SOCI 40258

Causal Mediation Analysis

Week 9: Robust Estimation

Outline

- Model misspecification
- Robust/DML estimation of total effects
- Robust/DML estimation of natural effects
- Unobserved confounding
- Sensitivity analysis

The problem of misspecification

- All the parametric approaches to estimation that we've covered thus far hinge on the assumption of no model misspecification
- In practice, however, model misspecification is likely, if not certain, which will lead to bias and inconsistency
- Triangulating results from different models and estimation procedures can help to allay concerns about misspecification, but standard inferential statistics do not properly account for this specification search

Multiply robust estimation

- Most of the estimators we have covered require correct parametric models for the outcome, the mediator(s), and/or the exposure

- Multiply robust estimators only require correct models for a subset of these variables in order to remain consistent
  - When implemented with parametric models, multiply robust estimators provide a degree of protection against misspecification bias
  - When implemented with data-adaptive machine learning (ML) methods, multiply robust estimators provide even greater protection against problems associated with model misspecification

Robust estimation of total effects

- We will begin by reviewing on robust estimation of average total effects in order to establish foundational principles
- In particular, we will focus on constructing a robust estimator for the ATE using its efficient influence function
- The resulting estimator will be *doubly robust*, that is, consistent if either a model for the exposure or a model for the outcome is correctly specified

Efficient Influence Functions (EIFs)

- An efficient influence function provides a mechanism for understanding the best possible performance that an estimator can achieve under certain conditions

- In general, an influence function captures how a single data point affects the overall estimate of a given target parameter (e.g., the ATE)

- An efficient influence function, then, is the influence function corresponding to an estimator that converges the lowest possible asymptotic variance
  - Essentially, it represents the optimal way that an estimator can be influenced by the data

EIF for the ATE

- With a binary treatment $D \in \{0,1\}$ and a set of baseline confounders C, the EIF for the average total effect, $ATE(1,0) = E\big(Y(1) - Y(0)\big)$, can be expressed as follows:

$$EIF_{ATE} = \frac{1}{n}\sum \left[\left[\left[\frac{D}{\pi_1(C)} - \frac{1-D}{1-\pi_1(C)}\right] \times \big(Y - \mu_D(C)\big)\right] + [\mu_1(C) - \mu_0(C)]\right] - ATE(1,0)$$

where $\pi_d(C) = P(D = d|C)$, $\mu_d(C) = E(Y|C, D = d)$, and $Y - \mu_D(C) = Y - E(Y|C, D)$ is a residual term for the outcome

Robust estimation for the ATE

- Setting the EIF_{ATE} equal to zero and solving yields an efficient, doubly-robust (DR) estimator for the ATE:

$$\widehat{ATE}(1,0)^{\text{dr}} = \frac{1}{n}\sum\left[\left[\frac{D}{\hat{\pi}_1(C)} - \frac{1-D}{1-\hat{\pi}_1(C)}\right] \times \left(Y - \hat{\mu}_D(C)\right)\right] + [\hat{\mu}_1(C) - \hat{\mu}_0(C)]$$

where the hats are used to denote estimates of the conditional means and probabilities

Robust estimation for the ATE

- Setting the EIF_{ATE} equal to zero and solving yields an efficient, doubly-robust (DR) estimator for the ATE:

$$\widehat{ATE}(1,0)^{\text{dr}} = \frac{1}{n}\sum \underbrace{\left[\left[\frac{D}{\hat{\pi}_1(C)} - \frac{1-D}{1-\hat{\pi}_1(C)}\right] \times \left(Y - \hat{\mu}_D(C)\right)\right]}_{\text{IPW update based on residual confounding}} + \underbrace{[\hat{\mu}_1(C) - \hat{\mu}_0(C)]}_{\text{regression imputation}}$$

Robust estimation for the ATE with parametric models

- With a DR estimator for the ATE,

$$\widehat{ATE}(1,0)^{\text{dr}} = \frac{1}{n}\sum\left[\left[\frac{D}{\hat{\pi}_1(C)} - \frac{1-D}{1-\hat{\pi}_1(C)}\right] \times \left(Y - \hat{\mu}_D(C)\right)\right] + [\hat{\mu}_1(C) - \hat{\mu}_0(C)],$$

  estimation with parametric models just involves fitting GLMs for $P(D|C)$ and $E(Y|C,D)$ and then using these models to compute all the terms in $\widehat{ATE}(1,0)^{\text{dr}}$

- If either the GLM for $P(D|C)$ or the GLM for $E(Y|C,D)$ is correctly specified, this estimator will be consistent and asymptotically normal, providing a degree of protection against model misspecification

Robust estimation for the ATE with parametric models

- The DR estimator can be implemented with parametric models through the following series of steps:

1. Fit a GLM for $P(D|C)$
   - Use the fitted model to compute $\hat{\pi}_1(C)$

2. Fit a GLM for $E(Y|C,D)$
   - Use the fitted model to compute $\hat{\mu}_1(C)$, $\hat{\mu}_0(C)$, and $Y - \hat{\mu}_D(C)$

3. Plug these terms into $\widehat{ATE}(1,0)^{\mathrm{dr}}$ for each sample member and solve

Robust estimation for the ATE with ML models

- With a DR estimator for the ATE,

$$\widehat{ATE}(1,0)^{\text{dr}} = \frac{1}{n}\sum\left[\left[\frac{D}{\hat{\pi}_1(C)} - \frac{1-D}{1-\hat{\pi}_1(C)}\right] \times \left(Y - \hat{\mu}_D(C)\right)\right] + [\hat{\mu}_1(C) - \hat{\mu}_0(C)],$$

  estimation via ML just involves training ML models for $P(D|C)$ and $E(Y|C,D)$, and then using these models to compute all the terms in $\widehat{ATE}(1,0)^{\text{dr}}$

- If all the ML models possess sufficiently fast rates of convergence, then the DR estimator will be consistent, efficient, and asymptotically normal, providing additional protection against misspecification
  - Generally, each of the ML models must be at least $n^{1/4}$-consistent (i.e., they must converge to their target estimand at a $n^{1/4}$ rate)

Robust estimation for the ATE with ML models

- The DR estimator can be implemented with ML models through the following series of steps:

1. Train a ML model for $P(D|C)$
   - Use the trained model to compute $\hat{\pi}_1(C)$
2. Train a ML model for $h(Y|C,D)$
   - Use the trained model to compute $\hat{\mu}_1(C)$, $\hat{\mu}_0(C)$, and $Y - \hat{\mu}_D(C)$
3. Plug these terms into $\widehat{ATE}(1,0)^{\mathrm{dr}}$ for each sample member and solve

Robust estimation for the ATE with ML models

- Candidate ML models for $P(D|C)$ and $E(Y|C, D)$ might include:
  - Regularized GLMs (LASSO, ridge, elasticnet)
  - Classification and regression trees (CARTs)
  - Random forests (RFs)
  - Gradient boosted trees (GBTs)
  - Artificial neural networks (ANNs)
  - Super learners (SLs)

