Lab 2: OLS and Matching

Ray Caraher

2025-02-24

Section 1

Setup

Setting up our script

Before we get into any real coding, let's make sure that the preamble for our code looks good. Here is how I set it up:

A bunch of text detailing how the loading of the packages should print when this is run.

```
## Load packages

library(haven)
library(lmtest)
library(stargazer)
library(tidyverse)
## Set options

options(scipen = 999)

## Clear environment

rm(list = ls())

## Set directories

base_directory <- '/Users/rcaraher/Library/CloudStorage/OneDrive-UniversityofMassachusetts/Academic/Teaching/EC
data_directory <- file.path(base_directory, 'Data')
results_directory <- file.path(base_directory, 'Results')
```

A note about Working Directories

In R, I generally recommend you set your *working directory to the source file location*.

- You can do this in RStudio by going to: Session -> Set Working Directory -> To Source File Location. This folder is usually associate with a Git repo.

I save the data and results files on cloud storage. I use the `base_directory` object to tell R where that location is as a file path. The `data_directory` and `results_directory` add one more layer to the file path (to a Data and Results folder, respectively) to save a typing when loading data/saving results.

- If you have a similar workflow, the only line you should need to change is the file path for the `base_directory` object.
- The `file.path()` function provides an OS-agnostic way to specify file paths.

Section 2

OLS in R

Loading Data

Let's read in the Census data and work with that for our lab.

```
census <- read_dta(file.path(data_directory, "census_sample_30_50.dta"))
```

Glimpse variables

The `glimpse()` function allows us to view the variables, the class, and the first few rows of data.

```
glimpse(census)
```

```
## Rows: 65,741
## Columns: 109
## $ year          <dbl+lbl> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000~
## $ hhwt          <dbl> 25, 16, 26, 3, 22, 2, 7, 21, 8, 17, 7, 36, 14, 6, 10, 29~
## $ region        <dbl+lbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ stateicp      <dbl+lbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ statefips     <dbl+lbl> 42,  6, 47,  6, 13, 12, 39,  6, 53, 53, 42, 48, 34, ~
## $ metro         <dbl+lbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ metaread      <dbl+lbl> 3240, 4482,    0, 8780,    0,    0,    0, 6920,    0~
## $ city          <dbl+lbl>    0, 2430,    0,    0,    0,    0,    0,    0,    0~
## $ citypop       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ puma          <dbl> 3102, 7100, 600, 3503, 3300, 800, 1600, 1505, 300, 2003,~
## $ pumasupr      <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ conspuma      <dbl> 405, 45, 444, 50, 88, 78, 372, 40, 505, 498, 409, 466, 3~
## $ cntry         <dbl+lbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ gq            <dbl+lbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ perwt         <dbl> 21, 16, 21, 2, 8, 5, 6, 24, 5, 26, 7, 42, 14, 4, 13, 29,~
## $ age           <dbl+lbl> 45, 45, 33, 46, 37, 48, 45, 37, 40, 38, 39, 32, 33, ~
## $ sex           <dbl+lbl> 2, 2, 1, 2, 1, 2, 1, 1, 2, 2, 1, 1, 1, 2, 1, 2, 2, 1~
```

Cleaning Data

Let's say we want to estimate the wage penalty of being a foreign-born worker. In other words, we want to run the following regression:

$$y_i = \beta_0 + \beta_1 FB_i + X_i\Omega + \epsilon_i$$

where y is log wage, FB is a indicator variable (i.e., 0 or 1) if the worker is a non-citizen, X_i is a vector of covariates, and ϵ_i is the error term.

We first need to make sure our data is cleaned and in the correct format to run these regressions.

Looking at the outcome variable

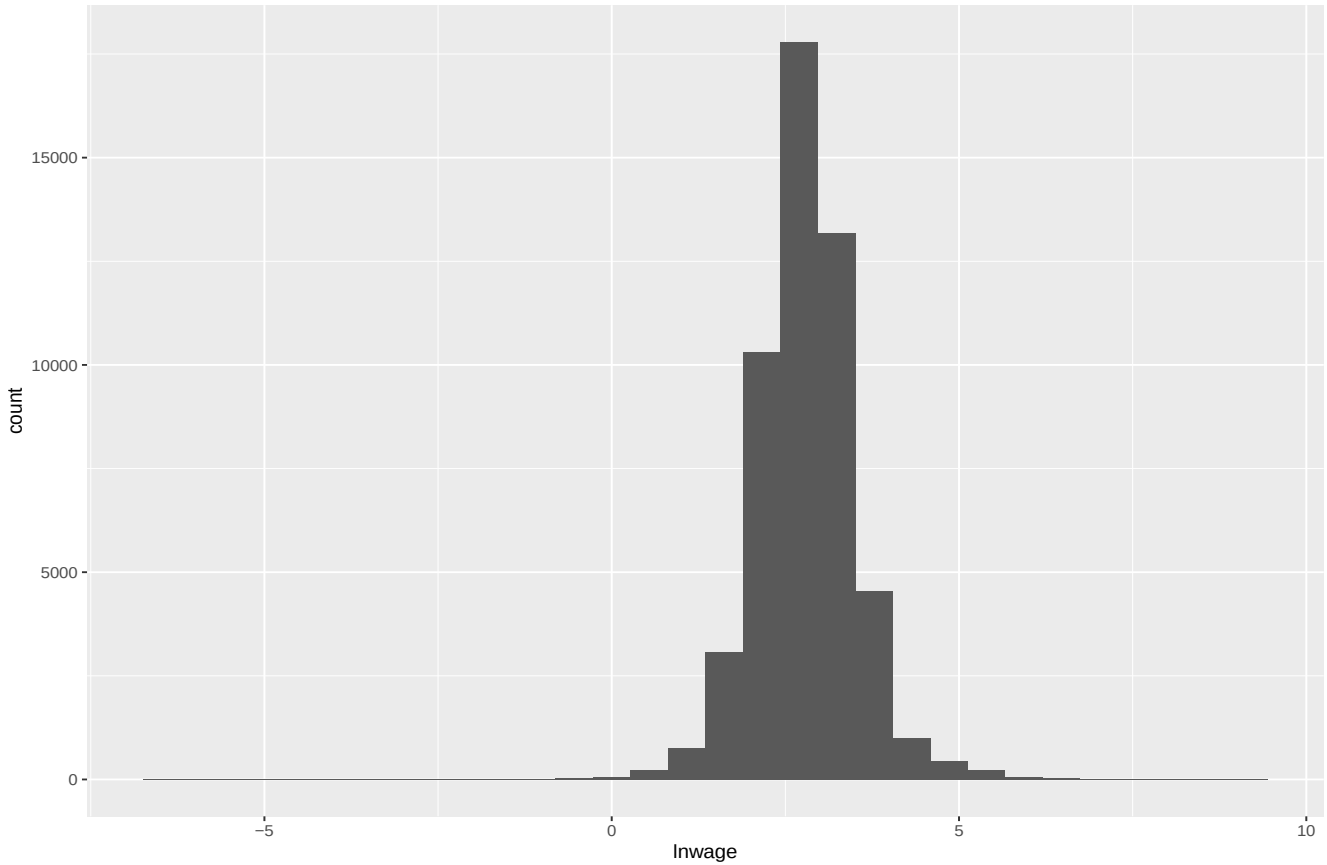
Let's do a quick histogram of the outcome variable to make sure it looks reasonable. We can use the ggplot commands here to generate the figure.

```
ggplot(data = census) +
  geom_histogram(aes(x = lnwage))
```

