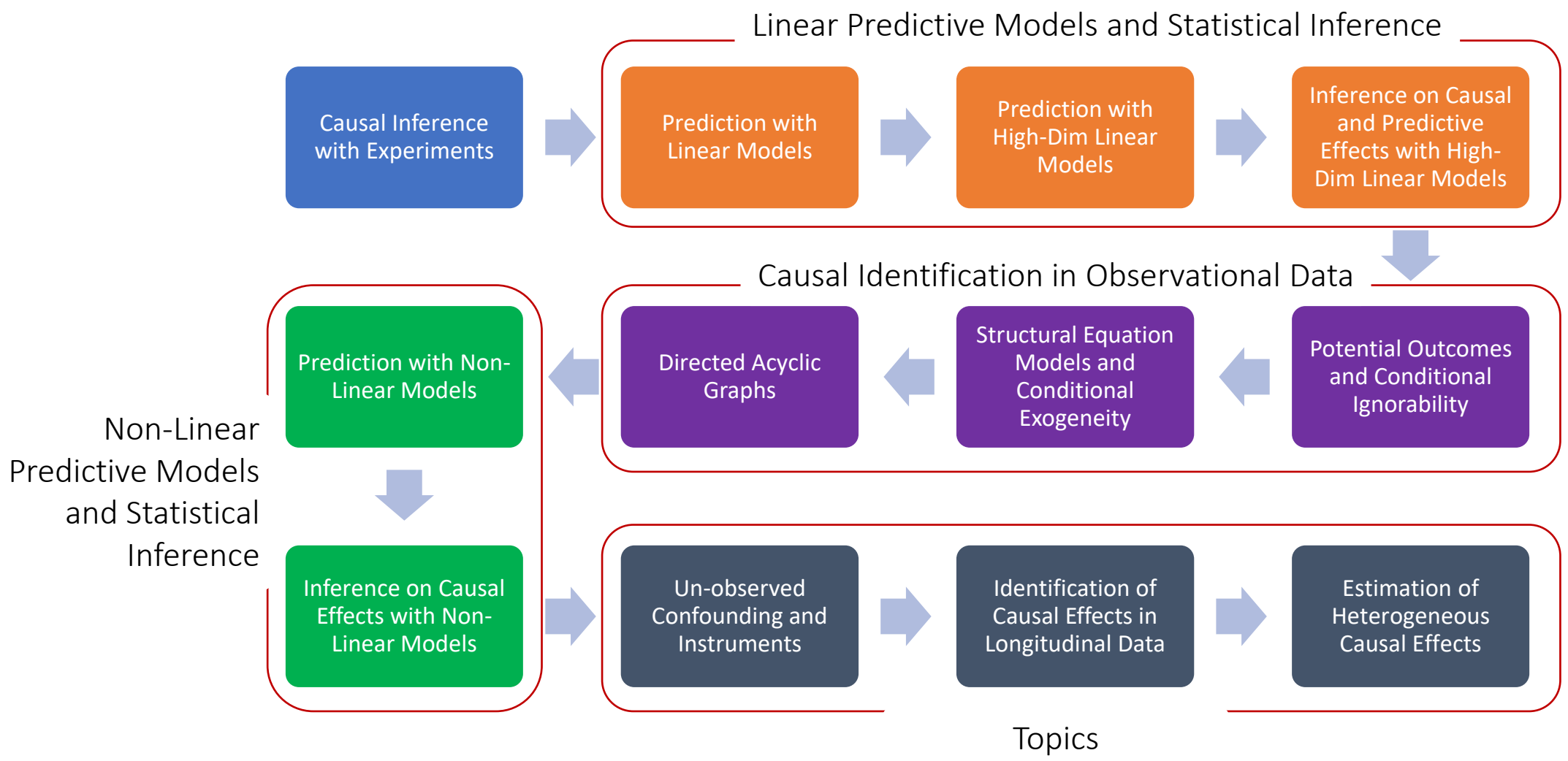
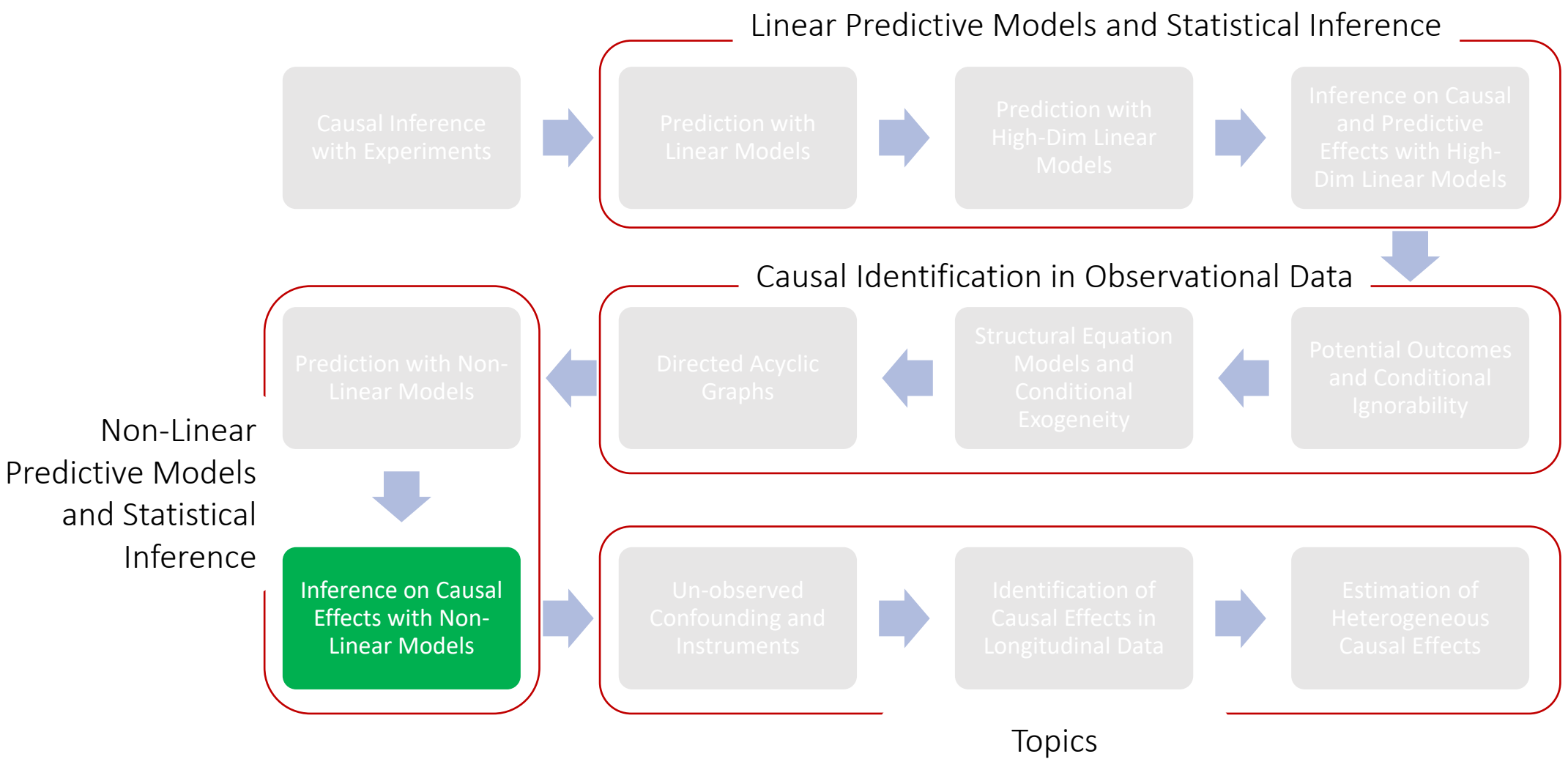
MS&E 228: Inference with Modern Non-Linear Prediction

