ddml: Double/debiased machine learning in Stata

Achim Ahrens (ETH Zürich)

Mark E Schaffer (Heriot-Watt University, IZA)
Christian B Hansen (University of Chicago)
Thomas Wiemann (University of Chicago)

Package website: `https://statalasso.github.io/`

Latest version available here

November 18, 2022

Swiss Stata Group Meeting

Introduction

A rich and growing literature exploits machine learning to facilitate causal inference.

A central focus: *high-dimensional* controls and/or instruments, which can arise if

- we observe many controls/instruments
- controls/instruments enter through an unknown function

Belloni, Chernozhukov, and Hansen (2014) and Belloni et al. (2012) propose estimators *relying on the Lasso* that allow for high-dimensional controls/instruments.

Available via `pdslasso` in Stata (Ahrens, Hansen, and Schaffer, 2020)

Introduction

What if we don't want to use the lasso?

- The Lasso might not be the *best-performing machine learner* for a particular problem.
- The Lasso relies on the *approximate sparsity assumption*, which might not be appropriate in some settings.

Chernozhukov et al. (2018) propose *Double/Debiased Machine Learning* (DDML) which allow to exploit machine learners other than the Lasso.

Our contribution:

- We introduce `ddml`, which implements DDML for Stata.
- We provide simulation evidence on the finite sample performance of DDML.
- Our recommendation is to use DDML in combination with Stacking.

Background

Motivating example. The partial linear model:

$$y_i = \underbrace{\theta d_i}_{\text{causal part}} + \underbrace{g(\boldsymbol{x}_i)}_{\text{nuisance}} + \varepsilon_i.$$

How do we account for confounding factors $\boldsymbol{x}_i$? — The standard approach is to assume linearity $g(\boldsymbol{x}_i) = \boldsymbol{x}_i \beta$ and consider alternative combinations of controls.

Problems:

- Non-linearity & unknown interaction effects
- High-dimensionality: we might have "many" controls
- We don't know which controls to include

Background

Motivating example. The partial linear model:

$$y_i = \underbrace{\theta d_i}_{\text{causal part}} + \underbrace{g(\boldsymbol{x}_i)}_{\text{nuisance}} + \varepsilon_i.$$

Post-double selection (Belloni, Chernozhukov, and Hansen, 2014) and *post-regularization* (Chernozhukov, Hansen, and Spindler, 2015) provide data-driven solutions for this setting.

Both "double" approaches rely on the *sparsity assumption* and use two auxiliary lasso regressions: y_i $\boldsymbol{x}_i$ and d_i $\boldsymbol{x}_i$. lasso PDS

Related approaches exist for *optimal IV* estimation (Belloni et al., 2012) and/or *IV with many controls* (Chernozhukov, Hansen, and Spindler, 2015).

Background

These methods have been implemented for Stata in `pdslasso` (Ahrens, Hansen, and Schaffer, 2020), `dsregress` (StataCorp) and R (`hdm`; Chernozhukov, Hansen, and Spindler, 2016).

Example syntax:

```
. pdslasso $Y $D (c.($X)#c.($X)), robust
```

Example 1 (`pdslasso`) allows for high-dimensional controls.

Example 2 (`ivlasso`) treats `avexpr` as endogenous and exploits `logem4` as an instrument. (More details in the `pds/ivlasso` help file.)

Background

There are **advantages** of relying on lasso:

- intuitive assumption of (approximate) sparsity
- computationally relatively cheap (due to plugin lasso penalty; no cross-validation needed)
- Linearity has its advantages (e.g. extension to fixed effects; Belloni et al., 2016)

But there are also **drawbacks**:

- What if the sparsity assumption is not plausible?
- There is a wide set of machine learners at disposable—Lasso might not be the best choice.
- Lasso requires careful feature engineering to deal with non-linearity & interaction effects.

Review of DDML

The partial linear model:

$$
\begin{aligned}
y_i &= \theta d_i + g(\boldsymbol{x}_i) + \varepsilon_i \\
d_i &= m(\boldsymbol{x}_i) + v_i
\end{aligned}
$$

Naive idea: We estimate conditional expectations $\ell(\boldsymbol{x}_i) = E[y_i|\boldsymbol{x}_i]$ and $m(\boldsymbol{x}_i) = E[d_i|\boldsymbol{x}_i]$ using ML and partial out the effect of $\boldsymbol{x}_i$ (in the style of Frisch-Waugh-Lovell):

$$
\hat{\theta}_{DDML} = \left(\frac{1}{n}\sum_i \hat{v}_i^2\right)^{-1} \frac{1}{n}\sum_i \hat{v}_i(y_i - \hat{\ell}),
$$

where $\hat{v}_i = d_i - \hat{m}_i$.