Looking at the outcome variable

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

## Warning: Removed 13925 rows containing non-finite outside the scale range
## (`stat_bin()`).
```



Cleaning the treatment variable

We can use the `count()` function to take a look at the treatment variable and see how it is organized.

```
count(census, citizen)
```

```
## # A tibble: 4 x 2
##   citizen                                    n
##   <dbl+lbl>                              <int>
## 1 0 [N/A]                                58406
## 2 1 [Born abroad of American parents]      497
## 3 2 [Naturalized citizen]                 2823
## 4 3 [Not a citizen]                       4015
```

Cleaning the treatment variable

Let's set all those who are not a citizen (value of 3) as our foreign-born binary indicator We can use the `mutate()` and `case_when` functions to re-code variables (look at the documentation for these!).

```
census <- census |>
  mutate(fb = case_when(citizen == 3 ~ 1,
                        TRUE ~ 0))

count(census, citizen, fb)
```

```
## # A tibble: 4 x 3
##   citizen                                  fb     n
##   <dbl+lbl>                             <dbl> <int>
## 1 0 [N/A]                                   0 58406
## 2 1 [Born abroad of American parents]       0   497
## 3 2 [Naturalized citizen]                   0  2823
## 4 3 [Not a citizen]                         1  4015
```

Cleaning the control variables

We also want to control for marital status, experience, race, ethnicity, education, English-speaking, and gender. We can use a similar method to get these variables in the format appropriate for a linear regression.

```
census <- census |>
  mutate(english = case_when(speakeng == 1 ~ 0,
                             speakeng == 6 ~ 0,
                          TRUE ~ 1))

count(census, speakeng, english)
```

```
## # A tibble: 5 x 3
##   speakeng                          english     n
##   <dbl+lbl>                           <dbl> <int>
## 1 1 [Does not speak English]              0   700
## 2 3 [Yes, speaks only English]            1 56082
## 3 4 [Yes, speaks very well]               1  5151
## 4 5 [Yes, speaks well]                    1  2214
## 5 6 [Yes, but not well]                   0  1594
```

Regression and factor variables

When doing regression in R, it can be helpful to have categorical variables (such as race) as the factor class. When factor variables are included in a regression, R will automatically create dummy variables for each possible value (minus an omitted one). Let's make our (non-binary) categorical variables into factors.

```
count(census, racesingd)
```

```
## # A tibble: 16 x 2
##    racesingd                                          n
##    <dbl+lbl>                                      <int>
##  1 10 [White]                                     53434
##  2 12 ['Other race', Hispanic]                     2719
##  3 20 [Black]                                      6403
##  4 30 [AI (American Indian)]                        713
##  5 31 [AN (Alaskan Native)]                         217
##  6 32 [AI/AN (American Indian/Alaskan Native)]      197
##  7 40 [Asian Indian]                                305
##  8 41 [Chinese]                                     460
##  9 42 [Filipino]                                    337
## 10 43 [Japanese]                                    142
## 11 44 [Korean]                                      183
## 12 45 [Asian]                                       445
```

Section 3

Running a Regression

The `lm()` function

The default way to run a linear regression in R is with the `lm()` function. We can then examine the results using the `summary()` function on the object. Let's do a bivariate estimation.

```
bi_ols <- lm(lnwage ~ fb, data = census)
summary(bi_ols)
```

Results

```
##
## Call:
## lm(formula = lnwage ~ fb, data = census)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.2645 -0.4137 -0.0053  0.4001  6.3820
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)  2.790872   0.003242  860.82 <0.0000000000000002 ***
## fb          -0.185049   0.013922  -13.29 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7177 on 51814 degrees of freedom
##   (13925 observations deleted due to missingness)
## Multiple R-squared:  0.003398,   Adjusted R-squared:  0.003379
## F-statistic: 176.7 on 1 and 51814 DF,  p-value: < 0.00000000000000022
```

Robust OLS

The default OLS standard error reported will not correctly adjust for heteroskedasticity. Using the `coeftest()` function from the `sandwhich` package, we can get adjust the var-cov matrix.

```
bi_ols_r <- coeftest(bi_ols, vcov. = vcovHC(bi_ols, type = "HC1"))
bi_ols_r
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value                Pr(>|t|)
## (Intercept)  2.7908717  0.0032249 865.404 < 0.00000000000000022 ***
## fb          -0.1850494  0.0150832 -12.268 < 0.00000000000000022 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Multivariate OLS

Let's add the additional control variables:

```
mul_ols <- lm(lnwage ~ fb + married + exp + exp2 + race +
              hisp + edu + english + gender, data = census)
```

