Recitation Week 4: Matching

Licheng Liu

Massachusetts Institute of Technology

Causal Inference (6.S059/15.C08/17.C08)
March 8, 2024

Today's Goals

1. **Admin**
2. **Matching**
3. **R and Python tips**

Admin

Reminders

- *Pset 2* is due on March 18th.
- Lab session (matching and regression for causal inference) on March 18th.

What does matching give us?

Matching is an estimation strategy (like regression).
Remember: "Identification precedes estimation."

What does matching give us?

Matching is an estimation strategy (like regression).
Remember: "Identification precedes estimation."

"Matching is simply a tool and cannot compensate for a poor identification strategy." - Keele, The Statistics of Causal Inference

Still have conditional ignorability as our key identification assumption!

Matching

So then what do we gain over regression?

- Less extrapolation beyond common support.
- Matching used to pre-process data can improve regression fit (so it's not necessarily matching *or* regression).
- Matching provides a non-parametric way to estimate causal effects when conditional ignorability holds.

Matching

So then what do we gain over regression?

- Less extrapolation beyond common support.
- Matching used to pre-process data can improve regression fit (so it's not necessarily matching *or* regression).
- Matching provides a non-parametric way to estimate causal effects when conditional ignorability holds.

But be careful:

- Matching can induce bias (more on this later).
- Losing observations (larger standard error).
- Balance after matching is comforting, but can give illusion of validity.

Implementation

1. Choose a distance metric.

Implementation

1. Choose a distance metric.
2. Choose the number of matches (one or many).

Implementation

1. Choose a distance metric.
2. Choose the number of matches (one or many).
3. Find matches & drop observations without matches.

Implementation

1. Choose a distance metric.
2. Choose the number of matches (one or many).
3. Find matches & drop observations without matches.
4. Check balance.

Implementation

1. Choose a distance metric.
2. Choose the number of matches (one or many).
3. Find matches & drop observations without matches.
4. Check balance.
5. Repeat until balance is achieved.

Implementation

1. Choose a distance metric.
2. Choose the number of matches (one or many).
3. Find matches & drop observations without matches.
4. Check balance.
5. Repeat until balance is achieved.
6. Estimate ATE/ATT/ATC in the matched dataset.

Distance metrics

- Exact matching
- Euclidean distance
- Mahalanobois distance
- Genetic matching
- Propensity score
- And others ...

Distance metrics

- Exact matching
- Euclidean distance
- Mahalanobois distance
- Genetic matching
- Propensity score
- And others ...

But note that none can compensate for insufficient overlap between treatment and control!

Exact matching

Effectively requires distance between observations to be zero

Let $\tilde{X}_i$ be a discrete variable with unique values for all possible combinations of k covariates

- age *in* $\{18 \ldots 25\}$
- gender $\in \{\text{male, female}\}$
- occupation $\in \{\text{farmer, teacher, plumber}\}$

$\tilde{X}_i$ would have $8 \times 2 \times 3 = 48$ discrete values

Exact matching

Effectively requires distance between observations to be zero.
Let $\tilde{X}_i$ be a discrete variable with unique values for all possible combinations of k covariates

For each treated unit:

- Find the set of control units j such that $\tilde{X}_i = \tilde{X}_j$

Exact matching

Effectively requires distance between observations to be zero.
Let $\tilde{X}_i$ be a discrete variable with unique values for all possible combinations of k covariates

For each treated unit:

- Find the set of control units j such that $\tilde{X}_i = \tilde{X}_j$
- Randomly select one of these units to be the match

Exact matching

Effectively requires distance between observations to be zero.
Let $\tilde{X}_i$ be a discrete variable with unique values for all possible combinations of k covariates

For each treated unit:

- Find the set of control units j such that $\tilde{X}_i = \tilde{X}_j$
- Randomly select one of these units to be the match
- Discard unmatched control units

Drawbacks:

Exact matching

Effectively requires distance between observations to be zero.
Let $\tilde{X}_i$ be a discrete variable with unique values for all possible combinations of k covariates

For each treated unit:

- Find the set of control units j such that $\tilde{X}_i = \tilde{X}_j$
- Randomly select one of these units to be the match
- Discard unmatched control units

Drawbacks:

- Doesn't work well when k is large
- Doesn't work well when covariates are continuous
- Curse of dimensionality!

