EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Causal Machine Learning - A Hands-on Tutorial with Double Machine Learning

EMAC annual conference 2023, Odense/Denmark May 26, 2023

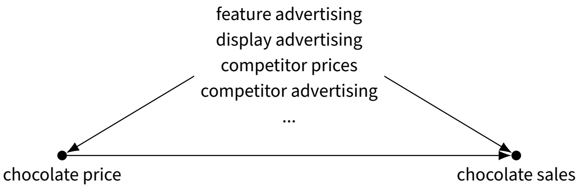
Jonathan Fuhr
Dominik Papies
School of Business and Economics
University of Tübingen

machine learning
new perspectives
for science

Funded by the DFG – EXC number 2064/1
– Project number 390727645

Why causal inference and ML?

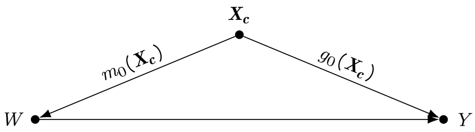
1 | Causal questions are at the heart of marketing research



- For many causal questions, we only have observational data
- Under certain assumptions, we can do causal inference from observational data
- Machine learning can potentially help to relax some of these assumptions

How DML works

2 | Double machine learning is one of the most popular methods using ML for CI



Partially linear model:

$$Y = \beta W + g_0(\boldsymbol{X_c}) + V_y$$
$$W = m_0(\boldsymbol{X_c}) + V_w$$

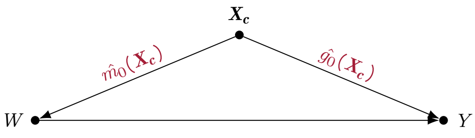
- W: Treatment
- Y: Outcome
- $\boldsymbol{X_c} \in \mathbb{R}^p$: Observed confounders

The problem with traditional approaches

- Traditional approach: Estimate linear outcome model $Y = \beta W + \boldsymbol{\gamma X_c} + V_y$
- Problem: High-dimensional nuisance parameters
  - → Functions $g_0(\boldsymbol{X_c})$ and $m_0(\boldsymbol{X_c})$ are potentially complex and nonlinear
  - → Traditional approaches break down for $p > n$

Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W., & Robins, J. (2018). Double/debiased machine learning for treatment and structural parameters. The Econometrics Journal, 21(1), C1–C68.

2 | DML uses ML to adjust flexibly for nonlinear/high-dimensional confounding



Estimates equations:

$$Y = \hat{g}_0(\boldsymbol{X_c}) + V_y$$

$$W = \hat{m}_0(\boldsymbol{X_c}) + V_w$$

- W: Treatment
- Y: Outcome
- $\boldsymbol{X_c} \in \mathbb{R}^p$: Observed confounders

How DML works

- Models $g_0(\boldsymbol{X_c})$ and $m_0(\boldsymbol{X_c})$ with flexible ML methods
- Predicts and takes residuals of treatment and outcome
  - → "Eliminates" confounding influence
- Regresses residual of Y on residual W
  - → Effect estimate

Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W., & Robins, J. (2018). Double/debiased machine learning for treatment and structural parameters. The Econometrics Journal, 21(1), C1–C68.

2 | The DML algorithm for the partially linear model

1. Split the data into K folds
2. Train two machine learning models on $K-1$ folds:
   a. Outcome: W; features: $\boldsymbol{X_c}$
   b. Outcome: Y; features: $\boldsymbol{X_c}$
3. Use the models to make predictions ($\hat{W}$ and $\hat{Y}$) on the held-out fold
4. Compute residuals as $\hat{V}_W = W - \hat{W}$ and $\hat{V}_Y = Y - \hat{Y}$
5. Use a linear regression to estimate coefficient from residuals:
   Regress $\hat{V}_Y$ on $\hat{V}_W$, obtain the coefficient on $\hat{V}_W$
6. Repeat for all folds, average resulting coefficients to obtain the final causal estimate
7. For more robustness w.r.t. the random partitioning in finite samples:
   Repeat the algorithm S (e.g., 100) times for different splits, then report the median estimate

