Matching and Regression

Alex Alekseev

```r
library(modelsummary)
options(modelsummary_factory_default = 'data.frame')

library(estimatr)
library(Matching)
library(MatchIt)
library(causalweight)

library(tidyverse)
```

Intro

In this session, we will be working with the datasets from LaLonde (1986) and Dehejia and Wahba (2002) that examine the effect of a job training program on earnings. The program was called the National Supported Work Demonstration (NSW) job-training program and was run in the 1970s. The NSW was designed to help individuals with low earning potential (e.g., addicts, released offenders, and people without high school education) find jobs by giving them work experience and counseling. The program was unique in that it randomized potential participants into actual participation. We will evaluate the treatment effect of the program on earnings (1978 real earnings) using the original data from NSW, as well as the dataset in which the control group was replaced by a random sample from the Current Population Survey (CPS). The former dataset will provide a benchmark for the true treatment effect, since the assignment was random. The latter dataset will allow us to examine how various conditioning strategies can approximate that benchmark in case when you do not have random assignment.

First, let's load the datasets. The first dataset, `dw_exper.csv`, contains the NSW data. The second dataset, `dw_cps.csv` contains potential control observations from the CPS. Make sure to specify the right path to your files.

```r
df_exper <- read_csv("path_to_file/dw_exper.csv")
df_cps <- read_csv("path_to_file/dw_cps.csv")
```

The dependent variable in both datasets in `re78` (real 1978 earnings), the treatment variable is `treat`, and the rest of the variables, which include socio-demographic characteristics and prior earnings, can be used as controls.

	0		1			
	Mean	Std. Dev.	Mean	Std. Dev.	Diff. in Means	Std. Error
age	25.1	7.1	25.8	7.2	0.8	0.7
educ	10.1	1.6	10.3	2.0	0.3	0.2
black	0.8	0.4	0.8	0.4	0.0	0.0
hisp	0.1	0.3	0.1	0.2	0.0	0.0
marr	0.2	0.4	0.2	0.4	0.0	0.0
nodegree	0.8	0.4	0.7	0.5	−0.1	0.0
re74	2107.0	5687.9	2095.6	4886.6	−11.5	503.5
re75	1266.9	3103.0	1532.1	3219.3	265.1	305.0
re78	4554.8	5483.8	6349.1	7867.4	1794.3	671.0

Experimental data

First, let's get a benchmark for the treatment effect from the experimental data. Since the treatment assignment was random, the independence assumption holds and we can compute the treatment effect by running a simple regression. Recall that if independence holds, all three treatments effects (ATE, ATT, ATU) are identical and also equal to the NTE.

```r
reg <- lm(re78 ~ treat, data = df_exper)
coef(reg)["treat"]
```

```
##    treat
## 1794.342
```

Thus, the program increased the participants' real 1978 earnings by about $1800.

We know that the program had random assignment, but what if we include controls anyway. Would it change our estimate of the treatment effect? Here we will include controls variables in a simple linear way.

```r
reg <- lm(re78 ~ treat
          + age + educ + black + hisp + marr + nodegree + re74 + re75
          , data = df_exper
)
coef(reg)["treat"]
```

```
##    treat
## 1676.343
```

The effect decreased. Not dramatically, but not trivially either. Even under random assignment it is possible that there will be correlation between individuals' characteristics and treatment status in a finite sample. Another possibility is that despite program administrators' efforts, some selection still happened. Whatever the cause, we can check whether the observable characteristics of the treatment and control groups are similar.

	0		1			
	Mean	Std. Dev.	Mean	Std. Dev.	Diff. in Means	Std. Error
age	33.2	11.0	25.8	7.2	−7.4	0.5
educ	12.0	2.9	10.3	2.0	−1.7	0.1
black	0.1	0.3	0.8	0.4	0.8	0.0
hisp	0.1	0.3	0.1	0.2	0.0	0.0
marr	0.7	0.5	0.2	0.4	−0.5	0.0
nodegree	0.3	0.5	0.7	0.5	0.4	0.0
re74	14 016.8	9569.8	2095.6	4886.6	−11 921.2	367.2
re75	13 650.8	9270.4	1532.1	3219.3	−12 118.7	247.8
re78	14 846.7	9647.4	6349.1	7867.4	−8497.5	583.4

```r
datasummary_balance(~ treat,  data = df_exper)
```

We do see some difference between the treatment and control groups, e.g., there are more Hispanic people in the control group than in the treatment group. Overall the differences are pretty minor. In the next sections, we will use the estimate of the treatment effect from the simple regression as our benchmark.