Vasilis Syrgkanis

MS&E, Stanford


Linear Predictive Models and Statistical Inference
Causal Inference with Experiments
Prediction with Linear Models
Prediction with High-Dim Linear Models
Inference on Causal and Predictive Effects with High-Dim Linear Models
Causal Identification in Observational Data
Potential Outcomes and Conditional Ignorability
Structural Equation Models and Conditional Exogeneity
Directed Acyclic Graphs
Non-Linear Predictive Models and Statistical Inference
Prediction with Non-Linear Models
Inference on Causal Effects with Non-Linear Models
Un-observed Confounding and Instruments
Identification of Causal Effects in Longitudinal Data
Estimation of Heterogeneous Causal Effects
Topics


Linear Predictive Models and Statistical Inference
Causal Inference with Experiments
Prediction with Linear Models
Prediction with High-Dim Linear Models
Inference on Causal and Predictive Effects with High-Dim Linear Models
Causal Identification in Observational Data
Potential Outcomes and Conditional Ignorability
Structural Equation Models and Conditional Exogeneity
Directed Acyclic Graphs
Prediction with Non-Linear Models
Non-Linear Predictive Models and Statistical Inference
Inference on Causal Effects with Non-Linear Models
Un-observed Confounding and Instruments
Identification of Causal Effects in Longitudinal Data
Estimation of Heterogeneous Causal Effects
Topics

Goals for Today

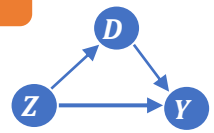
- Revisiting and expanding on theory from last time
- Sample-splitting and cross-fitting
- Example application

Recap of Last Lecture

Causal Inference Pipeline

Theory

DAGs/ASEMs



a_d
=
$E[E[Y|D=d,Z]]$

Construct estimate $\hat{g}(d,z)$ of predictive model
$E[Y \mid D=d, Z=z]$
Estimate mean counterfactual
$\hat{a}_d = E_n[\hat{g}(d,Z)]$

$a_d \in \hat{a}_d \pm 2\hat{\sigma}$

Practice

Data Collection → Domain Assumption Elicitation → Identification → Estimation (training) → Validation (testing) → Inference (Confidence Intervals) → Sensitivity Analysis → Decision Policy → Adaptive Experiments → Data Collection

There is no way to validate my causal estimate $\hat{\alpha}_d$, given the data that I have! I cannot observe counterfactuals. The best alternative is to **quantify uncertainty of my estimate** to see how confident I should be in the result.

$a_d \in \hat{a}_d \pm (2\hat{\sigma} + \epsilon)$

Semi-Parametric Moment Restrictions

- Observe samples $Z_1, \ldots, Z_n$ i.i.d. from data distribution D
- Distribution D satisfies vector of moment restrictions

$$M(\theta_0, g_0) \coloneqq E_{Z \sim D}[m(Z; \theta_0, g_0)] = 0$$

- $\theta_0 \in R^d$ finite dimensional target parameter of interest
- $g_0 \in G$ potentially infinite dimensional (an un-known function) we don't care (nuisance)
- g_0 is un-known and needs to be estimated from data

Natural Estimation Algorithm (sample-splitting)

- Split the data in half
- On first half, estimate $\hat{g}$ of g_0
- On second half, solution w.r.t. $\hat{\theta}$ of empirical plug-in moment equation:

$$M_n(\hat{\theta}, \hat{g}) := \frac{1}{n_2} \sum_{i \in S_2} m(Z_i; \hat{\theta}, \hat{g}) = 0$$

Natural Estimation Algorithm (cross-fitting)

- Split data in K parts, $S_1, \ldots, S_K$
- For each part k, estimate $\hat{g}_k$ using data from all parts except S_k
- Return solution $\hat{\theta}$ to cross-fitted empirical moment equation:

$$\frac{1}{n}\sum_{k=1}^{K}\sum_{i\in S_k} m\left(Z_i;\hat{\theta},\hat{g}_k\right) = 0$$

Formal Definition

- Moment $M(\theta, g)$ is Neyman orthogonal if for any $\nu \in G - g_0$:

$$D_g M(\theta_0, g_0)[\nu] := \frac{\partial}{\partial t} M(\theta_0, g_0 + t\,\nu)\Big|_{t=0} = 0$$

