Finite sample results for lasso and stepwise partialling out Poisson

David M. Drukker *
Sam Houston State University
dxd070@shsu.edu

Di Liu
Stata
dliu@stata.com

November 10, 2020

1 Introduction

High-dimensional models that include many covariates which might potentially affect an outcome are increasingly common. One approach to high-dimensional models makes a sparsity assumption which requires that the number of covariates that must be included in the model is small relative to the sample size.[1] As discussed below, this sparse approach to high-dimensional models uses covariate selection and moment conditions that are robust to the mistakes made by the covariate-selection technique to produce reliable inference for some of the model parameters. We call this sparse approach the partialing-out (PO) approach, because it extends the classic partialing-out technique for obtaining some regression coefficients after removing the impact of other covariates. The PO approach was derived in Belloni, Chen, Chernozhukov, and Hansen (2012), Belloni, Chernozhukov, and Hansen (2014), and Belloni, Chernozhukov, and Wei (2016).

This paper makes four contributions to the rapidly evolving literture on PO estimators for the parameters of interest. First, it presents simulation evidence that there are DGPs for

*We thank Enrique Pinzon and Joerg Luedicke for discussions and for their work in running previous simulations. We also thank Fang Wang for speeding up the coordinate-descent algorithm that does the numerical optimization. For comments along the way, we thank seminar participants at Boston College, Harvard School of Public Health, Queen's University, Sam Houston State University, Syracuse Univeristy, Tilberg University, University of Leicester, University of Maryland, participants of the Bank of Canada conference on Microeconometrics and data science conference, and participants of the 25th Texas Camp Econometrics.

[1]Another approach removes the many-covariate bias and uses many-covariate robust methods to estimate the asymptotic variance of the bias corrected estimator. See Cattaneo, Jansson, and Newey (2018b) and Cattaneo, Jansson, and Ma (2018a) for this approach.

which a BIC-stepwise version of the Belloni, Chernozhukov, and Wei (2016) PO estimator performs well but the usual lasso-based PO estimators fail. Second, this paper the extends the Belloni, Chen, Chernozhukov, and Hansen (2012) algorithm for selecting the lasso tuning parameters to the Poisson model and it presents simulation evidence that a version of the Belloni, Chernozhukov, and Wei (2016) PO estimator that uses a Poisson lasso with these tuning parameters performs well. Third, this paper presents simulation evidence that selecting the lasso tuning parameters using the BIC, produces a version of the Belloni, Chernozhukov, and Wei (2016) PO estimator that performs about as well as the plugin estimator. Fourth, this paper presents simulation evidence that selecting the lasso tuning parameters using cross-validation, produces a version of the Belloni, Chernozhukov, and Wei (2016) PO estimator that performs worse than the PO estimator that uses the plugin-based lasso, the BIC-based lasso, or BIC-based stepwise.

2 PO estimator for parameters in an HDM

2.1 High-dimensional models

A cross-sectional high-dimensional generalized linear model (GLM) can be written as

$$\mathbf{E}[y_i|\mathbf{d}_i, \mathbf{x}_i] = G(\mathbf{d}_i\boldsymbol{\alpha}_0' + \mathbf{x}_i\boldsymbol{\beta}_0')$$

where y is the outcome, $\mathbf{d}_i$ are the covariates of interest, $\mathbf{x}_i$ are the control covariates that potentially need to be included in the model, $\boldsymbol{\alpha}_0$ are the coefficients on $\mathbf{d}_i$, and $\boldsymbol{\beta}_0$ are the coefficients on $\mathbf{x}_i$. $G()$ maps the linear index $\mathbf{d}_i\boldsymbol{\alpha}_0' + \mathbf{x}_i\boldsymbol{\beta}_0'$ to the conditional mean. Although there are many other possibilities, three common models are when $G()$ is the identity function for linear models, when $G()$ is the standard logistic distribution for logit models, or when $G()$ is the exponential function for Poisson or exponential conditional mean models.

The number of potential covariates in $\mathbf{x}_i$ ($p_{\mathbf{x}}$) can be larger than the sample size n. We are interested in the case in which $p_{\mathbf{x}}$ is too large for a GLM regression of y on $\mathbf{d}$ and $\mathbf{x}$ to produce reliable results for $\boldsymbol{\alpha}$, but the number of covariates in $\mathbf{x}$ that belong in the model ($s_{\mathbf{x}}$) is not too large. Belloni, Chen, Chernozhukov, and Hansen (2012) and Belloni, Chernozhukov, and Wei (2016) derive rates that must bind $s_{\mathbf{x}}$ as a function of n and $p_{\mathbf{x}}$. The assumption that $s_{\mathbf{x}}$ is not too large is known as a sparsity assumption.

The goal is to obtain reliable estimation and inference for $\boldsymbol{\alpha}_0$. The number of covariates in $\mathbf{d}_i$ is assumed to be fixed and small relative to n. The variables in $\mathbf{d}_i$ must be specified a-priori.

The key features of a high-dimensional model are that we are only interested in estimating $\boldsymbol{\alpha}_0$, that there are too many covariates in $\mathbf{x}$ to to reliably estimate $\boldsymbol{\alpha}_0$ using a quasi-maximum-likelihood (QML) estimator of y on $\mathbf{d}$ and $\mathbf{x}$, and that a sparsity assumption holds.

The sparsity assumption makes the problem feasible and implies that we have a covariate selection problem. Let $\tilde{\mathbf{x}}_n$ be the subset of $\mathbf{x}$ that we need to include for a QML estimator of y on $\mathbf{d}$ and $\tilde{\mathbf{x}}_n$ to be a root-N consistent and asyptotically normal estimator for $\boldsymbol{\alpha}_0$. We view our model as an approximation to reality and allow the number of covariates $\tilde{\mathbf{x}}_n$ to change with the sample size. Belloni, Chen, Chernozhukov, and Hansen (2012) and Belloni,

Chernozhukov, and Wei (2016) provide formal statements and analyses of how to allow for and how to bind the approximation error.

For our purposes, we note that allowing the model to approximate reality make its results more believable and more robust. But this robustness comes at a cost. Approximators tend to have some coeficients that are small in magnitude.

Algorithm 1 is a "naive" estimator for $\boldsymbol{\alpha}_0$ that has been used previously in the literature.[2]

Algorithm 1: Naive estimator for $\boldsymbol{\alpha}_0$

1. Use a feasible covariate-selection technique to select the subset of $\mathbf{x}$ that should be included in the model. Call these selected covariates $\check{\mathbf{x}}$.

2. Use a QML Poisson estimator of y on $\mathbf{d}$ and $\check{\mathbf{x}}$ to estimate $\boldsymbol{\alpha}_0$.

Leeb and Pötscher (2006), Leeb and Pötscher (2008), and Pötscher and Leeb (2009) show that naive estimators like the one in algorithm 1 do not have an asymptotic normal distribution and can perform poorly in finite samples when some of the coefficients are small in magnitude. In repeated samples, which of the covariates with small coefficients are included is random. This random inclusion causes small amounts of omitted-variable bias to be randomly added to the estimator. This random omitted-variable bias makes the distribution of the naive estimator have a nonnormal, asymptotic distribution. Using a normal distribution to approximate this nonnormal distribution can produce exceptionally poor results in finite samples.

The PO estimators were explicitly designed to provide valid inference for $\boldsymbol{\alpha}$ when some of the coefficients in the model are small in magnitude. See Belloni, Chen, Chernozhukov, and Hansen (2012), Belloni, Chernozhukov, and Wei (2016), and Chernozhukov, Hansen, and Spindler (2015) for formal results. Instead of naively using the covariates selected in a model of y on $\mathbf{d}$ and $\mathbf{x}$, PO estimators use moment conditions that are robust to the inevitable mistakes that covariate selection methods make. The PO estimators use multiple covariate-selection steps to form a moment condition for $\boldsymbol{\alpha}$ that is orthogonal to $\mathbf{x}$. This process can be viewed as an extension of partialling-out in a linear a model, hence the PO moniker. At a deeper level, PO estimators use the efficient influence function (EIF) approach to create a robust moment condition for $\boldsymbol{\alpha}$. See Bickel, Klaassen, Ritov, and Wellner (1998) and Tsiatis (2006) for discussions of the EIF approach. See Chernozhukov, Hansen, and Spindler (2015) for formal discussions of how to use the EIF approach to create PO estimators.