Non-experimental controls

We will now replace the control group with a random sample from the CPS. This replacement simulates a typical situation where treatment exposure is not random. We will try to condition on the observable characteristics and see if we can get close to the treatment effect estimates we obtained with the experimental dataset.

```r
df <-
  bind_rows(
    df_exper |> filter(treat == 1)
    , df_cps
  )
```

Let's check the background characteristics in this dataset.

```r
datasummary_balance(~ treat,  data = df)
```

We see large differences between our new control group drawn from the general population and the treatment group. The treatment group is younger, less educated, has a higher proportion of Black individuals, fewer married individuals. The before-treatment earnings in the treatment group are much lower than the earnings of the control group. This should not be surprising because the NSW program specifically targeted disadvantaged people.

Our goal will be to try and adjust for these differences in observable characteristics using regression and matching.

Regression

Let's start with a naive regression. We can expect that the result will be biased. But how bad is it?

```
reg <- lm(re78 ~ treat, data = df)
coef(reg)["treat"]
```

```
##     treat
## -8497.516
```

The naive treatment effect is actually negative! Again, this should not be surprising since people who selected into the program had lower earning potential to begin with. Recall that the selection (or baseline) bias in the NTE decomposition is $E[Y^0 \mid D = 1] - E[Y^0 \mid D = 0]$. The baseline (control state) earnings of the non-treated are likely to be much higher than the earnings of the treatment group. Hence, the difference is negative. The second part of the bias, the differential treatment effect bias, could be positive or negative.

Our first attempt at adjusting for observables could be including them in a linear manner.

```
reg <- lm(re78 ~ treat
          + age + educ + black + hisp + marr + nodegree + re74 + re75
          , data = df)
coef(reg)["treat"]
```

```
##    treat
## 699.1317
```

The estimate is no longer negative, which is an improvement, however, it is a far cry from the numbers we obtained using the experimental data.

We can try to create a more flexible adjustment. Let's include all the binary variables as fully interacted terms, add polynomials of the continuous variables, and add the interactions of education and prior earnings.

```
reg <- lm(re78 ~ treat
          + black*hisp*marr*nodegree
          + age + I(age^2) + I(age^3) + educ + I(educ^2) + I(educ^3)
          + re74 + I(re74^2) + I(re74^3) + re75 + I(re75^2) + I(re75^3)
          + educ*re74 + educ*re75
          , data = df)
coef(reg)["treat"]
```

```
##    treat
## 1654.351
```

This number is pretty close to the one that we got from the experimental data when using controls. However, we have to ask which treatment effect does this regression give us? Recall that in general, it will not be any of the treatment effects, but rather a conditional-variance-weighted average of conditional ATEs.

What if we try an even more flexible specification for the controls? Let's interact *all* the variables and add polynomials of all continuous variables. Notice that this regression will have a lot of coefficients and might take some time to compute.

```r
reg <- lm(re78 ~ treat
          + age*educ*re74*re75*black*hisp*marr*nodegree
          + I(age^2) + I(age^3) + I(educ^2) + I(educ^3)
          + I(re74^2) + I(re74^3) + I(re75^2) + I(re75^3)
          , data = df)
coef(reg)["treat"]
```

```
##    treat
## 1267.855
```

Interestingly, the resulting effect is further away from the experimental treatment effect than the estimated effect in the previous specification. A priori it would appear that a more flexible specification should remove the biases better than a less flexible specification. And yet it is not the case here. Without having the experimental benchmark we would not be able to tell which specification is better, though.