Multivariate OLS Results

```
summary(mul_ols)
```

```
##
## Call:
## lm(formula = lnwage ~ fb + married + exp + exp2 + race + hisp +
##     edu + english + gender, data = census)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5400 -0.3374  0.0063  0.3441  6.6215
##
## Coefficients:
##                                                 Estimate  Std. Error t value
## (Intercept)                                   1.73602463  0.05732371  30.285
## fb                                           -0.05992353  0.01574691  -3.805
## married                                       0.09813295  0.00621820  15.782
## exp                                           0.02827203  0.00394290   7.170
## exp2                                         -0.00046747  0.00008901  -5.252
## race'Other race', Hispanic                   -0.02922727  0.01965982  -1.487
## raceBlack                                    -0.02547833  0.01014746  -2.511
## raceAI (American Indian)                     -0.12327925  0.02878191  -4.283
## raceAN (Alaskan Native)                       0.20213808  0.04966400   4.070
## raceAI/AN (American Indian/Alaskan Native)    0.08305657  0.05598223   1.484
## raceAsian Indian                              0.13064643  0.04381043   2.982
## raceChinese                                   0.09138875  0.03554342   2.571
## raceFilipino                                  0.09752215  0.03823153   2.551
## raceJapanese                                  0.06221922  0.06371841   0.976
## raceKorean                                    0.02130718  0.05953784   0.358
## raceAsian                                     0.00853463  0.03661039   0.233
## raceHawaiian                                  0.26516461  0.14251969   1.861
## racePI (Pacific Islander)                     0.00958198  0.09154558   0.105
## raceAsian and PI (Pacific Islander)          -0.05236747  0.14250953  -0.367
## raceOther race, non-Hispanic                  0.00961830  0.10466211   0.092
## hisp                                          0.02485940  0.01426359   1.743
```

Big OLS Models

The built-in, default R method for linear regression is fine, but when we want to run high-dimensional models, it is much easier to work with other functions. Let's use the `fixest` package for this. Install it if you don't have it yet with `install.packages("fixest")`.

```
library(fixest)
```

The FEOLS function

The `feols` function is the high-dimensional version of `lm()`. Let's use it to run the same bivariate OLS to make sure the results are the same. The syntax for this function is different. Make sure to look at the documentation with `?feols`. We can also directly tell it to report a hetero. robust SE.

```
library(fixest)

bi_feols <- feols(lnwage ~ fb,
                  vcov = "HC1",
                  data = census)
```

```
## NOTE: 13,925 observations removed because of NA values (LHS: 13,925).
```

FEOLS Results

```
summary(bi_feols)
```

```
## OLS estimation, Dep. Var.: lnwage
## Observations: 51,816
## Standard-errors: Heteroskedasticity-robust
##              Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)  2.790872   0.003225 865.4042 < 2.2e-16 ***
## fb          -0.185049   0.015083 -12.2685 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 0.717702   Adj. R2: 0.003379
```

Fully-Saturated Controls

To run a fully-saturated OLS model, it means we include all controls and interactions.

We could do this by explicitly multiplying the columns (i.e., race-black * married * edu_College * . . .) but there is a short cut!

We can create a new variable which assigns the same group number for each unique combination of our control variables!

Grouping

```
census <- census |>
  group_by(married, hisp, english, gender, race, edu) |>
  mutate(group = cur_group_id()) |>
  ungroup()

count(census, group)
```

```
## # A tibble: 1,097 x 2
##    group     n
##    <int> <int>
##  1     1     4
##  2     2     2
##  3     3     2
##  4     4     1
##  5     5     8
##  6     6     9
##  7     7     1
##  8     8     7
##  9     9     1
## 10    10     2
## # i 1,087 more rows
```

Grouping

We can now run the OLS model! We could run this regression a few ways (including making the group variable a factor), but another way is to include it as a fixed-effect (it is computationally faster!)

```
fs_ols <- feols(lnwage ~ fb + exp + exp2 |
                  group,
                  vcov = "HC1",
                  data = census)
```

```
## NOTE: 13,925 observations removed because of NA values (LHS: 13,925).
```

```
summary(fs_ols)
```

```
## OLS estimation, Dep. Var.: lnwage
## Observations: 51,816
## Fixed-effects: group: 972
## Standard-errors: Heteroskedasticity-robust
##       Estimate Std. Error  t value             Pr(>|t|)
## fb   -0.039374   0.017654 -2.23028 0.02573349125737646 *
## exp   0.028190   0.003896  7.23478 0.00000000000047284 ***
## exp2 -0.000469   0.000089 -5.28692 0.00000012490246345 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 0.641699     Adj. R2: 0.188036
```

Section 4

Matching

Intro to Matching

Including controls often isn't enough to ensure we are making apples-to-apples comparisons, especially if there is some systemic differences between the groups.

Matching methods are one way to make sure comparisons between the treated and control groups are more similar.

Propensity Score Matching

A common set of matching techniques use the propensity score, which estimates the **likelihood of treatment based on observable**.

The `MatchIt` package is able to do most types of modern matching methods.

```
library(MatchIt)
```

Propensity Score Matching in R

Let's make sure we drop all NA values for the variables we are using.

```
census_comp <- census |>
  dplyr::select(lnwage, fb, married, exp, exp2, race, hisp, edu, english, gender) |>
  na.omit()
```

Propensity Score Matching in R

Now let's use the `matchit()` function in R.

```
ps_matchout <- matchit(fb ~ married + exp + exp2 + race +
                         hisp + edu + english + gender,
                       data = census_comp,
                       method = "nearest",
                       link = "probit"
                       )
summary(ps_matchout)
```

```
##
## Call:
## matchit(formula = fb ~ married + exp + exp2 + race + hisp + edu +
##     english + gender, data = census_comp, method = "nearest",
##     link = "probit")
##
## Summary of Balance for All Data:
##                                          Means Treated Means Control
## distance                                        0.3997        0.0340
## married                                         0.6495        0.6714
## exp                                            20.3100       22.0917
## exp2                                          444.2075      522.7215
## raceWhite                                       0.4480        0.8439
## race'Other race', Hispanic                      0.2790        0.0247
## raceBlack                                       0.0673        0.0942
```