While the theory for PO estimators is rapidly evolving, we are unaware of a simulation study that evaluates the finite-sample properties of PO estimators for a nonlinear model using different covariate-selection methods. This paper fills this gap and it produces four important results. First, we find that the BIC-based stepwise should be further investigated as a method for covariate selection in PO estimators. Our simulations identify a DGP for which the BIC-stepwise version of the PO estimator performs well, but all the lasso versions of PO fail. Furthermore, the PO estimator that uses BIC-stepwise for covariate selection performs about as well as the best PO estimators that use a lasso for covariate selection.

[2]See for example table 4 in Allcott and Kessler (2019), table 7 in Banerjee, Chandrasekhar, Duflo, and Jackson (2019), figures 4 and 5 in Bennedsen, Tsoutsoura, and Wolfenzon (2019), table 2 in Kallestrup-Lamb, Kock, and Kristensen (2016), and table 4 in Paravisini, Rappoport, Schnabl, and Wolfenzon (2014).

Second, we find that using the BIC to select the lasso tuning parameters produces a PO estimator that performs as well as the PO estimator that uses the plugin method of selecting the lasso tuning parameters. Third, we find that the ubiquitous cross-validation-based lasso should be avoided. We find that using cross-validation (CV) to select the lasso tuning parameters produces a PO estimator that performs worse than the PO estimator that uses the BIC or the plugin method of selecting the lasso tuning parameters. Fourth, we provide the details of how to implement the plugin estimator for GLM models.

2.2 The lasso

The lasso is a widely used technique for covariate selection. Mechanically, the cross-sectional GLM lasso with given penalty parameter λ and given penalty loadings κ_j $j \in \{1, \ldots, p\}$ solves

$$\widehat{\boldsymbol{\delta}} = \arg\min_{\boldsymbol{\delta}} \left\{ \frac{1}{n} \sum_{i=1}^{n} Q(y_i, \mathbf{w}_i \boldsymbol{\delta}') + \lambda \sum_{j=1}^{p} \kappa_j |\boldsymbol{\delta}_j| \right\} \tag{1}$$

where $\boldsymbol{\delta} = (\boldsymbol{\alpha}, \boldsymbol{\beta})$, $\mathbf{w}_i = (\mathbf{d}_i, \mathbf{x}_i)$, and $Q(y_i, \mathbf{w}_i \boldsymbol{\delta}')$ is the negative of the contribution of the i(th) observation to the GLM quasi-maximum-likelihood (QML) function. See Hastie, Tibshirani, and Wainwright (2015), Friedman, Hastie, Höfling, and Tibshirani (2007), and Friedman, Hastie, and Tibshirani (2010) for descriptions of the coordinate-descent algorithm used to perform the minimization. The formulas for $Q()$ are given in appendix A.

The first term in the objective function in equation (1) is the usual QML objective function. The second term penalizes the objective function for allowing a coefficient to differ from zero. The kink in the absolute value function in the penalty term causes some elements of $\widehat{\boldsymbol{\delta}}$ to be exactly zero at the minimum, while others are not zero; see Hastie, Tibshirani, and Wainwright (2015) for details.

That some elements of $\widehat{\boldsymbol{\delta}}$ are exactly zero at the minimum is the basis of the lasso as a covariate selection technique. The j(th) covariate in $\mathbf{w}$ is included, if the j(th) element in $\widehat{\boldsymbol{\delta}}$ is not zero. Analogously, the j(th) covariate in $\mathbf{w}$ is excluded, if the j(th) element in $\widehat{\boldsymbol{\delta}}$ is zero.

The penalty parameter λ and the penalty loadings κ_j $j \in \{1, \ldots, p\}$ are known as the lasso tuning parameters. One must choose the lasso tuning parameters before using the lasso for covariate selection. The values of λ and the κ_j determine which covariates will have estimated coefficients that are not zero and which covariates will have estimated coefficients that are zero. Thus, the properties of the lasso-covariate-selection method depend on the method used to choose the tuning parameters.

The most widely used methods for selecting the tuning parameters are CV, plug-in methods, and minimizing an information criterion. We want a method that selects tuning parameters that produce a lasso that finds the important covariates, but does not include too many extra covariates. We discuss the plugin method, the BIC method, and CV for choosing the lasso tuning parameters below.

Belloni, Chen, Chernozhukov, and Hansen (2012), Belloni, Chernozhukov, and Hansen (2014), Belloni, Chernozhukov, and Wei (2016), and Chernozhukov, Chetverikov, Demirer, Duflo, Hansen, Newey, and Robins (2018) derive PO estimators that are robust to the

mistakes that the lasso makes in excluding covariates with small coefficients. Belloni, Chen, Chernozhukov, and Hansen (2012),Belloni, Chernozhukov, and Hansen (2014), and Belloni, Chernozhukov, and Wei (2016), rigorously show that some of these robust methods will perform well when the tuning parameters are selected using their plug-in method. Belloni, Chen, Chernozhukov, and Hansen (2012) derives a plug-in method for a linear model and provides an algorithm to implement it. Belloni, Chernozhukov, and Wei (2016) derives PO estimators for GLMs, but it does not provide a plug-in method or an algorithm for Poisson models.[3]

2.3 Choosing the lasso tuning parameters

We discuss three methods for choosing the lasso tuning parmaters and evaluate the finite-sample performance of PO estimators that use lassos with these methods in section 3. In the statistical literature on the lasso, CV is by far the most discussed method. Heuristically, CV finds the lasso tuning parameters that produce the lasso that predicts best out of sample. CV is a standard technique in nonparametric statistics. We provide the algorithm for completeness.

[3]Belloni, Chernozhukov, and Wei (2016) derives a plug-in method for the tuning parameters of logit models, but this method uses a fixed bound. Belloni, Chernozhukov, and Wei (2016) does not extend the algorithm in Belloni, Chen, Chernozhukov, and Hansen (2012) to the GLM case.

2.3.1 CV algorithm

Algorithm 2: Standard CV

This algorithm assumes that each x_j has been normalized to have mean 0 and variance 1. The penalty loadings $(\tilde{\kappa}_1, \ \ldots, \ \tilde{\kappa}_p)$ are all set to 1.
On exit, λ_{cv} contains the lasso penalty value selected by post-selection CV.

1. Let Λ be a grid of q points over the $(0, \lambda_{max}]$. We use the standard method discussed in Hastie, Tibshirani, and Wainwright (2015).

2. Randomly partition the data into K folds. We use the nearly ubiquitous $K = 10$ in our simulations.

3. For each $j \in \{1, \ldots, q\}$, estimate the out-of-sample deviance using the post-selection lasso estimates produced by setting $\lambda = \lambda_j$.

   (a) Using the data not in fold k, get the penalized Poisson lasso estimates $\widehat{\boldsymbol{\delta}}_{\lambda_j}$ that solve equation (1), when $\lambda = \lambda_j$.

   (b) Using the data in fold k, fill in the values for the observation-level out-of-sample deviance using the penalized estimates $\widehat{\boldsymbol{\delta}}_{\lambda_j}$, denoted by $D(\widehat{\boldsymbol{\delta}}_{\lambda_j})_i$

   (c) The standard CV function for λ_j is the mean of the observation-level contributions to the out-of-sample deviance:

$$CV(\lambda_j) = \frac{1}{n}\sum_{i=1}^{n} D(\widehat{\boldsymbol{\delta}}_{\lambda_j})_i$$

4. Standard CV selects the λ that minimizes the post-selection CV function:

$$\lambda_{cv} = \arg\min_{\lambda_j \in \Lambda}\{CV(\lambda_j)\}$$

2.3.2 Plug-in for generalized linear models

The penalty-loadings play an essential role in defining the penalty level λ that provides a good lasso estimator. Following Belloni, Chernozhukov, and Wei (2016), and Bickel, Ritov, and Tsybakov (2009), the GLM lasso will have good selection properties if

$$P\left(\lambda \geq c \max_{1\leq j\leq p}\left|\frac{1}{n}(1/\kappa_j)\sum_{i=1}^{n}\mathbf{h}_j(y_i, \mathbf{w}_i\boldsymbol{\delta}_0')\right|\right) \to_p 1 \tag{2}$$

where we have the following definitions.