Matching

The benefit of matching is that we do not have to worry about specifying the functional form of the model. A bigger worry with matching is the quality of controls, however, we will be able to quantify this problem and try to solve it. Another nice feature of matching is that we can explicitly specify which treatment effect we want to estimate. In our case, it probably makes the most sense to focus on the ATT, since we have a large pool of control observations that can be matched to the treatment observations.

We will be using the `Match` function from the `Matching` package to produce the estimates of the treatment effect and get the matched sample.

Exact matching

Let's attempt an exact matching with the Mahalanobis distance. By default, we have matching with replacement. We will also allow for ties. This means that we can find more than one match for each treatment unit, in which case each of the matched controls will be included in the matched sample and get an equal weight.

```r
match_exact <- Match(
  Y = df$re78
  , Tr = df$treat
  , X = model.matrix(
    re78 ~ age + educ + black + hisp + marr + nodegree + re74 + re75
    , data = df
  )
  , estimand = "ATT"
  , Weight = 2
  , exact = T
  , ties = T
)
```

	0		1			
	Mean	Std. Dev.	Mean	Std. Dev.	Diff. in Means	Std. Error
age	21.9	6.0	21.9	6.0	0.0	1.4
educ	10.8	1.5	10.8	1.5	0.0	0.3
black	0.8	0.4	0.8	0.4	0.0	0.1
hisp	0.0	0.0	0.0	0.0	0.0	0.0
marr	0.1	0.3	0.1	0.3	0.0	0.1
nodegree	0.7	0.5	0.7	0.5	0.0	0.1
re74	0.0	0.0	0.0	0.0	0.0	0.0
re75	0.0	0.0	0.0	0.0	0.0	0.0
re78	3020.4	5148.2	6077.9	6798.8	3057.5	1362.8

```r
summary(match_exact)
```

```
##
## Estimate...  3057.5
## AI SE......  319.91
## T-stat.....  9.5574
## p.val......  < 2.22e-16
##
## Original number of observations..............  16177
## Original number of treated obs...............  185
## Matched number of observations...............  29
## Matched number of observations  (unweighted).  60
##
## Number of obs dropped by 'exact' or 'caliper'  156
```

The estimated effect from the matched sample is positive but it is much larger than the effect from the experimental data. Notice, however, that the matching algorithm dropped a lot of treatment observations that had no matches. It is actually surprising it found any exact matches at all given that income variables `re74` and `re75` are continuous. This is probably due to the fact that we have relatively few treated units and a lot of control units.

Let's check the covariate balance. We will use the indices of the treatment and control observations that the algorithm included in the matched sample, as well as the weights, which are defined for each matched pair. Notice that since we are allowing for ties, some weights are not 0 or 1, so we have to include the weights to compute the correct weighted averages.

```r
df[c(match_exact$index.treated, match_exact$index.control), ] |>
  mutate(weights = c(match_exact$weights, match_exact$weights)) |>
  datasummary_balance(~ treat,  data = _)
```

There are no differences at all. This should not be surprising given that we are matching *exactly.*

You can also use the `MatchBalance` function to get a more rigorous look at the covariate balance. But the output is rather long, I will not include it here.

```
MatchBalance(
  treat ~ age + educ + black + hisp + marr + nodegree + re74 + re75
  , data = df, match.out = match_exact)
```

What can we do to increase the number of matches? We can, e.g., ask the matching algorithm to match exactly only on a subset of variables, while doing a nearest-neighbor matching on the rest. In this case, we will do exact matching only on binary variables.

```
match_partial_exact <- Match(
  Y = df$re78
  , Tr = df$treat
  , X = model.matrix(
    re78 ~ age + educ + black + hisp + marr + nodegree + re74 + re75 - 1
    , data = df
  )
  , estimand = "ATT"
  , Weight = 2
  , exact = c(F, T, T, T, T, T, F, F)
  , ties = T
)

summary(match_partial_exact)
```

```
##
## Estimate...  1934.9
## AI SE......  930.22
## T-stat.....  2.08
## p.val......  0.037523
##
## Original number of observations..............  16177
## Original number of treated obs...............  185
## Matched number of observations...............  185
## Matched number of observations  (unweighted).  225
##
## Number of obs dropped by 'exact' or 'caliper'  0
```

In this case, the algorithm matched all observations. The estimated ATT is now closer to the experimental treatment effect, although it still overshoots.