Sample splitting

- With ML models, using the same data for training the models and for estimating the ATE can sometimes lead to bias and complicates standard approaches to statistical inference

- To avoid these problems, we can pair DR estimation with a sample splitting algorithm

- The sample splitting algorithm proceeds as follows:

    1. Split the original sample into two separate subsamples

    2. Train the ML models for $P(D|C)$ and $E(Y|C,D)$ using the first subsample

    3. Apply the trained models to the second subsample to compute the $\widehat{ATE}(1,0)^{\mathrm{dr}}$

Repeated cross-fitting

- Sample splitting is inefficient, since each subset of the data is only used for training or effect estimation, but not both
- Cross-fitting resolves this problem by iterating the sample splitting procedure:
  1. Split the original sample into $4 \leq J \leq 10$ equally sized subsamples
  2. Train the ML models for $P(D|C)$ and $E(Y|C, D)$ using $J-1$ of the subsamples
  3. Apply the trained models to the remaining subsample j to compute $\widehat{ATE}(1,0)_j^{\text{dr}}$
  4. Iterate the previous two steps until each subsample is used for estimation a single time
  5. Compute $\widehat{ATE}(1,0)^{\text{dr}} = \frac{1}{J}\sum_{j=1}^{J} \widehat{ATE}(1,0)_j^{\text{dr}}$ by averaging estimates across the subsamples

Wald tests and CIs

- When computed using cross-fitting, the $\widehat{ATE}(1,0)^{\mathrm{dr}}$ is asymptotically normally distributed with a variance equal to $Var(EIF_{ATE})/n$

- This allows the use of standard inferential statistics, including…
  - Wald tests, based on $W = \frac{\widehat{ATE}(1,0)^{\mathrm{dr}} - ATE(1,0)_{H_0}}{\sqrt{\widehat{Var}(EIF_{ATE})/n}} \sim N(0,1)$ under H_0
  - Wald confidence intervals, given by $\widehat{ATE}(1,0)^{\mathrm{dr}} \pm Z_\alpha \times \sqrt{\widehat{Var}(EIF_{ATE})/n}$ where $Z \sim N(0,1)$

- The nonparametric bootstrap can also be used to test null hypotheses and construct confidence intervals for cross-fit ML estimators, but it is more computationally intensive

Example: NLSY79

- Does attending college reduce depression later in adulthood?
- 1979 National Longitudinal Study of Youth
  - Exposure (D)
    - sample member attended college before age 22
  - Outcome (Y):
    - standardized scores on the CES-D at age 40
  - Covariates (C):
    - Race and gender
    - Parental education, occupation, and income
    - Household size
    - AFQT scores in high school

Example: NLSY79

- Compute doubly robust estimates for the *ATE* using parametric models

```
### wk 9 nlsy tutorial ###
rm(list=ls())

# load/install libraries #
packages<-c("dplyr", "tidyr", "foreign", "margins", "survey", "ranger", "xgboost", "SuperLearner")
#install.packages(packages)

for (package.i in packages) {
    suppressPackageStartupMessages(library(package.i, character.only=TRUE))
    }

# load data #
datadir <- "C:/Users/Geoffrey Wodtke/Dropbox/D/courses/2023-24_UOFCHICAGO/SOCI_40258_CAUSAL_MEDIAT
nlsy <- read.dta(paste(datadir, "nlsy79.dta", sep=""))

nlsy <- nlsy[complete.cases(nlsy[, c("cesd_age40", "ever_unemp_age3539", "att22",
    "female", "black", "hispan", "paredu", "parprof", "parinc_prank", "famsize", "afqt3")]),]

nlsy$std_cesd_age40 <- (nlsy$cesd_age40-mean(nlsy$cesd_age40))/sd(nlsy$cesd_age40)

# parametric doubly robust estimate for ATE #
Ymodel.lm <- lm(std_cesd_age40 ~ att22*(female + black + hispan + paredu + parprof +
    parinc_prank*parinc_prank + famsize + afqt3), data=nlsy)

Dmodel.pr <- glm(att22 ~ female + black + hispan + paredu + parprof +
    parinc_prank + famsize + afqt3, data=nlsy, family=binomial(link="probit"))

```

Example: NLSY79

- Compute doubly robust estimates for the ATE using parametric models

```
gdata <- nlsy

gdata$att22 <- 1
yhat1 <- predict(Ymodel.lm, newdata=gdata)

gdata$att22 <- 0
yhat0 <- predict(Ymodel.lm, newdata=gdata)

resid <- Ymodel.lm$residuals

phatD1 <- predict(Dmodel.pr, type = "response")

ipw <- (nlsy$att22*mean(nlsy$att22)/phatD1) - (1-nlsy$att22)*(1-mean(nlsy$att22))/(1-phatD1)

eif <- ipw*resid + (yhat1-yhat0)
```

Example: NLSY79

- Compute doubly robust estimates for the *ATE* using parametric models

```r
ATEhat <- mean(eif)

VarATEhat <- var(eif)/length(ipw)

CI95pct <- c(ATEhat-1.96*sqrt(VarATEhat), ATEhat+1.96*sqrt(VarATEhat))

p <- (1-pnorm(abs(ATEhat/sqrt(VarATEhat)),0,1))*2

parResults <- data.frame(
    point.est=round(ATEhat, digits=3),
    se=round(sqrt(VarATEhat), digits=3),
    ll.95ci=round(CI95pct[1], digits=3),
    ul.95ci=round(CI95pct[2], digits=3),
    pval=round(p, digits=3))
rownames(parResults) <- c('ATE')

print(parResults)
```

```
> print(parResults)
    point.est    se ll.95ci ul.95ci pval
ATE     -0.15 0.022  -0.194  -0.107    0
>
```

Example: NLSY79

- Compute DML estimates for the *ATE* using super learners

```
# semi-parametric doubly robust estimate for ATE w/ cross-fitting #
set.seed(3308004)

nlsy$k <- runif(nrow(nlsy),0,1)
nlsy <- nlsy[order(nlsy$k),]
nlsy$k <- rep(1:5, length.out=nrow(nlsy))

cntrl.sl <- SuperLearner.CV.control(V=10)

confounders <- c("female", "black", "hispan", "paredu",
    "parprof", "parinc_prank", "famsize", "afqt3")
```

Example: NLSY79

- Compute DML estimates for the *ATE* using super learners

```
for (j in 1:5) {

    df.train <- nlsy[which(nlsy$k!=j),]
    df.est <- nlsy[which(nlsy$k==j),]

    Ymodel.sl <- SuperLearner(
        Y=df.train[,"std_cesd_age40"],
        X=df.train[,c(confounders, "att22")],
        SL.library=c("SL.lm", "SL.ranger", "SL.xgboost"),
        cvControl=cntrl.sl)

    Dmodel.sl <- SuperLearner(
        Y=df.train[,"att22"],
        X=df.train[,confounders],
        family = binomial(),
        SL.library=c("SL.glm", "SL.ranger", "SL.xgboost"),
        cvControl=cntrl.sl)
```

