Benchmarking Estimators for Natural Experiments: A Novel Dataset and Doubly Robust Algorithm

R. Teal Witter

New York University

Christopher Musco

New York University

Treatment Effect Estimation

There are n **observations**, each with **covariates** $\boldsymbol{x}_i \in \mathbb{R}^d$ for $i \in [n]$.

Observation i receives the "treatment" with **propensity** $p_i \in (0,1)$ and the "control" otherwise.

We then receive either the **treatment outcome** $y_i^{(1)}$ or the **control outcome** $y_i^{(0)}$, but not both.

Our goal is to estimate the **average treatment effect**:

$$\tau = \frac{1}{n} \sum_{i \in [n]} y_i^{(1)} - y_i^{(0)}$$

A Novel Treatment Effect Dataset

Reach Out and Read Colorado (RORCO) is an early childhood literacy nonprofit that gives free books to children at pediatric check-ups.

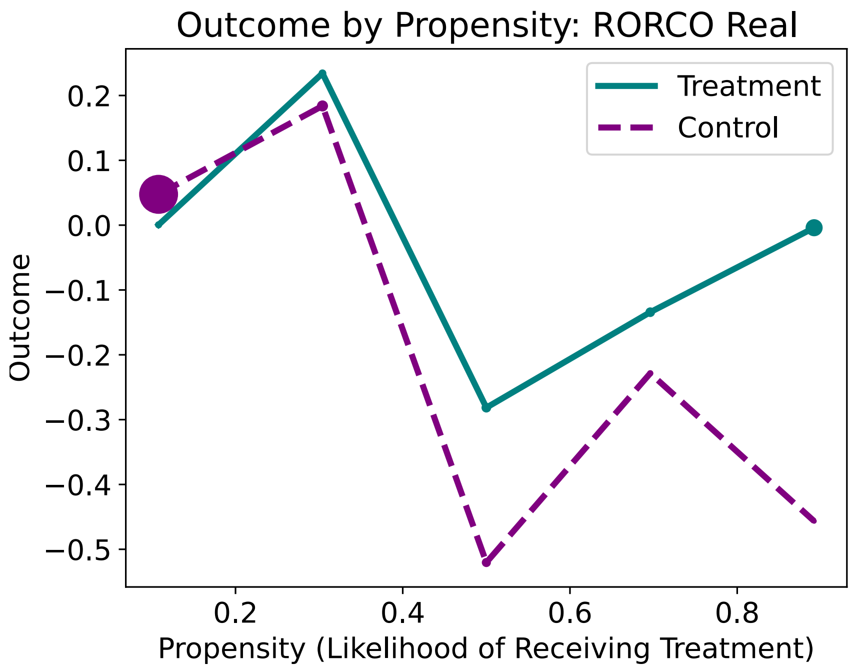
Dataset	**Size**	**Variables**	**Treated %**	**BCE**	$\mathbf{Corr}(\mathbf{y}^{(1)}, \mathbf{p})$	$\mathbf{Corr}(\mathbf{y}^{(0)}, \mathbf{p})$
JOBS	722	8	41.1	0.0856	0.0355	0.0541
TWINS	50820	40	49.4	0.499	-0.00311	-0.0036
IHDP	747	26	18.6	0.452	0.0967	0.0236
NEWS	5000	3	45.8	0.545	0.86	-0.565
ACIC 2016	4802	54	18.4	0.372	0.112	0.0383
ACIC 2017	4302	50	47.4	0.436	-0.269	-0.153
RORCO Real	4178	78	25.3	0.158	-0.000602	-0.0739
RORCO	21663	78	44.3	0.212	-0.986	-0.989