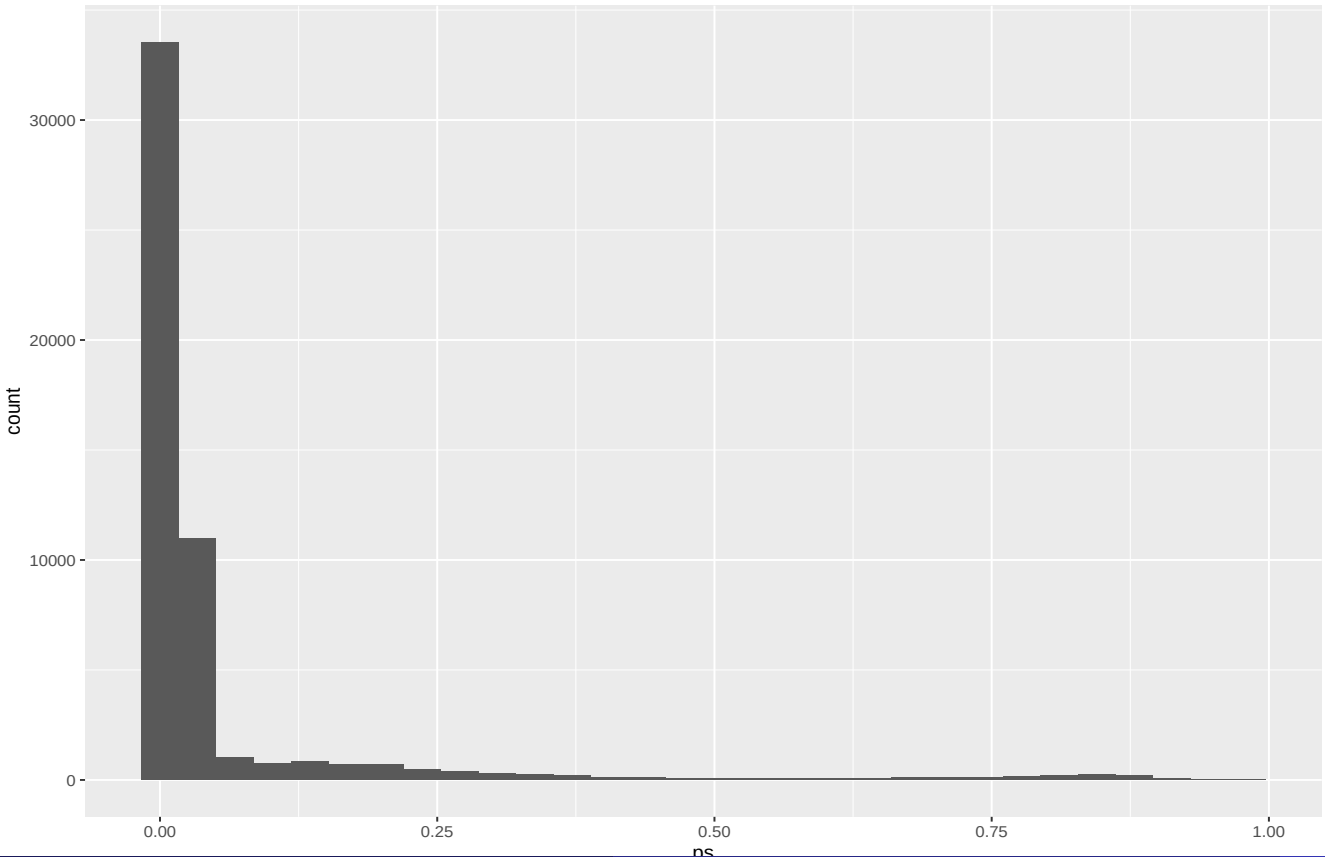
Propensity Score Matching in R

Let's look at the estimate probabilities:

```
census_comp <- census_comp |>
  bind_cols(ps = ps_matchout$distance)

ggplot(data = census_comp) +
  geom_histogram(aes(x = ps))
```

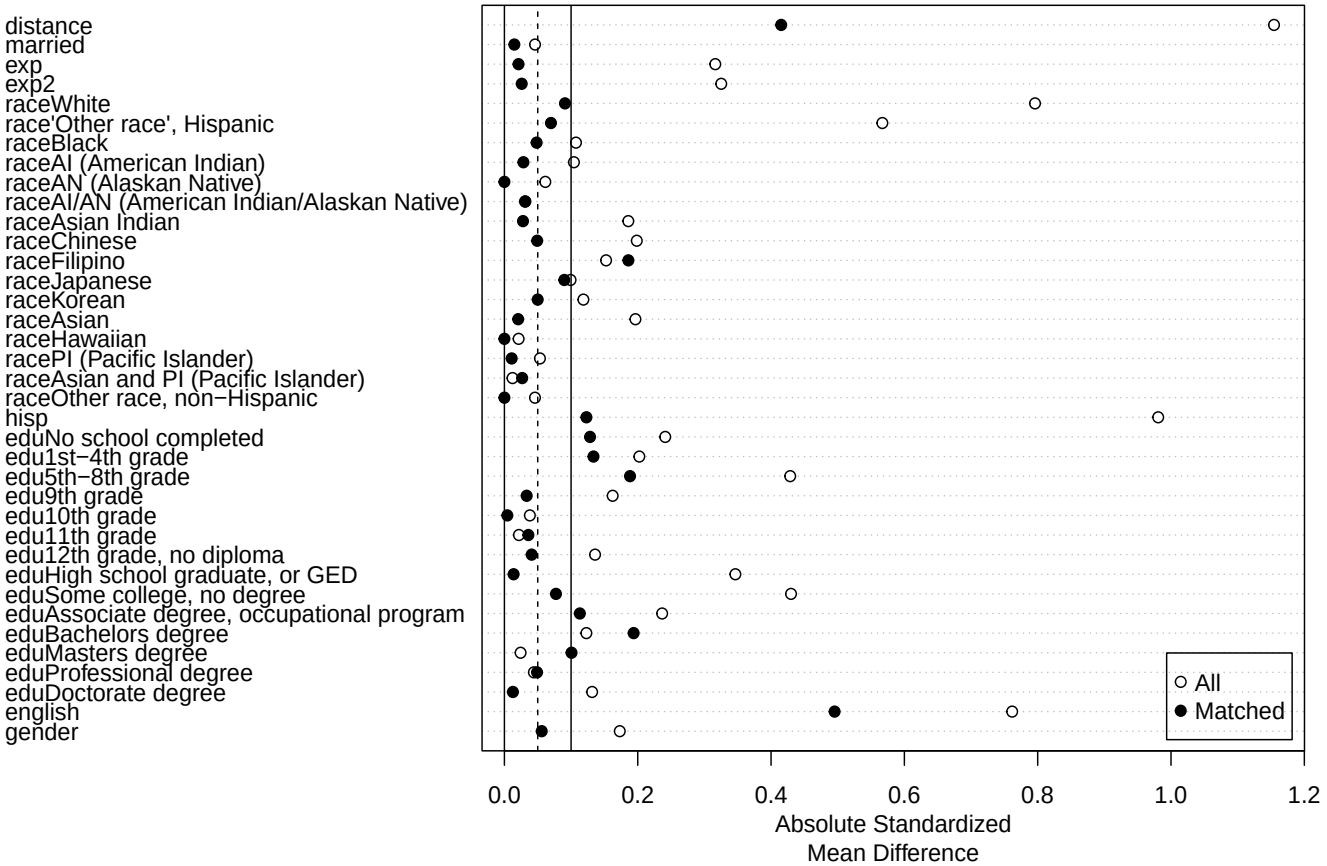
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We can also compare covariate balance before and after the matching.