Review of DDML

Yet, there is a problem: The estimation error $(\boldsymbol{x}_i) - \hat{}$ and v_i may be correlated due to **over-fitting**, leading to poor performance.

DDML, thus, relies on **cross-fitting** (sample splitting with swapped samples).

DDML for the partial linear model (DML 2)

We split the sample in K random folds of equal size denoted by I_k:

- For $k = 1, \ldots, K$, estimate $(\boldsymbol{x}_i)$ and $m(\boldsymbol{x}_i)$ using sample I_k^c and form out-of-sample predictions $\hat{}_i$ and $\hat{m}_i$ for all i in I_k.
- Construct estimator $\hat{\theta}$ as

$$\left(\frac{1}{n}\sum_i \hat{v}_i^2\right)^{-1} \frac{1}{n}\sum_i \hat{v}_i(y_i - \hat{}),$$

where $\hat{v}_i = d_i - \hat{m}_i$. $\hat{m}_i$ and $\hat{}_i$ are the cross-fitted predicted values.

DDML models

The DDML framework can be applied to other models (all implemented in `ddml`):

Interactive model

$$y_i = g(d_i, \boldsymbol{x}_i) + u_i \qquad E[u_i|\boldsymbol{x_i}, d_i] = 0$$
$$z_i = m(\boldsymbol{x}_i) + v_i \qquad E[u_i|\boldsymbol{x_i}] = 0$$

As in the Partial Linear Model, we are interested in the ATE, but do not assume that d_i (a binary treatment variable) and $\boldsymbol{x}_i$ are separable.

We estimate the conditional expectations $E[y_i|\boldsymbol{x}_i, d_i = 0]$ and $E[y_i|\boldsymbol{x}_i, d_i = 1]$ as well as $E[d_i|\boldsymbol{x}_i]$ using a supervised machine learner.

DDML models

The DDML framework can be applied to other models (all implemented in `ddml`):

Partial linear IV model

$$y_i = d_i\theta + g(\boldsymbol{x}_i) + u_i \qquad E[u_i|\boldsymbol{x_i}, z_i] = 0$$
$$z_i = m(\boldsymbol{x}_i) + v_i \qquad E[v_i|\boldsymbol{x_i}] = 0$$

where the aim is to estimate the average treatment effect θ using observed instrument z_i in the presence of controls $\boldsymbol{x}_i$. We estimate the conditional expectations $E[y_i|\boldsymbol{x_i}]$, $E[d_i|\boldsymbol{x}_i]$ and $E[z_i|\boldsymbol{x}_i]$ using a supervised machine learner.

DDML models

The DDML framework can be applied to other models (all implemented in `ddml`):

High-dimensional IV model

$$y_i = d_i\theta + g(\boldsymbol{x}_i) + u_i$$
$$d_i = h(\boldsymbol{z}_i) + m(\boldsymbol{x}_i) + v_i$$

where the parameter of interest is θ. The instruments and controls enter the model through unknown functions $g()$, $h()$ and $f()$.

We estimate the conditional expectations $E[y_i|\boldsymbol{x}_i]$, $E[\hat{d}_i|\boldsymbol{x}_i]$ and $\hat{d}_i := E[d_i|\boldsymbol{z}_i, \boldsymbol{x}_i]$ using a supervised machine learner. The instrument is then formed as $\hat{d}_i - \hat{E}[\hat{d}_i|\boldsymbol{x}_i]$ where $\hat{E}[\hat{d}_i|\boldsymbol{x}_i]$ denotes the estimate of $E[\hat{d}_i|\boldsymbol{x}_i]$.

DDML models

The DDML framework can be applied to other models (all implemented in `ddml`):

Interactive IV model

$$\begin{aligned} y_i &= \mu(\boldsymbol{x}_i, \boldsymbol{z}_i) + u_i & E[u_i|\boldsymbol{x}_i, z_i] = 0 \\ d_i &= m(z_i, \boldsymbol{x}_i) + v_i & E[v_i|\boldsymbol{x}_i, z_i] = 0 \\ z_i &= p(\boldsymbol{x}_i) + \xi_i & E[\xi_i|\boldsymbol{x}_i] = 0 \end{aligned}$$

where the aim is to estimate the local average treatment effect.