Main Theorem

- If moment is Neyman orthogonal and RMSE of $\hat{g}$ is $o_p\left(n^{-1/4}\right)$ *

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \rightarrow N\left(0, A^{-1}\Sigma\left(\mathrm{A}^{-1}\right)^{\mathsf{T}}\right)$$

- $A = \nabla_\theta M(\theta_0, g_0)$ and $\Sigma = E\left[m(Z; \theta_0, g_0)\, m(Z; \theta_0, g_0)^{\mathsf{T}}\right]$

*plug regularity conditions

Main Theorem (expanded)

Define RMSE: $\|h\|_{L^2} = \sqrt{E[h(X)^2]}$

- If moment is Neyman orthogonal and RMSE of $\hat{g}$ goes down at rate $n^{1/4}$, plus regularity conditions

$$n^{1/4}\|\hat{g} - g_0\|_{L^2} \approx 0$$

- Then the estimate $\hat{\theta}$ is *asymptotically linear*

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \approx \sqrt{n}\, E_n[\phi_0(Z)], \qquad \phi_0(Z) = -J_0^{-1}\, m(Z; \theta_0, g_0), \qquad J_0 \coloneqq \partial_\theta E[m(Z; \theta_0, g_0)]$$

Jacobian of moments with respect to parameter; relates to identification strength

influence function

- Consequently, it is *asymptotically normal*

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \sim_a N(0, V), \qquad V \coloneqq E[\phi_0(Z)\phi_0(Z)']$$

Covariance of the influence function

- *Confidence intervals* for any projection based on estimate of variance are asymptotically valid

$$\ell'\theta \in \left[\ell'\hat{\theta} \pm c\sqrt{\frac{\ell'\hat{V}\ell}{n}}\right], \qquad \hat{\mathrm{V}} = \mathrm{Var_n}\left(\hat{\phi}(Z)\right), \qquad \hat{\phi}(Z) \coloneqq -\hat{J}^{-1} m\left(Z; \hat{\theta}, \hat{g}\right), \qquad \hat{J} = \partial_\theta E_n\left[m\left(Z; \hat{\theta}, \hat{g}\right)\right]$$

Empirical variance of approximate influence

Approximate influence function

Empirical average of jacobian

Python Pseudocode

```
# General DML pseudocode
def general_dml(Z, ell, nfolds, moment, jacobian, nuisance_estimator):
    # construct out-of-fold predictions from the nuisance estimator
    ghat = cross_val_predict(nuisance_estimator, Z, cv=nfolds)
    # use these predictions to define the empirical moment equation
    # with respect to theta
    avg_moment = lambda theta: np.mean(moment(Z, theta, ghat), axis=0)
    # solve for the empirical moment equation equals zero
    thetahat = fsolve(avg_moment)
    # calculate empirical jacobian with respect to theta, evaluated
    # at the estimates thetahat and ghat
    Jhat = np.mean(jacobian(Z, thetahat, ghat), axis=0)
    # construct approximate influence function for each sample
    phihat = - moment(Z, thetahat, ghat) @ np.linalg.pinv(Jhat).T
    # variance estimate
    var = phihat.T @ phihat / phihat.shape[0]
    # estimate and standard error for any projection ell of the parameters
    point = ell @ thetahat
    stderr_ell = np.sqrt(ell.T @ var @ ell / phihat.shape[0])
    return point, stderr_ell
```

Main Theorem (linear moments)

- If moments are linear

$$m(Z;\theta,g) = \nu(Z;g) - a(Z;g)\theta$$

- Estimate is closed form:

$$\hat{\theta} = \hat{J}^{-1}E_n[\nu(Z;g)], \qquad \hat{J} = E_n[a(Z;g)]$$

- Then the estimate $\hat{\theta}$ is *asymptotically linear*

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \approx \sqrt{n}\, E_n[\phi_0(Z)], \qquad \phi_0(Z) = -J_0^{-1}\, m(Z;\theta_0,g_0), \qquad J_0 \coloneqq E[a(Z;\ g_0)]$$

- Consequently, it is *asymptotically normal*

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \sim_a N(0,V), \qquad V \coloneqq E[\phi_0(Z)\phi_0(Z)']$$

- *Confidence intervals* for any projection based on estimate of variance are asymptotically valid

$$\ell'\theta \in \left[\ell'\hat{\theta} \pm c\sqrt{\frac{\ell'\hat{V}\ell}{n}}\right], \qquad \hat{\mathrm{V}} = \mathrm{Var_n}\left(\hat{\phi}(Z)\right), \qquad \hat{\phi}(Z) \coloneqq -\hat{J}^{-1}m\left(Z;\hat{\theta},\hat{g}\right), \qquad \hat{J} = E_n[a(Z;\hat{g})]$$

Python Pseudocode

```
# General DML pseudocode for linear moments: m(Z; theta, g) = nu(Z;g) - alpha(Z;g) theta
def general_dml_linear_moment(Z, ell, nfolds, alpha, nu, jacobian, nuisance_estimator):
    # construct out-of-fold predictions from the nuisance estimator
    ghat = cross_val_predict(nuisance_estimator, Z, cv=nfolds)
    # use these predictions to define the empirical moment equation
    # and solve explicitly with respect to theta
    Jhat = np.mean(alpha(Z, ghat), axis=0)
    avg_nu = np.mean(nu(Z, ghat), axis=0)
    # solve for the empirical moment equation equals zero
    invJhat = np.linalg.pinv(Jhat)
    thetahat = invJhat @ avg_nu
    # construct approximate influence function for each sample
    phihat = - (nu(Z, ghat) - alpha(Z, ghat) @ thetahat) @ invJhat.T
    # variance estimate
    var = phihat.T @ phihat / phihat.shape[0]
    # estimate and standard error for any projection ell of the parameters
    point = ell @ thetahat
    stderr_ell = np.sqrt(ell.T @ var @ ell / phihat.shape[0])
    return point, stderr_ell
```

ATE under Conditional Exogeneity

- We observe n samples $Z_1, \ldots, Z_n$ where $Z_i = (X_i, D_i, Y_i)$

- Want to estimate average effect θ_0, which satisfies:

$$M(\theta_0, g_0) \coloneqq E[g_0(1, X) - g_0(0, X) - \theta_0] = 0$$

- Where:

$$g_0(D, X) \coloneqq E[\, Y \mid D, X \,]$$

- We want to be able to use ML to learn regression function g!

