Untitled

Spencer Braun

11/8/2020

```r
library(DOS2)
library(optmatch)
library(RItools)
library(rcbalance)

library(readstata13)
library(tidyverse)
library(haven)

source("utility.R")
```

Main Analysis

Here I do two things: define an analysis dataframe with the covariates used in the paper and define a larger analysis dataframe including more covariates. I will compare these approaches in generating propensity scores.

```r
main <- read.csv("processed_data/main.csv")
# main %>% head()
# main %>% names()


analysis.cols <- main %>% select(
  p2p,
  current_vote,
  transfersshare,
  donationsshare,
  gov,
  y2006,
  female,
  exminister,
  previous_vote,
  media_mentions,
  quality,
  years_served
)

covariates <- c(
  "transfersshare",
  "donationsshare",
  "gov",
  "y2006",
  "female",
  "exminister",
  "previous_vote",
  "media_mentions",
  "quality",
  "years_served"
)

sapply(analysis.cols, function(x) sum(is.na(x)))
```

```
##            p2p   current_vote transfersshare donationsshare            gov
##              0              0             29             29              0
##          y2006         female     exminister  previous_vote media_mentions
##              0              0              0              0              2
##        quality   years_served
##              0              0
```

```
# for now, drop NAs

analysis <- analysis.cols %>%
  drop_na() %>%
  rename(zb = p2p) %>%
  rename(y = current_vote)



analysis.cols.large <- main %>% select(
  p2p,
  current_vote,
  transfersshare,
  donationsshare,
  gov,
  y2006,
  female,
  exminister,
  previous_vote,
  media_mentions,
  quality,
  years_served,
  # province, #prov if need numeric
  election,
  winner,
  pop_per_km2,
  immigrants,
  citizens,
  unemployment_rate,
  median_family_income
)

analysis.large <- analysis.cols.large %>%
  drop_na() %>%
  rename(zb = p2p) %>%
  rename(y = current_vote)
```

```
main %>%
  group_by(id) %>%
  summarise(p2p06 = sum(p2p * y2006),
            p2p08 = sum(p2p * (1-y2006))) %>%
  filter(p2p06 != p2p08)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 146 x 3
##       id p2p06 p2p08
##    <int> <int> <dbl>
##  1     3     1     0
##  2     4     0     1
##  3     5     0     1
##  4     7     1     0
##  5     9     0     1
##  6    11     1     0
##  7    12     1     0
##  8    13     0     1
##  9    15     1     0
## 10    16     1     0
## # ... with 136 more rows
```

```
governing <- analysis %>% filter(gov == 1)
```

Note: matches stata output . gen SHARE=transferss+donationss (29 missing values generated)

Propensity Scores

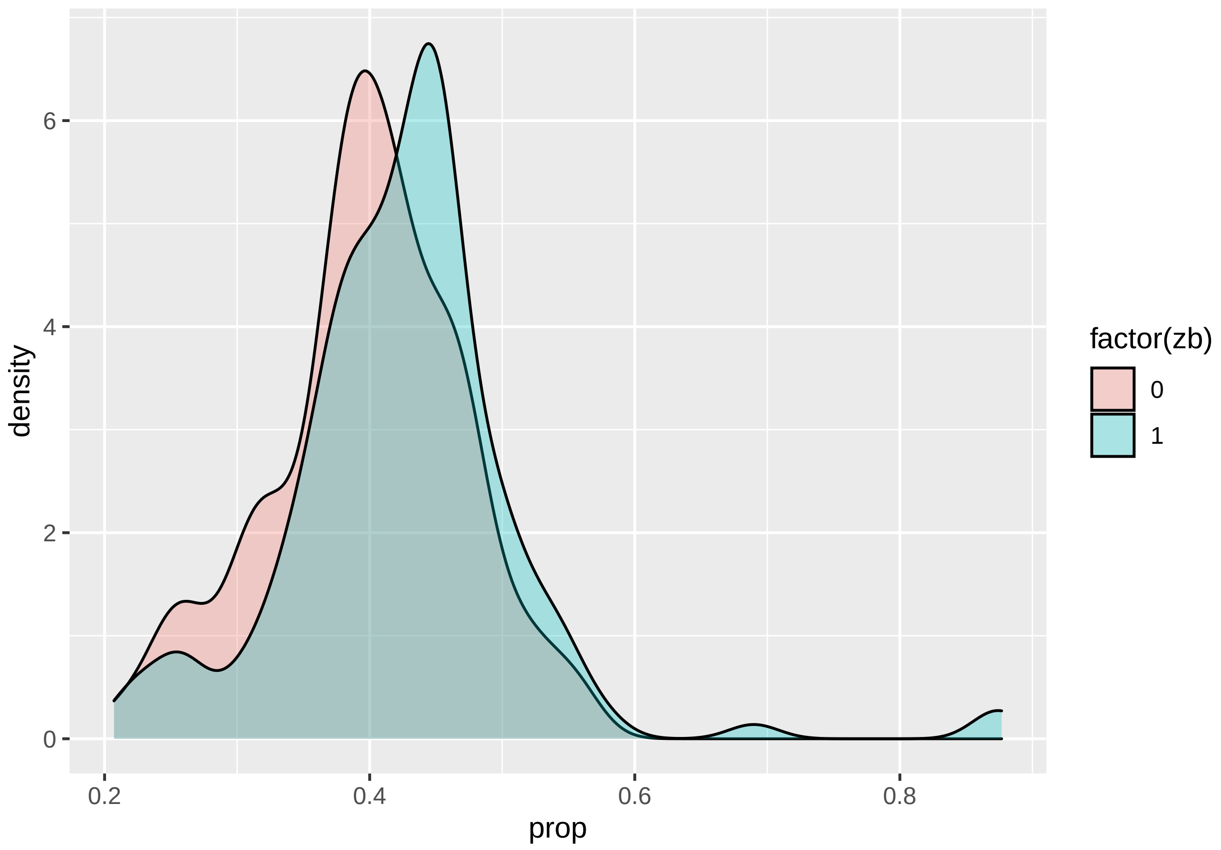
```
prop.scores <- glm(zb ~ . - y, family=binomial, data=analysis)
analysis$prop <- prop.scores$fitted.values


prop.scores.large <- glm(zb ~ . - y, family=binomial, data=analysis.large)
analysis.large$prop.large <- prop.scores.large$fitted.values


analysis$prop.large <- prop.scores.large$fitted.values
```

Imbens / Rubin page 282:

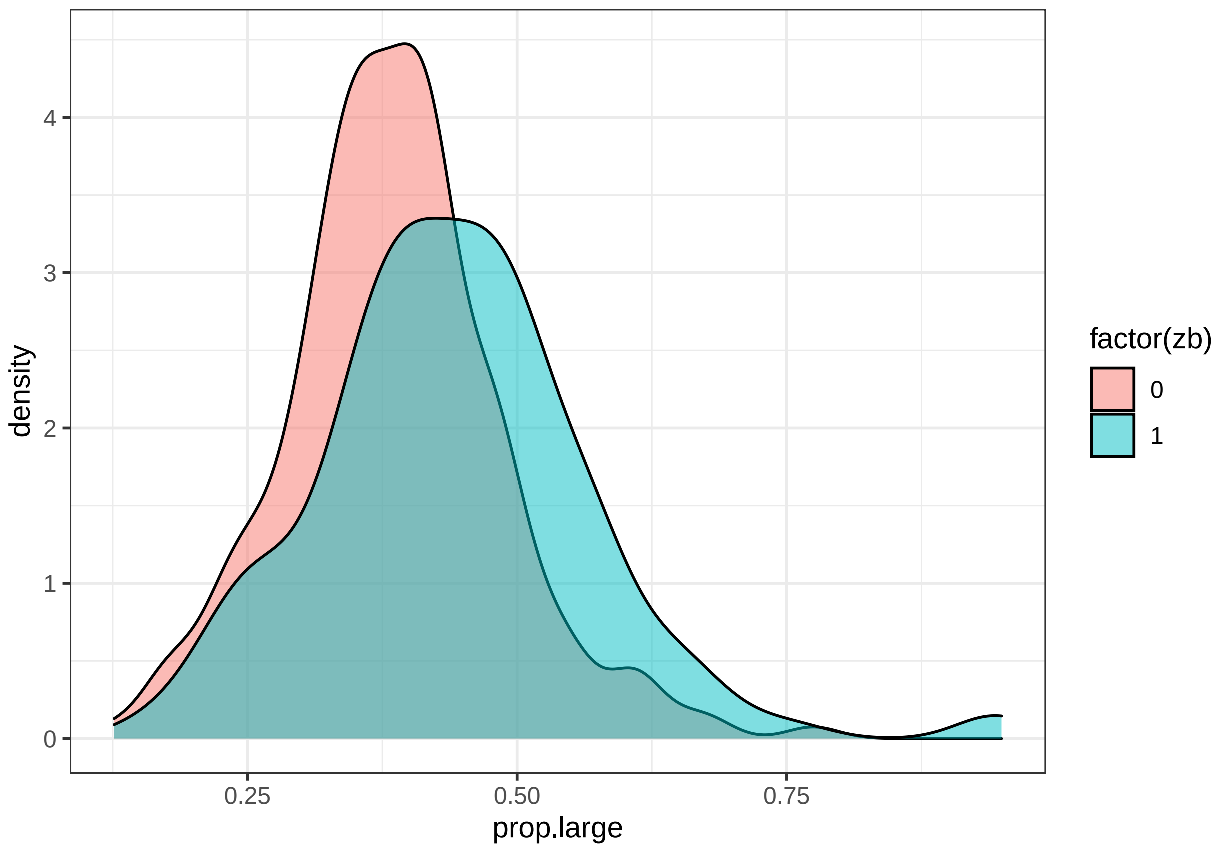
First, it is important to note, however, that the goal is not simply to get the best estimate of the propensity score in terms of mean-integrated-squared-error, or a similar criterion based on minimizing the difference between the estimated and true propensity score. Such a criterion would always suggest that using the true propensity score is preferable to using an estimated propensity score. In contrast, for our purposes, it is often preferable to use the estimated propensity score. The reason is that using the estimated score may lead to superior covariate balance in the sample compared to that achieved when using the true super-population propensity score.