We estimate, using a supervised machine learner, the following conditional expectations: $E[y_i|\boldsymbol{x}_i, z_i = 0]$ and $E[y_i|\boldsymbol{x}_i, z_i = 1]$; $E[D|\boldsymbol{x}_i, z_i = 0]$ and $E[D|\boldsymbol{x}_i, z_i = 1]$; $E[z_i|\boldsymbol{x}_i]$.

The `ddml` package

We introduce `ddml` for Stata:

- Compatible with various ML programs in Stata (e.g. `lassopack`, `pylearn`, `randomforest`).
  - → *Any* program with the classical "`reg y x`" syntax and post-estimation `predict` will work.
- Short (one-line) and flexible multi-line version
- 5 models supported: partial linear model, interactive model, interactive IV model, partial IV model, optimal IV.

Stacking regression

Which machine learner should we use?

`ddml` supports a range of ML programs: `pylearn`, `lassopack`, `randomforest`. — Which one should we use?

We don't know whether we have a sparse or dense problem; linear or non-linear. We don't know whether, e.g., lasso or random forests will perform better.

Stacking, as implemented in `pystacked`, provides a solution: We use an 'optimal' combination of base learners.

Stacking regression

Which machine learner should we use?

We don't know whether we have a sparse or dense problem; linear or non-linear; etc.

Stacking is an ensemble method that combines multiple base learners into one model. As the default, we use *constrained least squares*:

$$\boldsymbol{w} = \arg\min_{w_j} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{J} w_j \hat{y}_i^{(j)} \right)^2, \quad w_j \geq 0, \quad \sum_j w_j = 1$$

where $\hat{y}_i^{(j)}$ are cross-validated predictions of base learner j.

Voting regression is a special case with unweighted (or user-specified) weights.

Extended ddml syntax

Step 1: Initialise `ddml` and select model:

`ddml init` *model* [, `kfolds(`*integer*`) reps(`*integer*`) ...`]

where *model* is either 'partial', 'iv', 'interactive', 'ivhd', 'late'.

Step 2: Add supervised ML programs for estimating conditional expectations:

`ddml` *eq* [*newvarname*] [, *eqopt*] : *command depvar indepvars* [, *cmdopt*]

where *eq* selects the conditional expectations to be estimated. *command* is a ML program that supports the standard `reg y x`-type syntax. *cmdopt* are specific to that program.

Multiple estimation commands per equation are allowed.

Extended `ddml` syntax

Step 3: Cross-fitting

```
ddml crossfit [ , shortstack ]
```

Step 4: Estimation of causal effects

```
ddml estimate [ , robust ... ]
```

Additional auxiliary commands:

`ddml describe` (describe current model set up), `ddml save` & `ddml use` (to import/save `ddml` objects), `ddml extract` (to retrieve objects), `ddml export` (export in csv format).

Extended `ddml` syntax: Example

We demonstrate the use of `ddml` using the partially linear model by extending the analysis of 401(k) eligibility and total financial wealth of Poterba, Venti, and Wise (1995). The data consists of $n = 9915$ households from the 1991 SIPP.

Step 0: Load data, define globals

```
. use "sipp1991.dta", clear
. global Y net_tfa
. global X age inc educ fsize marr twoearn db pira hown
. global D e401
```

Step 1: Initialise `ddml` and select model:

```
. set seed 42
. ddml init partial, kfolds(4)
```

Extended `ddml` syntax: Example (cont'd.)

Step 2: Add supervised ML programs for estimating conditional expectations. We use OLS, Lasso and Random Forest.

```
. *** add learners for E[Y|X]
. ddml E[Y|X]: reg $Y $X
Learner Y1_reg added successfully.

. ddml E[Y|X]: cvlasso $Y c.($X)#c.($X), lopt postresults
Learner Y2_cvlasso added successfully.

. ddml E[Y|X], vtype(none): rforest $Y $X, type(reg)
Learner Y3_rforest added successfully.

. *** add learners for E[D|X]
. ddml E[D|X]: reg $D $X
Learner D1_reg added successfully.

. ddml E[D|X]: cvlasso $D c.($X)#c.($X), lopt postresults
Learner D2_cvlasso added successfully.

. ddml E[D|X], vtype(none): rforest $D $X, type(reg)
Learner D3_rforest added successfully.
```

Extended `ddml` syntax: Example (cont'd.)