Euclidean distance

Let $X_i = (X_{i1} \dots X_{ik})^T$ for k covariates

Euclidean distance between unit i and unit j:

$$D_{ij} = \sqrt{\sum_k \frac{(X_{ik} - X_{jk})^2}{\hat{\sigma}_k}}$$

Intuitively, this is just the sum of the standardized distances across all covariates

Euclidean distance

Let $X_i = (X_{i1} \dots X_{ik})^T$ for k covariates

Euclidean distance between unit i and unit j:

$$D_{ij} = \sqrt{\sum_k \frac{(X_{ik} - X_{jk})^2}{\hat{\sigma}_k}}$$

Intuitively, this is just the sum of the standardized distances across all covariates

Drawback: Doesn't account for covariance between variables

If X_{ik} and $X_{ik'}$ are both determined by a latent factor and are thus highly correlated, this latent factor is effectively double counted under Euclidean distance

Mahalanobois distance

Mahalanobois distance: Adjusts for covariance in the data

Intuition: If X_{ik} and $X_{ik'}$ are highly correlated, then their contribution to the distances should be lower.

- distances between highly correlated variables downweighted
- distances between uncorrelated variables upweighted

$$D_{ij} = \sqrt{(X_i - X_j)^T \sum_X^{-1} (X_i - X_j)}$$

where $\sum_X$ is the sample variance-covariance matrix of X_i

Drawback: Treats balance on all characteristics as equally important. (This is a drawback of exact and euclidian as well.)

Propensity score

Reduces matching to a single dimension:

$$\pi(X_i) \equiv Pr(D_i = 1|X_i)$$

Effectively defines "distance" in terms of the *strength of each covariate's relationship with treatment*

Propensity score

Reduces matching to a single dimension:

$$\pi(X_i) \equiv Pr(D_i = 1|X_i)$$

Effectively defines "distance" in terms of the *strength of each covariate's relationship with treatment*

Has the 'balancing property': $D_i \perp\!\!\!\perp X_i|\pi(X_i)$

> *Among units with the same propensity score, X_i is identically distributed between treated and untreated*

Implies the conditional ignorability assumption holds given just the propensity score:

$$\{Y_i(1), Y_i(0)\} \perp\!\!\!\perp D_i|\pi(X_i)$$

BUT requires correct specification of $\pi(X_i)$

Number of matches

Matching estimator for one-to-one matching:

$$\hat{\tau}_{ATT} = \frac{1}{n_1} \sum_{i:D_i=1} (Y_i - \tilde{Y}_i)$$

where $\tilde{Y}_i$ is the observed outcome of treated unit i's match

Number of matches

Matching estimator for one-to-one matching:

$$\hat{\tau}_{ATT} = \frac{1}{n_1} \sum_{i:D_i=1} (Y_i - \tilde{Y}_i)$$

where $\tilde{Y}_i$ is the observed outcome of treated unit i's match

Matching estimator for one-to-many matching:

$$\hat{\tau}_{ATT} = \frac{1}{n_1} \sum_{i:D_i=1} \left\{ Y_i - \left(\frac{1}{M_i} \sum_{m=1}^{M_i} \tilde{Y}_{i_m} \right) \right\}$$

Number of matches

Matching estimator for one-to-one matching:

$$\hat{\tau}_{ATT} = \frac{1}{n_1} \sum_{i:D_i=1} (Y_i - \tilde{Y}_i)$$

where $\tilde{Y}_i$ is the observed outcome of treated unit i's match

Matching estimator for one-to-many matching:

$$\hat{\tau}_{ATT} = \frac{1}{n_1} \sum_{i:D_i=1} \left\{ Y_i - \left(\frac{1}{M_i} \sum_{m=1}^{M_i} \tilde{Y}_{i_m} \right) \right\}$$

Bias-variance trade-off wrt choosing M_i:

- Small $M_i \Rightarrow$ small sample size, higher variance
- Large $M_i \Rightarrow$ worse matches, higher bias

Check balance

So there's a bit of trial and error in choosing the distance metric and number of matches

Goal is to achieve balance while maximizing sample size

Balance metrics:

- Difference in means
- Kolmogorov-Smirnov test of whether $X_{i:D_i=1,k}$ and $X_{i:D_i=0,k}$ are drawn from the same distribution
- Variance ratios

Check balance

So there's a bit of trial and error in choosing the distance metric and number of matches

Goal is to achieve balance while maximizing sample size

Balance metrics:

- Difference in means
- Kolmogorov-Smirnov test of whether $X_{i:D_i=1,k}$ and $X_{i:D_i=0,k}$ are drawn from the same distribution
- Variance ratios

Warning: Balance tests can be misleading if you can make everything insignificant by dropping lots of observations