- c is a constant greater than 1.
- $\mathbf{h}_j(y_i, \mathbf{w}_i\boldsymbol{\delta}_0')$ is the contribution of the i(th) observation to the score for the unpenalized QML estimator for the j(th) parameter evaluated at $\boldsymbol{\beta}_0$.

- Each penalty loading has its ideal value

$$\kappa_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{h}_j(y_i, \mathbf{w}_i\boldsymbol{\delta}^{0'})\right]^2}$$

To be more specific about the scores, the vector of scores for the unpenalized QML estimator is

$$\frac{\partial Q(y_i, \mathbf{w}_i\boldsymbol{\delta}')}{\partial \boldsymbol{\delta}} = \mathbf{h}(y_i, \mathbf{w}_i\boldsymbol{\delta}')$$

and the contribution of the i(th) observation to the j(th) score is $\mathbf{h}_j(y_i, \mathbf{w}_i\boldsymbol{\delta}')$, which is the j(th) element of $\mathbf{h}$.

Note that equation (2) can be written as

$$P\left(\lambda \geq c \max_{1\leq j\leq p}\left|\frac{\frac{1}{n}\sum_{i=1}^{n}\mathbf{h}_j(y_i, \mathbf{w}_i\boldsymbol{\delta}_0')}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{h}_j(y_i, \mathbf{w}_i\boldsymbol{\delta}_0')\right]^2}}\right|\right) \to_p 1 \tag{3}$$

In a series of analogous cases, Belloni, Chen, Chernozhukov, and Hansen (2012), Belloni, Chernozhukov, and Hansen (2014), and Belloni, Chernozhukov, and Wei (2016) use the self-normalized moderate deviation theory developed by Jing, Shao, and Wang (2003) and Peña, Lai, and Shao (2009) to show that

$$P\left(\sqrt{n}\max\left|\frac{1}{n}(1/\kappa_j)\sum_{i=1}^{n}\mathbf{h}_j(y_i\mathbf{w}_i\boldsymbol{\delta}_0')\right| \leq \Phi^{-1}(1-\gamma/(2p))\right) \geq 1-\gamma+o(1) \tag{4}$$

under reasonable conditions.

Using equations (2) and (4), Belloni, Chen, Chernozhukov, and Hansen (2012), Belloni, Chernozhukov, and Hansen (2014), and Belloni, Chernozhukov, and Wei (2016) define the ideal value for λ in equation (1) to be

$$\lambda = \frac{c}{\sqrt{n}}\Phi^{-1}\left[1-\gamma/(2p)\right] \tag{5}$$

in a series of analogous cases.

2.3.3 What is the logic behind equation 4

Getting a handle on where equation (4) comes from is essential to understanding the extension in this paper. Belloni, Chen, Chernozhukov, and Hansen (2012) use theorem 7.4 in Peña, Lai, and Shao (2009) to show that

$$P\left(\max_{1\leq j\leq p}|S_j| > \Phi^{-1}(1-\gamma/(2p))\right) \leq \gamma\left(1+A/\ell_n^3\right)$$

where

$$S_j = \frac{\sum_{i=1}^{n} U_{ij}}{\sqrt{\sum_{i=1}^{n} U_{ij}^2}}$$

each U_{ij} is an independent realization of a mean zero random variable.

For small γ, this can be shown to imply that

$$P\left(\max_{1\leq j\leq p} |S_j| \leq \Phi^{-1}(1-\gamma/(2p))\right) \approx 1-\gamma \tag{6}$$

Note that the definition of S_j in equation (6) uses sums over i that are not multiplied by $1/n$ but that the sums over i in equation (3) are multiplied by $1/n$. Extending Belloni, Chen, Chernozhukov, and Hansen (2012) to the GLM case, we let

$$U_{i,j} = \mathbf{h}_j(y_i, \mathbf{w}_i\boldsymbol{\delta}_0')$$

Conceptually, we have

$$\sqrt{n}(1/\kappa_j)1/n\sum_{i=1}^{n}\left[\mathbf{h}_j(y_i, \mathbf{w}_i\boldsymbol{\delta}_0')\right] \tag{7}$$

$$= \sqrt{n}\frac{1/n\sum_{i=1}^{n}\left[\mathbf{h}_j(y_i, \mathbf{w}_i\boldsymbol{\delta}_0')\right]}{\sqrt{1/n\sum_{i=1}^{n}\left[\mathbf{h}_j(y_i, \mathbf{w}_i\boldsymbol{\delta}_0')\right]^2}} \tag{8}$$

$$= \sqrt{n}\frac{1/n\sum_{i=1}^{n}\left[U_{i,j}\right]}{\sqrt{1/n\sum_{i=1}^{n}\left[U_{i,j}\right]^2}} \tag{9}$$

$$= S_j \tag{10}$$

Substituting the expression in (7) in for S_j in equation (6) yields equation (4).

2.3.4 Algorithm for the penalty loadings

We use an extension of the algorithm derived by Belloni, Chen, Chernozhukov, and Hansen (2012) to estimate plug-in values of λ and $\kappa_1, \ \ldots, \ \kappa_p$ that can be used to solve the lasso in equation (1).

Algorithm 3: Plug-in method for GLM lasso tuning parameters

This algorithm assumes that each x_j has been normalized to have mean 0 and variance 1.
On exit, λ contains the penalty value and the penalty loadings are in $(\tilde{\kappa}_1, \ \ldots, \ \tilde{\kappa}_p)$.

1. Set $\gamma = 0.1/\ln(\max(p, n))$ and set $c = 1.1$.
2. Set $\lambda = \frac{c}{\sqrt{n}}\Phi^{-1}\left[1 - \gamma/(2p)\right]$.
3. Find the five covariates that have the highest correlations with y. Denote the vector of them by $\tilde{\mathbf{x}}_0$ and let $\tilde{\mathbf{x}}_{0,i}$ be the i(th) observation of this vector of variables.
4. Estimate the coefficients $\tilde{\boldsymbol{\beta}}_0$ on $\tilde{\mathbf{x}}_0$ by unpenalized GLM QML.
5. For each $j \in \{1, \ldots, p\}$, set
$$\tilde{\kappa}_{0,j} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{h}_j(y_i, \tilde{\mathbf{x}}_{0,i}\tilde{\boldsymbol{\beta}}_0')\right]^2}$$
6. Set $k = 1$ and do the following loop. (It will be executed at most 15 times.)
   (a) Using λ and loadings $\{\tilde{\kappa}_{k-1,1}, \ \ldots, \ \tilde{\kappa}_{k-1,p}\}$ to solve (1) which produces estimates $\tilde{\tilde{\boldsymbol{\beta}}}_k$.
   (b) Let $\tilde{\mathbf{x}}_k$ be the covariates with nonzero coefficients in $\tilde{\tilde{\boldsymbol{\beta}}}_k$.
   (c) Estimate the coefficients $\tilde{\boldsymbol{\beta}}_k$ on $\tilde{\mathbf{x}}_k$ by unpenalized GLM QML.
   (d) For each $j \in \{1, \ldots, p\}$, set
$$\tilde{\kappa}_{k,j} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{h}_j(y_i, \tilde{\mathbf{x}}_{k,i}\tilde{\boldsymbol{\beta}}_k')\right]^2}$$
   where $\tilde{\mathbf{x}}_{k,i}$ is the i(th) observation on $\tilde{\mathbf{x}}_k$.
   (e) Set $k = k + 1$
   (f) If $k > 15$ or the variables in $\tilde{\mathbf{x}}_k$ are the same as those in $\tilde{\mathbf{x}}_{k-1}$ set each $\tilde{\kappa}_j = \tilde{\kappa}_{k,j}$ and exit; else go to step 6a.