Example: NLSY79

- Compute DML estimates for the ATE using super learners

```
    gdata <- df.est[,c(confounders, "att22")]

    gdata$att22 <- 1
    nlsy$yhat1[nlsy$k==j] <- predict(Ymodel.sl, newdata=gdata)$pred

    gdata$att22 <- 0
    nlsy$yhat0[nlsy$k==j] <- predict(Ymodel.sl, newdata=gdata)$pred

    nlsy$resid[nlsy$k==j] <- df.est$std_cesd_age40 -
        predict(Ymodel.sl, df.est[,c(confounders, "att22")])$pred

    phatD1 <- predict(Dmodel.sl, newdata=df.est[,confounders], type="prob")$pred

    nlsy$ipw[nlsy$k==j] <- (nlsy$att22[nlsy$k==j]/phatD1)
        - (1-nlsy$att22[nlsy$k==j])/(1-phatD1)
}
```

Example: NLSY79

- Compute DML estimates for the *ATE* using super learners

```r
eif <- nlsy$ipw*nlsy$resid + (nlsy$yhat1-nlsy$yhat0)

ATEhat <- mean(eif)

VarATEhat <- var(eif)/length(nlsy$ipw)

CI95pct <- c(ATEhat-1.96*sqrt(VarATEhat), ATEhat+1.96*sqrt(VarATEhat))

p <- (1-pnorm(abs(ATEhat/sqrt(VarATEhat)),0,1))*2

sparResults <- data.frame(
    point.est=round(ATEhat, digits=3),
    se=round(sqrt(VarATEhat), digits=3),
    ll.95ci=round(CI95pct[1], digits=3),
    ul.95ci=round(CI95pct[2], digits=3),
    pval=round(p, digits=3))
rownames(sparResults) <- c('ATE')

print(sparResults)
```

```
> print(sparResults)
    point.est    se ll.95ci ul.95ci  pval
ATE    -0.117 0.046  -0.208  -0.026 0.012
```

Robust estimation for natural effects

- Next, we will focus on multiply robust estimation of natural direct and indirect effects based on their efficient influence functions…

Robust estimation for natural effects

- Define the marginal means of the nested potential outcomes as follows:

$$\psi(d_1, d_2) = E\left(Y\big(d_1, M(d_2)\big)\right),$$

where d_1 and d_2 just denote two generic values of the exposure

- With this notation, then, we can define our target estimands as follows:

$$NDE(d, d^*) = \psi(d, d^*) - \psi(d^*, d^*) = E\left(Y\big(d, M(d^*)\big)\right) - E\left(Y\big(d^*, M(d^*)\big)\right)$$

$$NIE(d, d^*) = \psi(d, d) - \psi(d, d^*) = E\left(Y\big(d, M(d)\big)\right) - E\left(Y\big(d, M(d^*)\big)\right)$$

The EIF for ψ_{d_1,d_2}

- With a binary treatment $D \in \{0,1\}$ and set of baseline confounders C, the EIF for the marginal mean of the nested potential outcomes can be expressed as:

$$EIF_{\psi(d_1,d_2)} = \frac{1}{n}\sum\left[\left(\frac{I(D=d_1)}{\pi_{d_2}(C)}\right)\left(\frac{\pi_{d_2}(C,M)}{\pi_{d_1}(C,M)}\right)\left(Y - \mu_{d_1}(C,M)\right)\right] +$$

$$\left[\left(\frac{I(D=d_1)}{\pi_{d_2}(C)}\right)\left(\frac{\pi_{d_2}(C,M)}{\pi_{d_1}(C,M)}\right)\left(\mu_{d_1}(C,M) - \nu_{d_2}(C)\right)\right] +$$

$$\nu_{d_2}(C) - \psi(d_1,d_2)$$

where $\pi_{d_2}(C) = P(D = d_2|C)$, $\pi_{d_2}(C,M) = P(D = d_2|C,M)$ and $\pi_{d_1}(C,M)$ is defined analogously, $\mu_{d_1}(C,M) = E(Y|C,D = d_1,M)$, and $\nu_{d_2}(C) = E(E(Y|C,D = d_1,M)|C,D = d_2)$

Robust estimation for natural effects

- Setting the $EIF_{\psi_{d_1,d_2}}$ equal to zero and solving yields an efficient, multiply robust (MR) estimator for the mean of the nested potential outcomes:

$$\hat{\psi}(d_1, d_2)^{\mathrm{mr}} = \frac{1}{n}\sum\left[\left(\frac{I(D=d_1)}{\hat{\pi}_{d_2}(C)}\right)\left(\frac{\hat{\pi}_{d_2}(C,M)}{\hat{\pi}_{d_1}(C,M)}\right)\left(Y - \hat{\mu}_{d_1}(C,M)\right)\right] +$$

$$\left[\left(\frac{I(D=d_1)}{\hat{\pi}_{d_2}(C)}\right)\left(\frac{\hat{\pi}_{d_2}(C,M)}{\hat{\pi}_{d_1}(C,M)}\right)\left(\hat{\mu}_{d_1}(C,M) - \hat{v}_{d_2}(C)\right)\right] +$$

$$\hat{v}_{d_2}(C)$$

where the hats are used to denote estimates of the conditional means and probabilities

Robust estimation for natural effects

- Comparing different estimated marginal means for the nested potential outcomes yields MR estimators for the natural direct and indirect effects of interest:

$$\widehat{NDE}(1,0)^{\mathrm{mr}} = \hat{\psi}(1,0)^{\mathrm{mr}} - \hat{\psi}(0,0)^{\mathrm{mr}}$$

$$\widehat{NDE}(1,0)^{\mathrm{mr}} = \hat{\psi}(1,1)^{\mathrm{mr}} - \hat{\psi}(1,0)^{\mathrm{mr}}$$

Robust estimation for natural effects with parametric models

- With the MR estimator for $\psi(d_1, d_2)$, estimation with parametric models involves fitting GLMs for $P(D|C)$, $P(D|C,M)$, $E(Y|C,D,M)$, and $E(E(Y|C,D=d_1,M)|C,D)$
  - Then, these models are used to compute all the terms in $\hat{\psi}(d_1, d_2)^{\text{mr}}$
- If the models for (i) $P(D|C)$ and $P(D|C,M)$, (ii) for $P(D|C)$ and $E(Y|C,D,M)$, or (iii) for $E(Y|C,D,M)$ and $E(E(Y|C,D=d_1,M)|C,D)$ are correctly specified, this estimator will be consistent, efficient, and asymptotically normal
  - It is sometimes said to be "triply robust" because there are three distinct pathways to consistency

Robust estimation for natural effects with parametric models

- The MR estimator can be implemented with parametric models through the following series of steps:

