ A + W1 + W2 + W3, data = match_exact_att_data, weights = weights)
lm_exact_att_summ <- summary(lm_exact_att)
lm_exact_att_summ
```

```
##
## Call:
## lm(formula = Y_obs ~ A + W1 + W2 + W3, data = match_exact_att_data,
##     weights = weights)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -1.53547 -0.05442  0.01638  0.12633  0.58254
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.6799281  0.0151746  44.807  < 2e-16 ***
## A           -0.3822308  0.0068497 -55.802  < 2e-16 ***
## W1           0.0266851  0.0069572   3.836 0.000126 ***
## W2           0.0240483  0.0039450   6.096 1.13e-09 ***
## W3           0.0085523  0.0005877  14.553  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2947 on 9985 degrees of freedom
## Multiple R-squared:  0.2799, Adjusted R-squared:  0.2796
## F-statistic: 970.3 on 4 and 9985 DF,  p-value: < 2.2e-16
```

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

```
## [1] -0.3822308
```

k Nearest Neighbor Matching Example

Now let's perform a 2:1 nearest neighbor matching using (logistic regression) propensity scores on all three covariates. Remember that we can only estimate ATT in this case.

```r
match_ps_att <- matchit(formula = A ~ W1 + W2 + W3, data = df, method = "nearest", distance = "glm", lin
summary(match_ps_att)
```

```
##
## Call:
## matchit(formula = A ~ W1 + W2 + W3, data = df, method = "nearest",
##     distance = "glm", link = "logit", discard = "control", replace = FALSE,
##     ratio = 2)
##
## Summary of Balance for All Data:
##          Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean
## distance        0.2939        0.2305          0.7663     0.5832    0.1674
## W1              0.5526        0.5140          0.0729     1.0446    0.0129
## W2              0.3430        0.9382         -0.7117     0.5354    0.0992
## W3             28.9718       31.5276         -0.3545     0.9191    0.0995
##          eCDF Max
## distance   0.3023
## W1         0.0314
## W2         0.3006
## W3         0.1424
##
##
## Summary of Balance for Matched Data:
##          Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean
## distance        0.2939        0.2889          0.0610     1.0480    0.0221
## W1              0.5526        0.5386          0.0265     1.0438    0.0047
## W2              0.3430        0.3647         -0.0260     1.0027    0.0044
## W3             28.9718       29.2079         -0.0327     1.2777    0.0270
##          eCDF Max Std. Pair Dist.
## distance   0.0683          0.0613
## W1         0.0075          0.8180
## W2         0.0242          0.0605
## W3         0.0740          0.7599
##
## Percent Balance Improvement:
##          Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
## distance            92.0       91.3      86.8     77.4
## W1                  63.7        1.8      63.7     76.0
## W2                  96.3       99.6      95.5     92.0
## W3                  90.8     -190.5      72.9     48.0
##
## Sample Sizes:
##           Control Treated
## All          7539    2461
## Matched      4922    2461
## Unmatched    2612       0
## Discarded       5       0
```

```r
match_ps_att_data <- match.data(match_ps_att)
lm_ps_att <- lm(Y_obs ~ A + W1 + W2 + W3, data = match_ps_att_data, weights = weights)
lm_ps_att_summ <- summary(lm_ps_att)
lm_ps_att_summ
```

```
##
## Call:
## lm(formula = Y_obs ~ A + W1 + W2 + W3, data = match_ps_att_data,
##     weights = weights)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.00632 -0.05202  0.04179  0.15133  0.61351
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.6329517  0.0190306  33.260  < 2e-16 ***
## A           -0.3862028  0.0078235 -49.364  < 2e-16 ***
## W1           0.0358554  0.0087614   4.092 4.31e-05 ***
## W2           0.0372798  0.0048629   7.666 2.00e-14 ***
## W3           0.0099793  0.0007384  13.515  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3167 on 7378 degrees of freedom
## Multiple R-squared:  0.297,  Adjusted R-squared:  0.2966
## F-statistic: 779.2 on 4 and 7378 DF,  p-value: < 2.2e-16
```

```
ATT_ps <- lm_ps_att_summ$coefficients["A", "Estimate"]
ATT_ps
```

```
## [1] -0.3862028
```

Full Optimal Mahalanobis Matching Example

Now let's perform a full optimal matching on all three covariates using Mahalanobis distances. (We'll need to do this on a smaller subset of the data)

```
df_small <- sample_n(df, 1000) # SRS of 1000
```

```
match_full_ate <- matchit(formula = A ~ W1 + W2 + W3, data = df_small, method = "full", distance = "maha
summary(match_full_ate)
```

```
##
## Call:
## matchit(formula = A ~ W1 + W2 + W3, data = df_small, method = "full",
##     distance = "mahalanobis")
##
## Summary of Balance for All Data:
##    Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
## W1        0.5168        0.5315         -0.0280     1.0212    0.0063   0.0168
## W2        0.2479        1.0013         -1.0752     0.3569    0.1256   0.3708
## W3       28.2750       31.6957         -0.4647     0.9351    0.1282   0.2161
##
##
## Summary of Balance for Matched Data:
```