This makes the simpler prop score look better since they are more balanced between treatment and control.

```
analysis %>% ggplot() +
  geom_density(aes(x=prop, fill=factor(zb)), alpha=0.3)
```

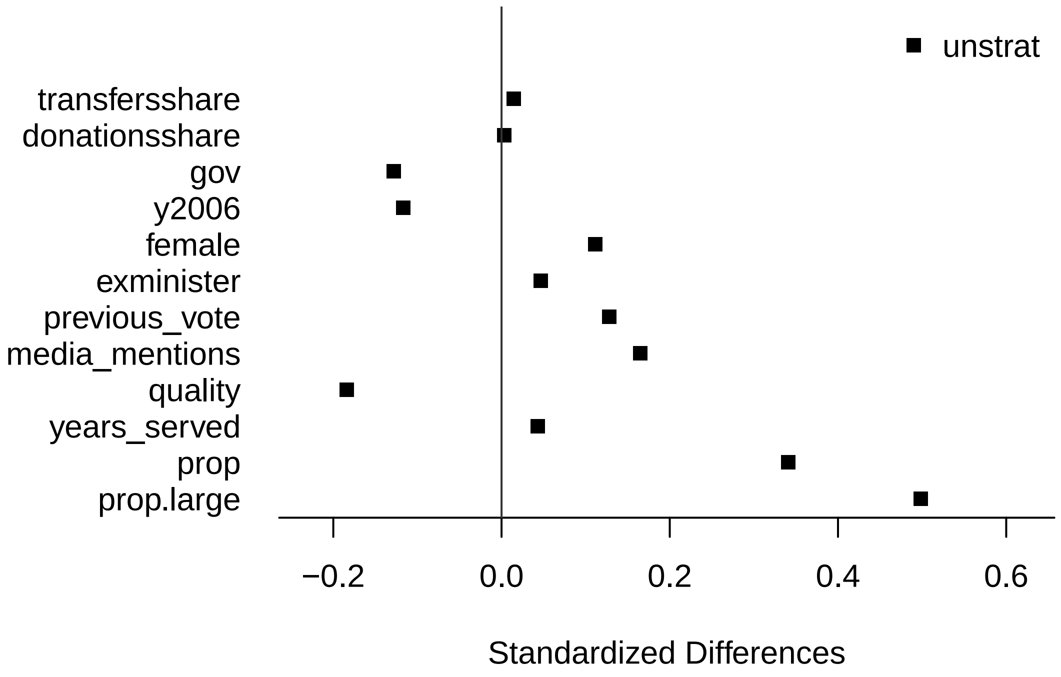


```
analysis  %>% ggplot() +
  geom_density(aes(x=prop.large, fill=factor(zb)), alpha=0.5) +
  theme_bw()
```



Matching

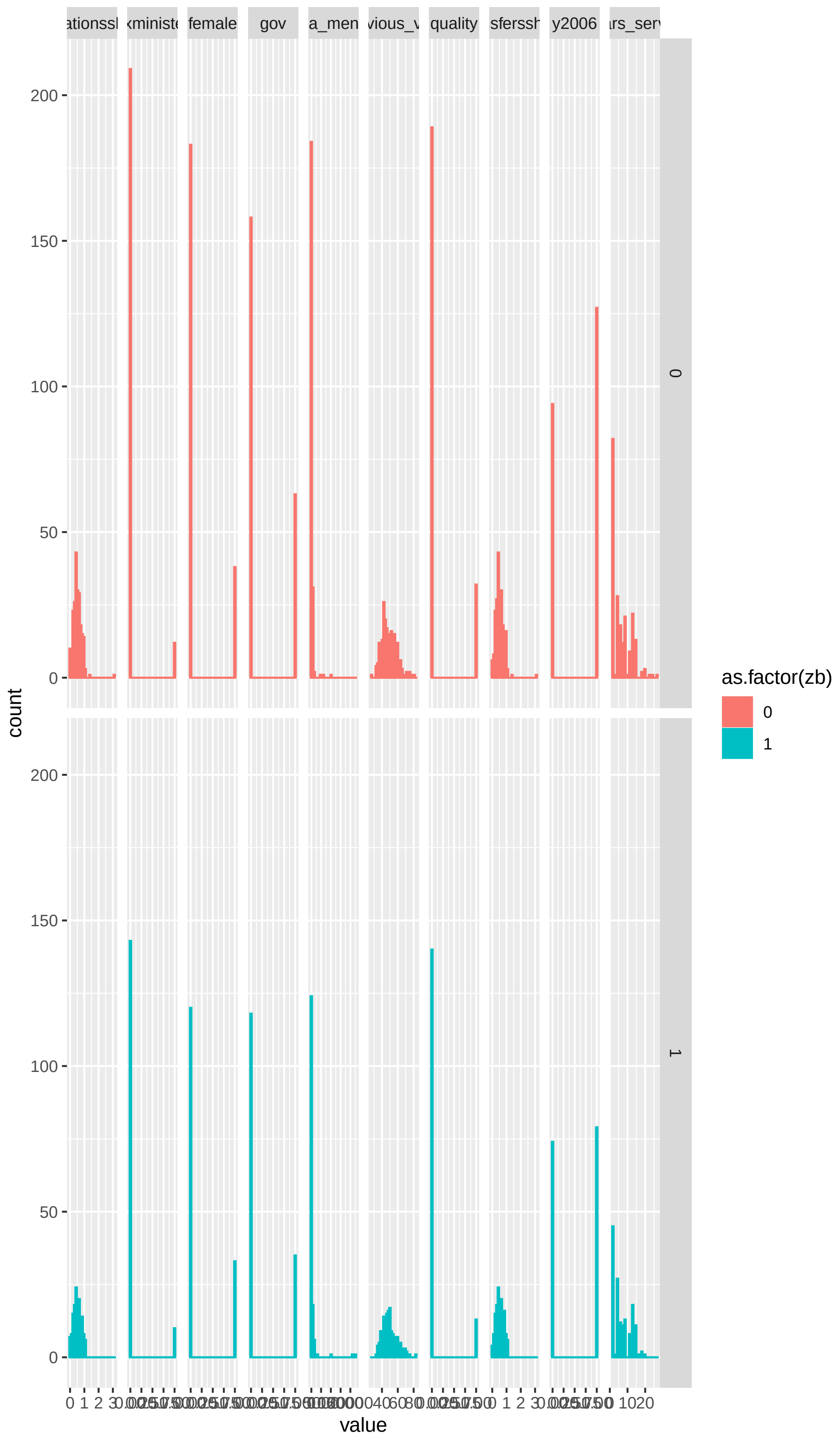
```r
plot(xBalance(zb ~ . - 1 -y, data=analysis))
```



```
# plot(xBalance(zb ~ . - 1 -y, data=analysis.large))
```