Implementing DML

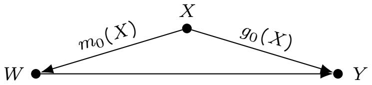
3 | Own implementation: Doing DML once

```
dml_once_rf <- function(dataset, idx1, idx2) {  # indices come from second function
  # Predicting treatment from the controls, using first part of the data
  rf1 <- randomForest(formula = w ~ . - y, data = df[idx1, ], ntree = 200)

  # Predicting outcome from the controls, using first part of the data
  rf2 <- randomForest(formula = y ~ . - w, data = df[idx1, ], ntree = 200)

  # Make predictions for treatment and outcome with trained model, using second half of the data
  W_hat <- predict(rf1, newdata = df[idx2, ])
  Y_hat <- predict(rf2, newdata = df[idx2, ])

  # Compute residuals for treatment and outcome
  W_resid <- df[idx2, "w"] - W_hat
  Y_resid <- df[idx2, "y"] - Y_hat

  # Regress residual of outcome on residual of treatment, obtain coefficient for treatment residual
  beta_w <- coef(lm(Y_resid ~ 0 + W_resid))
  return(beta_w)
}
```

3 | Own implementation: Cross-fitting

```
dml_cross_rf <- function(dataset, K = 5) {
  # Generate indices for K folds
  idx_k <- vector(mode = "list", length = K)
  idx <- 1:nrow(dataset)
  for (k in 1:(K - 1)) { # draw indices for each split randomly into a list
    idx_k[[k]] <- sample(idx, 1/K * nrow(dataset))
    idx <- setdiff(idx, idx_k[[k]])}
  idx_k[[K]] <- idx

  beta <- numeric(K)  # Empty vector to store the coefficients
  # Apply the dml_once() function for each fold: ML on larger, estimation on smaller sample
  for (k in 1:K) {
    # Split sample in two parts
    idx2 <- idx_k[[k]]
    idx1 <- setdiff(1:nrow(dataset), idx2)

    # Run dml_once()
    beta[k] <- dml_once_rf(dataset, idx1, idx2)
    }
  # Average the resulting coefficients
  return(mean(beta))
}
```

3 | A simple simulation to show DML's flexibility



Linear DGP

```r
library(randomForest); set.seed(314)
n <- 1000
x <- rnorm(n)
w <- x + rnorm(n)      # Linear DGP
y <- w + x + rnorm(n)  # Linear DGP
df <- data.frame(y, w, x)
coef(lm(formula = y ~ w + x, data = df))[["w"]]  # Standard linear regression
## [1] 0.9989102
dml_cross_rf(data = df)      # DML with random forests
## [1] 0.9792316
```

Nonlinear DGP

```r
w <- x^2 + rnorm(n)      # Nonlinear DGP
y <- w + x^2 + rnorm(n)  # Nonlinear DGP
df <- data.frame(y, w, x)
coef(lm(formula = y ~ w + x, data = df))[["w"]]  # Standard linear regression
## [1] 1.646125
dml_cross_rf(data = df)      # DML with random forests
## [1] 1.011167
```

3 | Implementing DML with the {DoubleML} package

Object-oriented implementation of DML in Python and R

- In Python: built on top of {scikit-learn}
- In R: built on top of the {mlr3}-family
- Implemented models: partially linear (IV) regression, interactive (IV) regression, and more

1. Construct data object

```
library(DoubleML)
dml_data <- DoubleMLData$new(df, y_col = "y", d_cols = "w", x_cols = c("x"))
```

2. Initialize ML models (choose any, here random forests)

```
library(mlr3); library(mlr3learners)
ml_l_rf <- lrn("regr.ranger", num.trees = 200)  # outcome model
ml_m_rf <- lrn("regr.ranger", num.trees = 200)  # treatment model
```

Bach, P., Chernozhukov, V., Kurz, M. S., & Spindler, M. (2023). DoubleML – An Object-Oriented Implementation of Double Machine Learning in R (arXiv:2103.09603). arXiv.

Bach, P., Chernozhukov, V., Kurz, M. S., & Spindler, M. (2022). DoubleML - An Object-Oriented Implementation of Double Machine Learning in Python. Journal of Machine Learning Research, 23(53), 1–6.