Better Moment for ATE

- Add a "debiasing" correction:

$$\widetilde{M}(\theta, g, a) = M(\theta, g) + E\left[a(D, X)\left(Y - g(D, X)\right)\right]$$

- What is a_0? Should be such that

$$E[a_0(D, X)\, g(D, X)] = E[g(1, X) - g(0, X)]$$

- The following works: inverse propensity scoring

$$a_0(D, X) = \frac{D}{\Pr[D = 1|X]} - \frac{1 - D}{\Pr[D = 0|X]}$$

- Doubly robust estimation algorithm

Main Theorem (expanded) Define RMSE: $\|h\|_{L^2} = \sqrt{E[h(X)^2]}$

- If moment is Neyman orthogonal and RMSE of $\hat{g}$ goes down at rate $n^{1/4}$, plus regularity conditions
$$n^{1/4}\|\hat{g} - g_0\|_{L^2} \approx 0$$

- Then the estimate $\hat{\theta}$ is *asymptotically linear*
$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \approx \sqrt{n}\, E_n[\phi_0(Z)], \qquad \phi_0(Z) = -J_0^{-1}\, m(Z; \theta_0, g_0), \qquad J_0 \coloneqq \partial_\theta E[m(Z; \theta_0, g_0)]$$

- Consequently, it is *asymptotically normal*
$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \sim_a N(0, V), \qquad V \coloneqq E[\phi_0(Z)\phi_0(Z)']$$

- *Confidence intervals* for any projection based on estimate of variance are asymptotically valid

$$\ell'\theta \in \left[\ell'\hat{\theta} \pm c\sqrt{\frac{\ell'\hat{\mathbb{V}}\ell}{n}}\right], \qquad \hat{\mathbb{V}} = \mathrm{Var_n}\left(\hat{\phi}(Z)\right), \qquad \hat{\phi}(Z) \coloneqq -\hat{J}^{-1}m\left(Z; \hat{\theta}, \hat{g}\right), \qquad \hat{J} = \partial_\theta E_n\left[m\left(Z; \hat{\theta}, \hat{g}\right)\right]$$

Inference with Doubly Robust Algorithm

- Moment is Neyman orthogonal

$$m(Z;\theta,g) \coloneqq g(1,X) - g(0,X) - a(D,X)\left(Y - g(D,X)\right) - \theta$$

- If RMSE of propensity and regression model goes down at rate $n^{1/4}$, plus regularity conditions
- Then the estimate $\hat{\theta}$ is *asymptotically linear*

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \approx \sqrt{n}\, E_n[\phi_0(Z)], \qquad \phi_0(Z) = -m(Z;\theta_0,g_0), \qquad J_0 \coloneqq 1$$

- Consequently, it is *asymptotically normal*

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \sim_a N(0,V), \qquad V \coloneqq E\left[m(Z;\theta_0,g_0)^2\right]$$

- *Confidence intervals* for any projection based on estimate of variance are asymptotically valid

$$\ell'\theta \in \left[\ell'\hat{\theta} \pm c\sqrt{\frac{\ell'\hat{V}\ell}{n}}\right], \qquad \hat{\mathrm{V}} = \mathrm{Var_n}\left(\hat{\phi}(Z)\right), \qquad \hat{\phi}(Z) \coloneqq -m\left(Z;\hat{\theta},\hat{g}\right), \qquad \hat{J} = 1$$

Python Pseudocode

```
cv = KFold(n_splits=nfolds, shuffle=True, random_state=123)
yhat0, yhat1 = np.zeros(y.shape), np.zeros(y.shape)
# we will fit a model E[Y| D, X] by fitting a separate model for D==0
# and a separate model for D==1.
for train, test in cv.split(X, y):
    # train a model on training data that received zero and predict on all test data
    yhat0[test] = modely.fit(X[train][D[train]==0], y[train][D[train]==0]).predict(X[test])
    # train a model on training data that received one and predict on all test data
    yhat1[test] = modely.fit(X[train][D[train]==1], y[train][D[train]==1]).predict(X[test])
# prediction for observed treatment
yhat = yhat0 * (1 - D) + yhat1 * D
# propensity scores
Dhat = cross_val_predict(modeld, X, D, cv=cv, method='predict_proba', n_jobs=-1)[:, 1]
Dhat = np.clip(Dhat, trimming, 1 - trimming)
# doubly robust quantity for every sample
drhat = yhat1 - yhat0 + (y - yhat) * (D/Dhat - (1 - D)/(1 - Dhat))
point = np.mean(drhat)
var = np.var(drhat)
stderr = np.sqrt(var / X.shape[0])
return point, stderr, yhat, Dhat, y - yhat, D - Dhat, drhat
```

Partially Linear Model

- Relevant in many applications: dose-response curve in healthcare, effect of price on demand, return-on-investment
- Assume conditional exogeneity

$$Y^{(d)} \perp D \mid X$$

- Assume partially linear response: $E[\, Y \mid D, X \,] = \theta_0 D + f_0(X)$

$$Y = \theta_0 D + f_0(X) + \epsilon, \qquad E[\epsilon | D, X] = 0$$

- Parameter of interest θ_0 is constant marginal effect of treatment
- Orthogonal moment are normal equations after residualization

$$\tilde{Y} = \theta_0 \tilde{D} + \epsilon, \qquad E\left[\epsilon \middle| \tilde{D}\right] = 0 \Rightarrow E\left[\left(\tilde{Y} - \theta_0 \tilde{D}\right)\tilde{D}\right] = 0$$

