hw6

```
library(tidyverse)
library(MatchIt)
library(ebal)
library(kbal)
library(here)
```

1

a

$$
\begin{aligned}
E[\hat{\tau}_{wdim}] &= E\left[\frac{1}{n_1}\sum_{D_i=1} w_i Y_i\right] - E\left[\frac{1}{n_0}\sum_{D_i=0} w_i Y_i\right] \\
&= E\Big[w(X)Y(1) \mid D=1\Big] - E\Big[w(X)Y(0) \mid D=0\Big] \\
&= E\Big[E[w(X)Y(1) \mid X,\ D=1] \;\Big|\; D=1\Big] - E\Big[E[w(X)Y(0) \mid X,\ D=0] \;\Big|\; D=0\Big] && \text{iterated E} \\
&= E\Big[w(X)E[Y(1) \mid X] \;\Big|\; D=1\Big] - E\Big[w(X)E[Y(0) \mid X] \;\Big|\; D=0\Big] && \text{CI} \\
&= E[w(X)f_1(X) \mid D=1] - E[w(X)f_0(X) \mid D=0] \\
&= E[f_1(X)] - E[f_0(X)] && \text{WC-ATE} \\
&= E[E[Y(1) \mid X]] - E[E[Y(0) \mid X]] \\
&= E[Y(1)] - E[Y(0)] \\
&= \text{ATE}
\end{aligned}
$$

b

Let $Y(0) = f_0(X) + \epsilon$ and $Y(1) = f_1(X) + \epsilon$

$$
\begin{aligned}
E[\hat{\tau}_{wdim} - \text{SATE}] &= E\left[\frac{1}{n_1}\sum_{D_i=1} w_i Y_i(1) - \frac{1}{n_0}\sum_{D_i=0} w_i Y_i(0) - \frac{1}{n}\sum_{i=1}^{n}\Big(Y_i(1) - Y_i(0)\Big)\right] \\
&= E\left[\frac{1}{n_1}\sum_{D_i=1} w_i f_1(X) - \frac{1}{n_0}\sum_{D_i=0} w_i f_0(X) - \frac{1}{n}\sum_{i=1}^{n} f_1(X) + \frac{1}{n}\sum_{i=1}^{n} f_0(X)\right] \\
&+ \underbrace{E\left[\frac{1}{n_1}\sum_{D_i=1} w_i \epsilon_i\right]}_{\substack{0 \text{ since } w_i \text{ is entirely determined by X and D}}} - \underbrace{E\left[\frac{1}{n_0}\sum_{D_i=0} w_i \epsilon_i\right]}_{0} - \underbrace{E\left[\frac{1}{n}\sum_{i=1}^{n} \epsilon_i\right]}_{0} + \underbrace{E\left[\frac{1}{n}\sum_{i=1}^{n} \epsilon_i\right]}_{0} \\
&= E\left[\frac{1}{n_1}\sum_{D_i=1} w_i f_1(X) - \frac{1}{n_0}\sum_{D_i=0} w_i f_0(X) - \frac{1}{n}\sum_{i=1}^{n} f_1(X) + \frac{1}{n}\sum_{i=1}^{n} f_0(X)\right] \\
&= E\left[\frac{1}{n}\sum_{i=1}^{n} f_1(X) - \frac{1}{n}\sum_{i=0}^{n} f_0(X) - \frac{1}{n}\sum_{i=1}^{n} f_1(X) + \frac{1}{n}\sum_{i=1}^{n} f_0(X)\right] \qquad \text{EWC-ATE} \\
&= 0
\end{aligned}
$$

Finally we know that $E[\hat{\tau}_{wdim} - \text{SATE}] = E[\hat{\tau}_{wdim} - \text{ATE}]$ because $E[\text{SATE}] = \text{ATE}$ and so we have that $\hat{\tau}_{wdim}$ is unbiased for the ATE under EWC-ATE

c

We can show that

$$
\begin{aligned}
E[w(X)f_1(X) \mid D = 1] &= \int f_1(X) w(X) P(X \mid D = 1) dx \\
&= \int f_1(X) \frac{P(D = 1)}{P(D = 1 \mid X)} P(X \mid D = 1) dx \\
&= \int f_1(X) \frac{P(D = 1 \mid X) P(X)}{P(D = 1 \mid X)} dx \\
&= \int f_1(X) p(X) dx \\
&= E[f_1(X)]
\end{aligned}
$$

and similarly

$$
\begin{aligned}
E[w(X)f_0(X) \mid D=0] &= \int f_0(X)w(X)P(X \mid D=0)dx \\
&= \int f_0(X)\frac{P(D=0)}{P(D=0 \mid X)}P(X \mid D=0)dx \\
&= \int f_0(X)\frac{P(D=0 \mid X)P(X)}{P(D=0 \mid X)}dx \\
&= \int f_0(X)p(X)dx \\
&= E[f_0(X)]
\end{aligned}
$$