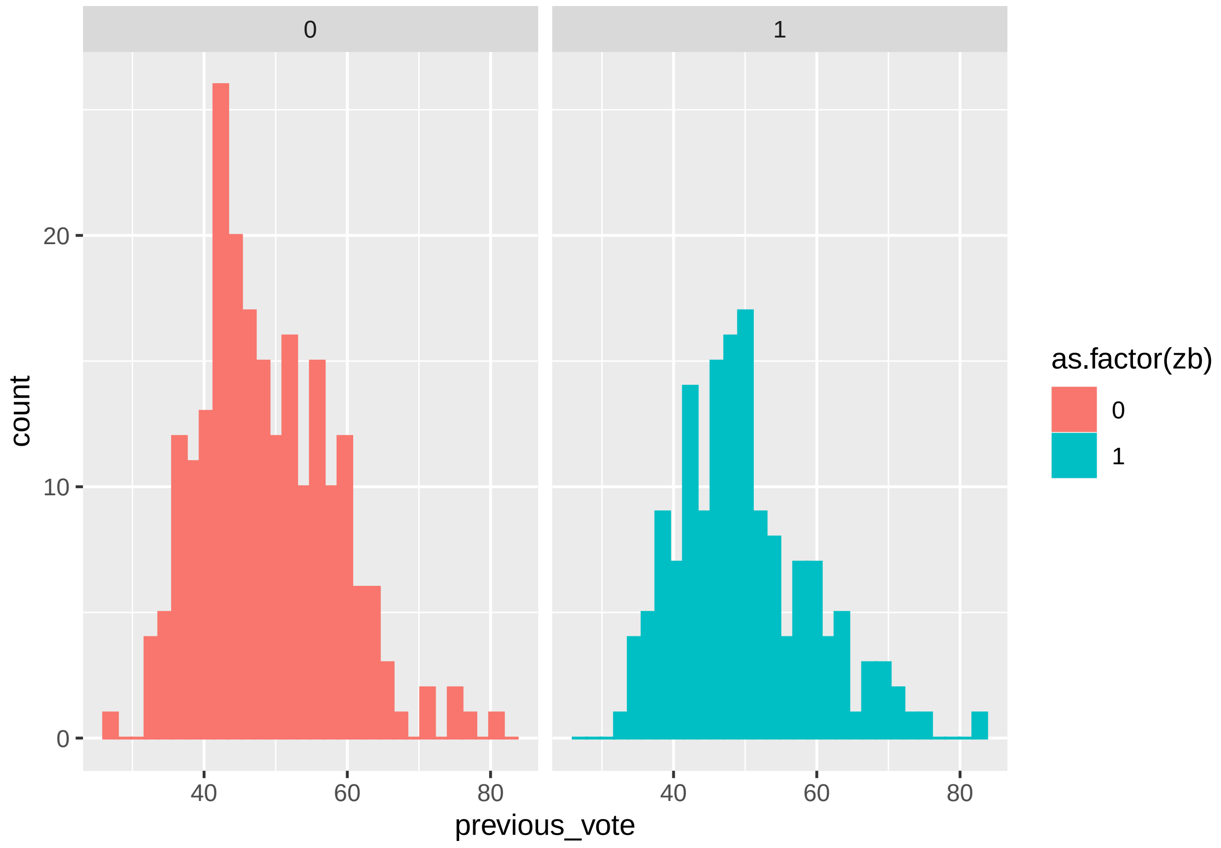
```
analysis %>%
  dplyr::select(zb, covariates) %>%
  pivot_longer(-zb, names_to="covariate", values_to="value") %>%
  ggplot(aes(x = value, color = as.factor(zb), fill = as.factor(zb))) +
  geom_histogram(bins=30) +
  facet_grid(cols=vars(covariate), rows=vars(zb), scales='free_x')
```

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

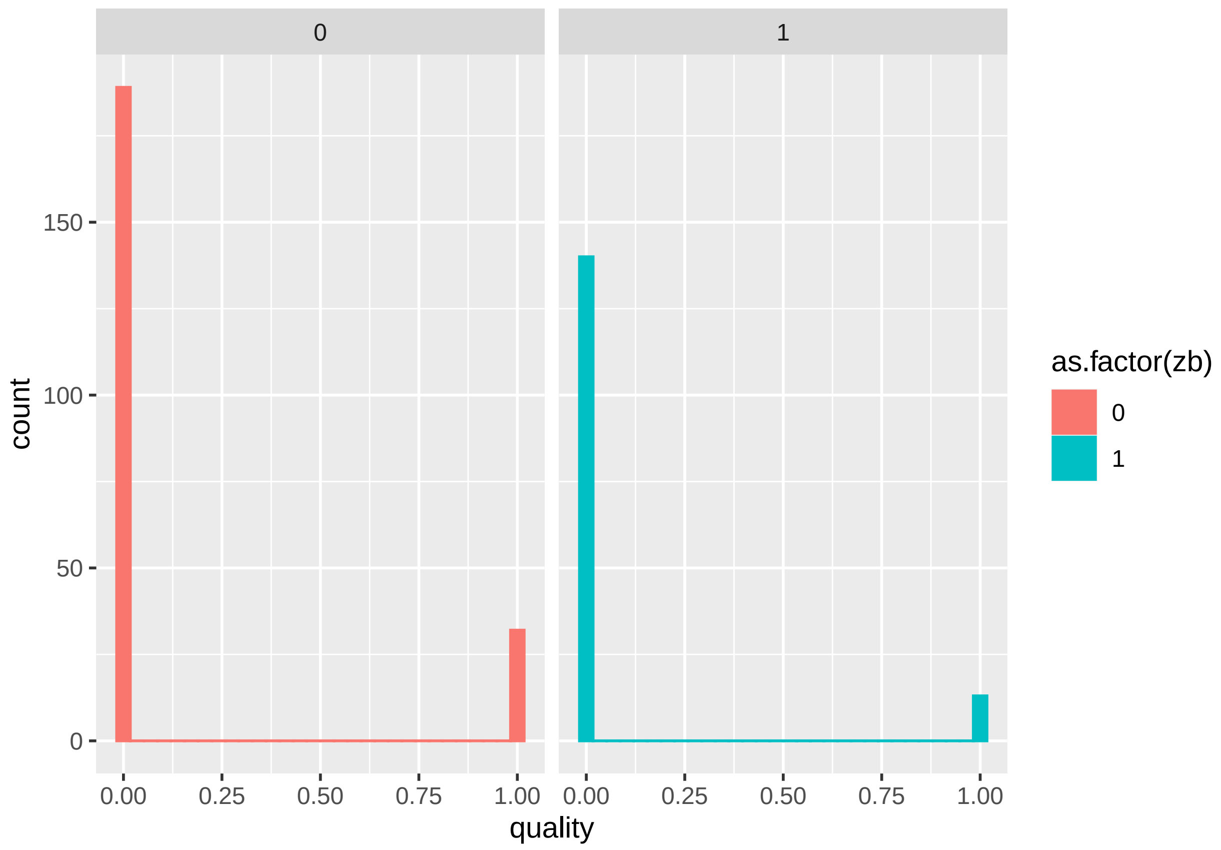


ationssl
xministe
female
gov
a_men
vious_v
quality
sferssh
y2006
rs_serv
0
1
count
value
200
150
100
50
0
as.factor(zb)
0
1

```
analysis.large %>%
  ggplot(aes(x = previous_vote, color = as.factor(zb), fill = as.factor(zb))) +
  geom_histogram(bins=30) +
  facet_wrap(.~zb)
```



```
analysis %>%
  ggplot(aes(x = quality, color = as.factor(zb), fill = as.factor(zb))) +
  geom_histogram(bins=30) +
  facet_wrap(.~zb)
```

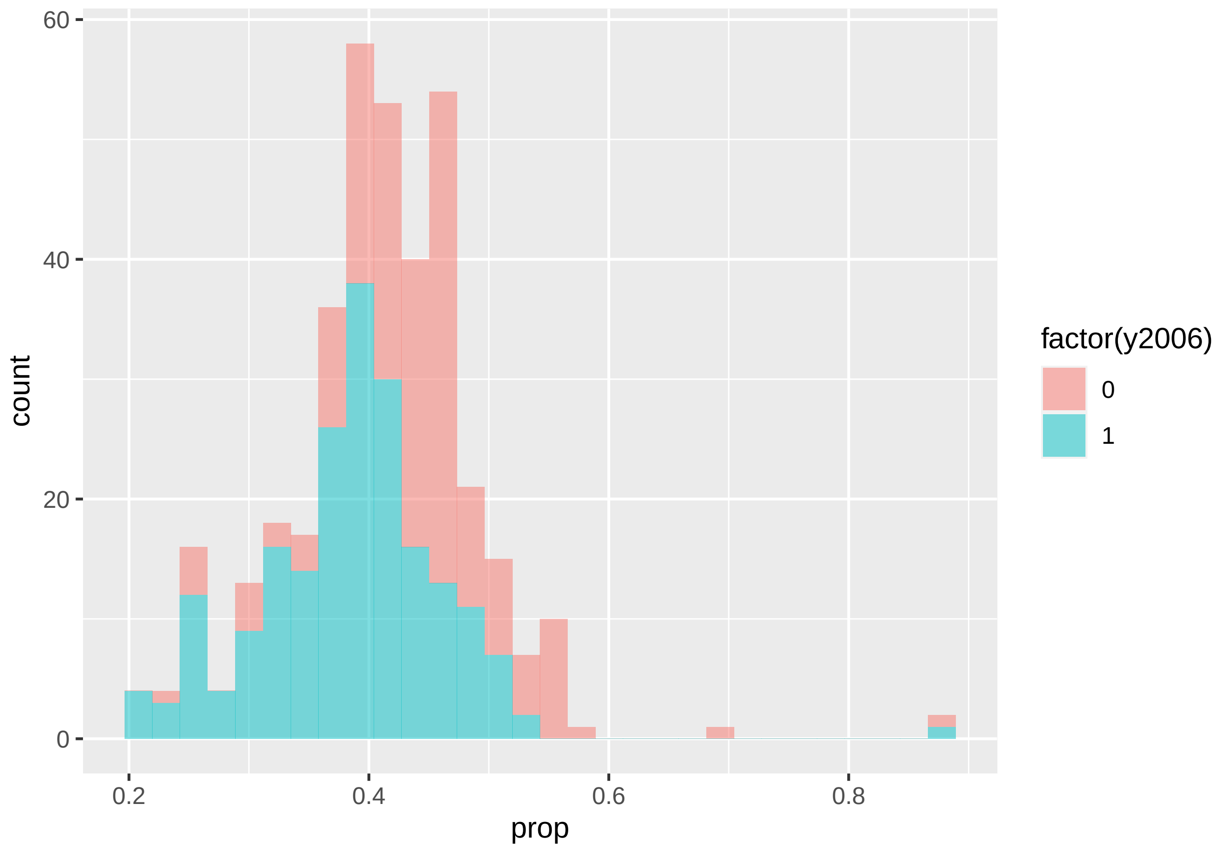


```r
names(analysis)
```

```
##  [1] "zb"              "y"               "transfersshare" "donationsshare"
##  [5] "gov"             "y2006"           "female"         "exminister"
##  [9] "previous_vote"   "media_mentions"  "quality"        "years_served"
## [13] "prop"            "prop.large"
```

```r
analysis %>%
  ggplot() +
  geom_histogram(aes(x=prop, fill=factor(y2006)), alpha=0.5)
```