Orthogonal Method: Double ML

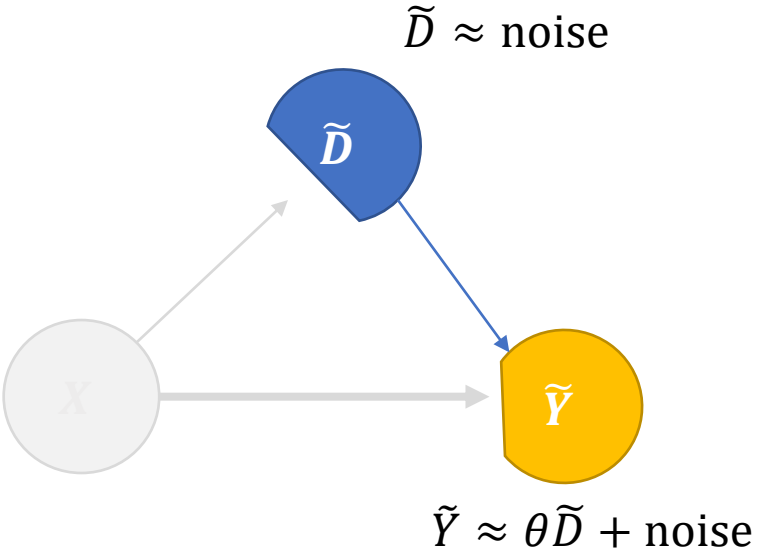
- Double ML. Split samples in half
  - Regress $Y \sim X$ with ML on first half, to get estimate $\hat{h}(S)$ of $E[Y|X]$
  - Regress $D \sim X$ with ML on first half, to get estimate $\hat{p}(S)$ of $E[D|X]$
  - Construct residuals on other half, $\widehat{D} \coloneqq D - \hat{p}(X)$ and $\hat{Y} \coloneqq Y - \hat{h}(X)$
  - Run OLS on residuals: $\hat{Y} \sim \widehat{D}$ to get $\hat{\theta}$

- OLS equivalent to solving moment condition:

$$E_n\left[\left(\hat{Y} - \theta\widehat{D}\right)\widehat{D}\right] = 0$$

- Orthogonal Moment condition:

$$M(\theta, h, p) = E\left[\left(Y - h(X) - \theta\left(D - p(X)\right)\right)\ \left(D - p(X)\right)\right]$$



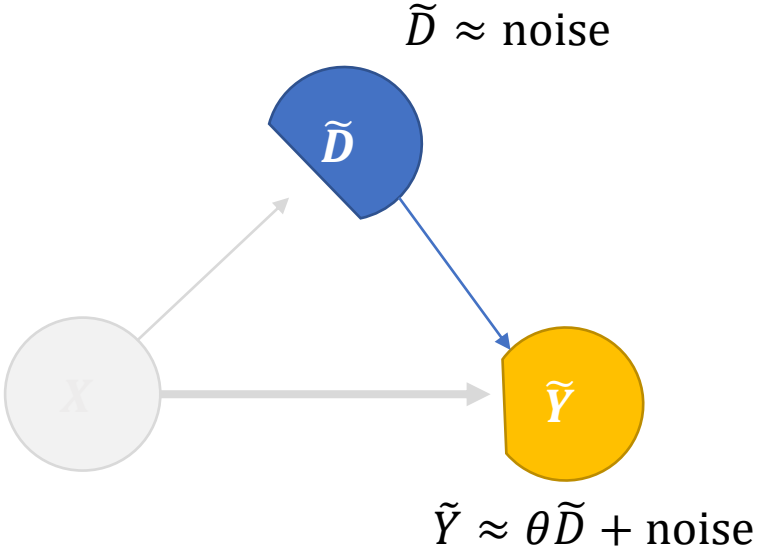
Orthogonal Method: Double ML

- OLS equivalent to solving moment condition:

$$E\left[\left(\tilde{Y} - \theta\tilde{D}\right)\tilde{D}\right] = 0$$

- Orthogonal Moment condition:

$$M(\theta, h, p) = E\left[\left(Y - h(X) - \theta\left(D - p(X)\right)\right)\ \left(D - p(X)\right)\right]$$



Verifying orthogonality

- Directional derivative with respect to h

$$\partial_t E\left[\left(Y - h_0(X) + t\,\nu(X) - \theta_0\left(D - p_0(X)\right)\right)\ \left(D - p_0(X)\right)\right]\Big|_{t=0} = 0$$

- Directional derivative with respect to p

$$\partial_t E\left[\left(Y - h_0(X) - \theta_0\left(D - p_0(X) - t\,\nu(X)\right)\right)\ \left(D - p_0(X) - t\,\nu(X)\right)\right]\Big|_{t=0} = 0$$

Main Theorem (expanded) Define RMSE: $\|h\|_{L^2} = \sqrt{E[h(X)^2]}$

- If moment is Neyman orthogonal and RMSE of $\hat{g}$ goes down at rate $n^{1/4}$, plus regularity conditions
$$n^{1/4}\|\hat{g} - g_0\|_{L^2} \approx 0$$

- Then the estimate $\hat{\theta}$ is *asymptotically linear*
$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \approx \sqrt{n}\, E_n[\phi_0(Z)], \qquad \phi_0(Z) = -J_0^{-1}\, m(Z; \theta_0, g_0), \qquad J_0 \coloneqq \partial_\theta E[m(Z; \theta_0, g_0)]$$

- Consequently, it is *asymptotically normal*
$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \sim_a N(0, V), \qquad V \coloneqq E[\phi_0(Z)\phi_0(Z)']$$

- *Confidence intervals* for any projection based on estimate of variance are asymptotically valid

$$\ell'\theta \in \left[\ell'\hat{\theta} \pm c\sqrt{\frac{\ell'\hat{\mathbb{V}}\ell}{n}}\right], \qquad \hat{\mathbb{V}} = \mathrm{Var_n}\left(\hat{\phi}(Z)\right), \qquad \hat{\phi}(Z) \coloneqq -\hat{J}^{-1}m\left(Z; \hat{\theta}, \hat{g}\right), \qquad \hat{J} = \partial_\theta E_n\left[m\left(Z; \hat{\theta}, \hat{g}\right)\right]$$