3 | Estimating with {DoubleML}

3. Initialize DoubleML object

```
dml_plr_rf <- DoubleMLPLR$new(dml_data, ml_l = ml_l_rf, ml_m = ml_m_rf)
```

4. Estimate the model

```
dml_plr_rf$fit()
```

5. Show the estimates

```
dml_plr_rf$coef
##         w
## 0.9902597

dml_plr_rf$summary()
## Estimates and significance testing of the effect of target variables
##   Estimate. Std. Error t value Pr(>|t|)
## w   0.99026    0.03387   29.24   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Applying DML

4 | We ran a battery of simulations to evaluate DML

- Baseline simulation with nonlinear confounding
- Varying the functional form of the confounding
- Varying the confounding strength
- Varying the number of observed confounders
- Varying the sample size
- Including varying numbers of noise variables
- Including variables only related to outcome
- Including variables only related to treatment
- Including an unobserved confounder
- Defining covariates as colliders instead of confounders
- Varying the number of folds in cross-fitting
- Varying the number of DML repetitions
- Including nonlinear transformations of all confounders

4 | Practical considerations and recommendations

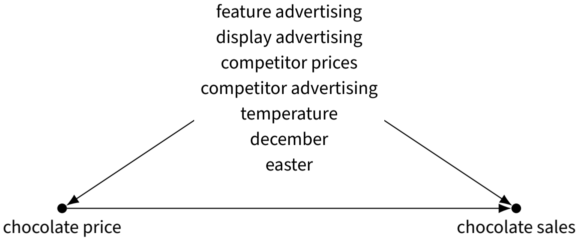
When should we use DML?

- If the effect is causally identified based on observables, e.g.,
  - → Unconfoundedness/conditional exogeneity
  - → Instrumental variables with conditionally exogeneous instrument
- Statistical guarantees only for cross-sectional data, panel data more complicated

How should we use DML?

- Which variables should we include?
  - → Confounders, outcome influencers, (noise variables), no instruments, no bad controls
- Use flexible ML models, e.g., random forests, XGBoost, neural networks
  - → Tune hyperparameters to improve predictive accuracy and avoid overfitting
  - → Choose ML method based on predictive accuracy in first stage
- Choose number of folds K based on sample size:
  - → Small sample: $K = 5 - 10$
  - → Larger sample: $K = 2 - 5$
- Choose number of algorithm repetitions S based on stability of estimates in repeated runs

4 | We apply DML to estimate the price elasticity for Verhouten chocolate



Leeflang, P. S. H., Wieringa, J. E., Bijmolt, T. H. A., & Pauwels, K. H. (2015). Modeling Markets: Analyzing Marketing Phenomena and Improving Marketing Decision Making. Springer.

4 | We use sales data to estimate the price elasticity for Verhouten chocolate

```
## Rows: 68
## Columns: 19
## $ price2     <dbl> 1.559941, 1.558169, 1.553499, 1.556005, 1.561368, 1.256176, 1.353876, 1.469375,~
## $ price3     <dbl> 1.748575, 1.376562, 1.403592, 1.478236, 1.583986, 1.654707, 1.701853, 1.741389,~
## $ price4     <dbl> 1.277969, 1.355054, 1.488910, 1.727150, 1.797017, 1.882738, 1.892483, 1.949858,~
## $ feature1   <dbl> 0.00, 0.00, 0.00, 0.10, 0.20, 0.00, 0.44, 0.24, 0.08, 0.02, 0.60, 0.19, 0.05, 0~
## $ feature2   <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.29, 0.03, 0.03, 0.00, 0.00, 0.00, 0.05, 0~
## $ feature3   <dbl> 0.00, 0.00, 0.00, 0.29, 0.08, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0~
## $ feature4   <dbl> 0.35, 0.54, 0.14, 0.05, 0.00, 0.02, 0.00, 0.00, 0.02, 0.02, 0.00, 0.00, 0.00, 0~
## $ display1   <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0~
## $ display2   <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0~
## $ display3   <dbl> 0.00, 0.82, 0.85, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0~
## $ display4   <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.33, 0~
## $ fand1      <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0~
## $ fand3      <dbl> 0.00, 0.47, 0.86, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0~
## $ fand4      <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.91, 0~
## $ temp       <int> 2, -1, 3, 2, 5, 3, 1, 2, 4, 7, 5, 6, 9, 8, 8, 13, 12, 10, 12, 14, 15, 17, 20, 1~
## $ december   <int> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ easter     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0~
## $ log_sales  <dbl> 4.196090, 4.451186, 4.494493, 4.592833, 4.633573, 4.661668, 6.274526, 4.904650,~
## $ log_price1 <dbl> 0.41181597, 0.41184043, 0.41214721, 0.41064019, 0.41010533, 0.41027694, 0.02646~
```

Leeflang, P. S. H., Wieringa, J. E., Bijmolt, T. H. A., & Pauwels, K. H. (2015). Modeling Markets: Analyzing Marketing Phenomena and Improving Marketing Decision Making. Springer.

4 | Estimation with traditional methods

Naive: simple linear regression