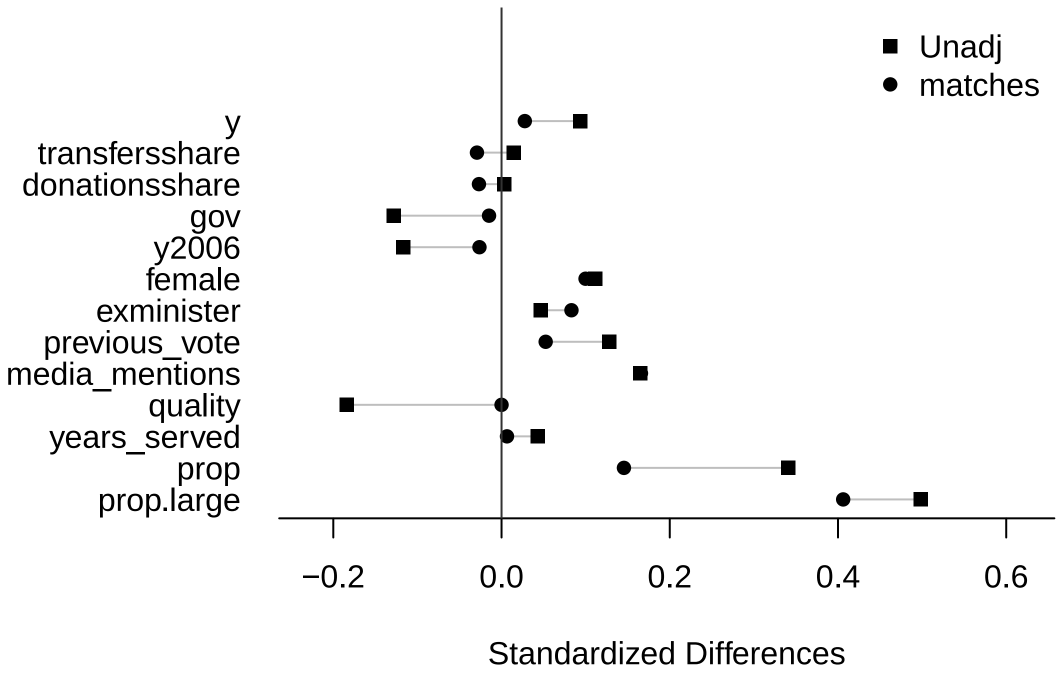
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



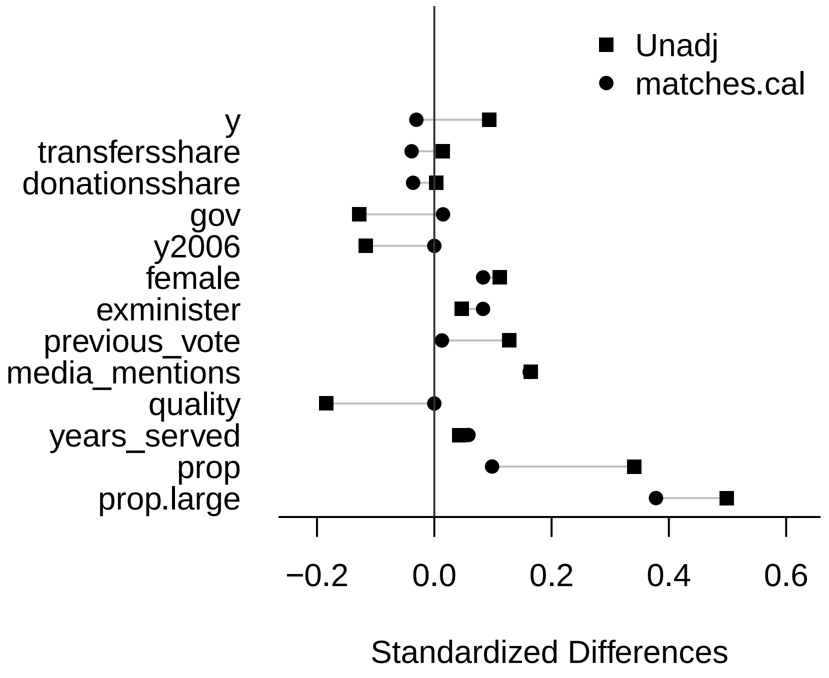
1:1 Exact Matching Note: we have the potential to match members to themselves, since ~150 had power to propose in one of the two years. However y2006 is a covariate in the distance formula and imbalance is low, meaning we are mostly matching within the same year. We could add the specification that it has to match exactly on y2006 so no member matches to themselves.

Here I do matching with an without a propensity caliper, but I end up using the propensity caliber one since the prop scores are still one of the worse balanced measures between groups.

```
z <- analysis$zb
X <- analysis %>% dplyr::select(-c(zb, prop, y, prop.large))

distance <- smahal(z, X)
distance.cal <- addcaliper(distance, z=analysis$zb, p=analysis$prop, caliper=0.1)

# analysis %>%
#   group_by(zb) %>%
#   summarise(count=n())


matches <- pairmatch(distance, data=analysis)
plot(xBalance(zb ~ . - 1 + strata(matches) , data=analysis))
```



```
matches.cal <- pairmatch(distance.cal, data=analysis)
matches.df <- summarize.match(analysis, matches.cal)

plot(xBalance(zb ~ . + strata(matches.cal) - 1, data=analysis))
```



I checked on the NAs in the matches - turns out these are just the extra control units. All treatment units are 1:1 matched with a control unit, so we should be 100% good here.

```
sum(is.na(matches)) #NAs are just extra controls !
```

```
## [1] 68
```

```r
sum(!is.na(matches))
```

```
## [1] 306
```

```r
306/2 == analysis %>% summarise(sum(zb))
```

```
##      sum(zb)
## [1,]    TRUE
```

```r
# analysis[which(is.na(matches)),]

sum(is.na(matches.cal))
```

```
## [1] 68
```

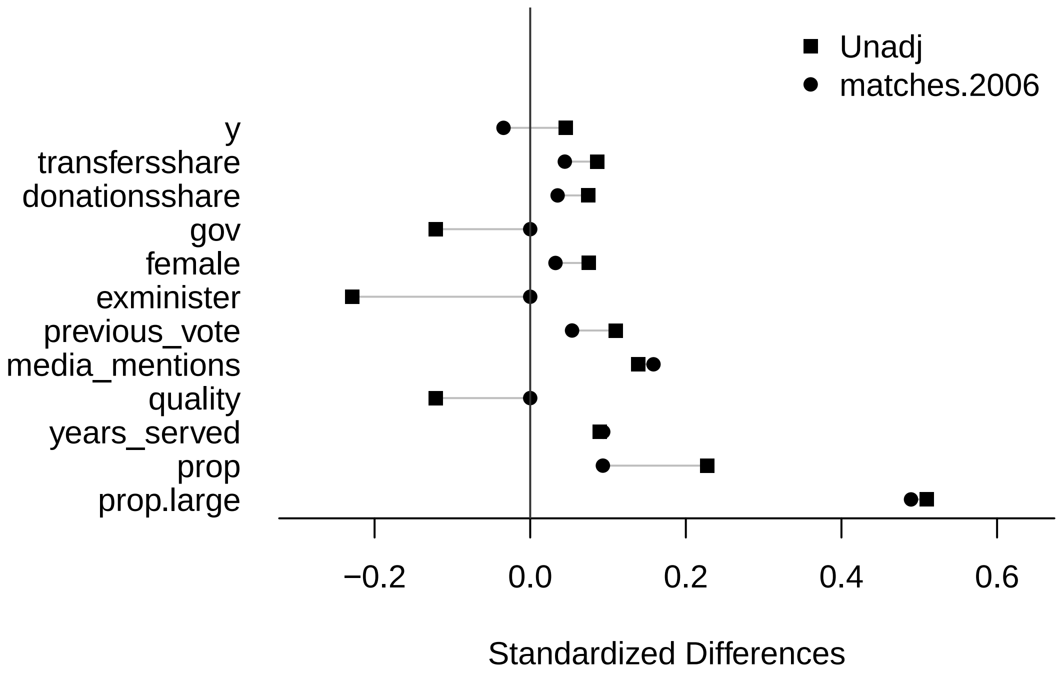
```r
sum(!is.na(matches.cal))
```

```
## [1] 306
```