Step 3: Cross-fitting with 5 folds

```
. ddml crossfit, shortstack
Cross-fitting E[Y|X] equation: net_tfa
Cross-fitting fold 1 2 3 4 ...completed cross-fitting...completed short-stacking
Cross-fitting E[D|X] equation: e401
Cross-fitting fold 1 2 3 4 ...completed cross-fitting...completed short-stacking
```

Extended `ddml` syntax: Example (cont'd.)

Step 4: Estimation of causal effects

```
. ddml estimate, robust

DDML estimation results:
spec  r     Y learner     D learner         b        SE
   1  1        Y1_reg        D1_reg  5964.151(1522.426)
   2  1        Y1_reg    D2_cvlasso  8390.126(1356.633)
   3  1        Y1_reg    D3_rforest  8054.667(1271.281)
   4  1    Y2_cvlasso        D1_reg  9350.056(1381.641)
*  5  1    Y2_cvlasso    D2_cvlasso  9570.601(1318.880)
  ss  1  [shortstack]          [ss]  9401.724(1300.628)
   ...  <-click or type ddml estimate, replay full to display full summary
* = minimum MSE specification for that resample.

Min MSE DDML model, specification 5
y-E[y|X]  = Y2_cvlasso_1                           Number of obs   =      9915
D-E[D|X,Z]= D2_cvlasso_1
------------------------------------------------------------------------------
             |               Robust
     net_tfa | Coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
        e401 |   9570.601    1318.88     7.26   0.000     6985.644    12155.56
------------------------------------------------------------------------------
```

`qddml` example: Partial linear model

`qddml` is the one-line ('quick') version of `ddml` and uses a syntax similar to `pds/ivlasso`.

```
. qddml $Y $D (c.($X)#c.($X)), model(partial) ///
>                              cmd(cvlasso) cmdopt(lopt postresults) ///
>                              robust

DDML estimation results:
spec  r     Y learner     D learner         b        SE
   1  1        Y1_reg        D1_reg  9504.777(1368.314)
*  2  1        Y1_reg    D2_cvlasso  9512.796(1357.514)
   3  1    Y2_cvlasso        D1_reg  9534.451(1373.390)
   4  1    Y2_cvlasso    D2_cvlasso  9483.607(1361.398)
* = minimum MSE specification for that resample.

Min MSE DDML model, specification 2
y-E[y|X]  = Y1_reg_1                                 Number of obs   =      9915
D-E[D|X,Z]= D2_cvlasso_1
------------------------------------------------------------------------------
             |               Robust
     net_tfa | Coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
        e401 |   9512.796   1357.514     7.01   0.000     6852.117    12173.47
------------------------------------------------------------------------------
```

Simulation I: Advantages of Stacking

Simulation set-up

We consider a *linear DGP* and a *non-linear DGP*, and compare performance of OLS, PDS-Lasso and various machine learners, including stacking.

We would expect that stacking performs well under both settings, while linear approaches only perform well if the DGP is linear.

Simulation I: Advantages of DDML+Stacking

Calibrated simulation based on Poterba, Venti, and Wise (1995), who estimate the causal effect of 401(k) eligibility on wealth.

1. Construct the partial residuals $y_i^{(r)} = y_i - \hat{\tau}_{OLS} d_i, \forall i$ where $\hat{\tau}_{OLS}$ is the full sample OLS estiamte.
2. We predict $y_i^{(r)}$ with the controls x_i either using
   - linear regression (*Linear DGP*)
   - gradient boosting (*Non-Linear DGP*)

   and call the fitted estimator $\tilde{h}$.
3. Similarily, predict d_i given x_i and call the estimator $\tilde{g}$.
3. We draw bootstrap sample $\mathcal{D}_b$ of size n_s from the data
4. To generate 401(k) eligibility and log wealth, we calculate

$$\tilde{d}_i^{(b)} = \mathbb{1}\{\tilde{h}(x_i) + \nu_i \geq 0.5\}, \quad \nu_i \overset{iid}{\sim} \mathcal{N}(0, \kappa_1)$$
$$\tilde{y}_i^{(b)} = \tau_0 \tilde{d}_i^{(b)} + \tilde{g}(x_i) + \varepsilon_i, \quad \varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, \kappa_2), \quad \forall i \in \mathcal{D}_b$$

where κ_1 and κ_2 are chosen to match distributions of d_i and y_i.

