ML for SS: Causal Machine Learning

Session 8

Dr. Richard M. Crowley
rcrowley@smu.edu.sg
http://rmc.link/

Overview

Papers

Chernozhukov et al. 2017 AAER

- Introduces a ML-based method for causal identification useful in standard DID and IV approaches
  - Focused on calculating ATE and ATTE

Gentzkow, Shapiro, and Taddy 2019 Econometrica

- A paper showing the methodological benefits that can come from careful merging of econometrics and machine learning

Technical Discussion

Focus on the DoubleML method

Python

- Use the `DoubleML` library

R

- The doubleML library is available in R as well
- The AAER paper's source code is also available
  - It's all R code!

Both languages work well for this

Double ML: Theory

Background

- There are a number of relevant papers published in economics in recent years developing and using Double ML
- The method is developed largely from:
  - Chernozhukov et al. (2017 AER), “Double/debiased/Neyman machine learning of treatment effects”
  - Chernozhukov et al. (2018 Econometrics J), “Double/debiased machine learning for treatment and structural parameters.”

> Impact or overlap with methodological work by Susan Athey, Matthew Gentzkow, Trevor Hastie, Guido Imbens, Matt Taddy, and Stefan Wager

What is Double ML?

1. Split your sample as you would for K-fold cross validation, into sets $\{I_k\}_{k\in\{1,\ldots,K\}}$
   - K sample of N/K observations each
   - Let $I_k^c = \cup\{I_j\}_{j\neq k}$
2. Construct K estimators using a machine learning estimator over nuisance parameters (e.g., controls) applied to the data I_K^c
3. Average the K estimators to obtain a final estimator
   - This average estimator is approximately unbiased and normally distributed
   - The estimator is also asymptotically efficient

> And repeat. Bootstrap this out and take the mean or median of the estimators

Where Double ML excels: Endogenous treatment

- Suppose a policy affects a subset of individuals (people, corporations, etc.)
- Suppose individuals have the ability to alter their treatment status
  - E.g., state laws (move), labor laws, etc.
- Linear controls may be insufficient to claim causality of the treatment on anything

> There are a lot of older methods that try to address this, though incompletely

1. Linear controls
2. Propensity score adjustments (e.g., weighting)
3. Matching methods
4. “doubly-robust” estimators

Why is machine learning needed?

- Suppose a true form of a specification is as follows
  - T is a treatment indicator, C is a vector of controls

$$Y = g_0(T, C) + \varepsilon_1$$
$$T = m_0(C) + \varepsilon_2$$

- We often assume g_0 to be something like $\alpha + \theta_0\ T + \gamma \cdot C$
- We often assume m_0 to be a constant (i.e., assume that T is exogenous)

> We know these assumptions aren't true!

Why is machine learning needed?

> How can we estimate a more general form for g_0 and m_0?

- We could use a more flexible econometric approach, such as including interactions between T and C
  - This is still very restrictive – purely linear
- We could include transformations of C and its interactions
  - This is still restrictive – T is additive separable
- We could use a nonparametric estimator!
  - This is where machine learning is very useful: efficient and reasonably accurate nonparametric estimation
    - LASSO, random forest, XGBoost, etc.

Model variants

- The models described in the last few slides are referred to as the “Interactive regression model” or IRM
- If you can separate your treatment effect from the controls but suspect nonlinear effects of controls, the “Partially linear regression model” or PLR is appropriate
  - Solves $Y = \theta_0 T + g_0(C) + \varepsilon_0$ and $T = m_0(C) + \varepsilon_2$
- There are also instrumental variable variants of both IRM and PLR

Reconciling these slides notation with the paper

- These slides use a somewhat simpler oriented notation.
- Reconciliation from slides to papers:
  - T is D
  - C is X
  - ε_0 is U or ζ depending on the paper
  - ε_1 is V

Implementing DoubleML

Walking through an implementation of DoubleML

> Problem: How does 401k participation impact wealth?

- This problem is walked through in Chernozhukov et al. (2017 AER, Web Appendix)
  - The R code for the AER paper is available from AER as well
    - Quite clean code at that!
- We will implement this in python using the `DoubleML` library
  - Which Chernozhukov was involved in the development of

Importing the data

- Conveniently, the data is available from the DoubleML package

```python
# Grab the dataset
import doubleml.datasets
df = dml.datasets.fetch_401K('DataFrame')
df
```

```
##           nifa  net_tfa        tw  age      inc  ...  twoearn  e401  p401  pira  hown
## 0          0.0      0.0    4500.0   47   6765.0  ...        0     0     0     0     1
## 1       6215.0   1015.0   22390.0   36  28452.0  ...        0     0     0     0     1
## 2          0.0  -2000.0   -2000.0   37   3300.0  ...        0     0     0     0     0
## 3      15000.0  15000.0  155000.0   58  52590.0  ...        1     0     0     0     1
## 4          0.0      0.0   58000.0   32  21804.0  ...        0     0     0     0     1
## ...        ...      ...       ...  ...      ...  ...      ...   ...   ...   ...   ...
## 9910   98498.0  98858.0  157858.0   52  73920.0  ...        0     1     1     0     1
## 9911     287.0   6230.0   15730.0   41  42927.0  ...        1     1     1     1     1
## 9912      99.0   6099.0    7406.0   40  23619.0  ...        0     1     0     1     0
## 9913       0.0    -32.0    2468.0   47  14280.0  ...        0     1     1     0     0
## 9914    4000.0   5000.0    8857.0   33  11112.0  ...        0     1     1     0     0
##
## [9915 rows x 14 columns]
```