So WC-ATE is satisfied and thus by part (a) $\hat{\tau}_{wdim}$ is unbiased for the ATE.

d

When $f_0(X)$ and $f_1(X)$ are linear functions of X

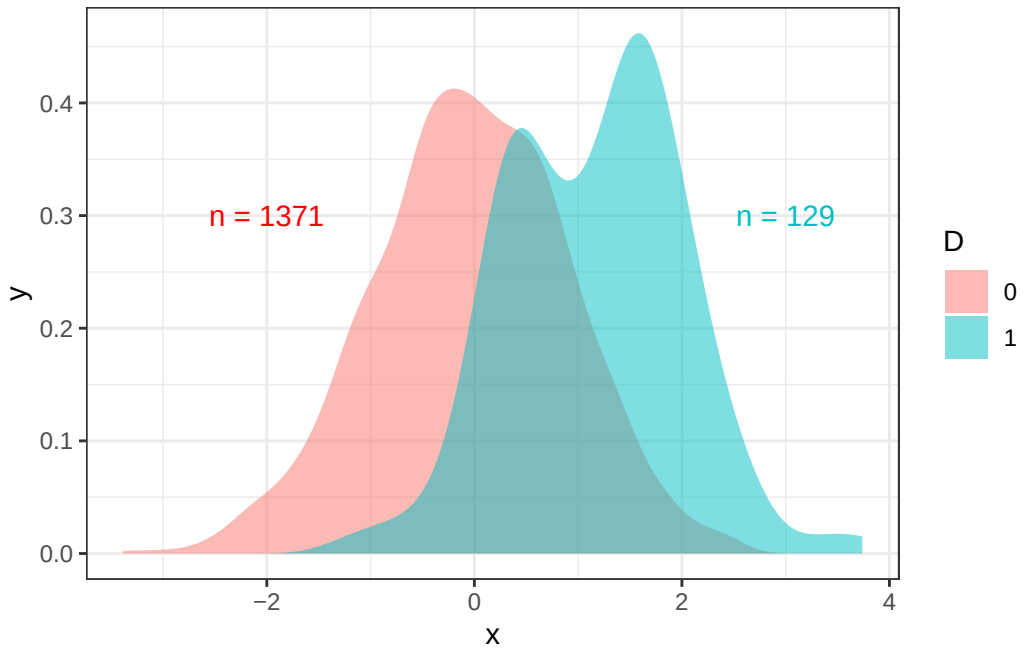
2

```
dgp1 <- function(n) {
  x <- rnorm(n)

  prob <- exp(1.5*x-3.25) / (1 + exp(1.5*x-3.25))
  d <- 1*(runif(n)<prob)

  y <- x + rnorm(n, sd=0.75)

  tibble(
    x = x,
    d = d,
    y = y
  )
}
```

```
set.seed(394)

n <- 1500
data <- dgp1(n)
```

a

```
data %>%
  ggplot(aes(x = x, fill = factor(d))) +
  geom_density(
    alpha = 0.5,
    color = NA
    ) +
  annotate("text", x = -2, y  = 0.3, label = "n = 1371", color = "red") +
  annotate("text", x = 3, y  = 0.3, label = "n = 129", color = "#00BFC4") +
  labs(
    fill = "D"
  ) +
  theme_bw()
```



We can see that the distribution of X on the treated group seems to be shifted farther to the right than for the control group and also appears to have a bimodal structure that is not present in the control group. What's more we see a large discrepancy in sample size between the treatment and control groups with the control group having more than 10x the number of observations as the treatment group