2.4 Minimizing the BIC

Following Zhang, Li, and Tsai (2010), we define the degrees of freedom in the BIC to be the number of nonzero coefficient estimates in the GLM lasso for a particular value of λ. Our implementation finds the λ_q in the grid of candidate values Λ that produces the smallest value of

$$BIC = -2\sum_{i=1}^{n} Q(y_i, \tilde{\mathbf{w}}_i\tilde{\boldsymbol{\delta}}) + s_{\lambda_q}\ln(n)$$

where the notation in this equation is as follows.

- $\tilde{\mathbf{w}}_i$ is the i(th) observation on the vector of covariates that have nonzero coefficients in the GLM lasso in equation (1), when λ is set to λ_q and $\kappa_j = 1$.
- s_{λ_q} is the number of covariates in $\tilde{\mathbf{w}}_i$.
- $\sum_{i=1}^{n} Q(y_i, \tilde{\mathbf{w}}_i\tilde{\boldsymbol{\delta}})$ is the value of the unpenalized GLM log likelihood function that includes only the covariates in $\tilde{\mathbf{w}}_i$
- $\tilde{\boldsymbol{\delta}}$ are the coefficients on $\tilde{\mathbf{w}}_i$ that maximize the unpenalized log likelihood function.

2.5 BIC stepwise

As discussed by Belloni and Chernozhukov (2011), the lasso can be viewed a convex approximation to the computationally infeasible problem of finding the subset of covariates that best approximates a conditional expectation function. The family of forward-stepwise methods are another approach to solving this best-subset regression problem. Forward-stepwise methods are computationally feasbile for many HDMs, but they do take much longer than lasso methods and do become infeasible for very high-dimentional problems.

We conjectured that there should exist families of DGPs for which the lasso performs a poor job of covariate selection but a forward-stepwise algorithm performs a good job of covariate selection. For a DGP from one of these families, the lasso-based PO estimator would perform poorly, but the stepwise based PO estimator would perform well. In section 3, we present finite-sample evidence that such a family does exist.

Many implementations of stepwise methods using hypothesis-testing method to find the covariates at each step. Kozbur (2020) formally derives rates of convergence for a testing-based forward-stepwise method for a linear model. These rates could be combined with the approach in Chernozhukov, Hansen, and Spindler (2015) to formally justify a PO estimator for a linear model. As discussed by Kozbur (2020), the testing-based method can get mired in the sticky issue of choosing the level of signficance. By using a BIC-based stepwise algorithm, we avoid the problems of choosing a signficance level.

Algorithm 4 provides the details for the stepwise BIC method.

Algorithm 4: BIC stepwise

1. Define the initial conditions.
   (a) Define the regression model, which can be linear, logit, Poisson, or probit.
   (b) Define the dependent variable as $\mathbf{y}$.
   (c) Define the minimum BIC, $\mathbf{bic_{min}}$, to be a missing value.
   (d) Define the included variables, $\mathbf{x_{in}}$, to be an empty set.
   (e) Define the candidate variables, $\mathbf{x_{out}}$, to have all the potential controls.
2. Repeat the following steps until BIC no longer decreases.
   (a) Get the next potential best BIC, and potential variables to be included.
      i. For each variable $\mathbf{x_j}$ in $\mathbf{x_{out}}$, run a regression of $\mathbf{y}$ on $\mathbf{x_{in}}$ and $\mathbf{x_j}$; denote $\mathbf{bic_{list}}$ as the vector that includes BIC for each regression.
      ii. Choose the minimum BIC in $\mathbf{bic_{list}}$, denote the chosen BIC as $\mathbf{bic_{next}}$, and denote the corresponding selected variable as $\mathbf{x_{next}}$.
   (b) Determine if BIC can decrease anymore.
      i. If $\mathbf{bic_{next}} < \mathbf{bic_{best}}$, update $\mathbf{bic_{best}} = \mathbf{bic_{next}}$, add $\mathbf{x_{next}}$ into $\mathbf{x_{in}}$, and remove $\mathbf{x_{next}}$ from $\mathbf{x_{out}}$. Go back to Step 2a.
      ii. If $\mathbf{bic_{next}} \geq \mathbf{bic_{best}}$, it means BIC cannot be improved any more, exit the loop, and jump to Step 3.
3. The selected variables are $\mathbf{x_{in}}$, and the minimum BIC is $\mathbf{bic_{best}}$.

2.6 PO estimator

Belloni, Chernozhukov, and Wei (2016) derived lasso-based PO estimators for GLM model. We evaluate the finite-sample performance of several versions of their PO estimator for the Poisson regression model. The versions that we implement differ in how they perform the covariate selection steps. We implement three different versions of lasso-based covariate selection. These versions use CV, the plugin, and the BIC to choose the lasso tuning parameters. We also implement a version of their PO estimator that uses BIC-based stepwise for covariate selection.

Algorithm 5 provides details about theses version of the Belloni, Chernozhukov, and Wei (2016) PO estimator.

Algorithm 5: PO Poisson estimation

1. In a Poisson model of y on $\mathbf{d}$ and $\mathbf{x}$, use covariate selection to find the subset of the $\mathbf{x}$ covariates that have nonzero estimated coefficients. Denote this subset by $\widetilde{\mathbf{x}}$.

   - Our results for the plug-in PO Poisson estimator use our version of the plug-in method to select the lasso tuning parameters in this lasso. Similarly, our results for the CV PO Poisson estimator, and the BIC PO Poisson estimator respectively use CV, or minimizing the BIC to select the lasso tuning parameters.
   - For the BIC-Stepwise PO estimator, we find the subset of the $\mathbf{x}$ that BIC-stepwise includes.

2. Use the unpenalized QML Poisson regression estimator to estimate the coefficients $\widetilde{\boldsymbol{\alpha}}$ and $\widetilde{\boldsymbol{\beta}}$ in a Poisson model of y on $\mathbf{d}$ and $\widetilde{\mathbf{x}}$.

3. Let $\widetilde{s}_i = \widetilde{\mathbf{x}}_i\widetilde{\boldsymbol{\beta}}'$ be the ith observation of the predicted value of the linear index $\mathbf{x}\boldsymbol{\beta}'$.

4. Let $\omega_i = G'(\mathbf{d}_i\widetilde{\boldsymbol{\alpha}}' + \widetilde{s}_i)$ be the ith observation of the predicted value of the derivative of $G(\cdot)$.

5. For each $j \in \{1, \ldots, J\}$, use a linear lasso or a linear BIC-stepwise of the jth variable in $\mathbf{d}$ on $\mathbf{x}$ using observation-level weights ω_i, and let $\check{\mathbf{x}}_j$ be the selected covariates.

   - The three lasso-based PO estimators use a weighted lasso for covariate selection.
   - The BIC-stepwise based PO estimator uses a weighted BIC-stepwise for covariate selection.

6. For each $j \in \{1, \ldots, J\}$, run a linear, ordinary least squares regression of the jth variable in $\mathbf{d}$ on $\check{\mathbf{x}}_j$ with observation-level weights ω_i. Let $\widetilde{d}_j$ be the unweighted residuals from this regression and let $\widetilde{d}_{j,i}$ be the ith observation on $\widetilde{d}_j$.

7. Create the vector instrumental variables $\mathbf{z} = (\widetilde{d}_1, \ldots, \widetilde{d}_J)$ and $\mathbf{z}_i$ be the ith observation on this vector of instrumental variables. Note that $\mathbf{z}_i = (z_{1,i}, \ldots, z_{J,i}) = (\widetilde{d}_{1,i}, \ldots, \widetilde{d}_{J,i})$.

8. Compute $\widehat{\boldsymbol{\alpha}}$ by solving the J sample-moment equations

$$\frac{1}{n}\sum_{i=1}^{n}\left[y_i - G(\mathbf{d}_i\boldsymbol{\alpha}' + \widetilde{s}_i)\right]\mathbf{z}_i = \mathbf{0}$$

   We use the standard robust estimator for the asymptotic variance of a method-of-moments estimator.