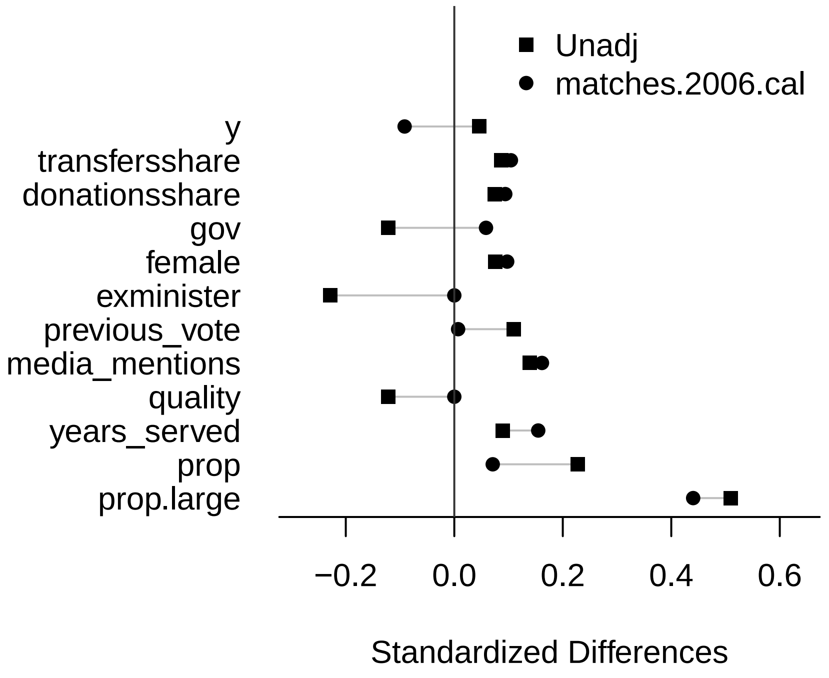
Thought doing the matching separately might be a good thing to look at - to see the effect of potentially matching members to themselves. I don't think we need to use this approach but we could mention that it was considered.

Just 2006

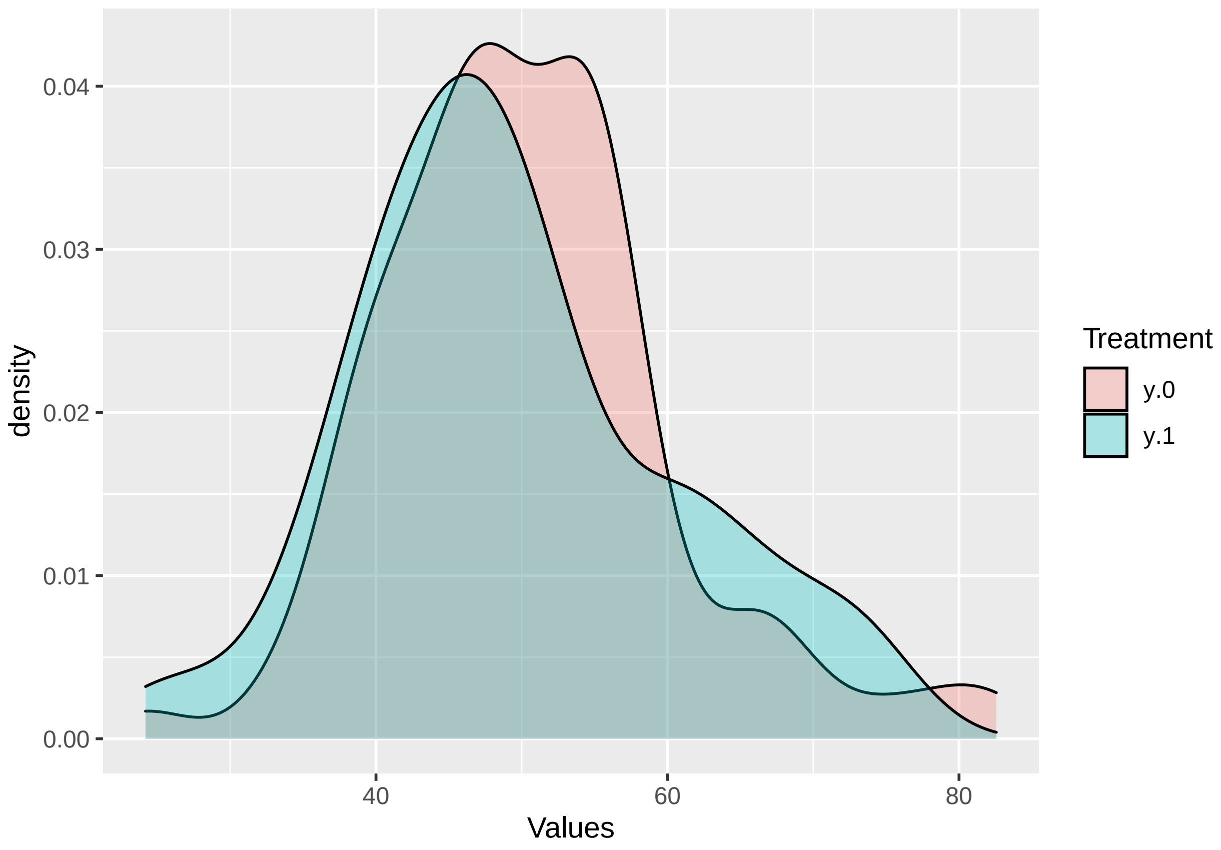
```r
analysis.2006 <- analysis %>% filter(y2006==1) %>% select(-y2006)

z <- analysis.2006$zb
X <- analysis.2006 %>% dplyr::select(-c(zb, prop, y, prop.large))



distance.2006 <- smahal(z, X)
distance.cal.2006 <- addcaliper(distance.2006, z=analysis.2006$zb, p=analysis.2006$prop, caliper=0.1)

matches.2006 <- pairmatch(distance.2006, data=analysis.2006)
plot(xBalance(zb ~ . - 1 + strata(matches.2006) , data=analysis.2006))
```



```
matches.2006.cal <- pairmatch(distance.cal.2006, data=analysis.2006)
matches.df.2006 <- summarize.match(analysis.2006, matches.2006.cal)

plot(xBalance(zb ~ . + strata(matches.2006.cal) - 1, data=analysis.2006))
```



```
matches.df.2006 %>%
  select(y.1, y.0) %>%
  pivot_longer(c(y.1, y.0),names_to='Treatment', values_to='Values') %>%
  ggplot() +
  geom_density(aes(x=Values, fill=Treatment), alpha=0.3)
```

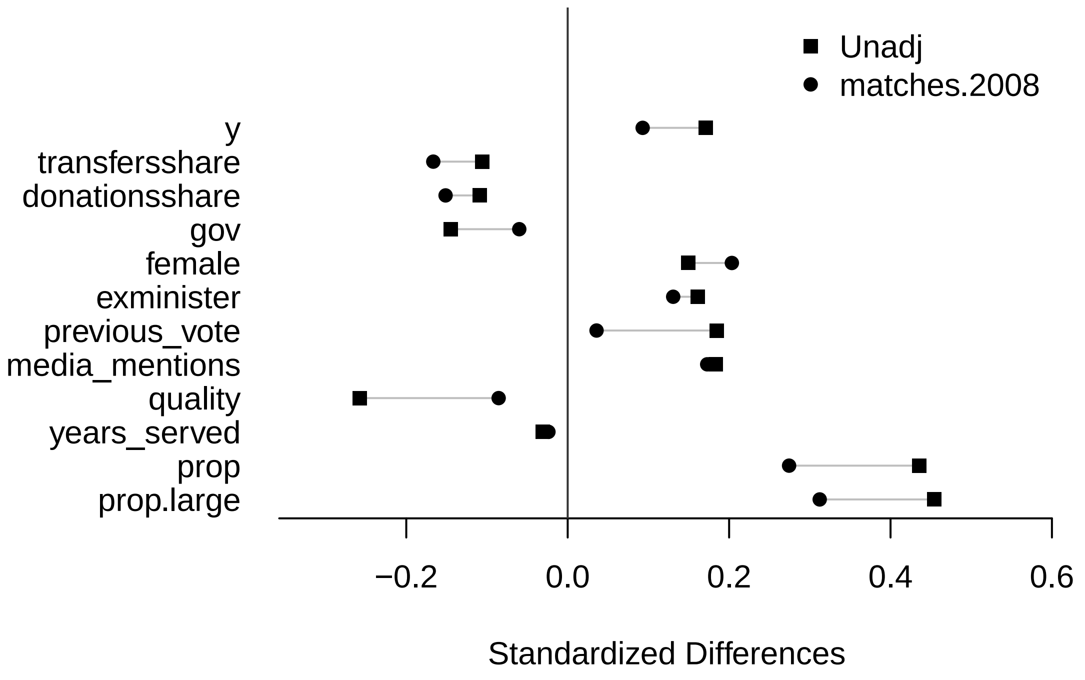


```
# DIM current assignment
T.obs.2006 <- matches.df.2006 %>%
  summarise(mean(y.1) - mean(y.0)) %>%
  pull(.)

T.obs.2006
```