Using your own data

- We can also do this manually, by importing the Stata file from AER
- We then need to prep the data into the format `DoubleML` expects
  - This is fairly straightforward, just defining our Y, treatment, and control variables

```python
df = pd.read_stata('../../Data/S8_sipp1991.dta')

y = 'net_tfa'
treat = 'e401'
controls = [x for x in df.columns.tolist() if x not in [y, treat]]

df_dml = dml.DoubleMLData(df, y_col=y, d_cols=treat, x_cols=controls)
```

What is the data format used by DoubleML?

```python
print(df_dml)
```

```
## ================== DoubleMLData Object ==================
##
## ------------------ Data summary      ------------------
## Outcome variable: net_tfa
## Treatment variable(s): ['e401']
## Covariates: ['nifa', 'tw', 'age', 'inc', 'fsize', 'educ', 'db', 'marr', 'twoearn', 'p401', 'pira', 'hown']
## Instrument variable(s): None
## No. Observations: 9915
##
## ------------------ DataFrame info    ------------------
## <class 'pandas.core.frame.DataFrame'>
## Int64Index: 9915 entries, 0 to 9914
## Columns: 14 entries, nifa to hown
## dtypes: float32(4), int8(10)
## memory usage: 329.2 KB
```

- Pandas dataframe
- A pre-specified outcome variable
- One or more treatment indicators
- One or more controls
- Optional instruments

Set up the Nuisance functions

- Recall that there are two functions, m_0 and g_0 that need to be solved for this method
- We can specify any form for these that we want, so long as they are consistent with Scikit-learn

g_0: *Continuous GBM*

```python
g_0 = GradientBoostingRegressor(
  loss='ls',
  learning_rate=0.01,
  n_estimators=1000,
  subsample=0.5,
  max_depth=2
  )
```

m_0: *Binary GBM*

```python
m_0 = GradientBoostingClassifier(
  loss='exponential',
  learning_rate=0.01,
  n_estimators=1000,
  subsample=0.5,
  max_depth=2
  )
```

Run the DML model: Average Treatment Effects

```python
# Fix the random number generator for replicability
np.random.seed(1234)
# Run the model
dml_model_irm = dml.DoubleMLIRM(df_dml, g_0, m_0)
# Output the model's findings
print(dml_model_irm.fit())
```

```
## ================== DoubleMLIRM Object ==================
##
## ------------------ Data summary      ------------------
## Outcome variable: net_tfa
## Treatment variable(s): ['e401']
## Covariates: ['nifa', 'tw', 'age', 'inc', 'fsize', 'educ', 'db', 'marr', 'twoearn', 'p401', 'pira', 'hown']
## Instrument variable(s): None
## No. Observations: 9915
##
## ------------------ Score & algorithm ------------------
## Score function: ATE
## DML algorithm: dml2
##
## ------------------ Resampling        ------------------
## No. folds: 5
## No. repeated sample splits: 1
## Apply cross-fitting: True
##
## ------------------ Fit summary       ------------------
##             coef     std err         t          P>|t|        2.5 %        97.5 %
## e401  3320.43343  383.604082  8.655887  4.890947e-18  2568.583245  4072.283614
```