Simulation I: Advantages of DDML+Stacking

Table: Average Stacking Weights

	Linear DGP		Non-Linear DGP	
	$Y\|X$	$D\|X$	$Y\|X$	$D\|X$
	(1)	(2)	(3)	(4)
OLS	0.06	0.29	0.00	0.02
cv-Lasso	0.35	0.17	0.01	0.00
cv-Ridge	0.35	0.17	0.01	0.00
Series Lasso (w/o interactions)	0.11	0.11	0.08	0.22
Series Lasso (w/ interactions)	0.07	0.13	0.31	0.30
Gradient boosting (low regularization)	0.03	0.05	0.30	0.23
Gradient boosting (high regularization)	0.02	0.05	0.29	0.20
Random forest (low regularization)	0.01	0.01	0.01	0.01
Random forest (high regularization)	0.01	0.01	0.01	0.01
Neural Network	0.00	0.00	0.00	0.00

Simulation I: Advantages of DDML+Stacking

Table: Coefficient Estimates

	n_s = 9915		n_s = 99150	
Panel (A): Linear DGP	Bias (1)	Rate (2)	Bias (3)	Rate (4)
OLS	-89.7	0.95	-0.3	0.94
DDML:				
cv-Lasso	-88.2	0.95	-0.4	0.94
Gradient boosting	-103.6	0.95	-7.2	0.94
Ensemble (stacking)	-112.5	0.94	-2.5	0.94
Panel (B): Non-Linear DGP	Bias (5)	Rate (6)	Bias (7)	Rate (8)
OLS	-2580.0	0.54	-2599.4	0.00
DDML:				
cv-Lasso	-2615.7	0.53	-2604.0	0.00
Gradient boosting	-50.7	0.94	-0.4	0.98
Ensemble (stacking)	248.8	0.94	-1.2	0.98

Simulation II: Small Sample Performance

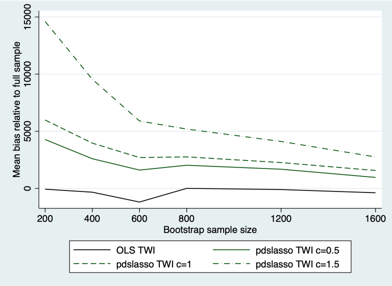
Wüthrich and Zhu (2021, henceforth WZ) demonstrate that PDS-Lasso suffers from a large finite sample bias and tends to underselect; again using the application of Poterba, Venti, and Wise (1995) and Belloni et al. (2017).

They use two specifications:

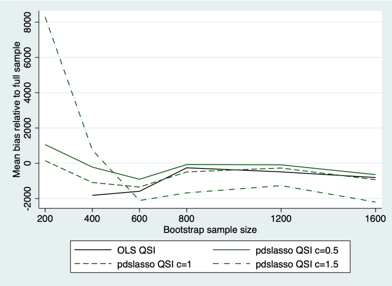
- two-way interactions (TWI) (as in Chernozhukov and Hansen, 2004); $p = 167$
- quadratic splines & interactions (QSI) (as in Belloni et al., 2017); $p = 272$

WZ run their simulations on bootstrap samples of the data ($n_b = \{200, 400, 800, 1600\}$) and calculate the bias as the mean difference to the full sample estimate ($N = 9915$).

Simulation II: Small Sample Performance



(a) Bias (TWI specification)

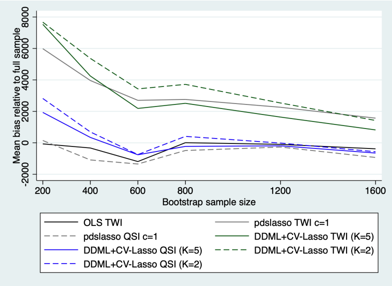


(b) Bias (QSI specification)

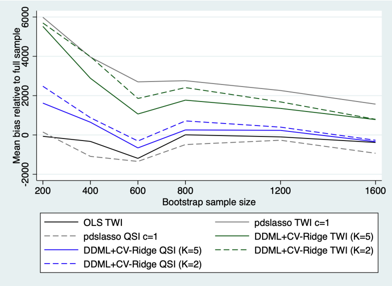
Notes: The figures report the mean bias calculated as the mean di□erence to the full sample estimates. Following WZ, we draw 600 bootstrap samples of size $n_b = \{200, 400, 600, 800, 1200, 1600\}$. 'TWI' indicates that the predictors have been expanded by two-way interactions. 'QSI' refers to the quadratic spline & interactions specification of Belloni et al. (2017).