3 Simulation results

This section outlines our simulation designs and summarizes our simulation results. The details for the designs are in section 3.1. The tables with the simulation results are in section 5. We use three designs of DPG's, we denote them by `tpos`, `parm`, and `alt`. All of them include some small coefficients and some large covariates. All of them cause the naive estimators to fail. The designs use different DGPs for how y depends on $\mathbf{d}$ and $\mathbf{x}$ and for how $\mathbf{d}$ depends on $\mathbf{x}$.

While we discuss the details in section 3.1, we briefly describe the structures here. `tpos` is an adaptation of designs used in Belloni, Chernozhukov, and Wei (2016). This is the easiest design. The $\mathbf{d}$ and the $\mathbf{x}$ have a Toeplitz design. The plugin-lasso tends to do exceptionally well at the finding the $\mathbf{x}$ that are related to the $\mathbf{d}$ in this type of design.

The `parm` design is more challenging. In the `parm` design, we generated the $\mathbf{x}$ from a Toeplitz design but then generated the $\mathbf{d}$ using linear parametric models from some of the $\mathbf{x}$ variables. In these models, some of the $\mathbf{x}$ have large coefficients, some have small coefficients, and the vast majority have zero coefficients, just like in the model for y.

The `alt` design is a version of the `parm` design that has cofficients that alternate in sign. We originally constructed this design as way of allowing for many covariates that would not cause the R-squared of the regression to increase as we added covariates. In finite samples, this design causes the plugin method to select a λ that is close to the values that will exclude all covariates. None the lasso-based PO estimators perform acceptably on this design, but the BIC-stepwise-based PO estimator does perform well on it.

Here are the most important results of simulations.

First, the `alt` design is an example of a DGP for which, in finite samples, the lasso-based PO estimators fail, but BIC-stepwise-based PO estimators perform well. In our simulations, all the lasso-based PO estimators perform unacceptably poorly on the `alt` designs. In contrast, the BIC-stepwise PO estimator performs well on the `alt` designs.

Second, BIC-stepwise-based PO estimators perform about as well as the BIC-lasso-based PO estimator and the plugin-lasso-based PO estimator in the `parm` and the `tpos` designs.

Third, we note that the BIC-stepwise PO estimates take an extraordinarily long time to compute. Clearly, there are datasets for which the lasso-based PO estimator is computationally feasible but the BIC-stepwise PO estimator is not. The good performance of the BIC-stepwise PO estimator comes at the prices of a massive increase in computational time. For example, where there are $1,000$ observations and $1,000$ parameters , it takes SWPO 80 minutes to converge, while it only takes plugin-lasso-based PO 3 seconds.

Fourth, in the designs that do not defeat the lasso in finite samples, the results indicate that our extension of the Belloni, Chen, Chernozhukov, and Hansen (2012) plugin method for the lasso tuning parameters performs well. Our simulation results are for the Poisson QML model, but we have written the formulas so that it is trivial to extend the method to GLM model.

Fifth, in our results the CV-lasso-based PO estimator performed worse than the plugin-lasso-based PO estimator or the BIC-lasso-based PO estimator. We attribute its poor performance to its tendency to include many covariates whose coefficients are zero. These results can also be viewed as supporting the theoretical results of Chetverikov, Liao, and Chernozhukov (2020) who found that the CV-lasso converges at a slower rate than the

plugin-lasso.

Sixth, we note that the BIC-lasso PO estimator performs as well as the plugin-lasso based PO-estimator. We know of no results for the rate of convergence or the sparsity bound of the BIC-lasso. Our results indicate that these formal results could be very useful.

3.1 Details about the DGPs

In this section, we discuss the details of how the DGPs were constructed.

The three designs are `tpos`, `parm`, and `alt`. In each case, y_i is drawn from a Poisson distribution with mean m_i. In each case

$$m_i = \alpha_1 d_{1,i} + \alpha_2 d_{2,i} + 0 d_{3,i} + \beta_b' \mathbf{x}_{b,i} + \beta_s' \mathbf{x}_{s,i}$$

where

- $d_{1,i}$ is the i(th) observation on the first covariate of interest
- α_1 is the coefficient on d_1
- $d_{2,i}$ is the i(th) observation on the second covariate of interest
- α_2 is the coefficient on d_2
- $d_{3,i}$ is the i(th) observation on the third covariate of interest
- $\mathbf{x}_{b,i}$ is the i(th) observation on the vector of covariates with large coefficients
- β_b is the vector of large coefficients
- $\mathbf{x}_{s,i}$ is the i(th) observation on the vector of covariates with small coefficients
- β_s is the vector of small coefficients

The designs vary how the d variables and x variables are generated. The values of the coefficients vary over the designs and the cases within the designs.

In all cases, there 7 covariates in $\mathbf{x}_b$ and 7 covariates in xb_s.

There are p covariates $x_1, \ldots, x_p$. In all cases, the x variables are generated from a Toeplitz structure. For speed, we used an autogressive structure over the covariates. The i(th) observation on covariate j is generated as

$$x_{j,i} = .5x_{j-1,i} + .2x_{j-4} + .1x_{j-8,i} + .1x_{j-10,i} + \eta_i$$

where η_i is $\chi^2(20)$ with its mean of 20 removed. We used this distribution because it is distinctly not normal, but the right tail is not too long. We generated and discarded 10 burn-in covariates to set up the autoregressive structure over the covariates.

The `tpos` design used the following structure, which is similar to designs used by Belloni, Chernozhukov, and Wei (2016).

- d_1, d_2, d_3 are the first 3 of the x variables.

- The next 7 x variables are the covariates with the large coefficients.
- The next 7 x variables are the covariates with the small coefficients.

All the small coefficients have the same value. This value was set to about about twice the standard error of coefficient in the true model for y given d_1, d_2, d_3 and the x covariates. The precise values are in the results tables.

All the large coefficients have the same value. This value was set to about about twice the small coefficient value. The precise values are in the results tables.

In this structure, the plugin lasso does an exceptional job of finding the few covariates that are sufficiently correlated with each d_j. Even when the plugin lasso does not include covariates with large coefficients that affect y, removing all the covariates related to each d_j creates a reliable PO estimator. For these reasons, the plugin lasso does exceptionally well in this case.

Instead of using the simple Toeplitz structure for each d_j, the `parm` design uses a linear parametric model to create each d_j from the x variables with large coefficients and with small coefficients in the equation for y. The linear model for each d_j has large coefficients on the covariates that have large coefficients in the equation for y and it has small coefficients on the covariates that have small coefficients in the equation for y.

In the equation for each d_j, each of the small coefficients is set to about twice the standard error of coefficient in the true model for that d_j. The large coefficients are set to twice the small coefficients. The precise values are in the results tables.

The `alt` structure is like the `parm` structure, except that the coefficients alternate in sign. This structure has the advantage of keeping the R^2 fixed, when adding many covariates with the same sized coefficients. The `alt` is extremely challenging for lasso-based covariate selection. In a linear model, in finite samples, the plugin selects tuning parameters that will include zero covariates. Our simulations indicate that a similar result holds for GLM models. To distinguish the BIC-based lasso from the plugin-based lasso, we used values for the small coeficients that are larger than twice their standard errors in the true model. The large coefficients are still set to twice the small coefficient value.

4 Conclusions and further research

This paper has made four contributions to the rapidly evolving literture on PO estimators for the parameters of interest. First, it presented simulation evidence that there are DGPs for which a BIC-stepwise version of the Belloni, Chernozhukov, and Wei (2016) PO estimator performs well but the usual lasso-based PO estimators fail. Second, this paper the extended the Belloni, Chen, Chernozhukov, and Hansen (2012) algorithm for selecting the lasso tuning parameters to the Poisson model and it presented simulation evidence that a version of the Belloni, Chernozhukov, and Wei (2016) PO estimator that uses a Poisson lasso with these tuning parameters performs well. Third, this paper presented simulation evidence that selecting the lasso tuning parameters using the BIC, produces a version of the Belloni, Chernozhukov, and Wei (2016) PO estimator that performs about as well as the plugin estimator. Fourth, this paper presented simulation evidence that selecting the lasso tuning parameters using cross-validation, produces a version of the Belloni, Chernozhukov, and Wei

(2016) PO estimator that performs worse than the PO estimator that uses the plugin-based lasso, the BIC-based lasso, or BIC-based stepwise.