```
##    Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
## W1        0.5168        0.5158          0.0020     1.0004    0.0004   0.0011
## W2        0.2479        0.2862         -0.0547     0.8833    0.0076   0.0261
## W3       28.2750       28.3682         -0.0127     1.0068    0.0064   0.0236
##    Std. Pair Dist.
## W1          0.0146
## W2          0.1317
## W3          0.1619
##
## Percent Balance Improvement:
##    Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
## W1            92.8       98.3      94.4     93.7
## W2            94.9       88.0      93.9     93.0
## W3            97.3       89.8      95.0     89.1
##
## Sample Sizes:
##               Control Treated
## All            762.       238
## Matched (ESS)  268.39     238
## Matched        762.       238
## Unmatched        0.         0
## Discarded        0.         0
```

```r
match_full_ate_data <- match.data(match_full_ate)
lm_full_ate <- lm(Y_obs ~ A + W1 + W2 + W3, data = match_full_ate_data, weights = weights)
lm_full_ate_summ <- summary(lm_full_ate)
lm_full_ate_summ
```

```
##
## Call:
## lm(formula = Y_obs ~ A + W1 + W2 + W3, data = match_full_ate_data,
##     weights = weights)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -2.19121 -0.03945  0.01050  0.10057  0.59623
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.697541   0.047969  14.541  < 2e-16 ***
## A           -0.411817   0.022926 -17.963  < 2e-16 ***
## W1           0.010991   0.025075   0.438    0.661
## W2           0.025497   0.015117   1.687    0.092 .
## W3           0.008194   0.001976   4.147 3.65e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3087 on 995 degrees of freedom
## Multiple R-squared:  0.278,  Adjusted R-squared:  0.2751
## F-statistic: 95.77 on 4 and 995 DF,  p-value: < 2.2e-16
```

```r
ATE_full <- lm_full_ate_summ$coefficients["A", "Estimate"]
ATE_full
```

```
## [1] -0.4118175
```

```r
match_full_att <- matchit(formula = A ~ W1 + W2 + W3, data = df_small, method = "full", distance = "maha
summary(match_full_att)
```

```
##
## Call:
## matchit(formula = A ~ W1 + W2 + W3, data = df_small, method = "full",
##     distance = "mahalanobis")
##
## Summary of Balance for All Data:
##    Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
## W1        0.5168        0.5315         -0.0280     1.0212    0.0063   0.0168
## W2        0.2479        1.0013         -1.0752     0.3569    0.1256   0.3708
## W3       28.2750       31.6957         -0.4647     0.9351    0.1282   0.2161
##
##
## Summary of Balance for Matched Data:
##    Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
## W1        0.5168        0.5158          0.0020     1.0004    0.0004   0.0011
## W2        0.2479        0.2862         -0.0547     0.8833    0.0076   0.0261
## W3       28.2750       28.3682         -0.0127     1.0068    0.0064   0.0236
##    Std. Pair Dist.
## W1          0.0146
## W2          0.1317
## W3          0.1619
##
## Percent Balance Improvement:
##    Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
## W1            92.8       98.3      94.4     93.7
## W2            94.9       88.0      93.9     93.0
## W3            97.3       89.8      95.0     89.1
##
## Sample Sizes:
##               Control Treated
## All            762.       238
## Matched (ESS)  268.39     238
## Matched        762.       238
## Unmatched        0.         0
## Discarded        0.         0
```

```r
match_full_att_data <- match.data(match_full_att)
lm_full_att <- lm(Y_obs ~ A + W1 + W2 + W3, data = match_full_att_data, weights = weights)
lm_full_att_summ <- summary(lm_full_att)
lm_full_att_summ
```

```
##
## Call:
## lm(formula = Y_obs ~ A + W1 + W2 + W3, data = match_full_att_data,
##     weights = weights)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -2.19121 -0.03945  0.01050  0.10057  0.59623
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.697541   0.047969  14.541  < 2e-16 ***
## A           -0.411817   0.022926 -17.963  < 2e-16 ***
## W1           0.010991   0.025075   0.438    0.661
## W2           0.025497   0.015117   1.687    0.092 .
## W3           0.008194   0.001976   4.147 3.65e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3087 on 995 degrees of freedom
## Multiple R-squared:  0.278,  Adjusted R-squared:  0.2751
## F-statistic: 95.77 on 4 and 995 DF,  p-value: < 2.2e-16
```

```
ATT_full <- lm_full_att_summ$coefficients["A", "Estimate"]
ATT_full
```

```
## [1] -0.4118175
```

Question 15: Perform a matching algorithm of your own choosing. Report the estimated ATE or ATT where available. (Note: If your chosen algorithm takes too long to run on `df` you may instead use `df_small`)

```
# Your code here
```

Question 16: Compare the estimates of ATE and ATT found above with the true values (saved as `ATE_true` and `ATT_true`). Which method was most accurate? Considering the pros and cons of different methods we have discussed, which method do you prefer?

```
ATE_true
```

```
## [1] -0.3029
```

```
c(ATE_exact, ATE_full)
```

```
## [1] -0.2994747 -0.4118175
```

```
ATT_true
```

```
## [1] -0.3864283
```

```
c(ATT_exact, ATT_ps, ATT_full)
```

```
## [1] -0.3822308 -0.3862028 -0.4118175
```

Your answer here.

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

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2943670/

https://academic.oup.com/aje/article/173/7/761/103691