Let's look at the estimate probabilities:

```
plot(summary(ps_matchout))
```



Propensity Score Matching in R

Note: These are the same estimate propensity score that you would get from just running the probit regression:

```
probit_mod <- glm(fb ~ married + exp + exp2 + race +
                        hisp + edu + english + gender,
                      data = census_comp,
                family = binomial(link = "probit"))

census_comp <- census_comp |>
  mutate(ps2 = predict(probit_mod, type = "response"))

fivenum(census_comp$ps - census_comp$ps2)
```

```
##      1 12954 25908 38862 51816
##      0     0     0     0     0
```

Running the regression

We can now run the regression using matched propensity scores!

First we need to create the matched data object.

Then, we can do the regression.

Regression Results

```r
match1_data <- match.data(ps_matchout)

ps_lm1 <- lm(lnwage ~ fb + married + exp + exp2 + race +
                      hisp + edu + english + gender,
             data = match1_data)
summary(ps_lm1)
```

```
##
## Call:
## lm(formula = lnwage ~ fb + married + exp + exp2 + race + hisp +
##     edu + english + gender, data = match1_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9058 -0.3782 -0.0151  0.3503  4.4634
##
## Coefficients:
##                                                Estimate Std. Error t value
## (Intercept)                                   1.8681177  0.1389454  13.445
## fb                                           -0.0812118  0.0189349  -4.289
## married                                       0.1273640  0.0192070   6.631
## exp                                           0.0333812  0.0124825   2.674
## exp2                                         -0.0006698  0.0002910  -2.302
## race'Other race', Hispanic                   -0.0345887  0.0254826  -1.357
## raceBlack                                    -0.0157729  0.0379669  -0.415
## raceAI (American Indian)                      0.0483157  0.1649588   0.293
## raceAI/AN (American Indian/Alaskan Native)   -0.0172194  0.1523438  -0.113
## raceAsian Indian                              0.0893885  0.0492324   1.816
## raceChinese                                   0.0092814  0.0454886   0.204
## raceFilipino                                  0.0445163  0.0466622   0.954
## raceJapanese                                  0.0243702  0.0715646   0.341
## raceKorean                                   -0.0399359  0.0670611  -0.596
## raceAsian                                    -0.0815057  0.0462727  -1.761
## racePI (Pacific Islander)                     0.0008986  0.1335067   0.007
```

10-Nearest Neighbors

Now let's try matching with the 10-nearest neighbors. It is also easy to do with `matchit()`.

```
ps10_matchout <- matchit(fb ~ married + exp + exp2 + race +
                           hisp + edu + english + gender,
                         data = census_comp,
                         method = "nearest",
                         link = "probit",
                         ratio = 10
                         )
summary(ps10_matchout)
```

```
##
## Call:
## matchit(formula = fb ~ married + exp + exp2 + race + hisp + edu +
##     english + gender, data = census_comp, method = "nearest",
##     link = "probit", ratio = 10)
##
## Summary of Balance for All Data:
##                                        Means Treated Means Control
## distance                                      0.3997         0.0340
## married                                       0.6495         0.6714
## exp                                          20.3100        22.0917
## exp2                                        444.2075       522.7215
```

Regression Results

```r
match10_data <- match.data(ps10_matchout)

ps_lm10 <- lm(lnwage ~ fb + married + exp + exp2 + race +
                       hisp + edu + english + gender,
              data = match10_data)
summary(ps_lm10)
```

```
##
## Call:
## lm(formula = lnwage ~ fb + married + exp + exp2 + race + hisp +
##     edu + english + gender, data = match10_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9518 -0.3416  0.0018  0.3387  6.5775
##
## Coefficients:
##                                                Estimate Std. Error t value
## (Intercept)                                   1.6566025  0.0656115  25.249
## fb                                           -0.0586327  0.0158846  -3.691
## married                                       0.1257161  0.0082032  15.325
## exp                                           0.0368904  0.0051903   7.108
## exp2                                         -0.0006589  0.0001219  -5.406
## race'Other race', Hispanic                   -0.0258613  0.0198663  -1.302
## raceBlack                                    -0.0439011  0.0114842  -3.823
## raceAI (American Indian)                      0.0686631  0.0785642   0.874
## raceAI/AN (American Indian/Alaskan Native)    0.0911084  0.0739437   1.232
## raceAsian Indian                              0.1185954  0.0442123   2.682
## raceChinese                                   0.0790666  0.0359419   2.200
## raceFilipino                                  0.0793211  0.0387470   2.047
## raceJapanese                                  0.0486162  0.0643481   0.756
## raceKorean                                    0.0044247  0.0601164   0.074
## raceAsian                                    -0.0040271  0.0370360  -0.109
## racePI (Pacific Islander)                     0.0001037  0.0923082   0.002
```

IPW re-weighting

There are some econometric issues with doing simple propensity score matching.

However, inverse propensity score weighting mitigates these issues.

Intuition: Those who were likely to be treated but did not are likely more similar to those who are treated.

Computing IPW weights for the ATT

```
census_comp <- census_comp |>
  mutate(ipw_wght = case_when(fb == 1 ~ 1,
                              TRUE ~ ps/(1 - ps)))
```

Density Plots

Let's make plots which show the propensity score density before and after re-weighting. The `cowplot` package has some useful functions for plotting figures side-by-side.

```
library(cowplot)
```

```
##
## Attaching package: 'cowplot'

## The following object is masked from 'package:lubridate':
##
##     stamp
```