Figure: Replication of Figure 8 in WZ

Simulation II: Small Sample Performance



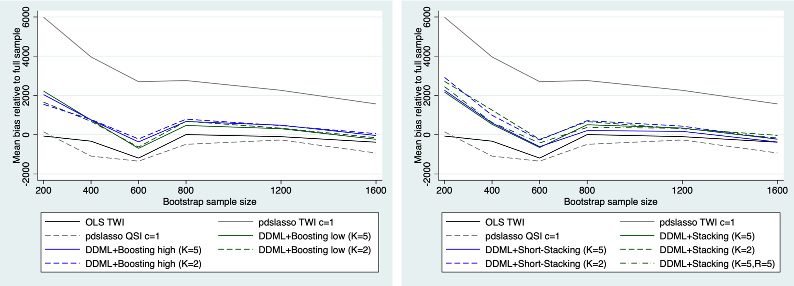
(a) CV-Lasso



(b) CV-Ridge

Figure: Mean bias relative to full sample

Simulation II: Small Sample Performance



(a) Boosted trees (b) Stacking

Figure: Mean bias relative to full sample

The small sample bias of stacking stabilizes for $n_b > 600$, suggesting that stacking may perform well for 'moderate' sample sizes.

Summary

- `ddml` implements Double/Debiased Machine Learning for Stata:
  - Compatible with various ML programs in Stata
  - Short (one-line) and flexible multi-line version
  - Uses Stacking Regression as the default machine learner; implemented via separate program `pystacked`
  - 5 models supported
- The advantage to `pdslasso` is that we can make use of almost any machine learner.
- *But which machine learner should we use?*
  - We suggest stacking. We don't know which learner is best suited for a particular problem.
  - Stacking allows to consider multiple learners in a joint framework, and thus reduces the risk of misspecification.
- We are in the final phase of development; hopefully we can make `ddml` available soon (following your feedback)

References I

Ahrens, Achim, Christian B. Hansen, and Mark E. Schaffer (2020). "lassopack: Model selection and prediction with regularized regression in Stata". In: *The Stata Journal* 20.1, pp. 176–235. url: https://doi.org/10.1177/1536867X20909697.

Belloni, A et al. (2017). "Program Evaluation and Causal Inference With High-Dimensional Data". In: *Econometrica* 85.1, pp. 233–298. url: https://onlinelibrary.wiley.com/doi/abs/10.3982/ECTA12723.

Belloni, Alexandre, Victor Chernozhukov, and Christian Hansen (2014). "Inference on treatment effects after selection among high-dimensional controls". In: *Review of Economic Studies* 81, pp. 608–650. url: https://doi.org/10.1093/restud/rdt044.

Belloni, Alexandre et al. (2012). "Sparse Models and Methods for Optimal Instruments With an Application to Eminent Domain". In: *Econometrica* 80.6. Publisher: Blackwell Publishing Ltd, pp. 2369–2429. url: http://dx.doi.org/10.3982/ECTA9626.

References II

Belloni, Alexandre et al. (2016). "Inference in High Dimensional Panel Models with an Application to Gun Control". In: *Journal of Business & Economic Statistics* 34.4. Genre: Methodology, pp. 590–605. url: https://doi.org/10.1080/07350015.2015.1102733 (visited on 02/14/2015).

Chernozhukov, Victor and Christian Hansen (Aug. 2004). "The effects of 401(K) participation on the wealth distribution: An instrumental quantile regression analysis". In: *The Review of Economics and Statistics* 86.3. tex.eprint: https://direct.mit.edu/rest/article-pdf/86/3/735/1614135/0034653041811734.pdf, pp. 735–751. url: https://doi.org/10.1162/0034653041811734.

Chernozhukov, Victor, Christian Hansen, and Martin Spindler (May 2015). "Post-Selection and Post-Regularization Inference in Linear Models with Many Controls and Instruments". In: *American Economic Review* 105.5, pp. 486–490. url: https://doi.org/10.1257/aer.p20151022.

– (2016). "High-dimensional metrics in R". In: 401, pp. 1–32.

References III

Chernozhukov, Victor et al. (2018). "Double/debiased machine learning for treatment and structural parameters". In: *The Econometrics Journal* 21.1. tex.ids= Chernozhukov2018a publisher: John Wiley & Sons, Ltd (10.1111), pp. C1–C68. url: https://onlinelibrary.wiley.com/doi/abs/10.1111/ectj.12097.

Poterba, James M, Steven F Venti, and David A Wise (1995). "Do 401 (k) contributions crowd out other personal saving?" In: *Journal of Public Economics* 58.1, pp. 1–32.

Wüthrich, Kaspar and Ying Zhu (2021). "Omitted variable bias of Lasso-based inference methods: A finite sample analysis". In: *Review of Economics and Statistics* 0.(0), pp. 1–47.