  1. Fit a GLM for $P(D|C)$
     - Use the fitted model to compute $\hat{\pi}_{d_2}(C)$
  2. Fit a GLM for $P(D|C,M)$
     - Use the fitted model to compute $\hat{\pi}_{d_1}(C,M)$ and $\hat{\pi}_{d_2}(C,M)$
  3. Fit a GLM for $E(Y|C,D,M)$
     - Use the fitted model to compute $\hat{\mu}_{d_1}(C,M)$
  4. Fit a GLM for $E(E(Y|C,D=d_1,M)|C,D)$
     - Use the fitted model to compute $\hat{\nu}_{d_2}(C)$
  5. Plug these terms into $\hat{\psi}(1,1)^{\mathrm{mr}}$, $\hat{\psi}(0,0)^{\mathrm{mr}}$, and $\hat{\psi}(1,0)^{\mathrm{mr}}$ and solve

Robust estimation for natural effects with ML models

- With the MR estimator for $\psi(d_1, d_2)$, estimation via ML just involves training ML models for $P(D|C)$, $P(D|C, M)$, $E(Y|C, D, M)$, and $E(E(Y|C, D = d_1, M)|C, D)$, and then using these models to compute $\hat{\psi}(d_1, d_2)^{\text{mr}}$
- If all the ML models possess sufficiently fast rates of convergence—generally, meaning they are at least $n^{1/4}$-consistent—then the MR estimator will be consistent, efficient, and asymptotically normal

Robust estimation for natural effects with ML models

- The MR estimator can be implemented with ML models through the following series of steps:

1. Train a ML model for $P(D|C)$
   - Use the trained model to compute $\hat{\pi}_{d_2}(C)$

2. Train a ML model for $P(D|C, M)$
   - Use the trained model to compute $\hat{\pi}_{d_1}(C, M)$ and $\hat{\pi}_{d_2}(C, M)$

3. Train a ML model for $E(Y|C, D, M)$
   - Use the trained model to compute $\hat{\mu}_{d_1}(C, M)$

4. Train a ML model for $E(E(Y|C, D = d_1, M)|C, D)$
   - Use the trained model to compute $\hat{v}_{d_2}(C)$

5. Plug these terms into $\hat{\psi}(1,1)^{\mathrm{mr}}$, $\hat{\psi}(0,0)^{\mathrm{mr}}$, and $\hat{\psi}(1,0)^{\mathrm{mr}}$ and solve

Repeated cross-fitting

- With ML models, using the same data for model training and for effect estimation can lead to bias and complicates standard approaches to statistical inference

- To avoid these problems, we can pair the estimation approach outlined previously with the cross-fitting algorithm:

    1. Split the original sample into $4 \leq J \leq 10$ equally sized subsamples
    2. Train the ML models using $J - 1$ of the subsamples
    3. Apply the trained models to the remaining subsample J to compute $\hat{\psi}(d_1, d_2)_j^{\mathrm{mr}}$
    4. Iterate the previous two steps until each subsample is used for estimation a single time
    5. Compute $\hat{\psi}(d_1, d_2)^{\mathrm{mr}} = \frac{1}{J}\sum_{j=1}^{J} \hat{\psi}(d_1, d_2)_j^{\mathrm{mr}}$ by averaging across the J subsamples

Wald tests and CIs for natural effects

- When computed using cross-fitting, the $\widehat{NDE}(1,0)^{\mathrm{mr}}$ is asymptotically normally distributed with a variance equal to $Var\left(EIF_{\psi(1,0)} - EIF_{\psi(0,0)}\right)/n$

- Similarly, the $\widehat{NIE}(1,0)^{\mathrm{mr}}$ is also asymptotically normal with a variance equal to $Var\left(EIF_{\psi(1,1)} - EIF_{\psi(1,0)}\right)/n$

- This suggests the use of standard inferential statistics, including Wald tests and confidence intervals, with the standard errors for $\widehat{NDE}(1,0)^{\mathrm{mr}}$ and $\widehat{NIE}(1,0)^{\mathrm{mr}}$ given by:

$$se\left(\widehat{NDE}(1,0)^{\mathrm{mr}}\right) = \sqrt{\widehat{Var}\left(EIF_{\psi(1,0)} - EIF_{\psi(0,0)}\right)/n}$$

$$se\left(\widehat{NIE}(1,0)^{\mathrm{mr}}\right) = \sqrt{\widehat{Var}\left(EIF_{\psi(1,1)} - EIF_{\psi(1,0)}\right)/n}$$

Example: NLSY79

- 1979 National Longitudinal Study of Youth
  - Exposure (D)
    - sample member attended college before age 22
  - Outcome (Y)
    - standardized scores on the CES-D at age 40
  - Covariates (C)
    - Race, gender, parental education, occupation, and income, household size, AFQT scores
  - A potential mediator (M)
    - unemployment between age 35-40

Example: NLSY79

- Many studies have documented that going to college seems to reduce the likelihood of becoming depressed later in life—but how does this effect come about?
- One possibility is that a more advanced education reduces depression by protecting its recipients from financially strenuous and mentally taxing spells of unemployment
- Does unemployment mediate the effect of college attendance on depression?

Example: NLSY79

- Compute triply robust estimates of the NDE and NIE using parametric models

```
# parametric multiply robust estimate for NDE and NIE #
Ymodel.lm <- lm(std_cesd_age40 ~ (att22 * ever_unemp_age3539) * (female + black + hispan +
    paredu + parprof + parinc_prank*parinc_prank + famsize + afqt3), data=nlsy)

Dmodel_1.pr <- glm(att22 ~ female + black + hispan + paredu + parprof +
    parinc_prank + famsize + afqt3, data=nlsy, family=binomial(link="probit"))

Dmodel_2.pr <- glm(att22 ~ ever_unemp_age3539 + female + black + hispan + paredu + parprof +
    parinc_prank + famsize + afqt3, data=nlsy, family=binomial(link="probit"))

phat_d1_C <- predict(Dmodel_1.pr, type = "response")
phat_d0_C <- 1-phat_d1_C

phat_d1_CM <- predict(Dmodel_2.pr, type = "response")
phat_d0_CM <- 1-phat_d1_CM

phat_d1 <- mean(nlsy$att22)
phat_d0 <- 1-phat_d1

f1 <- nlsy$att22*phat_d1 / phat_d1_C
f2 <- (1-nlsy$att22)*phat_d0 / phat_d0_C
f3 <- nlsy$att22*phat_d0 / phat_d0_C

s1 <- phat_d0_CM / phat_d1_CM

resid <- Ymodel.lm$residuals
```