There are no formal results for the exact PO estimator discussed in this paper. It seems clear that the approach of Belloni, Chernozhukov, and Wei (2016) could be extended to cover the plugin-based lasso for the Poisson model. We do not know of any formal results for a BIC-based lasso that could be used with the approach of Chernozhukov, Hansen, and Spindler (2015) to justify the BIC-lasso PO estimator. We do not know of any formal results for BIC-stepwise selection that could be used with the approach of Chernozhukov, Hansen, and Spindler (2015) to justify the BIC-stepwise PO estimator. The simulation results in this paper motivate working on these formal results.

5 Tables

This section contains the tables of results from our simulations.

Here we review the notation used in the tables.

The designs `tpos`, `parm` and `alt` were defined in section 3.1.

The methods are the estimators.

method	estimator
bic	PO estimator using the BIC to select the lasso tuning parameters
cv	PO estimator using CV to select the lasso tuning parameters
plugin	PO estimator using the plugin to select the lasso tuning parameters
swpo	PO estimator using the BIC-based stepwise algorithm to select covariates
bicnaive	Naive estimator using the BIC to select the lasso tuning parameters
cvnaive	Naive estimator using CV to select the lasso tuning parameters
pluginnaive	Naive estimator using the plugin to select the lasso tuning parameters
swbicnaive	Naive estimator using the BIC-based stepwise algorithm to select covariates
true	Poisson QML estimator using the true structure used to generate the data

We used two sample sizes, n=500 and n=1,000. For n=500, we considered the case with p=250 and with p=500 x covariates. For n=1,000, we considered the case with p=500 and with p=1,000 x covariates.

We report results for 3 coefficients. In each case, the coefficient on d_1 was a large coefficient. In each case, the coefficient on d_2 was a small coefficient. The coefficient on d_3 was always zero. The summary tables report the precise values of large and small coefficients in each case.

5.1 Rejection rate tables

This section contains the results for the rejection rate of a Wald test against the true null hypothesis at the .05 significance level. Ideally, each rejection rate should be .05.

5.1.1 Rejection rate results for tpos design

The rejection rates for bic, plugin, and swpo are acceptable, or nearly so for all cases. That bic and swpo perform as well as plugin indicates that more theoretical work should be done

to justify these methods. The formal theoretical work by Belloni, Chen, Chernozhukov, and Hansen (2012) and Belloni, Chernozhukov, and Wei (2016) justifies the plugin method.

It is interesting how close some of the naive methods are to producing acceptable results. All the naive methods fail dramatically in the other designs.

Design tpos, rejection rate for d1					
		n	500	500	1000	1000
method	obs	p	250	500	500	1000
bic	2030		0.0571	0.0626	0.0500	0.0537
cv	2030		0.1084	0.1114	0.0793	0.0887
plugin	2026		0.0721	0.0712	0.0568	0.0597
swpo	2030		0.0665	0.0665	0.0582	0.0665
bicnaive	2030		0.0768	0.0852	0.0556	0.0690
cvnaive	2030		0.0709	0.0739	0.0675	0.0744
pluginnaive	2026		0.1273	0.1448	0.0717	0.0845
swbicnaive	2030		0.1000	0.1084	0.0732	0.0906
true	2030		0.0433	0.0517	0.0464	0.0483

Design tpos, rejection rate for d2					
		n	500	500	1000	1000
method	obs	p	250	500	500	1000
bic	2030		0.0527	0.0601	0.0551	0.0502
cv	2030		0.0921	0.0971	0.0773	0.0734
plugin	2026		0.0775	0.0657	0.0614	0.0543
swpo	2030		0.0596	0.0645	0.0520	0.0557
bicnaive	2030		0.0813	0.0833	0.0562	0.0488
cvnaive	2030		0.0655	0.0778	0.0618	0.0675
pluginnaive	2026		0.1002	0.1043	0.0604	0.0627
swbicnaive	2030		0.1020	0.1143	0.0747	0.0685
true	2030		0.0522	0.0493	0.0433	0.0473

Design tpos, rejection rate for d3					
		n	500	500	1000	1000
method	obs	p	250	500	500	1000
bic	2030		0.0547	0.0532	0.0541	0.0571
cv	2030		0.0862	0.0922	0.0726	0.0798
plugin	2026		0.0696	0.0727	0.0666	0.0607
swpo	2030		0.0522	0.0611	0.0546	0.0502
bicnaive	2030		0.2714	0.2719	0.1824	0.1892
cvnaive	2030		0.0778	0.1025	0.0701	0.0847
pluginnaive	2026		0.3356	0.3643	0.2663	0.2890
swbicnaive	2030		0.1852	0.2099	0.1108	0.1310

5.1.2 Rejection rate results for parm design

When $n = 500$, plugin, bic, and swpo all under reject. However, when $n = 1000$, these three estimators perform well. We conclude that the under-rejection is due to the smaller sample size. In contrast, CV has a tendency to over reject regardless of the sample size. Note that all the naive estimators performs poorly.

Design parm, rejection rate for d1					
	n	500	500	1000	1000
method	obs p	250	500	500	1000
bic	2030	0.0340	0.0317	0.0468	0.0429
cv	2030	0.1059	0.1135	0.0857	0.0827
plugin	1868	0.0369	0.0363	0.0561	0.0512
swpo	2030	0.0261	0.0256	0.0488	0.0471
bicnaive	1808	0.7511	0.8994	0.2491	0.3015
cvnaive	1808	0.4801	0.7402	0.1315	0.1774
pluginnaive	1889	0.9995	1.0000	1.0000	1.0000
swbicnaive	2030	0.4222	0.4931	0.1576	0.1820
true	2030	0.0571	0.0576	0.0542	0.0402

Design parm, rejection rate for d2					
	n	500	500	1000	1000
method	obs p	250	500	500	1000
bic	2030	0.0310	0.0346	0.0463	0.0386
cv	2030	0.1034	0.1065	0.0872	0.0795
plugin	1868	0.0332	0.0330	0.0585	0.0587
swpo	2030	0.0227	0.0335	0.0399	0.0423
bicnaive	1808	0.7611	0.9029	0.2590	0.2962
cvnaive	1808	0.4801	0.7379	0.1310	0.1684
pluginnaive	1889	1.0000	0.9995	1.0000	1.0000
swbicnaive	2030	0.4034	0.5089	0.1665	0.1730
true	2030	0.0532	0.0493	0.0522	0.0545

Design parm, rejection rate for d3					
	n	500	500	1000	1000
method	obs p	250	500	500	1000
bic	2030	0.0355	0.0332	0.0448	0.0545
cv	2030	0.0980	0.0963	0.0759	0.0827
plugin	1868	0.0391	0.0369	0.0575	0.0592
swpo	2030	0.0167	0.0286	0.0507	0.0508
bicnaive	1808	0.7550	0.9251	0.2422	0.3074
cvnaive	1808	0.4939	0.7624	0.1275	0.1849
pluginnaive	1889	1.0000	1.0000	1.0000	1.0000
swbicnaive	2030	0.4291	0.5089	0.1522	0.1884

5.1.3 Rejection rate results for alt design

While swpo has a small tendency to over reject, it performs either acceptably or close to acceptably. All the other methods fail dramatically.