Let's check the covariate balance.

```
df[
  c(
    match_partial_exact$index.treated
    , match_partial_exact$index.control
  )
```

	0		1			
	Mean	Std. Dev.	Mean	Std. Dev.	Diff. in Means	Std. Error
age	25.6	8.2	25.8	7.2	0.2	0.8
educ	10.3	2.0	10.3	2.0	0.0	0.2
black	0.8	0.4	0.8	0.4	0.0	0.0
hisp	0.1	0.2	0.1	0.2	0.0	0.0
marr	0.2	0.4	0.2	0.4	0.0	0.0
nodegree	0.7	0.5	0.7	0.5	0.0	0.0
re74	2150.1	3958.6	2095.6	4884.3	−54.5	458.5
re75	1553.8	3032.9	1532.1	3217.7	−21.7	322.3
re78	4414.3	5516.9	6349.1	7863.6	1934.9	690.6

```
  , ] |>
  mutate(
    weights = c(match_partial_exact$weights, match_partial_exact$weights)
  ) |>
  datasummary_balance(~ treat,  data = _)
```

As expected, the binary variables are matched exactly, but there are some very minor differences in the continuous variables.

Nearest-neighbor matching

Suppose we want to try nearest-neighbor matching instead of exact matching. We have to specify the number of nearest neighbors to look for. Let's try three.

```
match_knn <- Match(
  Y = df$re78
  , Tr = df$treat
  , X = model.matrix(
    re78 ~ age + educ + black + hisp + marr + nodegree + re74 + re75
    , data = df
  )
  , estimand = "ATT"
  , Weight = 2
  , M = 3
  , exact = F
  , ties = T
)

summary(match_knn)
```

```
##
## Estimate...  1569.1
## AI SE......  837.11
```

	0		1			
	Mean	Std. Dev.	Mean	Std. Dev.	Diff. in Means	Std. Error
age	25.5	7.8	25.8	7.1	0.3	0.4
educ	10.3	1.9	10.3	2.0	0.0	0.1
black	0.8	0.4	0.8	0.4	0.0	0.0
hisp	0.1	0.2	0.1	0.2	0.0	0.0
marr	0.2	0.4	0.2	0.4	0.0	0.0
nodegree	0.7	0.5	0.7	0.5	0.0	0.0
re74	2249.2	4228.1	2095.6	4877.4	−153.6	272.7
re75	1649.0	3370.0	1532.1	3213.2	−117.0	196.7
re78	4780.0	6051.7	6349.1	7852.6	1569.1	415.5

```
## T-stat.....  1.8744
## p.val......  0.060872
##
## Original number of observations..............  16177
## Original number of treated obs...............  185
## Matched number of observations...............  185
## Matched number of observations  (unweighted).  602
##
## Number of obs dropped by 'exact' or 'caliper'  0
```

The estimated effect went down relative to the previous estimate and now closer the experimental treatment effect. Also notice a bump in the matched number of observations (unweighted).

How about the covariate balance?

```
df[c(match_knn$index.treated, match_knn$index.control), ] |>
  mutate(weights = c(match_knn$weights, match_knn$weights)) |>
  datasummary_balance(~ treat,  data = _)
```

The differences between the continuous variables are more pronounced now. This is what we should expect, since as we increase the number of required matches some of those will not be as good.

We can hedge against really bad matches by using a caliper. Let's limit the maximum difference by 0.5 standard deviations for each variable.

```
match_caliper <- Match(
  Y = df$re78
  , Tr = df$treat
  , X = model.matrix(
    re78 ~ age + educ + black + hisp + marr + nodegree + re74 + re75
    , data = df
  )
  , estimand = "ATT"
```