b

Results will be cached so I don't have to re-run this each time

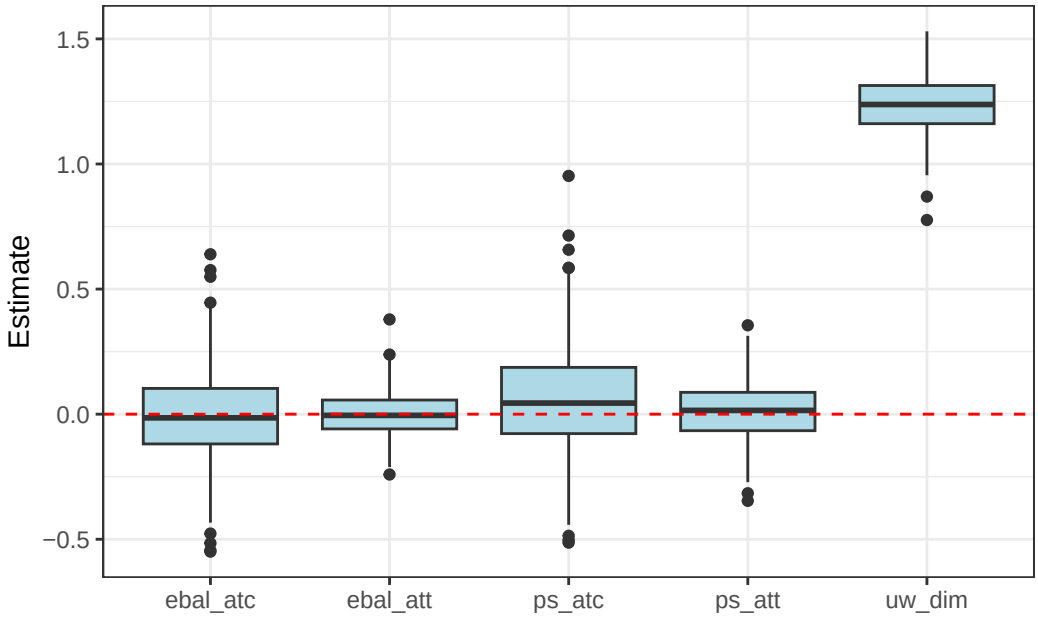
```
set.seed(100)

sim_data_sets <- list()
for(i in 1:500) {
  sim_data_sets[[i]] <- dgp1(1500)
}


sim_run <- function(df) {

  uw_dim <- lm(y ~ d, df)$coefficients[2]

  ps_att_w <- matchit(
    d ~ x,
    data = df,
    replace = T,
    ratio = 3,
    distance = "glm",
    link = "logit",
    estimand = "ATT"
  )

  ps_atc_w <- matchit(
    d ~ x,
    data = df,
    replace = T,
    ratio = 3,
    distance = "glm",
    link = "logit",
    estimand = "ATC"
  )

  ps_att <- lm(y ~ d, weights = ps_att_w$weights, df)$coefficients[2]
  ps_atc <- lm(y ~ d, weights = ps_atc_w$weights, df)$coefficients[2]

  # we get it, it converged within tolerance
  ebal_att_w <-  ebalance(Treatment=df$d, X=df$x)
  w_att <- rep(1, n)
```

```
  w_att[df$d == 0] <- ebal_att_w$w
  ebal_att <- lm(y ~ d, weights = w_att, df)$coefficients[2]

  ebal_atc_w <- ebalance(Treatment=1- df$d, X=df$x)
  w_atc <- rep(1, n)
  w_atc[df$d == 1] <- ebal_atc_w$w
  ebal_atc <- lm(y ~ d, weights = w_atc, df)$coefficients[2]

  tibble(
    uw_dim = uw_dim,
    ps_att = ps_att,
    ps_atc = ps_atc,
    ebal_att = ebal_att,
    ebal_atc = ebal_atc
  ) %>%
    pivot_longer(
      cols = everything(),
      names_to = "estimator",
      values_to = "val"
    )
}



sim_res <- sim_data_sets %>%
  map_dfr(sim_run)
```

We see that the unweighted difference in means estimator is very biased. The entropy balancing for the att and atc are both unbiased but for the atc we see higher variance. On the other hand the propensity score matching for the atc is biased while for the att it is not, and additionally for the atc we see higher variance.

```
sim_res <- read_csv(here("data", "hw6_p2.csv"))

sim_res %>%
  ggplot(aes(x = estimator, y = val)) +
  geom_boxplot(fill = "lightblue") +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(
    x = "",
    y = "Estimate"
```

```
) +
theme_bw()
```



c

In short, it's a lot easier to get close to satisfying EWC-ATT in this DGP because you get to use all of the units that were in the control group to do the matching and there are way more such units which means you're more likely to get good matching while also having minimal data loss. On the other hand it's harder to satisfy EWC-ATC in this DGP because you only can use units in the treatment group to do the matching and there are much fewer of these.

d

I think it has to do with the fact that entropy balancing weights are not trying to "do as much" as propensity score matching is. While it's difficult to equate the distributions of the covariates in estimating the ATC for this setting, it's much easier to make the means more similar. This is mostly because the functions in the DGP are actually linear, so good balance on the means is good enough to get low bias.