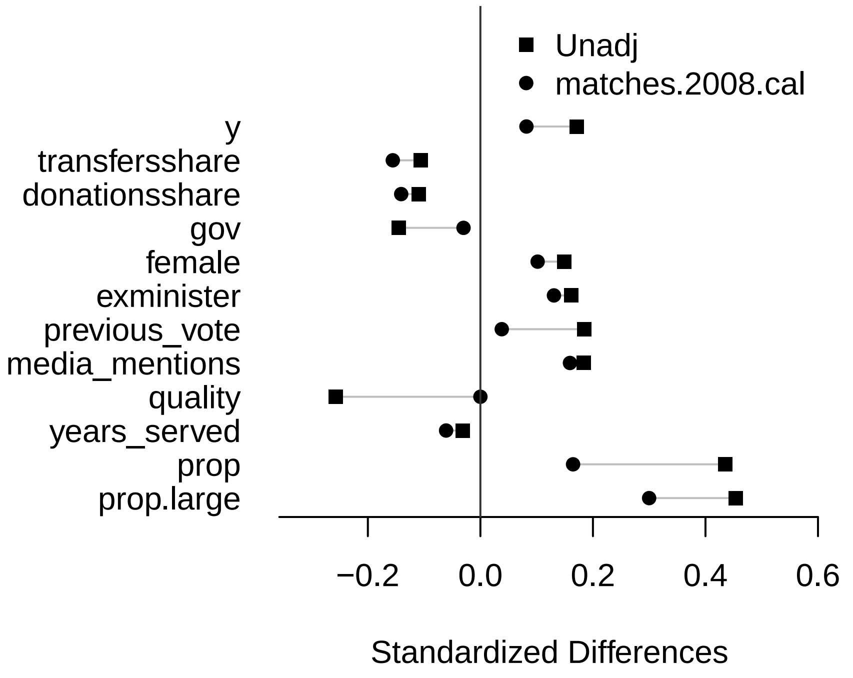
```
## [1] -0.9563288
```

Just 2008

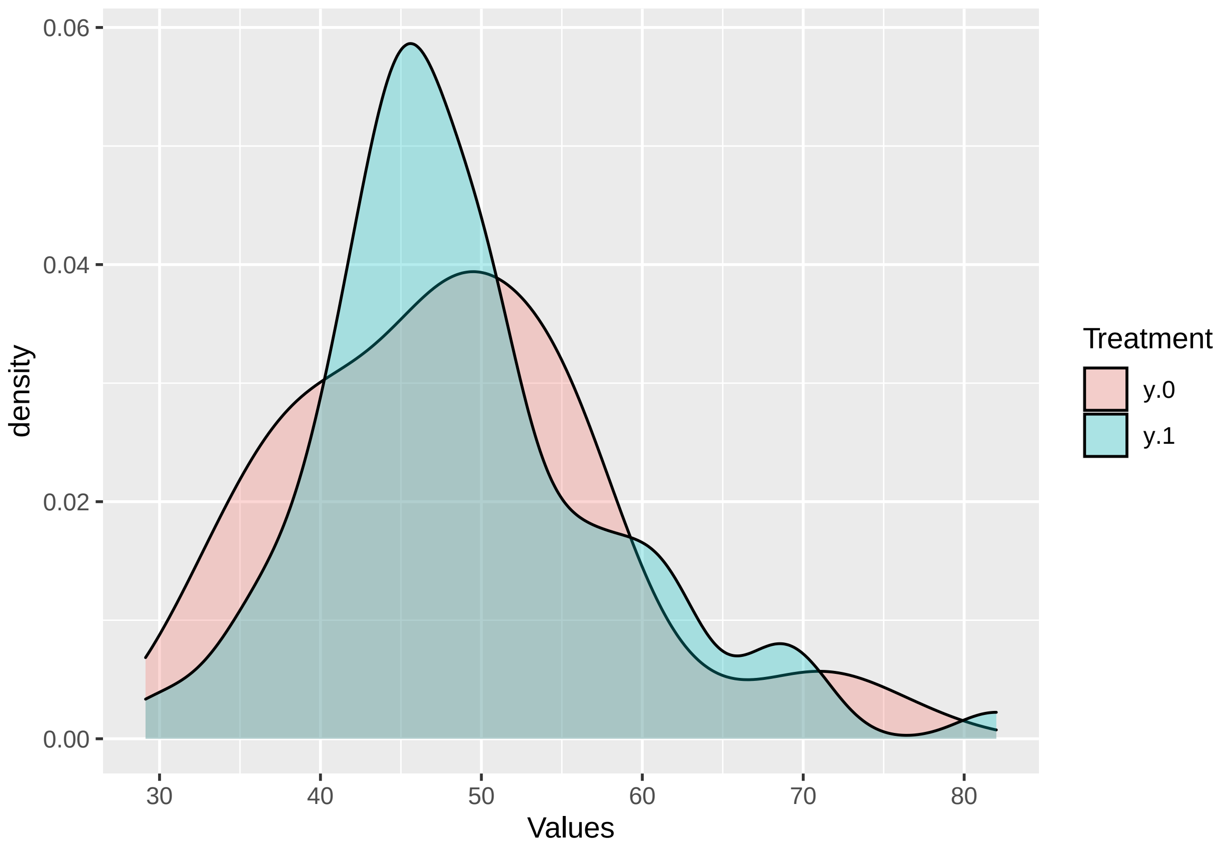
```
analysis.2008 <- analysis %>% filter(y2006==0) %>% select(-y2006)

z <- analysis.2008$zb
X <- analysis.2008 %>% dplyr::select(-c(zb, prop, y, prop.large))

distance.2008 <- smahal(z, X)
distance.cal.2008 <- addcaliper(distance.2008, z=analysis.2008$zb, p=analysis.2008$prop, caliper=0.1)

matches.2008 <- pairmatch(distance.2008, data=analysis.2008)
plot(xBalance(zb ~ . - 1 + strata(matches.2008) , data=analysis.2008))
```



```
matches.2008.cal <- pairmatch(distance.cal.2008, data=analysis.2008)
matches.df.2008 <- summarize.match(analysis.2008, matches.2008.cal)

plot(xBalance(zb ~ . + strata(matches.2008.cal) - 1, data=analysis.2008))
```



```
matches.df.2008 %>%
  select(y.1, y.0) %>%
  pivot_longer(c(y.1, y.0),names_to='Treatment', values_to='Values') %>%
  ggplot() +
  geom_density(aes(x=Values, fill=Treatment), alpha=0.3)
```



```
# DIM current assignment
T.obs.2008 <- matches.df.2008 %>%
  summarise(mean(y.1) - mean(y.0)) %>%
  pull(.)

T.obs.2008
```

```
## [1] 0.770676
```

```

analysis %>%
  group_by(zb) %>%
  summarise(count=n() / nrow(analysis))
```

1:2 Approximate Matching

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

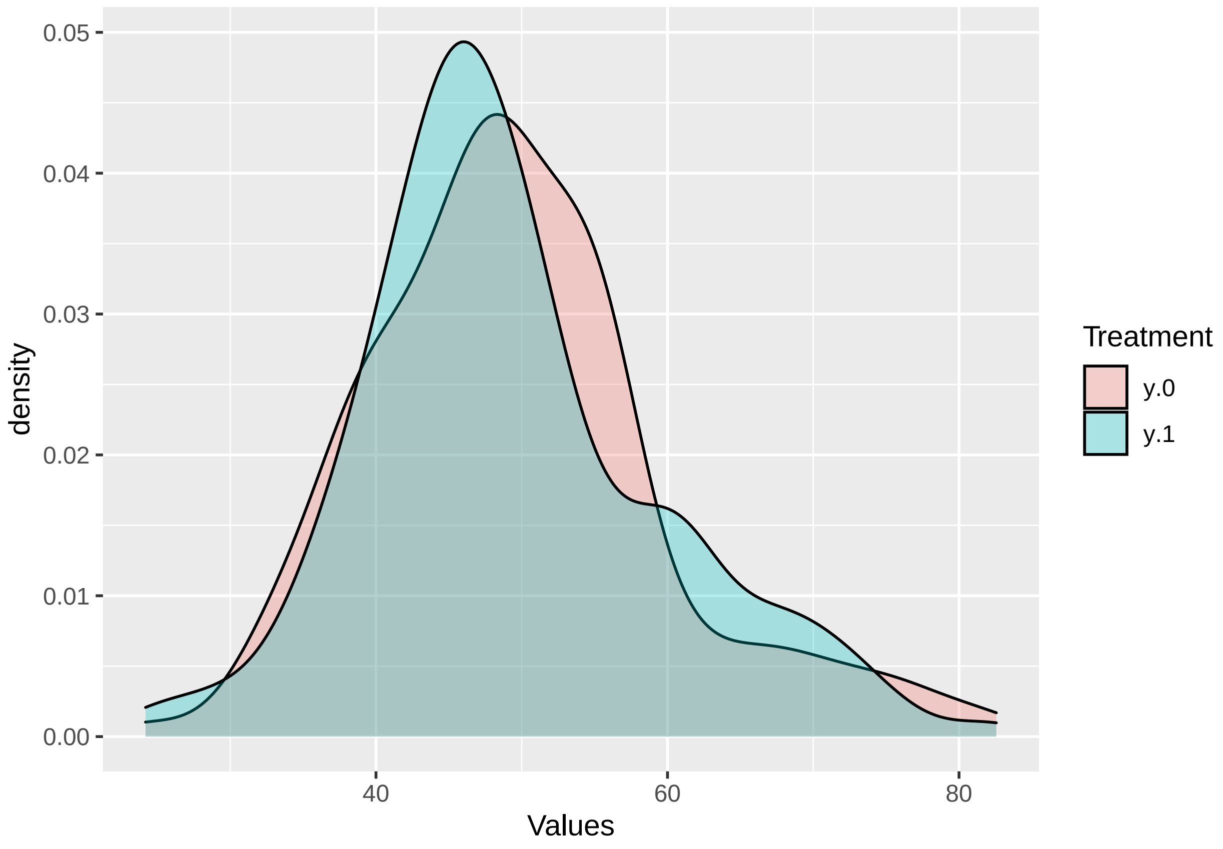
```
## # A tibble: 2 x 2
##      zb count
##   <int> <dbl>
## 1     0 0.591
## 2     1 0.409
```