Inference with DML in PLR Setting

- If RMSE of propensity model and outcome model goes down at rate $n^{1/4}$, plus regularity conditions
- Then the estimate $\hat{\theta}$ is *asymptotically linear*

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \approx \sqrt{n}\, E_n[\phi_0(Z)], \qquad \phi_0(Z) = -J_0^{-1}\left(\tilde{Y} - \theta_0 \tilde{D}\right)\tilde{D}, \qquad J_0 \coloneqq E\left[\tilde{D}^2\right]$$

- Consequently, it is *asymptotically normal*

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \sim_a N(0, V), \qquad V \coloneqq \frac{E\left[\left(\tilde{Y} - \theta_0 \tilde{D}\right)^2 \tilde{D}^2\right]}{E\left[\tilde{D}^2\right]^2}$$

- *Confidence intervals* for any projection based on estimate of variance are asymptotically valid

$$\ell'\theta \in \left[\ell'\hat{\theta} \pm c\sqrt{\frac{\ell'\hat{V}\ell}{n}}\right], \qquad \hat{\mathrm{V}} = \mathrm{Var_n}\left(\hat{\phi}(Z)\right), \qquad \hat{\phi}(Z) \coloneqq -\hat{J}^{-1}\left(\hat{Y} - \hat{\theta}\hat{D}\right)\hat{D}, \qquad \hat{J} = E_n\left[\hat{D}^2\right]$$

Python Pseudocode

```
cv = KFold(n_splits=nfolds, shuffle=True, random_state=123)
# out-of-fold predictions for y
yhat = cross_val_predict(modely, X, y, cv=cv, n_jobs=-1)
# out-of-fold predictions for D
Dhat = cross_val_predict(modeld, X, D, cv=cv, n_jobs=-1)
# calculate outcome and treatment residuals
resy = y - yhat
resD = D - Dhat
# final stage ols based point estimate and standard error
point = np.mean(resy * resD) / np.mean(resD**2)
epsilon = resy - point * resD
var = np.mean(epsilon**2 * D**2) / np.mean(D**2)**2
stderr = np.sqrt(var / X.shape[0])
return point, stderr
```

Practical Variants of Sample-Splitting

Cross-fitting and semi-cross-fitting

Cross-fitting

- Sample splitting is statistically lossy

- Only half of the data are used for the final parameter estimation
- Can we utilize all the data?

- *Cross-fitting:* analogous to cross-validation
- Use the second half to train g and predict on first half
- Then calculate parameter using all the data

Cross-fitting Estimation Algorithm

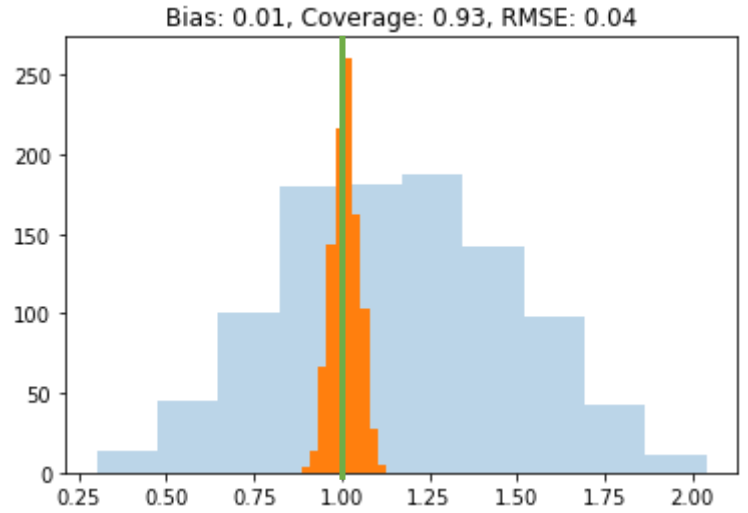
- Split the data in half
- On first half, estimate $\hat{g}_1$ of g_0 and predict on second half
- On second half, estimate $\hat{g}_2$ of g_0 and predict on first half
- On all data, solution $\hat{\theta}$ to empirical plug-in moment equation:

$$M_n(\hat{\theta}, \hat{g}) := \frac{1}{n}\sum_{i \in S_1} m(Z_i; \hat{\theta}, \hat{g}_2) + \frac{1}{n}\sum_{i \in S_2} m(Z_i; \hat{\theta}, \hat{g}_1) = 0$$

- In practice do this with $K \approx 3 \text{ to } 5$ folds: for each fold k train on all other folds and predict on fold k

Natural Algorithm (Draft 3) Gone Right

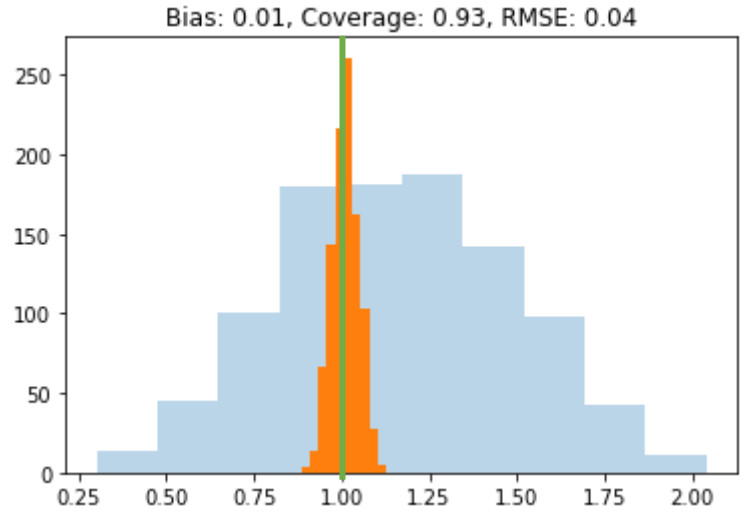
```
def dml2(X, D, y): # orthogonal dml with sample-splitting
    est_y = RandomForestRegressor(min_samples_leaf=20)
    yres = y - cross_val_predict(est_y, X, y, cv=3)
    est_t = RandomForestRegressor(min_samples_leaf=20)
    Dres = D - cross_val_predict(est_t, X, D, cv=3)
    theta = np.mean(yres * Dres) / np.mean(Dres**2)
    var = np.mean((Dres**2) * (yres - theta*Dres)**2) / np.mean(Dres**2)
    stderr = np.sqrt(var / X.shape[0])
    return theta, stderr
```