Estimating ATT/ATC/ATE

$$\hat{\tau}_{ATT} = \frac{1}{n_1} \sum_{i:D_i=1} \left\{ Y_i - \left(\frac{1}{M_i} \sum_{m=1}^{M_i} \tilde{Y}_{i_m} \right) \right\}$$

Estimating ATT/ATC/ATE

$$\hat{\tau}_{ATT} = \frac{1}{n_1} \sum_{i:D_i=1} \left\{ Y_i - \left(\frac{1}{M_i} \sum_{m=1}^{M_i} \tilde{Y}_{i_m} \right) \right\}$$

$$\hat{\tau}_{ATC} = \frac{1}{n_0} \sum_{i:D_i=0} \left\{ Y_i - \left(\frac{1}{M_i} \sum_{m=1}^{M_i} \tilde{Y}_{i_m} \right) \right\}$$

Estimating ATT/ATC/ATE

$$\hat{\tau}_{ATT} = \frac{1}{n_1} \sum_{i:D_i=1} \left\{ Y_i - \left(\frac{1}{M_i} \sum_{m=1}^{M_i} \tilde{Y}_{i_m} \right) \right\}$$

$$\hat{\tau}_{ATC} = \frac{1}{n_0} \sum_{i:D_i=0} \left\{ Y_i - \left(\frac{1}{M_i} \sum_{m=1}^{M_i} \tilde{Y}_{i_m} \right) \right\}$$

$$\hat{\tau}_{ATE} = \hat{\tau}_{ATT} Pr(D_i = 1) + \hat{\tau}_{ATC} Pr(D_i = 0)$$

Bias

- Where does bias come from?

Bias

- Where does bias come from?
- The difference between $X_{i,t}$ (covariates for a treated unit, i) and $X_{i,m}$ (covariates for a treated unit i's match(es)). In other words, the extent to which we are **NOT** able to perfectly match across all covariates

Bias

- Where does bias come from?
- The difference between $X_{i,t}$ (covariates for a treated unit, i) and $X_{i,m}$ (covariates for a treated unit i's match(es)). In other words, the extent to which we are **NOT** able to perfectly match across all covariates
- Therefore, exact matching is *unbiased*, but often *unfeasible*.

Matching in R

Plenty of different package options but we will look at `Matching` today.

Matching in R

Plenty of different package options but we will look at `Matching` today.

- To set up balance formula – `MatchBalance(formula, data, match.out)`: where `formula` is in the form "`treatment ~ X1 + X2 + X3 ...`", and `match.out` is output from the `Match` command

Matching in R

Plenty of different package options but we will look at `Matching` today.

- To set up balance formula – `MatchBalance(formula, data, match.out)`: where `formula` is in the form "`treatment ~ X1 + X2 + X3 ...`", and `match.out` is output from the `Match` command
- To get a balance table – `baltest.collect(mb.output, varnames, after)`: where `mb.output` is an object created with `MatchBalance`, `varnames` is a vector of variable names, and `after` is logical (`TRUE/FALSE`) indicating whether results are from before or after matching (you need to install package `ebal` to implement `baltest.collect()`) .

Matching in R

Pay attention to the arguments of the `Match` command (from the `Matching` package).

Matching in R

Pay attention to the arguments of the `Match` command (from the `Matching` package).

- `Y`: Outcome
- `Tr`: Treatment
- `X`: Variables we want to match on
- `estimand`: Causal estimand of interest (default is ATT)
- `M`: Number of matches
- `Weight`: What weighting scheme to use (2 is Mahalanobis)
- `exact`: logical (`TRUE/FALSE`) for whether exact matching should be used
- `BiasAdjust`: logical (`TRUE/FALSE`) for whether bias adjustment should be applied

Matching in Python using PsmPy

- Initialize the PsmPy class:

```
psm = PsmPy(df, treatment='treatment', indx='
    pat_id', exclude = [])
```

- Calculate propensity scores:

```
psm.logistic_ps(balance = False)
```

- One-to-one matching:

```
psm.knn_matched(matcher='propensity_logit',
    replacement=False, caliper=None,
    drop_unmatched=True)
```

- One-to-many matching:

```
psm.knn_matched_12n(matcher='propensity_logit',
     how_many=3)
```

Linear regression in R and Python

- R:

```
out <- lm(y ~ x + w, data = data)
```

- Python:

```
out = smf.ols(formula='y ~ x + w', data=data).
    fit()
```

(need to import `statsmodels.formula.api` as `smf`.)