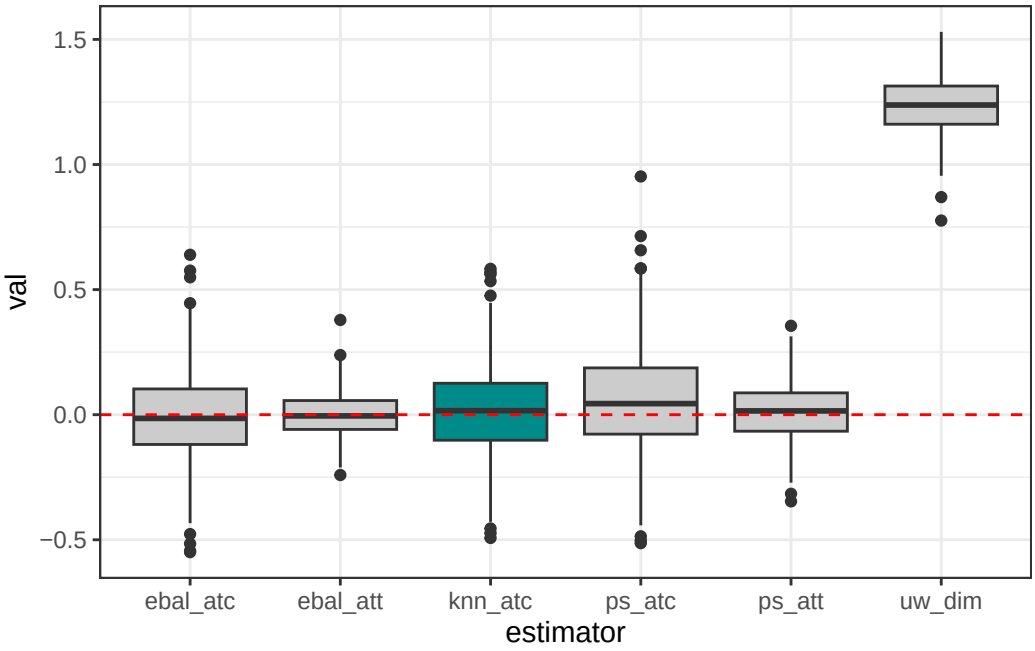
e

```
atc_better <- function(df) {
  ps_atc_w <- matchit(
    d ~ x,
    data = df,
    replace = T,
    ratio = 5,
    distance = "glm",
    link = "logit",
    estimand = "ATC",
    caliper = 0.05
  )

  match_atc <- lm(y ~ d, weights = ps_atc_w$weights, df)$coefficients[2]

  tibble(
    val = match_atc
  )

}

atc_res <- sim_data_sets %>%
  map_dfr(atc_better)

atc_res <- atc_res %>%
  mutate(estimator = "knn_atc")
```

We can see that we managed to get something with near zero bias using a different matching technique. In the end I only had to change a few things to get better performance. First I upped the ratio, and secondly I employed a caliper. The idea here was that since there are not a ton of units to match on, we want to avoid really bad matches that are just made because the algorithm needs a set number of matches. Importantly, while we brought down the bias, the variance is still higher than most of the other estimators.

```
atc_res <- read_csv(here("data", "hw6_p2-2.csv"))

sim_res %>%
  rbind(atc_res) %>%
  mutate(col = case_when(
    estimator == "knn_atc" ~ "h",
    T ~ "o"
```

```
)) %>%
ggplot(aes(x = estimator, y = val, fill = col))+
geom_boxplot(show.legend = F) +
scale_fill_manual(values = c("cyan4", "grey80")) +
geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
theme_bw()
```



f

I've learned that choosing an estimand is heavily dependent on the data set and in particular on the features of the covariates in that data set. The decisions you make about what estimators and estimands to use should be made after doing thorough exploratory data analysis. You really have to know what's going on with the data before you have a good sense for what estimands will be easy to estimate, and which estimators will do the best job.

3

```
load(here("data", "nsw_clean_v2.rda"))
```

a

```
exp_nsw <- nsw_clean_v2 %>%
  filter(group != "3. Non-Experimental Comparison")

exp_dim <- lm(re78 ~ treated, exp_nsw)$coefficients[2]

exp_dim
```

```
 treated
1794.342
```