Design alt, rejection rate for d1					
		n	500	500	1000	1000
method	obs	p	250	500	500	1000
bic	2030		0.2936	0.6509	0.1251	0.2344
cv	2030		0.7384	0.9601	0.2153	0.4180
plugin	1990		0.9593	0.9699	0.9828	0.9889
swpo	2030		0.0724	0.0995	0.0645	0.0704
bicnaive	1961		0.9092	0.9958	0.9360	0.9937
cvnaive	1961		0.9164	0.9953	0.6823	0.9503
pluginnaive	1981		0.9985	1.0000	1.0000	1.0000
swbicnaive	2030		0.2148	0.4852	0.3113	0.3974
true	2030		0.0409	0.0512	0.0483	0.0508

Design alt, rejection rate for d2					
		n	500	500	1000	1000
method	obs	p	250	500	500	1000
bic	2030		0.2557	0.6454	0.1355	0.2206
cv	2030		0.7039	0.9607	0.1700	0.3185
plugin	1990		0.9724	0.9831	0.9916	0.9921
swpo	2030		0.0675	0.1059	0.0704	0.0693
bicnaive	1961		0.9153	0.9990	0.9379	0.9915
cvnaive	1961		0.9072	0.9974	0.6906	0.9481
pluginnaive	1981		1.0000	0.9995	1.0000	1.0000
swbicnaive	2030		0.2148	0.5034	0.3197	0.4000
true	2030		0.0453	0.0409	0.0532	0.0402

Design alt, rejection rate for d3					
		n	500	500	1000	1000
method	obs	p	250	500	500	1000
bic	2030		0.2527	0.6010	0.1202	0.1974
cv	2030		0.7000	0.9532	0.1330	0.2386
plugin	1990		0.9809	0.9893	0.9911	0.9952
swpo	2030		0.0660	0.0951	0.0542	0.0757
bicnaive	1961		0.9286	0.9984	0.9389	0.9921
cvnaive	1961		0.9210	0.9969	0.6833	0.9571
pluginnaive	1981		1.0000	1.0000	1.0000	1.0000
swbicnaive	2030		0.2217	0.4970	0.3094	0.4095

5.2 Missing/Found tables

The tables that describe the selection ability of each method are in this section. Each table contains the results for one of the equations used by each PO estimator in a particular design.

The "Mean large missed" statistic is the sample mean number of variables with large coefficients that were missed by the method for that case. In all cases, 0 would be perfect covariate selection for the large-coefficient variables and 7 would be complete failure. The smaller the sample means, the better the selection ability.

The "Mean small missed" statistic is the sample mean number of variables with small coefficients that were missed by the method for that case. In all cases, 0 would be perfect covariate selection for the small-coefficient variables and 7 would complete failure. The smaller the sample means, the better the selection ability.

The "Mean extra found" statistic is the sample mean number of variables with zero coefficients that were included by the method for that case. In all cases, 0 would be best. The smaller the sample mean, the better the selection ability.

For the `parm` and `alt` design, there is a parametric structure that allows us to calculate the results for missing and extra covariates, in the equations for d_1, d_2 and d_3. There is no such structure for `tpos` case, so the missing and extra results are only reported for the equation for y that does have a parametric struture.

5.2.1 Missing and found results for tpos design

plugin performs the best in this design. cv has fewer misses, but a tendency to include way too way extra variables. bic and swpo have more misses than plugin, but do not include high numbers of extra variables.

Design tpos, Missing/found results for y					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	Mean large missed	0.807	0.842	0.161	0.152
bic	Mean small missed	3.196	3.207	2.135	2.077
bic	Mean extra included	0.264	0.302	0.217	0.351
cv	Mean large missed	0.047	0.072	0.002	0.003
cv	Mean small missed	1.044	1.099	0.362	0.388
cv	Mean extra included	11.319	13.724	13.597	15.373
plugin	Mean large missed	0.261	0.309	0.029	0.030
plugin	Mean small missed	1.567	1.667	0.642	0.636
plugin	Mean extra included	1.532	1.521	1.420	1.324
swpo	Mean large missed	1.970	2.098	0.643	0.709
swpo	Mean small missed	4.308	4.360	3.444	3.493
swpo	Mean extra included	1.918	3.370	2.436	4.285

5.2.2 Missing and found results for parm design

bic seems to perform best in this design, but plugin performs almost as well. CV again includes too many extra variables. swpo has more misses than bic or plugin, but does a

good job at not including extra variables.

Design parm, Missing/found results for y					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	Mean large missed	0.712	0.672	0.354	0.329
bic	Mean small missed	2.785	2.762	2.429	2.383
bic	Mean extra included	0.985	1.146	0.373	0.660
cv	Mean large missed	0.016	0.025	0.003	0.006
cv	Mean small missed	0.643	0.773	0.405	0.445
cv	Mean extra included	15.089	16.714	14.130	16.311
plugin	Mean large missed	0.713	0.871	0.083	0.118
plugin	Mean small missed	2.217	2.340	0.891	1.003
plugin	Mean extra included	2.180	2.476	1.472	1.420
swpo	Mean large missed	1.776	2.155	0.839	1.023
swpo	Mean small missed	4.153	4.384	3.638	3.759
swpo	Mean extra included	2.764	4.412	2.442	4.422

Design parm, Missing/found results for d1					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	Mean large missed	0.301	0.311	0.167	0.165
bic	Mean small missed	1.795	1.837	1.563	1.524
bic	Mean extra included	28.663	32.660	33.141	40.148
cv	Mean large missed	0.367	0.398	0.212	0.224
cv	Mean small missed	1.865	1.913	1.642	1.608
cv	Mean extra included	21.169	23.981	22.682	27.128
plugin	Mean large missed	0.577	0.562	0.288	0.277
plugin	Mean small missed	2.322	2.275	1.869	1.794
plugin	Mean extra included	6.346	9.349	6.538	9.584
swpo	Mean large missed	1.178	1.156	1.187	1.160
swpo	Mean small missed	3.383	3.329	3.577	3.504
swpo	Mean extra included	27.938	44.719	34.022	54.565

Design parm, Missing/found results for d2					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	Mean large missed	0.272	0.286	0.167	0.165
bic	Mean small missed	1.746	1.758	1.501	1.492
bic	Mean extra included	28.224	31.554	31.521	37.611
cv	Mean large missed	0.345	0.351	0.201	0.215
cv	Mean small missed	1.778	1.789	1.544	1.563
cv	Mean extra included	20.995	23.691	21.570	26.953
plugin	Mean large missed	0.520	0.509	0.259	0.246
plugin	Mean small missed	2.199	2.177	1.739	1.780
plugin	Mean extra included	6.041	8.796	6.057	8.770
swpo	Mean large missed	1.108	1.070	1.134	1.135
swpo	Mean small missed	3.298	3.313	3.520	3.451
swpo	Mean extra included	27.703	44.529	32.934	52.377

Design parm, Missing/found results for d3					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	Mean large missed	0.248	0.255	0.139	0.140
bic	Mean small missed	1.650	1.694	1.407	1.449
bic	Mean extra included	26.840	30.288	30.439	35.466
cv	Mean large missed	0.303	0.311	0.174	0.178
cv	Mean small missed	1.733	1.748	1.455	1.497
cv	Mean extra included	20.465	24.011	21.444	27.440
plugin	Mean large missed	0.476	0.468	0.234	0.224
plugin	Mean small missed	2.136	2.143	1.675	1.669
plugin	Mean extra included	5.756	8.493	5.733	8.363
swpo	Mean large missed	1.084	1.043	1.060	1.009
swpo	Mean small missed	3.276	3.271	3.473	3.451
swpo	Mean extra included	26.901	43.224	31.714	51.588

5.2.3 Missing and found results for alt design

The mean misses for plugin in greater than 5 out of 7 variables in almost all cases. swpo has the fewest misses, and does very well in the equation for y, although it does include many extra variables in the equations for d_1, d_2, and d_3. The poor performance of bic and cv also stands out.