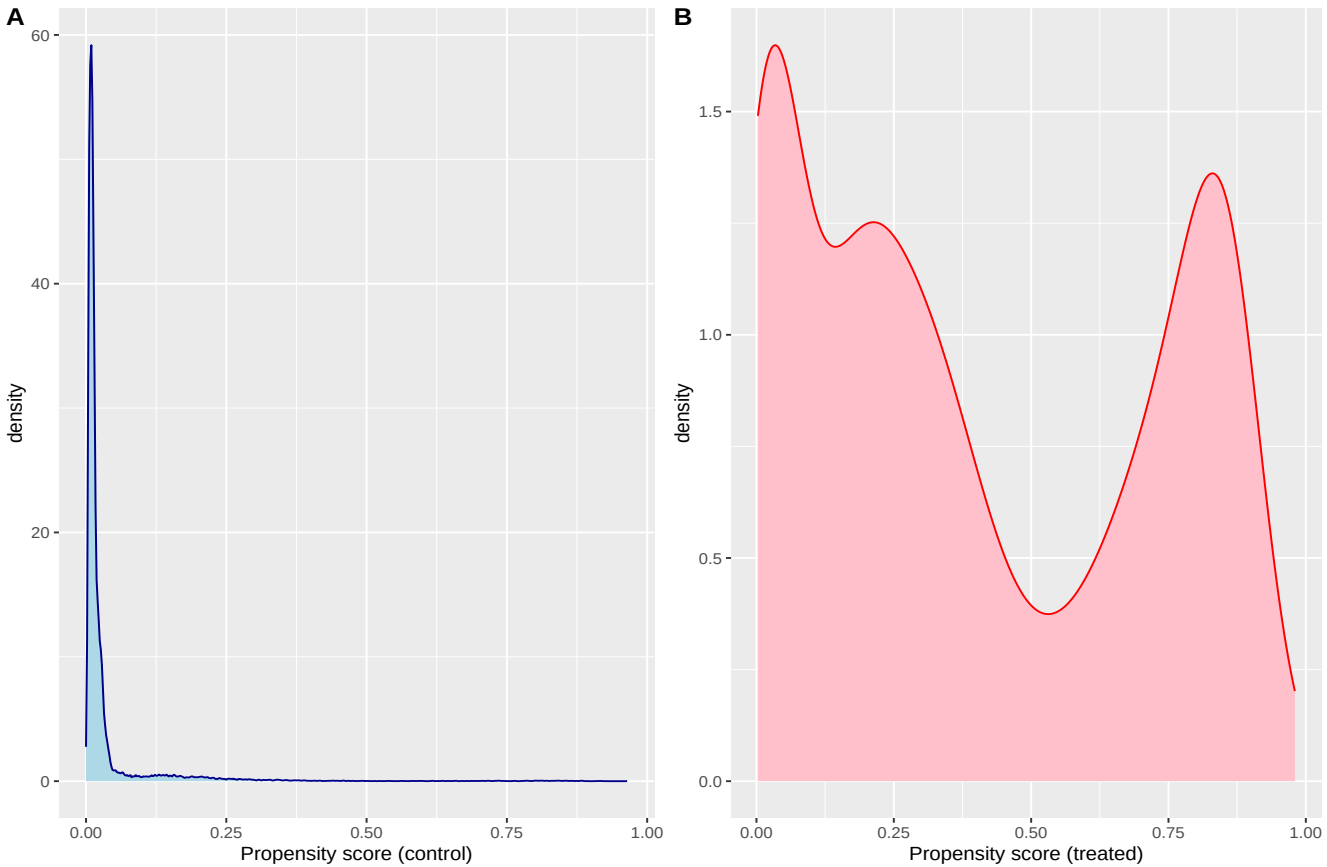
```
p1 <- ggplot(data = filter(census_comp, fb == 0)) +
  geom_density(aes(x = ps), color="darkblue", fill="lightblue") +
  labs(x = "Propensity score (control)")


p2 <- ggplot(data = filter(census_comp, fb == 1)) +
  geom_density(aes(x = ps), color="red", fill="pink") +
  labs(x = "Propensity score (treated)")

pg <- plot_grid(p1, p2, labels = c('A', 'B'),
                align = "h", nrow = 1, ncol = 2, scale = 1)
```

Density Plots Unweighted

```
print(pg)
```



Density Plots Weighted

Now, let's re-weight using the IPW weights and compare the density plots.

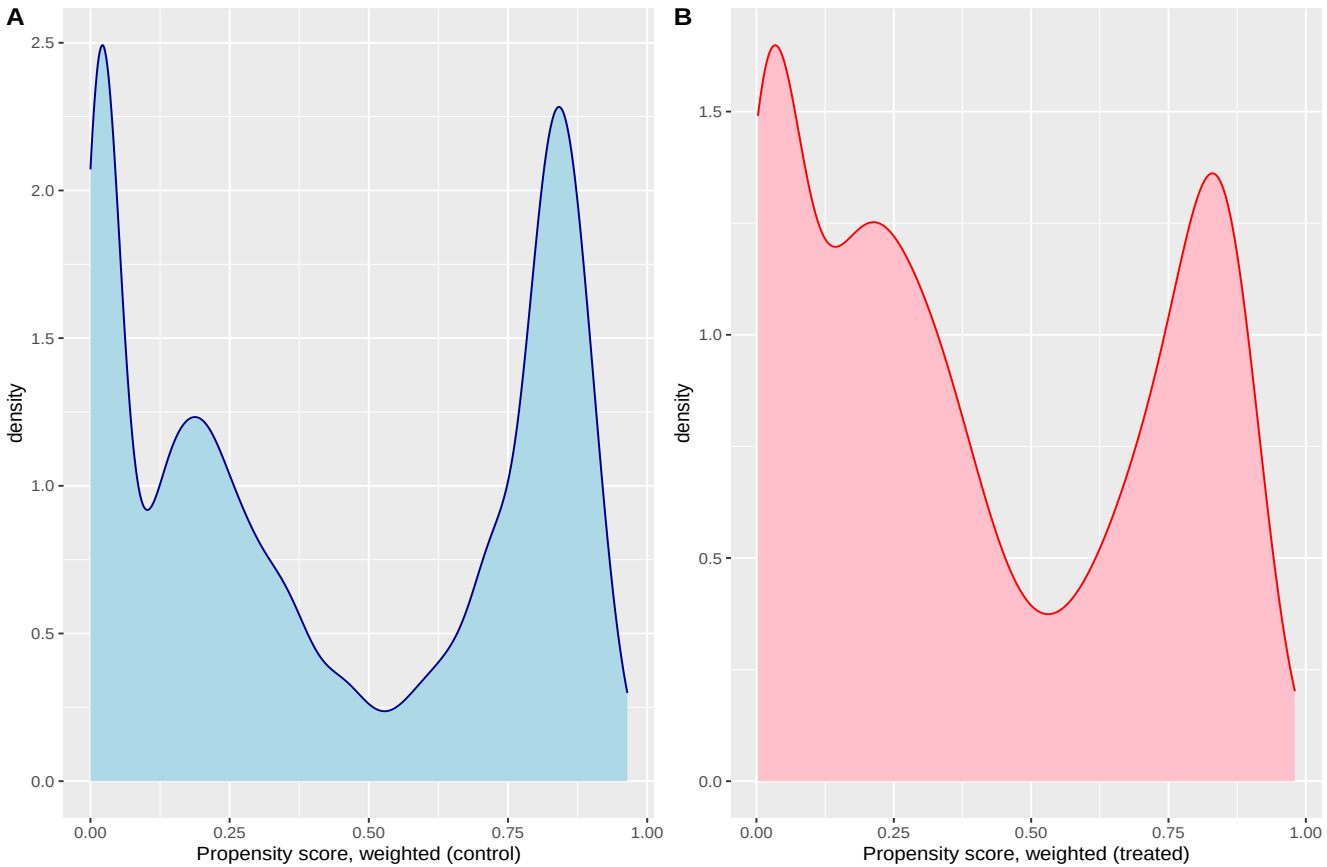
```
p3 <- ggplot(data = filter(census_comp, fb == 0)) +
  geom_density(aes(x = ps, weight = ipw_wght),
               color="darkblue", fill="lightblue", adjust = 30) +
  labs(x = "Propensity score, weighted (control)")

p4 <- ggplot(data = filter(census_comp, fb == 1)) +
  geom_density(aes(x = ps, weight = ipw_wght), color="red", fill="pink") +
  labs(x = "Propensity score, weighted (treated)")

pg2 <- plot_grid(p3, p4, labels = c('A', 'B'),
                 align = "h", nrow = 1, ncol = 2, scale = 1)
```