Natural Algorithm (Draft 3) Gone Right

```
from econml.dml import LinearDML

dml = LinearDML(model_y=RandomForestRegressor(min_samples_leaf=20),
                model_t=RandomForestRegressor(min_samples_leaf=20))
est.fit(y, D, W=X).effect_inference()
```



Stacking and Model Selection

- If we want to choose among many models or perform stacking, we can just use a stacked or automl model in place of each ML model

Stacking ML Models

```
def dml2(X, D, y): # orthogonal dml with sample-splitting
    est_y = StackingRegressor([rf, nnet, gbf, lasso])
    yres = y - cross_val_predict(est_y, X, y, cv=3)
    est_t = StackingRegressor([rf, nnet, gbf, lasso])
    Dres = D - cross_val_predict(est_t, X, D, cv=3)
    theta = np.mean(yres * Dres) / np.mean(Dres**2)
    var = np.mean((Dres**2) * (yres - theta*Dres)**2) / np.mean(Dres**2)
    stderr = np.sqrt(var / X.shape[0])
    return theta, stderr
```

AutoML Models

```
from flaml import AutoML

def dml2(X, D, y): # orthogonal dml with sample-splitting
    est_y = AutoML()
    yres = y - cross_val_predict(est_y, X, y, cv=3)
    est_t = AutoML()
    Dres = D - cross_val_predict(est_t, X, D, cv=3)
    theta = np.mean(yres * Dres) / np.mean(Dres**2)
    var = np.mean((Dres**2) * (yres - theta*Dres)**2) / np.mean(Dres**2)
    stderr = np.sqrt(var / X.shape[0])
    return theta, stderr
```

Stacking and Model Selection

- If we want to choose among many models or perform stacking, we can just use a stacked or automl model in place of each ML model
- Model selection or stacking done many times within each training fold
- Computationally expensive and statistically lossy

- Can we use all the data to at least select among models?

Semi-Cross-fitting Estimation Algorithm

- Split the data in half *(in practice K folds)*
- On first half, estimate $\hat{g}_1^{(1)}, \ldots, \hat{g}_1^{(\mathrm{L})}$ of g_0 and predict on second half
- On second half, estimate $\hat{g}_2^{(1)}, \ldots, \hat{g}_2^{(\mathrm{L})}$ of g_0 and predict on first half
- Choose the model $\ell \in \{1, \ldots, L\}$ that optimizes out-of-sample RMSE
- On all data, solution $\hat{\theta}$ to empirical plug-in moment equation:

$$M_n\left(\hat{\theta}, \hat{g}\right) := \frac{1}{n}\sum_{i \in S_1} m\left(Z_i; \hat{\theta}, \hat{g}_2^{(\ell)}\right) + \frac{1}{n}\sum_{i \in S_2} m\left(Z_i; \hat{\theta}, \hat{g}_1^{(\ell)}\right) = 0$$

Semi-Crossfitting

```
def dml2(X, D, y): # orthogonal dml with semi-crossfitting
    # cross val predict with many models
    est_y = [rf, gbf, lasso]
    yres = np.array([y - cross_val_predict(est, X, y, cv=3) for est in est_y])
    est_d = [rf, gbf, lasso]
    Dres = np.array([D - cross_val_predict(est, X, D, cv=3) for est in est_d])
    # select models with best out of fold performance
    best_y = np.argmin(np.mean(yres**2, axis=1))
    best_d = np.argmin(np.mean(Dres**2, axis=1))
    yres = yres[best_y]
    Dres = Dres[best_d]
    # go with their corresponding residuals
    theta = np.mean(yres * Dres) / np.mean(Dres**2)
    var = np.mean((Dres**2) * (yres - theta*Dres)**2) / np.mean(Dres**2)
    stderr = np.sqrt(var / X.shape[0])
    return theta, stderr
```

Semi-Crossfitting

- If the number of models L is small, then "spillover" is ok and approach still works. For practical purposes L should be thought as constant.
- Under further regularity, provably asymptotic normality holds if

$$\sqrt{\log(L)} = \mathrm{o}\left(\mathrm{n}^{1/4}\right)$$

Semi-Cross-fitting with Stacking

- Split the data in half *(in practice K folds)*
- On first half, estimate $\hat{g}_1^{(1)}, \ldots, \hat{g}_1^{(\mathrm{L})}$ of g_0 and predict on second half
- On second half, estimate $\hat{g}_2^{(1)}, \ldots, \hat{g}_2^{(\mathrm{L})}$ of g_0 and predict on first half
- Construct weights $\alpha_1, \ldots, \alpha_\ell$ on the models using all the data (stacking)

- On all data, solution $\hat{\theta}$ to empirical plug-in moment equation:

$$M_n\left(\hat{\theta}, \hat{g}\right) := \frac{1}{n}\sum_{i \in S_1} m\left(Z_i; \hat{\theta}, \hat{g}_2^{(\ell)}\right) + \frac{1}{n}\sum_{i \in S_2} m\left(Z_i; \hat{\theta}, \hat{g}_1^{(\ell)}\right) = 0$$

