Matching and Sub-Classification

Rahul bahadur

# Bias Correction:

Causal inference relies on finding the identical twin that was exposed to the treatment for the one which was not or vice-versa. However, that is not always possible. If the covariates are not exactly the same then matching them would lead to biases. This bias can be corrected as shown below.

import numpy as np   
import pandas as pd   
import statsmodels.api as sm   
import statsmodels.formula.api as smf   
from itertools import combinations   
import plotnine as p  
import ssl

# read data  
ssl.\_create\_default\_https\_context = ssl.\_create\_unverified\_context  
def read\_data(file):   
 return pd.read\_stata("https://raw.github.com/scunning1975/mixtape/master/" + file)  
  
training\_bias\_reduction = read\_data("training\_bias\_reduction.dta")

training\_bias\_reduction

|  | Unit | Y | D | X |
| --- | --- | --- | --- | --- |
| 0 | 1 | 5 | 1 | 11 |
| 1 | 2 | 2 | 1 | 7 |
| 2 | 3 | 10 | 1 | 5 |
| 3 | 4 | 6 | 1 | 3 |
| 4 | 5 | 4 | 0 | 10 |
| 5 | 6 | 0 | 0 | 8 |
| 6 | 7 | 5 | 0 | 4 |
| 7 | 8 | 1 | 0 | 1 |

## Steps for Bias correction

1. Find the closest matching unit with the treatment unit based on the covariate (IV). For eg. for `Unit` 1 with X=11, the matching un-treated unit is X=10, i.e. `Unit` 5. Similarly for others

training\_bias\_reduction['Y1'] = np.where(training\_bias\_reduction.D==1, training\_bias\_reduction.Y, 0)  
training\_bias\_reduction['Y0'] = np.where(training\_bias\_reduction.D==0, training\_bias\_reduction.Y, 0)

2. Create a column with the fitted data using a model for `Y ~ X`

fitted\_model = sm.OLS.from\_formula('Y ~ X', training\_bias\_reduction).fit()  
training\_bias\_reduction['fitted'] = fitted\_model.predict(training\_bias\_reduction.X)  
training\_bias\_reduction

|  | Unit | Y | D | X | Y1 | Y0 | fitted |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 5 | 1 | 11 | 5 | 0 | 3.888071 |
| 1 | 2 | 2 | 1 | 7 | 2 | 0 | 4.082474 |
| 2 | 3 | 10 | 1 | 5 | 10 | 0 | 4.179676 |
| 3 | 4 | 6 | 1 | 3 | 6 | 0 | 4.276878 |
| 4 | 5 | 4 | 0 | 10 | 0 | 4 | 3.936672 |
| 5 | 6 | 0 | 0 | 8 | 0 | 0 | 4.033873 |
| 6 | 7 | 5 | 0 | 4 | 0 | 5 | 4.228277 |
| 7 | 8 | 1 | 0 | 1 | 0 | 1 | 4.374080 |

Bias := it is the diff in the predicted values generated based on the covariates.` implying that, given the same model, and the same set of covariates, and no information on which units

are treated or not, the model should generate fitted values consistent with these assumptions.

Bias reduction method := It is the diff between the diff of Treated and Un-Treated covariate and the diff between Treated predicted value and un\_treated predicted value

ATT = np.mean((np.array(training\_bias\_reduction['Y'][training\_bias\_reduction.D==1]) - np.array(training\_bias\_reduction['Y'][training\_bias\_reduction.D==0])) -  
 (np.array(training\_bias\_reduction['fitted'][training\_bias\_reduction.D==1]) - np.array(training\_bias\_reduction['fitted'][training\_bias\_reduction.D==0])))  
  
print(ATT)

3.2864506627393233

## Computing the variance of the bias estimator

var\_att = (((np.array(training\_bias\_reduction.Y[training\_bias\_reduction.D==1]) - np.array(training\_bias\_reduction.Y[training\_bias\_reduction.D==0]) - ATT)) \*\*2).mean()  
var\_att, np.sqrt(var\_att)

(3.188828650814136, 1.7857291650231106)