Example: NLSY79

- Compute triply robust estimates of the NDE and NIE using parametric models

```
gdata <- nlsy

gdata$att22 <- 1
qhat1 <- predict(Ymodel.lm, newdata=gdata)

gdata$att22 <- 0
qhat0 <- predict(Ymodel.lm, newdata=gdata)

qhat0_CD <- lm(qhat0 ~ att22 * (female + black + hispan +
        paredu + parprof + parinc_prank + famsize + afqt3), data=nlsy)

gdata$att22 <- 0
EMgivC0qhat0 <- predict(qhat0_CD, newdata=gdata)

qhat1_CD <- lm(qhat1 ~ att22 * (female + black + hispan +
        paredu + parprof + parinc_prank + famsize + afqt3), data=nlsy)

gdata$att22 <- 1
EMgivC1qhat1 <- predict(qhat1_CD, newdata=gdata)

gdata$att22 <- 0
EMgivC0qhat1 <- predict(qhat1_CD, newdata=gdata)

eif_psi_11 <- f1*1*resid + f1*(qhat1 - EMgivC1qhat1) + EMgivC1qhat1
eif_psi_00 <- f2*1*resid + f2*(qhat0 - EMgivC0qhat0) + EMgivC0qhat0
eif_psi_10 <- f3*s1*resid + f3*s1*(qhat1 - EMgivC0qhat1) + EMgivC0qhat1
```

Example: NLSY79

- Compute triply robust estimates of the NDE and NIE using parametric models

```
NDEhat <- mean(eif_psi_10 - eif_psi_00)
NIEhat <- mean(eif_psi_11 - eif_psi_10)
ATEhat <- mean(eif_psi_11 - eif_psi_00)

VarNDEhat <- var(eif_psi_10 - eif_psi_00)/length(resid)
VarNIEhat <- var(eif_psi_11 - eif_psi_10)/length(resid)
VarATEhat <- var(eif_psi_11 - eif_psi_00)/length(resid)

NDE95pctCI <- c(NDEhat-1.96*sqrt(VarNDEhat), NDEhat+1.96*sqrt(VarNDEhat))
NIE95pctCI <- c(NIEhat-1.96*sqrt(VarNIEhat), NIEhat+1.96*sqrt(VarNIEhat))
ATE95pctCI <- c(ATEhat-1.96*sqrt(VarATEhat), ATEhat+1.96*sqrt(VarATEhat))

ATEp <- (1-pnorm(abs(ATEhat/sqrt(VarATEhat)),0,1))*2
NDEp <- (1-pnorm(abs(NDEhat/sqrt(VarNDEhat)),0,1))*2
NIEp <- (1-pnorm(abs(NIEhat/sqrt(VarNIEhat)),0,1))*2

parMedResults <- data.frame(
    point.est = round(c(ATEhat, NDEhat, NIEhat), digits = 3),
    se = round(c(sqrt(VarATEhat), sqrt(VarNDEhat), sqrt(VarNIEhat)), digits = 3),
    ll.95ci = round(c(ATE95pctCI[1], NDE95pctCI[1], NIE95pctCI[1]), digits = 3),
    ul.95ci = round(c(ATE95pctCI[2], NDE95pctCI[2], NIE95pctCI[2]), digits = 3),
    pval = round(c(ATEp, NDEp, NIEp), digits = 3))

rownames(parMedResults) <- c('ATE', 'NDE', 'NIE')

print(parMedResults)
```

Example: NLSY79

- Compute triply robust estimates of the NDE and NIE using parametric models

```
> print(parMedResults)
    point.est    se ll.95ci ul.95ci pval
ATE    -0.150 0.022  -0.194  -0.107 0.00
NDE    -0.152 0.042  -0.234  -0.069 0.00
NIE     0.001 0.024  -0.045   0.048 0.95
> |
```

Example: NLSY79

- Compute DML estimates of the *NDE* and *NIE* using super learners

```
# semi-parametric multiply robust estimate for NDE and NIE w/ cross-fitting #
set.seed(3308004)

nlsy$k <- runif(nrow(nlsy),0,1)
nlsy <- nlsy[order(nlsy$k),]
nlsy$k <- rep(1:5, length.out=nrow(nlsy))

cntrl.sl <- SuperLearner.CV.control(V=10)

confounders <- c("female", "black", "hispan", "paredu",
    "parprof", "parinc_prank", "famsize", "afqt3")

for (j in 1:5) {

    df.train <- nlsy[which(nlsy$k!=j),]
    df.est <- nlsy[which(nlsy$k==j),]

    Ymodel.sl <- SuperLearner(
        Y=df.train[,"std_cesd_age40"],
            X=df.train[,c(confounders, "att22", "ever_unemp_age3539")],
            SL.library=c("SL.lm", "SL.ranger", "SL.xgboost"),
            cvControl=cntrl.sl)

    Dmodel_1.sl <- SuperLearner(
        Y=df.train[,"att22"],
            X=df.train[,confounders],
        family = binomial(),
            SL.library=c("SL.glm", "SL.ranger", "SL.xgboost"),
            cvControl=cntrl.sl)

    Dmodel_2.sl <- SuperLearner(
        Y=df.train[,"att22"],
            X=df.train[,c(confounders, "ever_unemp_age3539")],
        family = binomial(),
            SL.library=c("SL.glm", "SL.ranger", "SL.xgboost"),
            cvControl=cntrl.sl)
```

Example: NLSY79

- Compute DML estimates of the NDE and NIE using super learners

```r
phat_d1_C <- predict(Dmodel_1.sl,
    newdata=df.est[,confounders], type="prob")$pred
phat_d0_C <- 1-phat_d1_C

phat_d1_CM <- predict(Dmodel_2.sl,
    newdata=df.est[,c(confounders, "ever_unemp_age3539")], type="prob")$pred
phat_d0_CM <- 1-phat_d1_CM

phat_d1 <- mean(df.est$att22)
phat_d0 <- 1-phat_d1

nlsy$f1[nlsy$k==j] <- nlsy$att22[nlsy$k==j]*phat_d1 / phat_d1_C
nlsy$f2[nlsy$k==j] <- (1-nlsy$att22[nlsy$k==j])*phat_d0 / phat_d0_C
nlsy$f3[nlsy$k==j] <- nlsy$att22[nlsy$k==j]*phat_d0 / phat_d0_C

nlsy$s1[nlsy$k==j] <- phat_d0_CM / phat_d1_CM

nlsy$resid[nlsy$k==j] <- df.est$std_cesd_age40 -
    predict(Ymodel.sl, df.est[,c(confounders, "att22", "ever_unemp_age3539")])$pred

gdata <- df.est[,c(confounders, "att22", "ever_unemp_age3539")]

gdata$att22 <- 1
nlsy$qhat1[nlsy$k==j] <- predict(Ymodel.sl, newdata=gdata)$pred

gdata$att22 <- 0
nlsy$qhat0[nlsy$k==j] <- predict(Ymodel.sl, newdata=gdata)$pred
```