Density Plots Weighted

```
print(pg2)
```



IPW Regression

Now let's run the regression using IPW weights.

```
ipw_lm <- lm(lnwage ~ fb + married + exp + exp2 + race +
                      hisp + edu + english + gender,
             weights = ipw_wght,
             data = census_comp)
summary(ipw_lm)
```

```
##
## Call:
## lm(formula = lnwage ~ fb + married + exp + exp2 + race + hisp +
##     edu + english + gender, data = census_comp, weights = ipw_wght)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2716 -0.0494 -0.0066  0.0343  6.8012
##
## Coefficients:
##                                                Estimate   Std. Error t value
## (Intercept)                                  1.95106910   0.04473928  43.610
## fb                                          -0.08431998   0.00589199 -14.311
## married                                      0.12856817   0.00625568  20.552
## exp                                          0.02655036   0.00408869   6.494
## exp2                                        -0.00049146   0.00009488  -5.179
## race'Other race', Hispanic                  -0.07839033   0.00806695  -9.717
## raceBlack                                   -0.02949151   0.01271862  -2.319
## raceAI (American Indian)                     0.03845865   0.04977876   0.773
## raceAN (Alaskan Native)                      0.51708625  45.81671133   0.011
## raceAI/AN (American Indian/Alaskan Native)  -0.10846268   0.04862937  -2.230
## raceAsian Indian                             0.13984796   0.01564662   8.938
## raceChinese                                 -0.00843883   0.01506582  -0.560
## raceFilipino                                 0.03417714   0.01733028   1.972
```

Collecting Regression Results

Let's now collect all of our regressions and export a nice table!

The `stargazer` package is great for exporting simple tables from data frames.

```
library(stargazer)
```

```
##
## Please cite as:

##  Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Statistics Tables.

##  R package version 5.2.3. https://CRAN.R-project.org/package=stargazer
```

```
coefficients <- c(bi_ols$coefficients[2], mul_ols$coefficients[2],
  fs_ols$coefficients[1], ps_lm1$coefficients[2], ps_lm10$coefficients[2],
  ipw_lm$coefficients[2]
  )


labels <- c("Bivariate OLS", "Multivariate OLS", "Fully saturated OLS",
            "Propensity score matching (regression)",
            "Propensity score with 10 nearest-neighbors",
            "Propensity score with inverse probability weights")

tab <- data.frame("Model" = labels, "Estimate" = coefficients)
```

Collecting Regression Results

```
stargazer(tab, summary = F,
          title = "Model estimates for problem 5",
          label = "tab:p5_2",
          type = "text", out = file.path(results_directory, "ps1_tabp5_2.tex")
)
```