	0		1			
	Mean	Std. Dev.	Mean	Std. Dev.	Diff. in Means	Std. Error
age	24.7	7.1	25.0	6.7	0.3	0.4
educ	10.6	1.7	10.6	1.7	0.0	0.1
black	0.8	0.4	0.8	0.4	0.0	0.0
hisp	0.1	0.2	0.1	0.2	0.0	0.0
marr	0.2	0.4	0.2	0.4	0.0	0.0
nodegree	0.7	0.5	0.7	0.5	0.0	0.0
re74	1938.0	3606.0	1729.0	3700.5	−209.0	230.9
re75	1334.6	2645.7	1259.4	2528.4	−75.2	163.5
re78	4576.3	5782.6	6288.8	7782.2	1712.5	429.4

```
  , Weight = 2
  , M = 3
  , caliper = 0.5
  , exact = F
  , ties = T
)

summary(match_caliper)
```

```
##
## Estimate...  1712.5
## AI SE......  772.85
## T-stat.....  2.2158
## p.val......  0.026706
##
## Original number of observations..............  16177
## Original number of treated obs...............  185
## Matched number of observations...............  165
## Matched number of observations  (unweighted).  542
##
## Number of obs dropped by 'exact' or 'caliper'  20
```

Using the caliper lead to a few treatment observations being unmatched. But the estimated ATT is now fairly close to the experimental ATT.

Did the caliper improve the covariate balance?

```
df[c(match_caliper$index.treated, match_caliper$index.control), ] |>
  mutate(weights = c(match_caliper$weights, match_caliper$weights)) |>
  datasummary_balance(~ treat,  data = _)
```

Not too much. The balance on `re74` actually got a bit worse, while the balance on `re75` did improve.

Coarsened exact matching

Another way to do exact matching would be to use the *coarsened exact matching* where we first coarsen the variables by binning their values into a few groups. This would make most sense for the continuous variables, although you can also bin together the values of categorical variables.

To do the coarsened exact matching, we will switch to a different package, `MatchIt`, and the `matchit` function that this package provides. It will not compute the treatment effect automatically, only the matching weights. We will have to compute the effect ourselves.

We will be coarsening the four continuous variables in our sample: `age`, `educ`, `re74`, `re75`. There are different ways to coarsen the values. We will make bins based on quantiles of the data. The trade-off here is that having fewer bins makes it more likely to find matches but lowers the quality of matches.

```
match_cem <- matchit(
  treat ~ age + educ + black + hisp + marr + nodegree + re74 + re75
  , data = df
  , method = "cem"
  , estimand = "ATT"
  , cutpoints = list(
    age = "q4"
    , educ = "q4"
    , re74 = "q5"
    , re75 = "q5"
  )
)
```

First, let's check the number of matches.

```
summary(match_cem)$nn
```

```
##                    Control Treated
## All (ESS)      15992.00000     185
## All            15992.00000     185
## Matched (ESS)     91.09724     172
## Matched         1799.00000     172
## Unmatched      14193.00000      13
## Discarded          0.00000       0
```

A few treatment observations were unmatched. This means that we could try to coarsen our variables a bit more until all the treatment observations get matched. This would probably come at the expense of poorer matches. On the other hand, the algorithm found a lot of matches among the control observations.

How about the covariate balance?

```
df |>
  mutate(weights = match_cem$weights) |>
  datasummary_balance(~ treat,  data = _)
```

There are some imbalances, most notably among the earnings variables `re74` and `re75`.

	0		1			
	Mean	Std. Dev.	Mean	Std. Dev.	Diff. in Means	Std. Error
age	25.3	8.8	25.6	7.2	0.4	0.7
educ	10.4	2.4	10.3	2.1	0.0	0.2
black	0.8	0.4	0.8	0.4	0.0	0.0
hisp	0.1	0.2	0.1	0.2	0.0	0.0
marr	0.2	0.4	0.2	0.4	0.0	0.0
nodegree	0.7	0.5	0.7	0.5	0.0	0.1
re74	1877.9	3711.3	1618.2	3598.6	−259.7	367.9
re75	1637.5	3127.0	1119.2	2458.6	−518.4	273.9
re78	4610.8	5725.3	6447.1	7726.6	1836.4	775.4

This is the consequence of coarsening. Larger bins imply pooling together different values.

You can get a more detailed analysis of the covariate balance if you use

```
summary(match_cem)
```

The output is a bit long, though, and I will not include it here.