Not sure we have enough controls to do more than a 1:1 matching

FRT

In matched pairs, the vote share received looks nearly identical.

```r
matches.df %>%
  select(y.1, y.0) %>%
  pivot_longer(c(y.1, y.0),names_to='Treatment', values_to='Values') %>%
  ggplot() +
  geom_density(aes(x=Values, fill=Treatment), alpha=0.3)
```



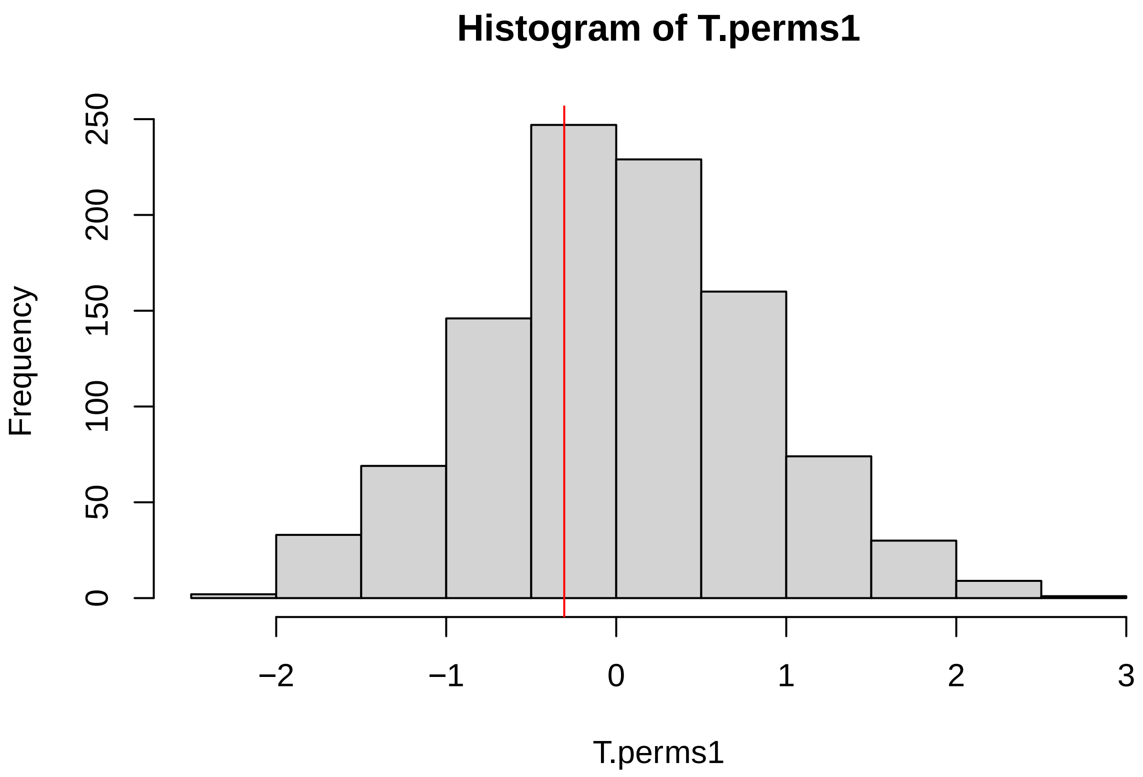
Run the FRT and our p-value is large, and tau (T.obs) small and opposite the expected sign.

```r
# DIM current assignment
T.obs1 <- matches.df %>%
  summarise(mean(y.1) - mean(y.0)) %>%
  pull(.)

T.obs1 #T obs small and negative!
```

```
## [1] -0.3057513
```

```r
genPermute <- function(x, matches) {
  treated_unit <- sample(c(0,1), nrow(matches), replace=TRUE)
  matches %>%
    select(y.0, y.1) %>%
    mutate(treated=treated_unit) %>%
    mutate(treated_y = ifelse(treated == 1, y.1, y.0),
           control_y = ifelse(treated == 1, y.0, y.1)) %>%
    summarise(mean(treated_y) - mean(control_y)) %>%
    pull(.)
}
```

```r
set.seed(123)
iters <- 1000
reps <- rep(NA,iters)

# Generate vector of test statistics under permutations
T.perms1 <- sapply(reps, function(x) genPermute(x, matches.df))
hist(T.perms1)
abline(v=T.obs1, col='red')
```

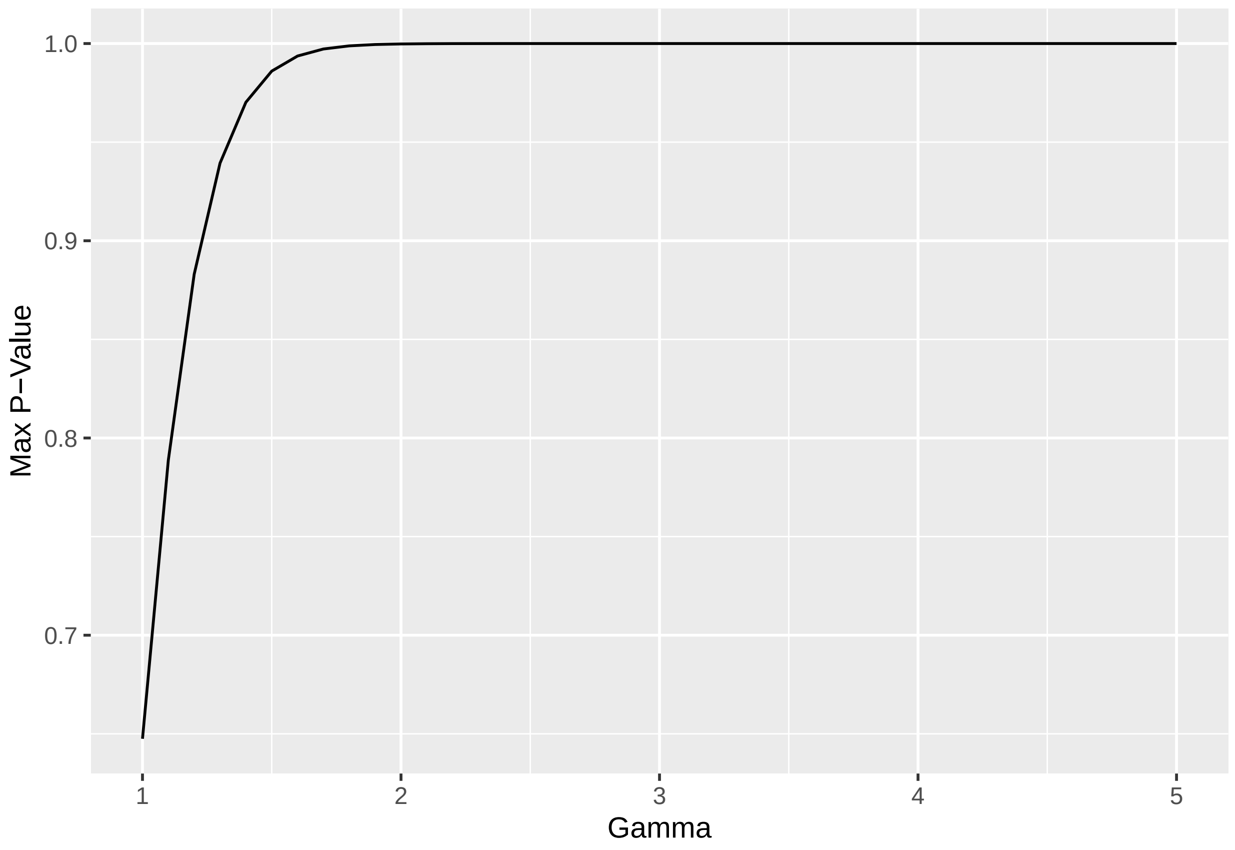


```r
pval <- (1/iters) * (sum(ifelse(T.perms1 >=  T.obs1, 1, 0)) ) #calculate one sided p-value
pval
```

```
## [1] 0.652
```