Example: NLSY79

- Compute DML estimates of the NDE and NIE using super learners

```
    qhat0_CD <- lm(qhat0 ~ att22 * (female + black + hispan +
        paredu + parprof + parinc_prank + famsize + afqt3),
        data=nlsy[which(nlsy$k==j),])

    gdata$att22 <- 0
    nlsy$EMgivC0qhat0[nlsy$k==j] <- predict(qhat0_CD, newdata=gdata)

    qhat1_CD <- lm(qhat1 ~ att22 * (female + black + hispan +
        paredu + parprof + parinc_prank + famsize + afqt3),
        data=nlsy[which(nlsy$k==j),])

    gdata$att22 <- 1
    nlsy$EMgivC1qhat1[nlsy$k==j] <- predict(qhat1_CD, newdata=gdata)

    gdata$att22 <- 0
    nlsy$EMgivC0qhat1[nlsy$k==j] <- predict(qhat1_CD, newdata=gdata)
    }

eif_psi_11 <- nlsy$f1*1*nlsy$resid +
    nlsy$f1*(nlsy$qhat1 - nlsy$EMgivC1qhat1) +
    nlsy$EMgivC1qhat1

eif_psi_00 <- nlsy$f2*1*nlsy$resid +
    nlsy$f2*(nlsy$qhat0 - nlsy$EMgivC0qhat0) +
    nlsy$EMgivC0qhat0

eif_psi_10 <- nlsy$f3*nlsy$s1*nlsy$resid +
    nlsy$f3*nlsy$s1*(nlsy$qhat1 - nlsy$EMgivC0qhat1) +
    nlsy$EMgivC0qhat1
```

Example: NLSY79

- Compute DML estimates of the NDE and NIE using super learners

```
NDEhat <- mean(eif_psi_10 - eif_psi_00)
NIEhat <- mean(eif_psi_11 - eif_psi_10)
ATEhat <- mean(eif_psi_11 - eif_psi_00)

VarNDEhat <- var(eif_psi_10 - eif_psi_00)/length(resid)
VarNIEhat <- var(eif_psi_11 - eif_psi_10)/length(resid)
VarATEhat <- var(eif_psi_11 - eif_psi_00)/length(resid)

NDE95pctCI <- c(NDEhat-1.96*sqrt(VarNDEhat), NDEhat+1.96*sqrt(VarNDEhat))
NIE95pctCI <- c(NIEhat-1.96*sqrt(VarNIEhat), NIEhat+1.96*sqrt(VarNIEhat))
ATE95pctCI <- c(ATEhat-1.96*sqrt(VarATEhat), ATEhat+1.96*sqrt(VarATEhat))

ATEp <- (1-pnorm(abs(ATEhat/sqrt(VarATEhat)),0,1))*2
NDEp <- (1-pnorm(abs(NDEhat/sqrt(VarNDEhat)),0,1))*2
NIEp <- (1-pnorm(abs(NIEhat/sqrt(VarNIEhat)),0,1))*2

sparMedResults <- data.frame(
    point.est = round(c(ATEhat, NDEhat, NIEhat), digits = 3),
    se = round(c(sqrt(VarATEhat), sqrt(VarNDEhat), sqrt(VarNIEhat)), digits = 3),
    ll.95ci = round(c(ATE95pctCI[1], NDE95pctCI[1], NIE95pctCI[1]), digits = 3),
    ul.95ci = round(c(ATE95pctCI[2], NDE95pctCI[2], NIE95pctCI[2]), digits = 3),
    pval = round(c(ATEp, NDEp, NIEp), digits = 3))

rownames(sparMedResults) <- c('ATE', 'NDE', 'NIE')

print(sparMedResults)
```

Example: NLSY79

- Compute DML estimates of the NDE and NIE using super learners

```
> print(sparMedResults)
    point.est    se ll.95ci ul.95ci  pval
ATE    -0.107 0.021  -0.147  -0.066 0.000
NDE    -0.115 0.038  -0.190  -0.039 0.003
NIE     0.008 0.022  -0.035   0.051 0.709
> |
```

Limitations

- MR/DML estimation is best suited for applications with a binary exposure
  - With an exposure that is continuous or otherwise has many values, MR estimation becomes unstable and practically difficult to implement

- MR estimation can also perform poorly when *none* of the component models are correctly specified, and sometimes, the misspecification bias can be even worse than with fully parametric approaches

- When implemented with parametric models, it is safer to compute inferential statistics using the bootstrap

Extensions

- Similar methods can also be used to construct MR/DML estimates for…
  - multivariate natural and path-specific effects (Miles et al. 2021, Zhou 2022)
  - interventional direct and indirect effects (Diaz et al. 2021, Rudolph et al. 2017)
- Targeted maximum likelihood (TMLE; Zheng and van der Laan 2012)
  - An alternative, two-step approach to MR estimation
- causal-Graphical Normalizing Flows (cGNFs; Balgi et al. 2024)
  - Learn entire causal systems represented as DAGs using highly flexible neural networks

The problem of confounding

- Another important concern in analyses of causal mediation is the problem of unobserved confounding
- If there are unobserved confounders for the exposure-outcome, mediator-outcome, or exposure-mediator relationships, then all the estimators that we have discussed previously are biased and inconsistent
- Confounding bias poses a significant challenge in analyses of causal mediation because it is very difficult to conduct experiments that ensure the relationships among key variables are unconfounded by design

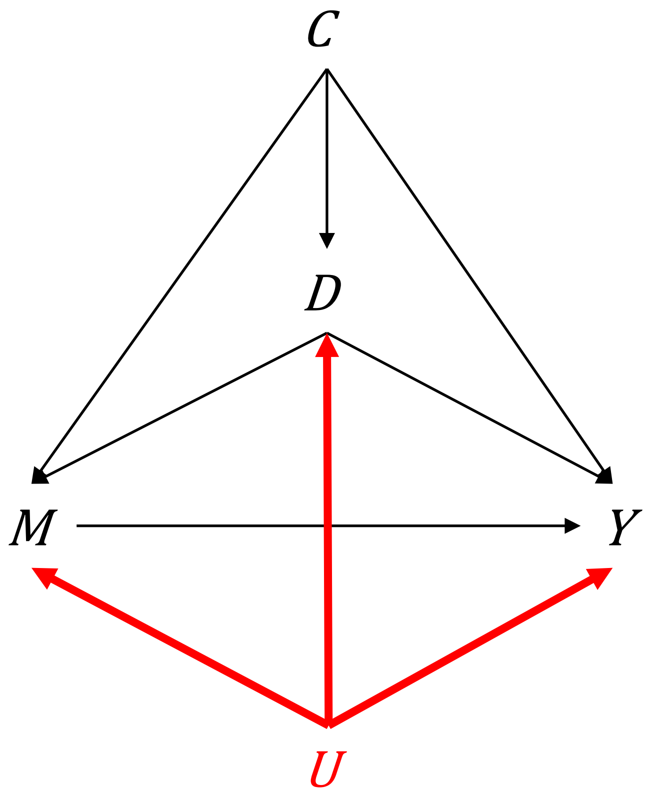
Sensitivity analysis

- When confounding cannot be controlled by experimental design, formal sensitivity analyses are useful for assessing potential biases and their impact on our inferences
- A sensitivity analysis involves postulating different, hypothetical patterns of unobserved confounding and then exploring how the resulting bias may alter our conclusions about causal mediation