Using the weights produced by the `matchit` function, we can compute the ATT (notice that we already know it, actually, from the last row of the table above). There are several ways to do this. First, we can simply compute it by hand using the formula. Recall the formula for the ATT

$$\widehat{ATT} = \frac{1}{n^1} \sum_{i \in I} \left[y_i - \sum_{j \in J} w_{ij} y_j \right].$$

The weights produces by the `matchit` function for the control units do not depend on i, hence we can re-write the formula as

$$\widehat{ATT} = \frac{1}{n^1} \left[\sum_{i \in I} y_i - n^1 \sum_{j \in J} w_j y_j \right] = \frac{1}{n^1} \sum_{i \in I} y_i - \sum_{j} w_j y_j.$$

The weights for the treatment units are either 1 (if a unit has a match) or 0 (a unit is unmatched), and they will sum up to the total number of matched observations. Notice that since the algorithm did drop a few treatment observations, strictly speaking, we will not get the ATT but a more narrowly defined ATT for those who were matched. The weights for the control units from the function output will sum up to the total number of matched control units. Thus in the formula above the weights w_j correspond to the weights from the function output that are divided by the total number of matched controls. In the end, the ATT will be simply the difference between the weighted averages of earnings among the treatment and control groups.

Thus, we get

```
(1/sum(match_cem$weights[df$treat == 1])
 * sum((df$re78 * match_cem$weights)[df$treat == 1])
 - 1/sum(match_cem$weights[df$treat == 0])
 * sum((df$re78 * match_cem$weights)[df$treat == 0])
)
```

```
## [1] 1836.355
```

This is very close to the experimental treatment effect.

There is an alternative way to get this number that involves using a... regression output. When you run a regression you can specify observation weights. Guess what would happen if we used the weights from the matching output.

```
reg <- lm(re78 ~ treat, data = df, weights = match_cem$weights)
  coef(reg)["treat"]
```

```
##    treat
## 1836.355
```

So in the end, regression and matching are not really that different.