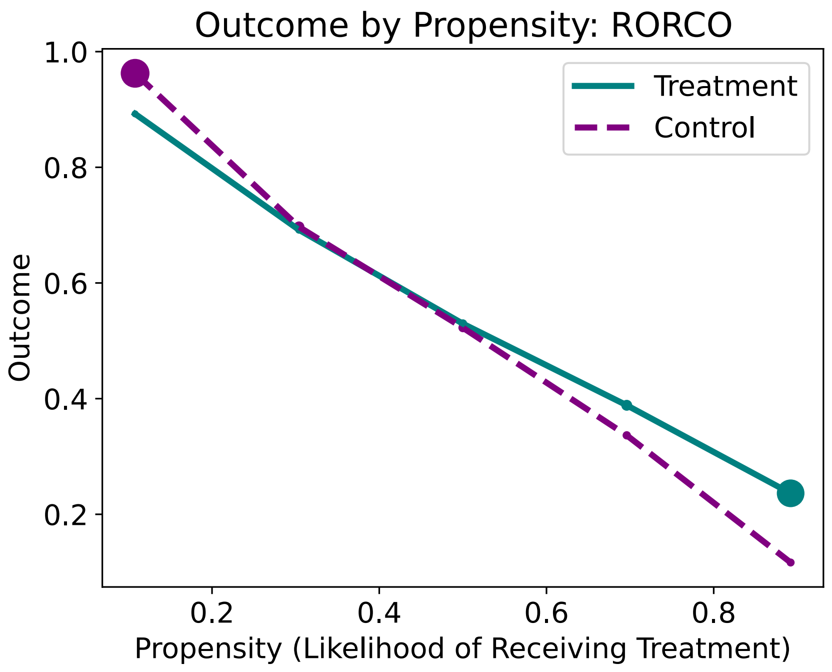
Challenging Confoundedness

RORCO prioritizes under-served children to maximize positive impact but this causes:

a) **Confounding between outcome and propensity** and

b) **Confounding between treatment effect and propensity**.





Benchmarking Treatment Effect Estimators

Method	Mean	1st Quartile	2nd Quartile	3rd Quartile	Time (s)
Regression Discontinuity	4.65e-03	2.72e-03	3.84e-03	5.52e-03	9.55e-04
Propensity Stratification	2.57e-03	1.52e-03	2.25e-03	3.29e-03	2.78e-03
Direct Difference	4.48e-01	3.57e-01	4.18e-01	5.79e-01	4.74e-04
Adjusted Direct	6.29e-03	5.25e-03	6.20e-03	7.14e-03	1.15e+01
Horvitz-Thompson	1.06e-02	4.29e-03	9.20e-03	1.44e-02	4.65e-04
TMLE	1.19e-01	7.21e-03	2.60e-02	7.43e-02	2.35e+01
Off-policy	3.17e-03	1.86e-03	2.86e-03	4.11e-03	1.14e+01
Double-Double	1.07e-05	1.06e-06	4.41e-06	1.45e-05	2.29e+01
Doubly Robust	9.98e-07	1.48e-07	5.42e-07	1.37e-06	9.89e+00
Direct Prediction	1.36e-02	3.60e-03	1.02e-02	1.94e-02	1.23e+01
SNet	2.57e-02	4.85e-03	1.21e-02	3.62e-02	3.49e+01
FlexTENet	1.15e-03	4.28e-05	1.09e-04	4.95e-04	1.56e+02
OffsetNet	1.10e-03	7.72e-04	9.90e-04	1.41e-03	1.30e+02
TNet	8.05e-04	6.39e-05	2.50e-04	4.37e-04	1.06e+02
TARNet	1.92e-04	2.70e-05	1.04e-04	2.38e-04	1.01e+02
DragonNet	2.18e-02	4.42e-03	1.71e-02	2.46e-02	6.88e+00
SNet3	1.80e-02	3.48e-03	9.80e-03	2.50e-02	2.36e+01
DRNet	5.00e-03	1.53e-04	6.01e-04	2.25e-03	1.14e+02
RANet	7.85e-04	3.67e-05	2.08e-04	7.06e-04	1.91e+02
PWNet	2.28e-01	7.02e-03	4.00e-02	2.82e-01	1.13e+02
RNet	2.96e-03	2.47e-03	2.84e-03	3.43e-03	5.83e+01
XNet	1.00e-03	3.08e-05	2.29e-04	9.26e-04	2.41e+02

Doubly Robust Estimators

We learn functions $f^{(1)}, f^{(0)}: \mathbb{R}^d \to \mathbb{R}$ to predict the treatment and control outcomes.

$$\hat{\tau}(\mathbf{z}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i^{(1)} - f^{(1)}(\mathbf{x}_i)}{p_i}\mathbb{1}_{z_i=1} - \frac{y_i^{(0)} - f^{(0)}(\mathbf{x}_i)}{1-p_i}\mathbb{1}_{z_i\neq 1} + f^{(1)}(\mathbf{x}_i) - f^{(0)}(\mathbf{x}_i)\right)$$

Doubly robust estimators are **asymptotically** correct if either:

a) The propensity scores are accurate (and the functions’ error is independent)

b) The functions are accurate (and the propensity scores’ error is independent).

Non-Asymptotic Setting

Why are doubly robust estimators so accurate in the finite setting?