Run the DML model: ATTE

- ATTE: Average Treatment Effects of the Treated

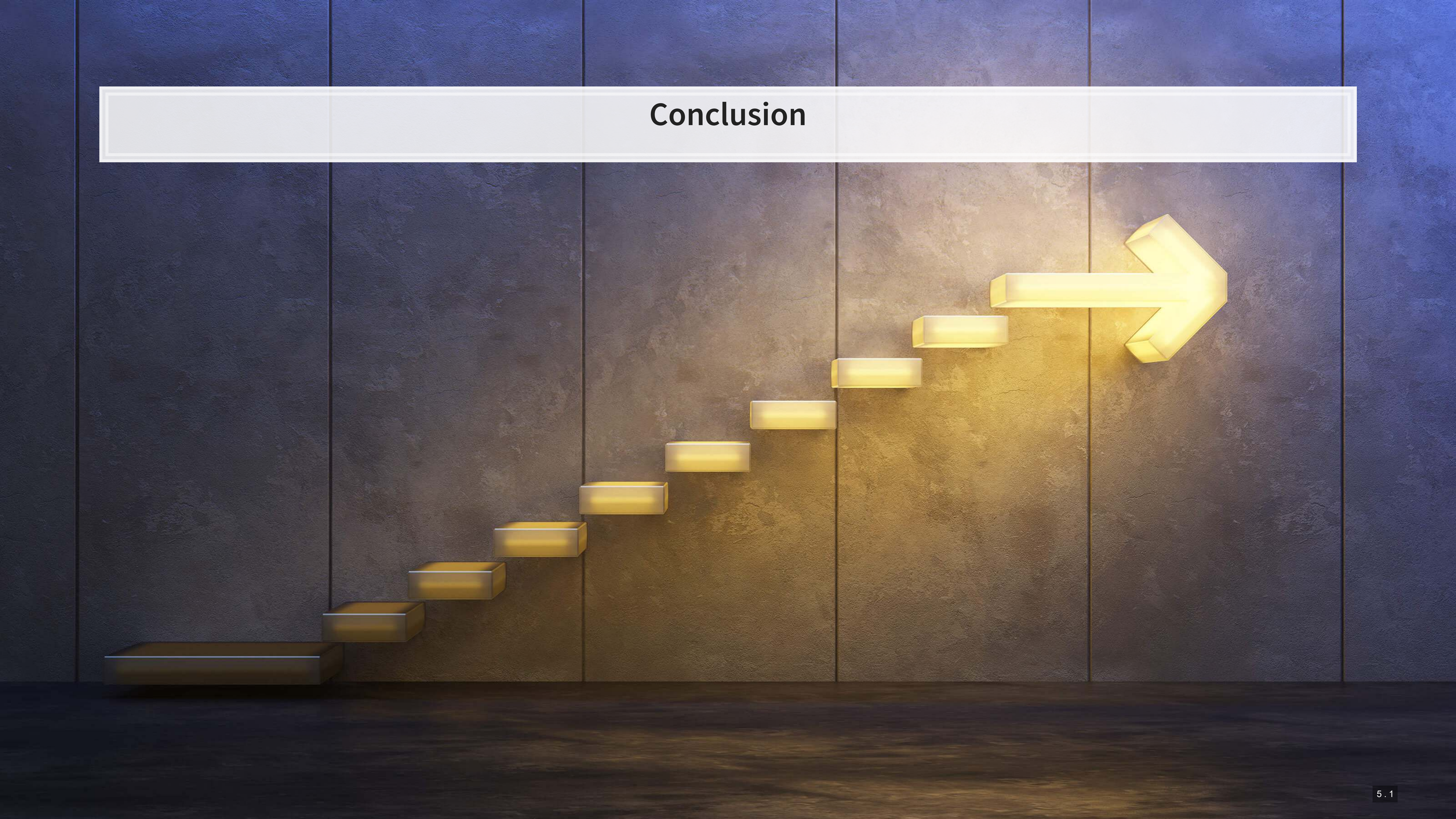
```python
# Run the model
dml_model_irm_ATTE = dml.DoubleMLIRM(df_dml, g_0, m_0, score='ATTE')
# Output the model's findings
print(dml_model_irm_ATTE.fit())
```

```
## ================== DoubleMLIRM Object ==================
##
## ------------------ Data summary      ------------------
## Outcome variable: net_tfa
## Treatment variable(s): ['e401']
## Covariates: ['nifa', 'tw', 'age', 'inc', 'fsize', 'educ', 'db', 'marr', 'twoearn', 'p401', 'pira', 'hown']
## Instrument variable(s): None
## No. Observations: 9915
##
## ------------------ Score & algorithm ------------------
## Score function: ATTE
## DML algorithm: dml2
##
## ------------------ Resampling        ------------------
## No. folds: 5
## No. repeated sample splits: 1
## Apply cross-fitting: True
##
## ------------------ Fit summary       ------------------
##               coef     std err          t          P>|t|        2.5 %        97.5 %
## e401  10081.312662  392.074708  25.712734  8.421563e-146  9312.860354  10849.764969
```

Other twists on the model

1. Change the machine learning backend
   - Our models used `dml2`
   - You can switch to `dml1` using `dml_procedure='dml1'`
   - `dml1` follows the math in these slides
     - Solve for a condition equal to zero for each model, and then average the estimators
     - `dml2` solves the for the average of the condition being equal to zero overall
2. Run multiple iterations of the model
   - The paper uses 100 iterations, emulate this by adding `n_rep=100`
3. Change the machine learning models fed to the DoubleML model
   - An example of using “Histogram-based Gradient Boosting” is in the Jupyter notebook
     - This is a much faster GBM-like model

Conclusion



Wrap-up

Double Machine Learning can help in cleaning identifying treatment effects

- Easy to implement as well!

ML and Econometrics are not at odds with one another

- You can use ML to strengthen an econometric framework

ML is essentially just another tool in the econometrics toolbox!

Packages used for these slides

Python

- doubleML
- numpy
- pandas

References

- Chernozhukov, Victor, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, and Whitney Newey. "Double/debiased/neyman machine learning of treatment effects." American Economic Review 107, no. 5 (2017): 261-65.
- Chernozhukov, Victor, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. "Double/debiased machine learning for treatment and structural parameters." (2018): C1-C68.
- Gentzkow, Matthew, Jesse M. Shapiro, and Matt Taddy. "Measuring group differences in high-dimensional choices: method and application to congressional speech." Econometrica 87, no. 4 (2019): 1307-1340.