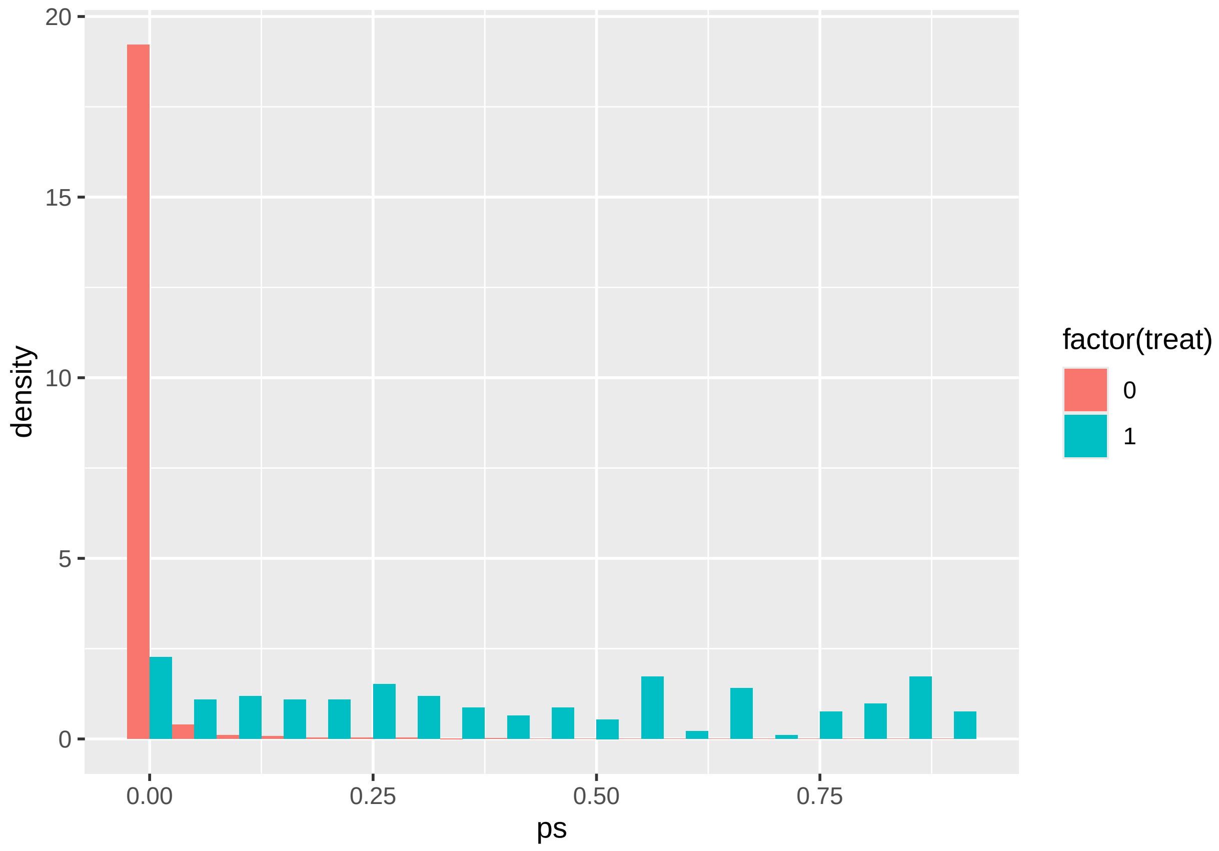
Propensity score matching

Instead of matching on covariates directly, we can use a propensity score (the probability of being treated) estimated using these covariates. First thing we would need to do then is to estimate the propensity score. There are a variety of ways to do that. We will use the most basic approach of estimating a logit model. We will include the covariates in the same flexible manner as we did with the regression.

```
ps_reg1 <- glm(
  treat
  ~ black*hisp*marr*nodegree
  + age + I(age^2) + I(age^3) + educ + I(educ^2) + I(educ^3)
  + re74 + I(re74^2) + I(re74^3) + re75 + I(re75^2) + I(re75^3)
  + educ*re74 + educ*re75
  , data = df
  , family = binomial(link = "logit")
)
```

After estimating the propensity score, a good idea is to check the overlap (common support assumption) between these scores among the treatment and control groups. We will do this visually using a histogram.

```
df |>
  mutate(ps = ps_reg1$fitted.values) |>
  ggplot(aes(ps)) +
  geom_histogram(
    aes(y = after_stat(density), fill = factor(treat))
    , binwidth = 0.05, position = "dodge"
  )
```



There is not that much overlap. This is not surprising because most people in the control group are unlike people in the treatment group, and thus unlike to have high predicted propensity scores.

Let's now try matching on the propensity score. A common recommendation when working with propensity scores is to trim the values that are below 0.1 and above 0.9 to avoid extreme values.

```r
ps_flt <- abs(ps_reg1$fitted.values - 0.5) < 0.4
df_flt <- df[ps_flt, ]

match_ps <- Match(
  Y = df_flt$re78
  , Tr = df_flt$treat
  , X = ps_reg1$fitted.values[ps_flt]
  , estimand = "ATT"
  , M = 1
  , ties = T
)

summary(match_ps)
```

```
##
## Estimate...  1688.7
## AI SE......  1209.6
## T-stat.....  1.3962
## p.val......  0.16267
```

	0		1			
	Mean	Std. Dev.	Mean	Std. Dev.	Diff. in Means	Std. Error
age	25.3	7.1	25.5	7.0	0.2	0.7
educ	10.4	1.6	10.3	1.9	−0.1	0.2
black	1.0	0.1	1.0	0.2	0.0	0.0
hisp	0.0	0.1	0.0	0.2	0.0	0.0
marr	0.2	0.4	0.2	0.4	0.0	0.0
nodegree	0.7	0.4	0.8	0.4	0.0	0.0
re74	1193.6	3391.2	1688.3	4706.6	494.7	417.9
re75	1103.8	1834.6	1036.2	1980.9	−67.6	194.6
re78	4724.3	5678.0	6413.1	8457.8	1688.7	782.2

```
##
## Original number of observations..............  406
## Original number of treated obs...............  143
## Matched number of observations...............  143
## Matched number of observations  (unweighted).  207
```

The estimated effect is fairly close to the experimental result. Notice, however, that trimming the data resulted in some treatment observations and a lot of control observations being dropped (this is not shown in the output of the function because we trim the data ourselves and then supply the trimmed data to the function). Again, keep in mind that whenever you drop observations, you are no longer estimating the ATT, but a more narrowly defined version of it.