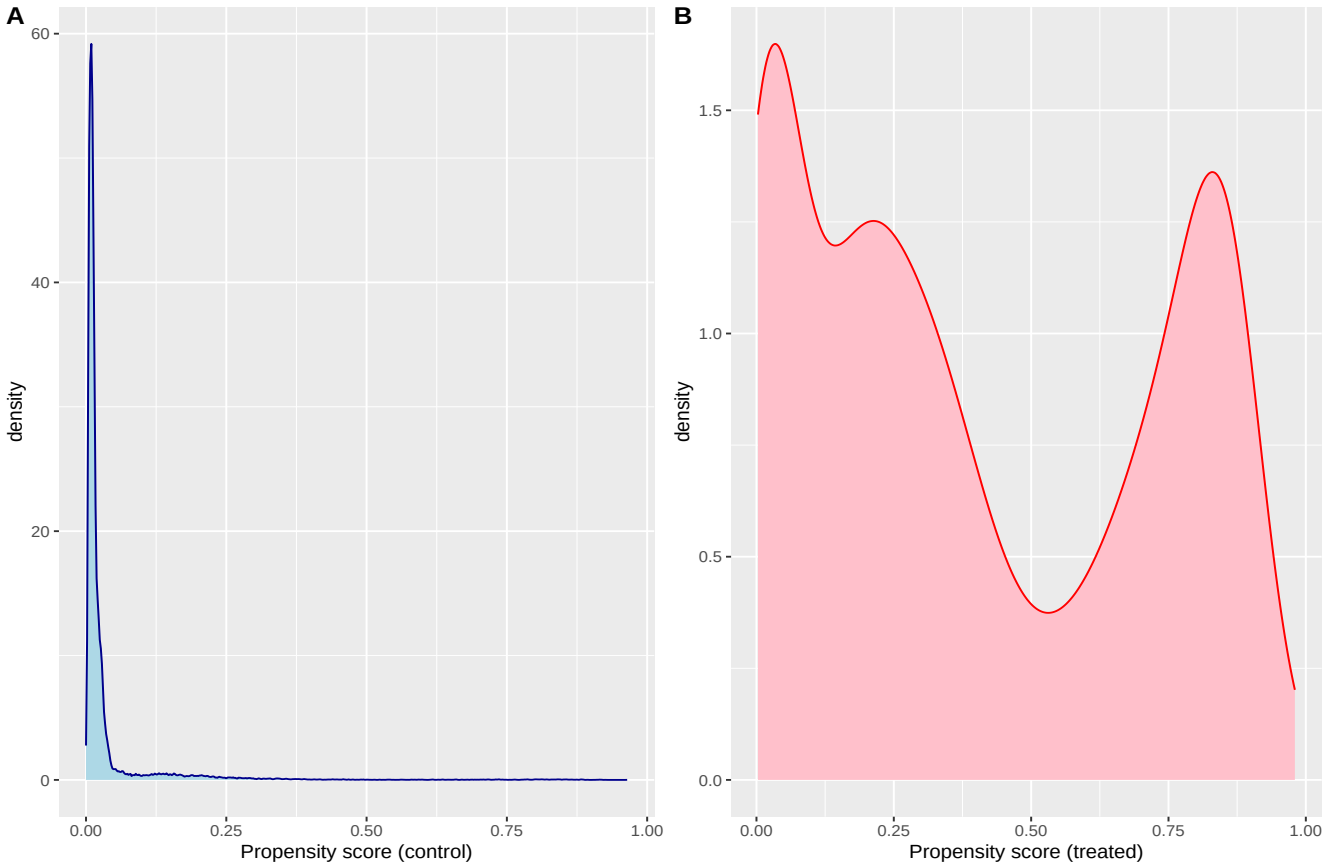
```
##
## Model estimates for problem 5
## ============================================================
##                           Model                          Estimate
## ------------------------------------------------------------
## 1                    Bivariate OLS                        -0.185
## 2                  Multivariate OLS                       -0.060
## 3                 Fully saturated OLS                     -0.039
## 4      Propensity score matching (regression)             -0.081
## 5    Propensity score with 10 nearest-neighbors           -0.059
## 6 Propensity score with inverse probability weights       -0.084
## ------------------------------------------------------------
```

Saving Figures

Let's also make sure to save our density plots!

We will use the `ggsave()` function after we print the plots.

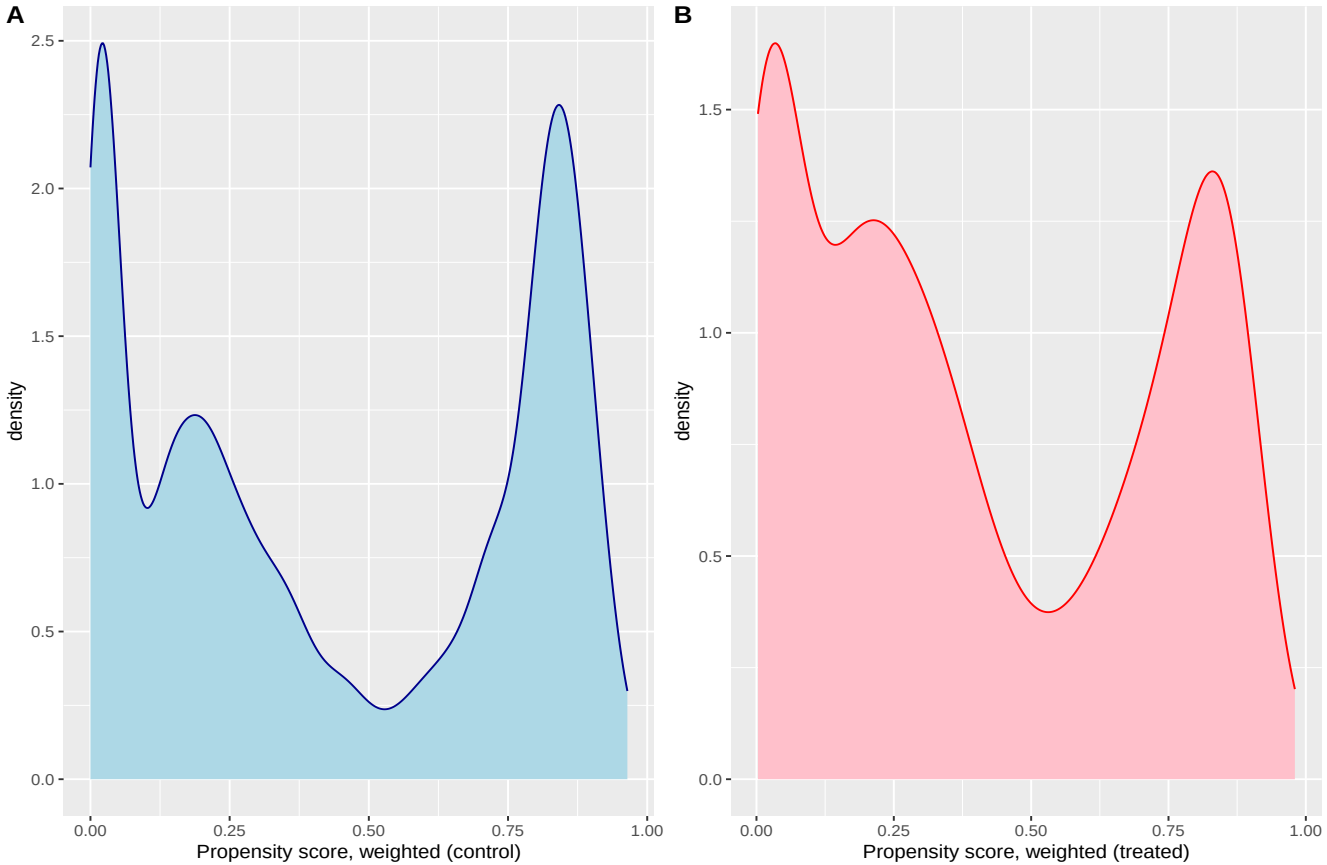
```
print(pg)
```




```
ggsave(file.path(results_directory, "ps1_dens_1.pdf"),
       width = 12, height = 8, units = "in")
```

Saving Figures

```
print(pg2)
```




```
ggsave(file.path(results_directory, "ps1_dens_2.pdf"),
       width = 12, height = 8, units = "in")
```