Sensitivity analysis

- Consider a scenario in which an unobserved variable, denoted by U, affects the exposure D, the mediator M, and the outcome Y

- Estimates of total, direct, and indirect effects would all be biased and inconsistent due to unobserved confounding



Sensitivity analysis

- Suppose that…
  - the unobserved confounder U is binary
  - $E(Y|c,d,m,U=1) - E(Y|c,d,m,U=0)$ is constant in c, d, and m
  - $P(U=1|c,d,m) - P(U=1|c,d^*,m)$ is constant in c and m
- In this scenario, the confounding bias in an estimator for the natural direct effect can be expressed as follows:

$$Bias\left(\widehat{NDE}(d,d^*)\right) = [E(Y|c,d,m,U=1) - E(Y|c,d,m,U=0)] \times [P(U=1|c,d,m) - P(U=1|c,d^*,m)]$$

$$= \delta\phi$$

Sensitivity analysis

- The confounding bias in an estimator for the natural direct effect:

$$Bias\left(\widehat{NDE}(d, d^*)\right) = \delta\phi$$

  - $\delta = E(Y|c, d, m, U = 1) - E(Y|c, d, m, U = 0)$ is the difference in the mean of the outcome associated with a unit increase in the unobserved confounder, conditional on the baseline confounders and mediator

  - $\phi = P(U = 1|c, d, m) - P(U = 1|c, d^*, m)$ is the difference in the probability of the unobserved confounder comparing level d versus d^* of the exposure, conditional on the confounders and mediator

Sensitivity analysis

- Suppose further that…
  - $E(Y|c,d,U=1) - E(Y|c,d,U=0)$ is constant in c and d
  - $P(U=1|c,d) - P(U=1|c,d^*)$ is constant in c
- In this scenario, the confounding bias in an estimator for the natural indirect effect can be expressed as follows:

$$Bias\left(\widehat{NIE}(d,d^*)\right) = [E(Y|c,d,U=1) - E(Y|c,d,U=0)] \times [P(U=1|c,d) - P(U=1|c,d^*)] - \delta\phi$$

$$= \eta\psi - \delta\phi$$

Sensitivity analysis

- The confounding bias in an estimator for the natural indirect effect:

$$Bias\left(\widehat{NIE}(d, d^*)\right) = \eta\psi - \delta\phi$$

  - δ and ϕ are defined as before
  - η is the difference in the mean of the outcome associated with a unit increase in the unobserved confounder, conditional only on the baseline confounders and the exposure
  - ψ is the difference in the probability of the unobserved confounder comparing level d versus d^* of the exposure, conditional only on the baseline confounders

Sensitivity analysis

- Under the same suppositions outlined previously…

$$Bias\left(\widehat{ATE}(d,d^*)\right) = Bias\left(\widehat{NDE}(d,d^*)\right) + Bias\left(\widehat{NIE}(d,d^*)\right)$$

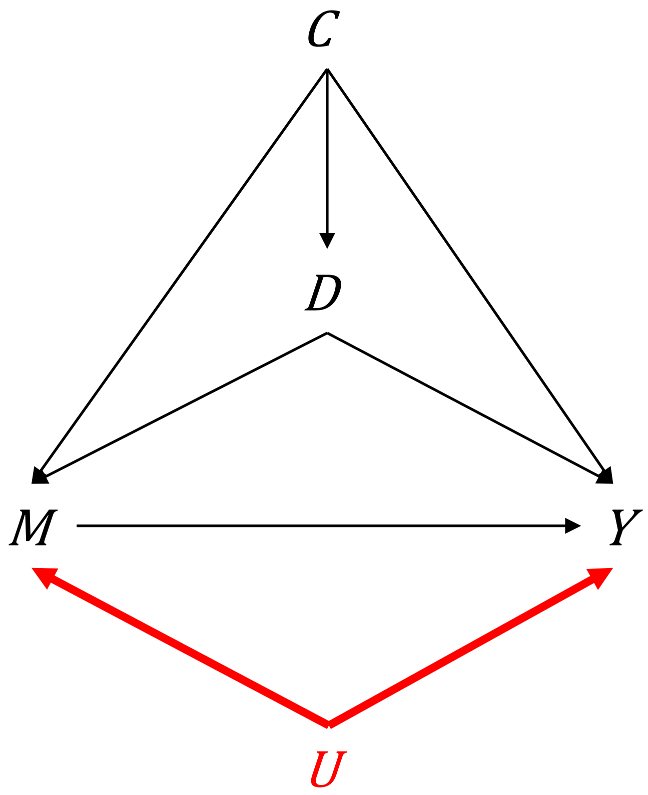
$$= \delta\phi + (\eta\psi - \delta\phi)$$

$$= \eta\psi$$

where $\eta = E(Y|c,d,U=1) - E(Y|c,d,U=0)$ and $\psi = P(U=1|c,d) - P(U=1|c,d^*)$

Sensitivity analysis

- Now consider a scenario in which an unobserved variable U affects only the mediator M and outcome Y
- Estimates of direct and indirect effects would be biased and inconsistent due to unobserved confounding
- Estimates of total effects, however, would still be unbiased and consistent
- This scenario is common in standard experiments, where only the exposure is randomized



Sensitivity Analysis

- In this scenario, and under the same assumptions about U as outlined previously, the bias in estimates of natural direct and indirect effects is given by the following expressions:

$$Bias\left(\widehat{NDE}(d, d^*)\right) = \delta\phi$$

$$Bias\left(\widehat{NIE}(d, d^*)\right) = -\delta\phi$$

where δ and ϕ are defined as before

Sensitivity Analysis

- With the bias formulas outlined previously, a sensitivity analysis proceeds by reevaluating a set of effect estimates across different hypothetical patterns of unobserved confounding

- To this end, we evaluate the bias formulas across a range of plausible values for their sensitivity parameters, and then construct bias-adjusted effect estimates by subtracting the bias from the corresponding point estimate:

$$\widehat{NDE}(d,d^*)^{adj} = \widehat{NDE}(d,d^*) - Bias\left(\widehat{NDE}(d,d^*)\right)$$

$$\widehat{NIE}(d,d^*)^{adj} = \widehat{NIE}(d,d^*) - Bias\left(\widehat{NIE}(d,d^*)\right)$$

$$\widehat{ATE}(d,d^*)^{adj} = \widehat{ATE}(d,d^*) - Bias\left(\widehat{ATE}(d,d^*)\right)$$