```
summary(lm(log_sales ~ log_price1, data = dchoc))$coef["log_price1", ]
##      Estimate    Std. Error       t value      Pr(>|t|)
## -3.937954e+00  2.153666e-01 -1.828489e+01  1.560591e-27
```

Adjusting for all observed covariates

```
form_full <- log_sales ~ log_price1 + price2 + price3 + price4 + feature1 + feature2 + feature3 +
                         feature4 + display1 + display2 + display3 + display4 + fand1 + fand3 +
                         fand4 + temp + december + easter

summary(lm(form_full, data = dchoc))$coef["log_price1", ]
##      Estimate    Std. Error       t value      Pr(>|t|)
## -3.852695e+00  3.205155e-01 -1.202031e+01  3.176756e-16
```

4 | The predictive accuracy of the ML methods can guide algorithm choice

- Note: We use more elaborate DML implementation (see notebook)
  → Computes standard errors
  → Report predictive accuracy of the first stage

DML with random forests

```
set.seed(314)
double_ml_rf(dchoc, form_full, outcome = "log_sales", treatment = "log_price1", K = 7)
##         beta           se        mse_w        mse_y
## -3.473462165  0.224939668  0.008029504  0.129608516
```

DML with linear regression

```
set.seed(314)
double_ml_lm(dchoc, form_full, outcome = "log_sales", treatment = "log_price1", K = 7)
##       beta         se      mse_w      mse_y
## -2.4886148  0.1673306  0.0109083  0.1746908
```

4 | Repeating DML in small samples makes estimates more robust

```
set.seed(12)
double_ml_rf(dchoc, form_full, outcome = "log_sales", treatment = "log_price1", K = 7)
##          beta           se        mse_w        mse_y
## -3.130125418  0.134088341  0.009297124  0.160322797
double_ml_rf(dchoc, form_full, outcome = "log_sales", treatment = "log_price1", K = 7)
##          beta           se        mse_w        mse_y
## -3.438799520  0.167773070  0.009186392  0.152462506
double_ml_rf(dchoc, form_full, outcome = "log_sales", treatment = "log_price1", K = 7)
##          beta           se        mse_w        mse_y
## -3.716021924  0.191319586  0.008911716  0.157043813
```

```
set.seed(12)
median(sapply(1:50, function(i) double_ml_rf(dchoc, form_full, outcome = "log_sales",
                                             treatment = "log_price1", K = 7))["beta",])
## [1] -3.373811
median(sapply(1:50, function(i) double_ml_rf(dchoc, form_full, outcome = "log_sales",
                                             treatment = "log_price1", K = 7))["beta",])
## [1] -3.412695
median(sapply(1:50, function(i) double_ml_rf(dchoc, form_full, outcome = "log_sales",
                                             treatment = "log_price1", K = 7))["beta",])
## [1] -3.403591
```

Conclusion and open questions

5 | DML is a very general framework for flexible effect estimation with ML

DML generalizes beyond the partially linear model

- Interactive model: binary treatment can fully interact with covariates
- Instrumental variable models (partially linear or interactive)
- Further recent developments: Difference-in-Differences, IV quantile regression, …
- Extension to **panel data** not obvious yet

What DML can and cannot do for us

- DML is a data-driven *estimation* method
  - → Can use any supervised ML method
  - → Allows adjusting flexibly for observed covariates: relaxes functional form assumptions
- DML is not a new *identification* strategy
  - → Given valid identification, it can improve estimation

Appendix

1 | DML in Stata

`ddml` package:

- Compatible with various ML programs in Stata
- Recommend stacking from `pystacked`: ensemble of multiple learners

Ahrens, A., Hansen, C. B., Schaffer, M. E., & Wiemann, T. (2023). ddml: Double/debiased machine learning in Stata (arXiv:2301.09397). arXiv. https://doi.org/10.48550/arXiv.2301.09397

1 | {DoubleML} implementation in Python

Object-oriented implementation of DML in Python and R

- In Python: built on top of {scikit-learn}
- In R: built on top of the {mlr3}-family
- Implemented models: partially linear (IV) regression, interactive (IV) regression, and more

1. Construct data object