Run sensitivity analysis over all values of gamma needed. Uninterestingly since it starts insignificant it only gets more so.

```r
senm.data1 <- cast.senm(analysis, matches.cal)
sen.out <- sensitivitymult::senm(senm.data1$y, senm.data1$z, senm.data1$mset, gamma=1.2, trim = Inf, in

gamma.compute <- function(gamma) sensitivitymult::senm(senm.data1$y, senm.data1$z, senm.data1$mset, gam
grange = seq(1,5,by=0.1)
sensitivity <- sapply(grange, gamma.compute)

data.frame(gamma=grange, pvalue=sensitivity) %>%
  ggplot(aes(x=gamma, y=sensitivity)) +
  geom_line() +
  labs(x="Gamma", y="Max P-Value")
```



IPW Estimators

Horvitz Thompson I fixed the HT estimator - the denominator was wrong. The nice thing about these IPW estimators is they do not use the matches, so kind of a separately analytical check that there is an issue with the raw difference in means.

```
horvitz.thompson <- analysis %>%
  mutate(ZY1 = (zb * y)/prop) %>%
  mutate(ZY0 = ((1-zb) * y)/(1-prop)) %>%
  summarise(
    Units = n(),
    YBar1 = sum(ZY1) / nrow(analysis),
    YBar0 = sum(ZY0) / nrow(analysis),
    .groups='drop'
    ) %>%
  mutate(DIM=(YBar1 - YBar0)) %>%
  summarise(sum(DIM)) %>%
  pull(.)

horvitz.thompson
```

```
## [1] 0.3081014
```

```
analysis %>%
  mutate(ZY1 = (zb * y)/prop.large) %>%
  mutate(ZY0 = ((1-zb) * y)/(1-prop.large)) %>%
  summarise(
    Units = n(),
```

```r
    YBar1 = sum(ZY1) / nrow(analysis),
    YBar0 = sum(ZY0) / nrow(analysis),
    .groups='drop'
    ) %>%
  mutate(DIM=(YBar1 - YBar0)) %>%
  summarise(sum(DIM)) %>%
  pull(.)
```

```
## [1] 0.5673222
```

```r
hayek <- analysis %>%
  mutate(ZY1 = (zb * y)/prop) %>%
  mutate(ZY0 = ((1-zb) * y)/(1-prop)) %>%
  summarise(
    Units = n(),
    YBar1 = sum(ZY1) / sum(zb / prop),
    YBar0 = sum(ZY0) / sum((1-zb)/(1-prop)),
    .groups='drop'
    ) %>%
  mutate(DIM=(YBar1 - YBar0)) %>%
  summarise(sum(DIM)) %>%
  pull(.)

hayek
```

Hayek

```
## [1] -0.148164
```

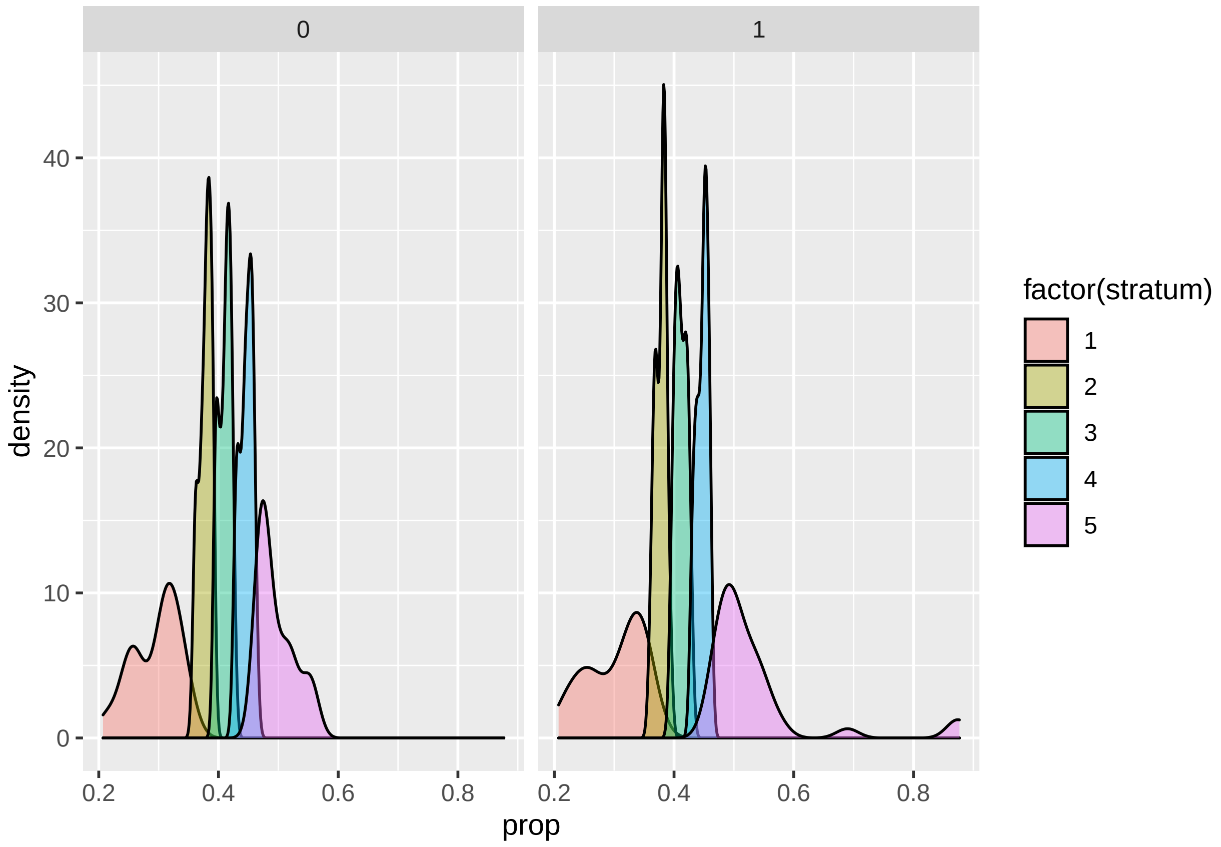
Subclassification with Neymanian CI's

```r
quant.vec <- quantile(analysis$prop, c(0.2, 0.4, 0.6, 0.8))
```

```r
stratified <- analysis %>%
  mutate(stratum = case_when(
    prop < quant.vec[1] ~ 1,
    (quant.vec[1] <= prop) & (prop < quant.vec[2]) ~ 2,
    (quant.vec[2] <= prop) & (prop < quant.vec[3]) ~ 3,
    (quant.vec[3] <= prop) & (prop < quant.vec[4]) ~ 4,
    prop >= quant.vec[4] ~ 5
  ))

# Number of units by stratum and treatment status
stratified %>%
  group_by(stratum) %>%
  summarise(
    treated = sum(zb),
    control = sum(1-zb),
    .groups='drop'
    )
```

```
## # A tibble: 5 x 3
##   stratum treated control
##     <dbl>   <int>   <dbl>
## 1         1       25       50
## 2         2       25       50
## 3         3       25       49
## 4         4       44       31
## 5         5       34       41
```

```
stratified %>%
  ggplot() +
  geom_density(aes(x=prop, fill=factor(stratum)), alpha=0.4) +
  facet_wrap(vars(zb))
```



```
tau_k <- stratified %>%
  mutate(ZY1 = zb * y) %>%
  mutate(ZY0 = (1-zb) * y) %>%
  group_by(stratum) %>%
  summarise(
    Units = n(),
    YBar1 = sum(ZY1) / sum(zb),
    YBar0 = sum(ZY0) / sum(1-zb),
    .groups='drop'
    ) %>%
  mutate(DIM=(YBar1 - YBar0) * Units/nrow(stratified)) %>%
  summarise(sum(DIM)) %>%
  pull(.)

tau_k
```

```
## [1] 0.1906014
```

```r
tau_k.var <- stratified %>%
  group_by(stratum, zb) %>%
  summarise(
    N = n(),
    Var = var(y),
    .groups='drop'
    ) %>%
  mutate(weighted_V = Var / N) %>%
  group_by(stratum) %>%
  summarise(
    stratum_var = sum(weighted_V) * (sum(N) / nrow(stratified))^2,
    .groups='drop'
    ) %>%
  summarise(sum(stratum_var)) %>%
  pull(.)

tau_k.var
```

```
## [1] 1.036314
```

Compute a 95% confidence interval

```r
normalCI <- function(tau, variance) {
  c(tau - sqrt(variance)*qnorm(.975), tau + sqrt(variance)*qnorm(.975))
}

normalCI(tau_k, tau_k.var)
```

```
## [1] -1.804632  2.185835
```

Null result