Example: NLSY79

- Compute bias-adjusted estimates for the natural direct and indirect effects of college attendance on depression, as mediated by unemployment

```
# sensitivity analysis for unobserved M-Y confounding #
sens.grid <- expand.grid(delta=seq(-0.2, 0.2, 0.02), phi=seq(-0.2, 0.2, 0.02))

adj.grid <- cbind(sens.grid,
    nde.adj=sparMedResults[2,1]-(sens.grid$delta*sens.grid$phi),
    nie.adj=sparMedResults[3,1]+(sens.grid$delta*sens.grid$phi))
```

Example: NLSY79

- Compute bias-adjusted estimates for the NDE and NIE

```r
nde.plot <- ggplot(adj.grid, aes(x=delta, y=phi, z=nde.adj, colour=stat(level))) +
        geom_contour(
            breaks=seq(round(min(adj.grid$nde.adj), 3),
            round(max(adj.grid$nde.adj), 3), 0.005), show.legend=FALSE) +
        scale_colour_distiller(palette="Greys", direction=1) +
        xlab(expression(delta)) +
        ylab(expression(phi)) +
        scale_x_continuous(breaks=seq(-0.2,0.2,0.04)) +
        scale_y_continuous(breaks=seq(-0.2,0.2,0.04)) +
        ggtitle("A. Bias-adjusted NDE(1,0) Estimates") +
        theme_bw(base_size=11) +
        theme(
            panel.grid.major=element_blank(),
            panel.grid.minor=element_blank()) +
        geom_text_contour(
            breaks=seq(
                round(min(adj.grid$nde.adj), 3),
                round(max(adj.grid$nde.adj), 3), 0.005),
            stroke=0.3,
            size=3,
            skip=0,
            color="black")
```

Example: NLSY79

- Compute bias-adjusted estimates for the NDE and NIE

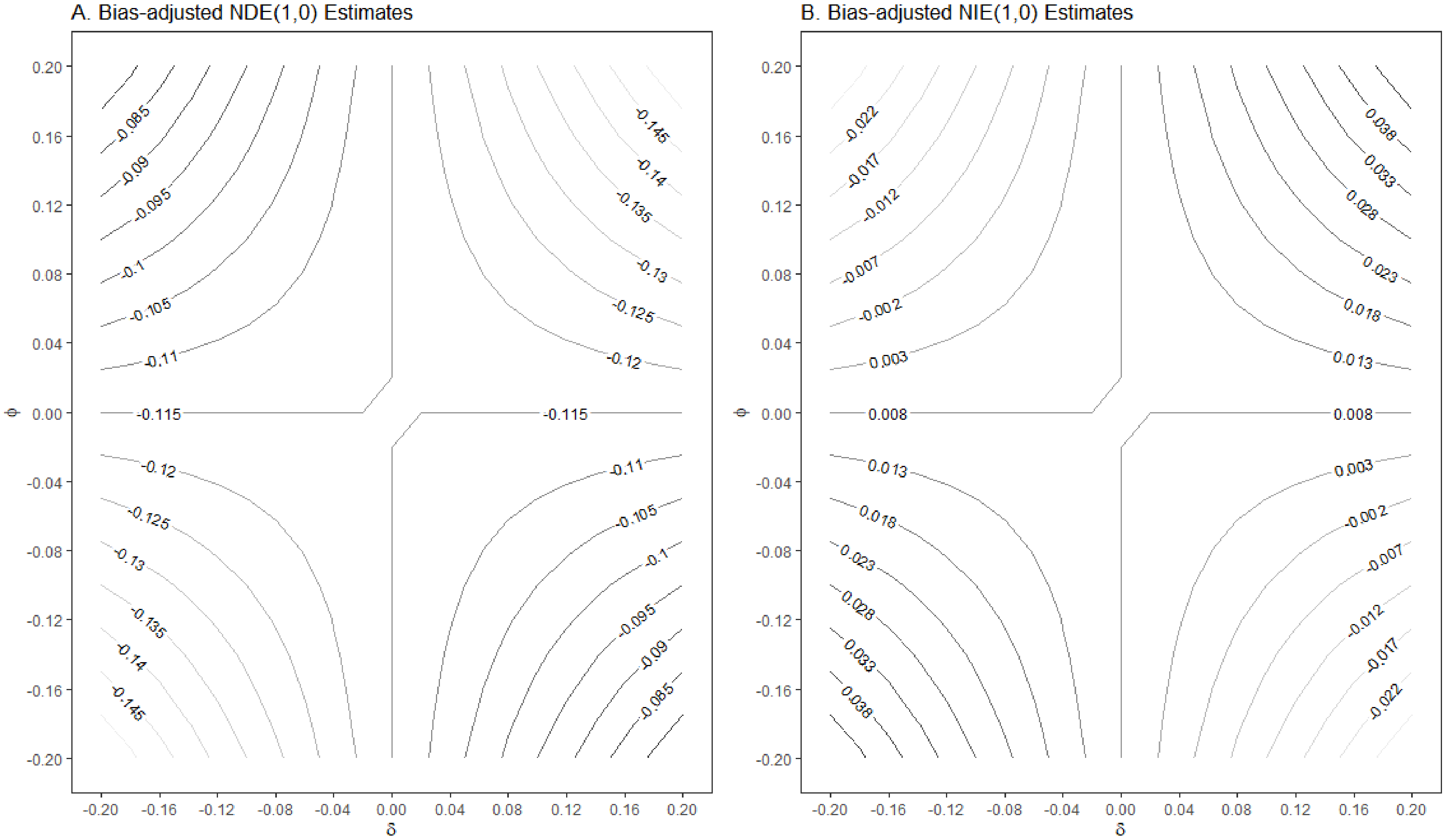
```
nie.plot <- ggplot(adj.grid, aes(x=delta,y=phi,z=nie.adj,colour=stat(level))) +
        geom_contour(
            breaks=seq(round(min(adj.grid$nie.adj),3),
            round(max(adj.grid$nie.adj),3),0.005),show.legend=FALSE) +
        scale_colour_distiller(palette="Greys",direction=1) +
        xlab(expression(delta)) +
        ylab(expression(phi)) +
        scale_x_continuous(breaks=seq(-0.2,0.2,0.04)) +
        scale_y_continuous(breaks=seq(-0.2,0.2,0.04)) +
        ggtitle("B. Bias-adjusted NIE(1,0) Estimates") +
        theme_bw(base_size=11) +
        theme(
            panel.grid.major=element_blank(),
            panel.grid.minor=element_blank()) +
        geom_text_contour(
            breaks=seq(
                round(min(adj.grid$nie.adj),3),
                round(max(adj.grid$nie.adj),3),0.005),
            stroke=0.3,
            size=3,
            skip=0,
            color="black")

comb.plot <- grid.arrange(nde.plot, nie.plot, ncol=2, nrow=1)

print(comb.plot)
```

Example: NLSY79

- Compute bias-adjusted estimates for the NDE and NIE



Extensions

- Similar methods can also be used to construct bias-adjusted estimates for…
  - multivariate natural and path-specific effects (Zhou 2022)
  - interventional direct and indirect effects (Wodtke and Zhou 2020)