b

```
nsw_use <- nsw_clean_v2 %>%
  filter(group != "2. Experimental Control")

# inverse propensity score weighting
psmod <- glm(
  treated ~ age + education + nodegree + married + black + hispanic + re74 + re75,
  family="binomial", data = nsw_use
  )
pi_hat <- psmod$fitted.values
n1 <- sum(nsw_use$treated == 1)
n0 <- sum(nsw_use$treated == 0)
w_ps <- (n0 / n1) * pi_hat / (1 - pi_hat)

w_ps[nsw_use$treated == 1] <- 1

w_ps_norm <- w_ps
w_ps_norm[nsw_use$treated==0] <- n0 * w_ps_norm[nsw_use$treated==0] / sum(w_ps_norm[nsw_us


ips <- lm(re78 ~ treated, weights = w_ps_norm, data = nsw_use)$coefficients[2]


# propensity score matching
out <- matchit(
  treated ~ age + education + re74 + re75 + black,
  data = nsw_use,
```

```
  replace = T,
  ratio = 3,
  distance = "glm",
  link = "logit",
  estimand = "ATT"
)

psm <- lm(re78 ~ treated, weights = out$weights, data = nsw_use)$coefficients[2]


# entropy balancing weights
X <- nsw_use %>%
  select(c(re74, re75, age, education)) %>%
  as.matrix()

output <- ebalance(Treatment = nsw_use$treated, X = X)
w_ebal <- rep(1, 2675)
w_ebal[nsw_use$treated == 0] <- output$w

ebal <- lm(re78 ~ treated, weights = w_ebal, data = nsw_use)$coefficients[2]

X_c <- nsw_use %>%
  select(re74, re75, age, education) %>%
  as.matrix()

# kernel balancing
full_kbal <- kbal(
  treatment = nsw_use$treated,
  allx = X_c,
  meanfirst = F
)

kbal.w <- full_kbal$w

kbal <- lm(re78 ~ treated, weights = kbal.w, data = nsw_use)$coefficients[2]
```

```
ests <- tibble(
  weights = c("inverse propensity score weights", "propensity score matching",
    "entropy balancing weights", "kernel balancing weights"),
  estimate = c(ips, psm, ebal, kbal)
)
```

```
ests
```

```
# A tibble: 4 x 2
  weights                          estimate
  <chr>                               <dbl>
1 inverse propensity score weights    1759.
2 propensity score matching           1512.
3 entropy balancing weights           1332.
4 kernel balancing weights            1348.
```

We can see that inverse propensity score weights gave us the closest estimate, propensity score matching was next closest, and the last two performed similarly in terms of distance from the unbiased estiamte in (a).

c

Effective sample sizes are astonishingly low implying that we have very extreme weights for some units in every estimator. I did not performing any trimming in the matching estimator so there is nothing to report there.

```
ess_calc <- function(wts) {
  ids <- which(nsw_use$treated == 0)
  wts_non_ex_comp <- wts[ids]
  num <- sum(wts_non_ex_comp)^2
  den <- sum(wts_non_ex_comp^2)
  return(num/den)
}


tibble(
  diagnostic = c("ess"),
  inverse_ps = c(ess_calc(w_ps_norm)),
  ps_match = c(ess_calc(out$weights)),
  ebal = c(ess_calc(w_ebal)),
  kbal = c(ess_calc(kbal.w))
)
```

```
# A tibble: 1 x 5
  diagnostic inverse_ps ps_match  ebal  kbal
  <chr>           <dbl>    <dbl> <dbl> <dbl>
```

```
1 ess              42.6     27.8  72.1  32.0
```

sorry lenny, got too tired on friday night so you won't get any SMD or variance ratio from me

d

We use the very simplistic weighted least squares augmented estimator here.

```
ips_new <- lm(re78 ~ treated + re75 + nodegree + age + married, weights = w_ps_norm, nsw_u
psm_new <- lm(re78 ~ treated + re75 + age + black + nodegree, weights = out$weights, nsw_u
ebal_new <- lm(re78 ~ treated + re75  + age + nodegree, weights = w_ebal, nsw_use)$coeffic
kbal_new <- lm(re78 ~ treated + re75  + age + nodegree, weights = kbal.w, nsw_use)$coeffic

tibble(
  weights = c("inverse propensity score weights", "propensity score matching",
    "entropy balancing weights", "kernel balancing weights"),
  aug_estimate = c(ips_new, psm_new, ebal_new, kbal_new)
) %>%
  left_join(ests) %>%
  rename(original_estimate = estimate)
```

```
# A tibble: 4 x 3
  weights                          aug_estimate original_estimate
  <chr>                                   <dbl>             <dbl>
1 inverse propensity score weights        1774.             1759.
2 propensity score matching               1763.             1512.
3 entropy balancing weights               1504.             1332.
4 kernel balancing weights                1529.             1348.
```

Ultimately we do see some improvements here, but I think this likely has to do with the fact that I knew the true value I was looking for and so I could tweak the variables being used to get closest to it. The results are very similar here although propensity score matching showed the most improvement.