```
from doubleml import DoubleMLData; df = r.df  # imports the simulated data from R
dml_data = DoubleMLData(df, y_col = "y", d_cols = "w", x_cols = ["x"])
```

2. Initialize ML models (choose any, here random forests)

```
from sklearn.ensemble import RandomForestRegressor
ml_l_rf = RandomForestRegressor(n_estimators = 200)  # outcome model
ml_m_rf = RandomForestRegressor(n_estimators = 200)  # treatment model
```

Bach, P., Chernozhukov, V., Kurz, M. S., & Spindler, M. (2023). DoubleML – An Object-Oriented Implementation of Double Machine Learning in R (arXiv:2103.09603). arXiv.
Bach, P., Chernozhukov, V., Kurz, M. S., & Spindler, M. (2022). DoubleML - An Object-Oriented Implementation of Double Machine Learning in Python. Journal of Machine Learning Research, 23(53), 1–6.

1 | Estimating with {DoubleML} in Python

3. Initialize DoubleML object

```
from doubleml import DoubleMLPLR
dml_plr_rf = DoubleMLPLR(dml_data, ml_l = ml_l_rf, ml_m = ml_m_rf)
```

4. Estimate the model

```
dml_plr_rf.fit()
```

5. Show the estimates

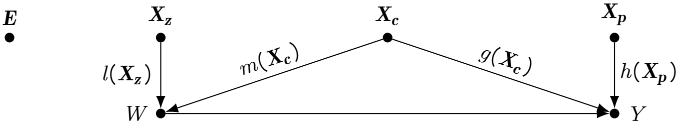
```
dml_plr_rf.coef

## array([1.01447626])
dml_plr_rf.summary
##        coef  std err          t          P>|t|     2.5 %    97.5 %
## w  1.014476  0.03304  30.704477  4.960021e-207  0.949719  1.079233
```

1 | Estimating the application with DoubleML in Python

```python
import numpy as np; np.random.seed(314)
from doubleml import DoubleMLData; dchoc = r.dchoc  # imports the data from R

dml_data = DoubleMLData(dchoc, y_col = 'log_sales', d_cols = 'log_price1',
                        x_cols = ['price2', 'price3', 'price4', 'feature1', 'feature2', 'feature3',
                                  'feature4', 'display1', 'display2', 'display3', 'display4',
                                  'fand1', 'fand3', 'fand4', 'temp', 'december', 'easter'])

from sklearn.ensemble import RandomForestRegressor
ml_l_rf = RandomForestRegressor(n_estimators = 200)  # outcome model
ml_m_rf = RandomForestRegressor(n_estimators = 200)  # treatment model

from doubleml import DoubleMLPLR
dml_plr_rf = DoubleMLPLR(dml_data, ml_l = ml_l_rf, ml_m = ml_m_rf, dml_procedure='dml1')

dml_plr_rf.fit()
```

```python
dml_plr_rf.summary
##                  coef   std err          t         P>|t|     2.5 %    97.5 %
## log_price1 -4.161086  0.278496 -14.941254  1.776019e-50 -4.706929 -3.615243
dml_plr_rf.rmses  # shows first-stage predictive accuracy (currently Python only)
## {'ml_l': array([[0.43265058]]), 'ml_m': array([[0.10683433]])}
```

1 | Which types of variables should we include in the algorithm?



- Noise variables E: doesn't significantly affect methods
  - → include if in doubt
- Variables influencing outcome X_p: helps precision of estimates (reduces variance)
  - → include
- Variables influencing treatment X_z: hurts precision of estimates (reduces variance in treatment)
  - → exclude if sure that no relationship with Y, include if unsure (less damage)