We exactly analyze doubly robust estimators that learn functions $f^{(1)}, f^{(0)}: \mathbb{R}^d \to \mathbb{R}$ separately from the data they are applied to. For these estimators, the variance is

$$\mathrm{Var}[\hat{\tau}(\mathbf{z}) - \tau] = \frac{1}{n^2} \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}, S_1, S_2} \left[\left((y_i^{(1)} - \hat{f}^{(1)}_{\mathbf{z}, S(i)}(\mathbf{x}_i)) \sqrt{\frac{1 - p_i}{p_i}} + (y_i^{(0)} - \hat{f}^{(0)}_{\mathbf{z}, S(i)}(\mathbf{x}_i)) \sqrt{\frac{p_i}{1 - p_i}} \right)^2 \right]$$
$$+ \frac{1}{n^2} \sum_{i \neq j} \mathbb{E}_{\mathbf{z}, S_1, S_2} \left[\left(\hat{y}_i(\mathbf{z}^{(j \to 1)}) - \hat{y}_i(\mathbf{z}^{(j \to 0)}) \right) \left(\hat{y}_j(\mathbf{z}^{(i \to 1)}) - \hat{y}_j(\mathbf{z}^{(i \to 0)}) \right) \right].$$

Non-Asymptotic Setting

Why are doubly robust estimators so accurate in the finite setting?

We exactly analyze doubly robust estimators that learn functions $f^{(1)}, f^{(0)}: \mathbb{R}^d \to \mathbb{R}$ separately from the data they are applied to. For these estimators, the variance is

$$\operatorname{Var}[\hat{\tau}(\mathbf{z}) - \tau] \approx \frac{1}{n^2} \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}, S_1, S_2} \left[\left((y_i^{(1)} - \hat{f}^{(1)}_{\mathbf{z}, S(i)}(\mathbf{x}_i)) \sqrt{\frac{1 - p_i}{p_i}} + (y_i^{(0)} - \hat{f}^{(0)}_{\mathbf{z}, S(i)}(\mathbf{x}_i)) \sqrt{\frac{p_i}{1 - p_i}} \right)^2 \right]$$

Non-Asymptotic Setting

Why are doubly robust estimators so accurate in the finite setting?

We exactly analyze doubly robust estimators that learn functions $f^{(1)}, f^{(0)}: \mathbb{R}^d \to \mathbb{R}$ separately from the data they are applied to. For these estimators, the variance is

$$\mathrm{Var}[\hat{\tau}(\mathbf{z}) - \tau] \approx \frac{1}{n^2} \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z},S_1,S_2}\left[\left((y_i^{(1)} - \hat{f}^{(1)}_{\mathbf{z},S(i)}(\mathbf{x}_i))\sqrt{\frac{1-p_i}{p_i}} + (y_i^{(0)} - \hat{f}^{(0)}_{\mathbf{z},S(i)}(\mathbf{x}_i))\sqrt{\frac{p_i}{1-p_i}}\right)^2\right]$$

What if we learn the functions to minimize this error?

⇒ The Double-Double Algorithm

Double-Double Algorithm

Method	Mean	1st Quartile	2nd Quartile	3rd Quartile	Time (s)
Doubly Robust	9.98e-07	1.48e-07	5.42e-07	1.37e-06	9.89e+00
DR + Weighting	4.02e-06	5.46e-07	2.62e-06	5.57e-06	9.81e+00
DR + 2x Weighting	3.80e-06	2.48e-07	9.71e-07	3.82e-06	9.80e+00
DR + Split	9.82e-05	3.27e-06	1.21e-05	3.65e-05	2.22e+01
DR + Split + Weight	1.12e-04	2.19e-06	1.03e-05	2.41e-05	2.22e+01
Double-Double	1.07e-05	1.06e-06	4.41e-06	1.45e-05	2.29e+01

Doubly robust estimators without the training split perform better but they effectively use twice as much data *as doubly robust estimators with the training split*

Natural Experiments Package

```
import naturalexperiments as ne

X, y, z = ne.dataloaders['RORCO']()                       # Load datasets

estimator = ne.methods['Double-Double']                   # Load estimator

p = ne.estimate_propensity(X, z)                          # Estimate propensity

estimated_effect = estimator(X, y, z, p, ne.train)
```

Thank you!

Codebase: github.com/rtealwitter/naturalexperiments

arXiv: arxiv.org/abs/2409.04500

Email: rtealwitter@gmail.com