How does the propensity score matching do in terms of covariate balance?

```
df_flt[c(match_ps$index.treated, match_ps$index.control), ] |>
  mutate(weights = c(match_ps$weights, match_ps$weights)) |>
  datasummary_balance(~ treat,  data = _)
```

There are some imbalances, most notably for the `re74` variable.

Another way to use the propensity score is to do inverse probability weighting. There are several ways to do this in R. First, since we already estimated the propensity score, we can compute the ATT ourselves. Second, we can use the `causalweight` package.

Let's first compute the ATT by hand. We will first define the indices of treatment and control observations that will be used in weighting. As before, we will trim the observations with propensity scores below 0.1 or above 0.9. The sums of these vectors will tell us the number of observations that remain in both groups after trimming. This is something we want to keep track of.

```
index_treat <- df$treat == 1 & ps_flt
index_control <- df$treat == 0 & ps_flt

sum(index_treat)
```

```
## [1] 143
```

```
sum(index_control)
```

```
## [1] 263
```

Recall the formula for the ATT that uses inverse probability weighting:

$$\widehat{ATT} = \frac{1}{n^1}\left(\sum_{i \in I} y_i - \sum_{j \in J} \frac{p_j}{1 - p_j} y_j\right)$$

It will be convenient to compute the weights $p_j/(1 - p_j)$ for the control observations.

```
weight_control <-
  (ps_reg1$fitted.values/(1 - ps_reg1$fitted.values))[index_control]

sum(weight_control)
```

```
## [1] 151.3331
```

Notice, however, that these weights do not sum up to the number of observations in the treatment group, unlike the weights for the treatment observations (they are all equal to 1). These weights are, in other words, not normalized. We can fix that by rescaling them by a factor of $n^1 / \sum_j (p_j/(1 - p_j))$.

```
weight_control <- weight_control*sum(index_treat)/sum(weight_control)

sum(weight_control)
```

```
## [1] 143
```

Then the ATT will be a simple difference in weighted averages.

```
1/sum(index_treat) *
  (sum(df$re78[index_treat])
   - sum((weight_control*df$re78[index_control]))
  )
```

```
## [1] 2356.085
```

This estimate is quite a bit higher than the experimental treatment effect.

Now let's use the `treatweight` function from the `causalweight` package. We will have to specify the formula to estimate the propensity score, because the function needs to compute it.

```
ipw <- treatweight(
  y = df$re78
  , d = df$treat
  , x = model.matrix(
    re78
    ~ black*hisp*marr*nodegree
    + age + I(age^2) + I(age^3) + educ + I(educ^2) + I(educ^3)
    + re74 + I(re74^2) + I(re74^3) + re75 + I(re75^2) + I(re75^3)
    + educ*re74 + educ*re75
    , data = df
  )
  , ATET = T
  , logit = T
  , trim = 0.1
  , boot = 3
)

ipw$effect
```

```
## [1] 1725.066
```

The estimate is pretty close to the benchmark. Why does it differ from the answer we got by computing the effect by hand? The reason is that the function uses a different trimming rule. When computing the ATT, it only trims observations with the propensity score larger than 0.9. In our calculation, we also trimmed observations with the propensity scores smaller than 0.1.

We can verify this.

```
index_treat <- df$treat == 1 & ps_reg1$fitted.values < 0.9
index_control <- df$treat == 0 & ps_reg1$fitted.values < 0.9

weight_control <-
  (ps_reg1$fitted.values/(1 - ps_reg1$fitted.values))[index_control]
weight_control <- weight_control*sum(index_treat)/sum(weight_control)

1/sum(index_treat) *
  (sum(df$re78[index_treat])
   - sum((weight_control*df$re78[index_control]))
  )
```

```
## [1] 1725.066
```