Semi-Crossfitting with Stacking

```
def dml2(X, D, y): # orthogonal dml with semi-crossfitting and stacking
    # cross val predict with many models
    est_y = [rf, gbf, lasso]
    ypreds = np.array([cross_val_predict(est, X, y, cv=3) for est in est_y]).T
    est_d = [rf, gbf, lasso]
    Dpreds = np.array([cross_val_predict(est, X, D, cv=3) for est in est_d]).T
    # calculate stacked residuals by finding optimal coefficients
    # and weigthing out-of-sample predictions by these coefficients
    yres = y - LinearRegression().fit(ypreds, y).predict(ypreds)
    Dres = D - LinearRegression().fit(Dpreds, D).predict(Dpreds)
    # go with the stacked residuals
    theta = np.mean(yres * Dres) / np.mean(Dres**2)
    var = np.mean((Dres**2) * (yres - theta*Dres)**2) / np.mean(Dres**2)
    stderr = np.sqrt(var / X.shape[0])
    return theta, stderr
```

Semi-Crossfitting

- If the number of models L is small, then "spillover" is ok and approach still works. For practical purposes L should be thought as constant.
- Under further regularity, provably asymptotic normality holds if

$$\sqrt{L} = \mathrm{o}\left(\mathrm{n}^{1/4}\right)$$

Equivalent view of cross-fitting with stacking (lens of FWL theorem)

- Construct out of fold predictions based on many ML models
- Use these predictions as engineered features X in a simple OLS regression on D, X
- Use the coefficient and standard error of D from this final OLS

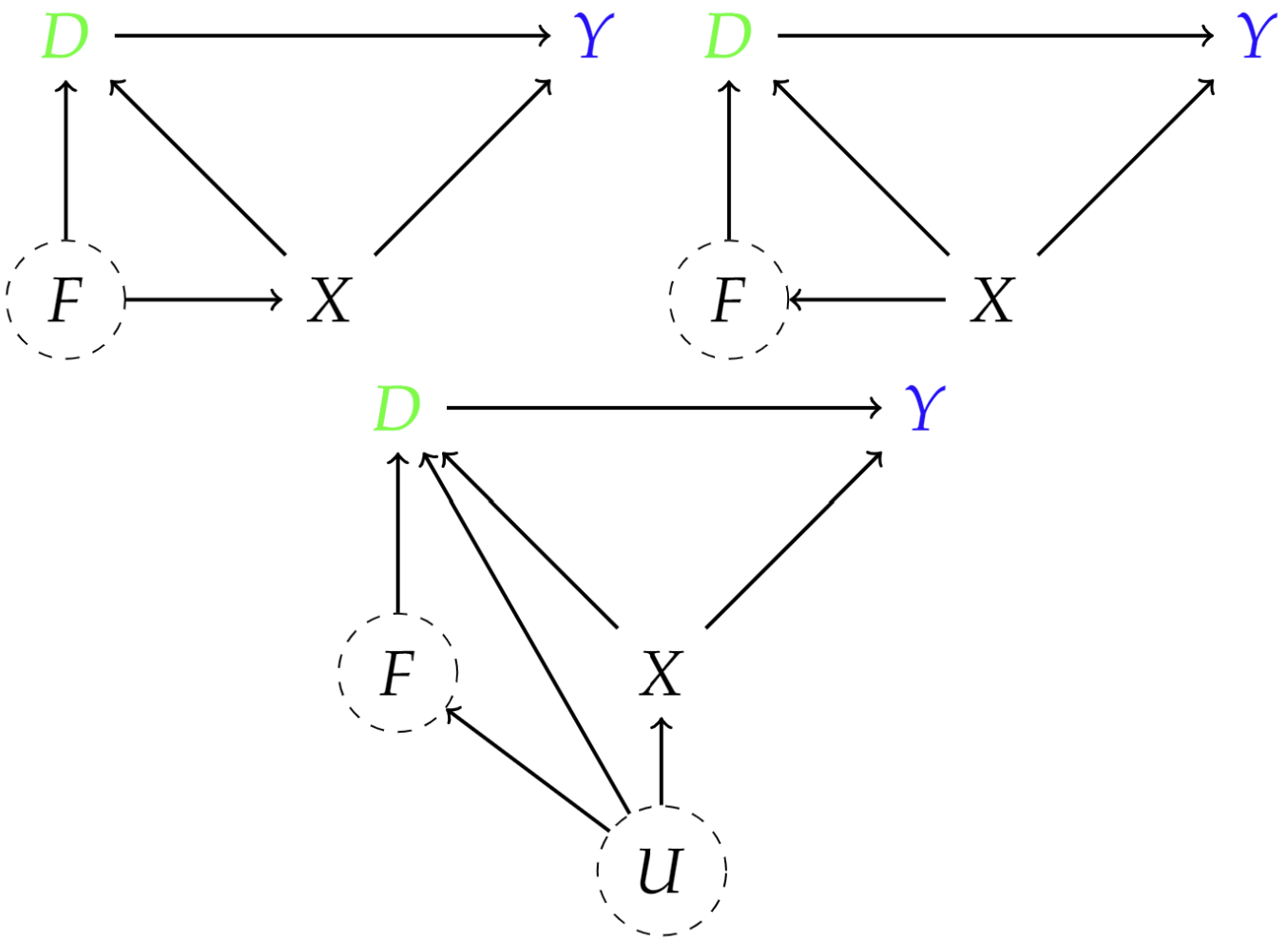
Application: Effect of 401k

Problem

- Impact of 401k eligibility on financial assets
- Working for a firm that offers access to 401k eligibility not randomly assigned
- After conditioning on important characteristics of the worker and job, assignment can be thought as exogenous
- Most important: income
- At least around 1991 (early in 401k deployment), workers would not really base their employment decision on 401k eligibility

Causal Diagrams

- F: firm characteristics
- D: eligibility for 401k
- Y: net financial assets
- X: age, income, family size, years of education, a married indicator, a two-earner status indicator, a defined benefit pension status indicator, an IRA participation indicator, and a home ownership indicator



Let's try it out!

Possible Violations

- F: firm characteristics
- D: eligibility for 401k
- Y: net financial assets
- X: age, income, family size, years of education, a married indicator, a two-earner status indicator, a defined benefit pension status indicator, an IRA participation indicator, and a home ownership indicator
- M: match amount