Design alt, Missing/found results for y					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	Mean large missed	0.919	2.967	2.254	3.745
bic	Mean small missed	3.761	5.452	5.610	6.259
bic	Mean extra included	21.733	20.749	9.410	8.404
cv	Mean large missed	1.597	3.647	0.270	1.629
cv	Mean small missed	2.227	5.193	1.981	4.153
cv	Mean extra included	52.244	28.183	68.773	59.549
plugin	Mean large missed	6.306	6.407	6.308	6.382
plugin	Mean small missed	6.861	6.905	6.989	6.986
plugin	Mean extra included	0.861	1.163	0.009	0.005
swpo	Mean large missed	0.383	1.389	0.615	1.042
swpo	Mean small missed	0.867	2.526	3.629	4.156
swpo	Mean extra included	7.081	12.551	3.469	6.595

Design alt, Missing/found results for d1					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	Mean large missed	1.312	2.451	0.242	0.556
bic	Mean small missed	3.503	4.734	3.724	4.415
bic	Mean extra included	44.634	47.839	30.193	40.277
cv	Mean large missed	6.045	6.479	1.030	2.390
cv	Mean small missed	6.515	6.772	2.640	4.194
cv	Mean extra included	6.437	4.709	68.979	63.764
plugin	Mean large missed	5.383	5.642	4.976	5.187
plugin	Mean small missed	6.257	6.420	6.562	6.648
plugin	Mean extra included	9.843	14.461	3.014	4.733
swpo	Mean large missed	0.692	0.801	0.027	0.016
swpo	Mean small missed	3.487	3.773	2.688	2.824
swpo	Mean extra included	34.004	53.441	23.154	40.224

Design alt, Missing/found results for d2					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	Mean large missed	0.892	2.110	0.153	0.390
bic	Mean small missed	3.294	4.613	3.613	4.350
bic	Mean extra included	43.490	46.081	25.828	34.750
cv	Mean large missed	5.776	6.314	0.462	1.388
cv	Mean small missed	6.339	6.701	1.862	3.222
cv	Mean extra included	8.517	5.893	75.921	84.086
plugin	Mean large missed	5.327	5.561	5.006	5.159
plugin	Mean small missed	6.299	6.424	6.579	6.670
plugin	Mean extra included	8.719	12.762	1.916	3.080
swpo	Mean large missed	0.351	0.426	0.006	0.004
swpo	Mean small missed	3.223	3.463	2.386	2.524
swpo	Mean extra included	32.006	51.779	19.757	35.576

Design alt, Missing/found results for d3					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	Mean large missed	0.669	1.752	0.081	0.249
bic	Mean small missed	3.043	4.370	3.513	4.267
bic	Mean extra included	41.950	44.754	22.024	29.519
cv	Mean large missed	5.283	6.102	0.149	0.658
cv	Mean small missed	6.021	6.586	1.284	2.358
cv	Mean extra included	11.965	7.435	80.775	97.883
plugin	Mean large missed	5.211	5.505	4.953	5.125
plugin	Mean small missed	6.284	6.409	6.649	6.705
plugin	Mean extra included	7.705	11.431	1.287	1.983
swpo	Mean large missed	0.167	0.229	0.006	0.002
swpo	Mean small missed	2.717	3.001	2.074	2.214
swpo	Mean extra included	30.284	49.314	17.423	31.953

5.3 Summary results for estimates

The tables in this subsection report the following results for each estimator, design, and case.

stat	result
true	the true coefficient value
mean	the mean of the estimated coefficients
SD	the standard deviation of the estimated coefficients
SE	the mean of the standard errors

The close the mean is to true value, the better the estimator. The closer the SE is to the SD, the more reliable the inference.

In each design, the true value on d_1 is the value of a large coefficient in that design. In each design, the true value on d_2 is the value of a small coefficient in that design.

5.3.1 Summary results for tpos design

Design tpos, summary for d1					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	true	0.100	0.100	0.100	0.100
bic	mean	0.099	0.100	0.100	0.101
bic	SD	0.040	0.043	0.027	0.028
bic	SE	0.039	0.041	0.027	0.028
cv	true	0.100	0.100	0.100	0.100
cv	mean	0.101	0.100	0.100	0.101
cv	SD	0.038	0.040	0.024	0.026
cv	SE	0.032	0.033	0.022	0.023
plugin	true	0.100	0.100	0.100	0.100
plugin	mean	0.101	0.101	0.100	0.101
plugin	SD	0.036	0.036	0.023	0.024
plugin	SE	0.032	0.033	0.023	0.023
swpo	true	0.100	0.100	0.100	0.100
swpo	mean	0.100	0.100	0.100	0.100
swpo	SD	0.045	0.053	0.029	0.035
swpo	SE	0.042	0.050	0.029	0.033
bicnaive	true	0.100	0.100	0.100	0.100
bicnaive	mean	0.107	0.107	0.102	0.103
bicnaive	SD	0.035	0.036	0.022	0.023
bicnaive	SE	0.032	0.032	0.022	0.022
cvnaive	true	0.100	0.100	0.100	0.100
cvnaive	mean	0.104	0.105	0.102	0.104
cvnaive	SD	0.037	0.038	0.024	0.025
cvnaive	SE	0.034	0.034	0.023	0.023
pluginnaive	true	0.100	0.100	0.100	0.100
pluginnaive	mean	0.114	0.116	0.103	0.105
pluginnaive	SD	0.050	0.052	0.029	0.027
pluginnaive	SE	0.032	0.032	0.022	0.022
swbicnaive	true	0.100	0.100	0.100	0.100
swbicnaive	mean	0.108	0.108	0.103	0.105
swbicnaive	SD	0.036	0.038	0.024	0.025
swbicnaive	SE	0.032	0.032	0.022	0.022
true	true	0.100	0.100	0.100	0.100
true	mean	0.101	0.101	0.100	0.101
true	SD	0.032	0.033	0.021	0.022
true	SE	0.032	0.032	0.022	0.022

Design tpos, summary for d2					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	true	0.050	0.050	0.050	0.050
bic	mean	0.050	0.050	0.049	0.050
bic	SD	0.045	0.048	0.031	0.032
bic	SE	0.044	0.046	0.030	0.032
cv	true	0.050	0.050	0.050	0.050
cv	mean	0.051	0.051	0.050	0.049
cv	SD	0.043	0.045	0.028	0.029
cv	SE	0.037	0.038	0.025	0.026
plugin	true	0.050	0.050	0.050	0.050
plugin	mean	0.051	0.050	0.050	0.050
plugin	SD	0.039	0.041	0.026	0.027
plugin	SE	0.036	0.037	0.025	0.026
swpo	true	0.050	0.050	0.050	0.050
swpo	mean	0.050	0.050	0.049	0.050
swpo	SD	0.050	0.061	0.034	0.039
swpo	SE	0.048	0.058	0.033	0.038
bicnaive	true	0.050	0.050	0.050	0.050
bicnaive	mean	0.053	0.053	0.051	0.051
bicnaive	SD	0.039	0.040	0.025	0.025
bicnaive	SE	0.035	0.035	0.024	0.024
cvnaive	true	0.050	0.050	0.050	0.050
cvnaive	mean	0.052	0.053	0.051	0.051
cvnaive	SD	0.041	0.042	0.027	0.027
cvnaive	SE	0.038	0.038	0.025	0.025
pluginnaive	true	0.050	0.050	0.050	0.050
pluginnaive	mean	0.057	0.056	0.051	0.051
pluginnaive	SD	0.048	0.046	0.028	0.028
pluginnaive	SE	0.035	0.035	0.025	0.024
swbicnaive	true	0.050	0.050	0.050	0.050
swbicnaive	mean	0.054	0.054	0.051	0.051
swbicnaive	SD	0.041	0.043	0.027	0.027
swbicnaive	SE	0.035	0.035	0.024	0.024
true	true	0.050	0.050	0.050	0.050
true	mean	0.050	0.050	0.050	0.050
true	SD	0.034	0.034	0.022	0.022
true	SE	0.033	0.033	0.023	0.023

Design tpos, summary for d3					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	true	0.000	0.000	0.000	0.000
bic	mean	-0.001	0.000	0.000	-0.001
bic	SD	0.044	0.046	0.029	0.032
bic	SE	0.043	0.045	0.029	0.031
cv	true	0.000	0.000	0.000	0.000
cv	mean	0.002	0.003	0.002	0.001
cv	SD	0.041	0.043	0.027	0.028
cv	SE	0.036	0.037	0.025	0.025
plugin	true	0.000	0.000	0.000	0.000
plugin	mean	0.000	0.001	0.001	0.000
plugin	SD	0.038	0.039	0.026	0.027
plugin	SE	0.035	0.037	0.025	0.026
swpo	true	0.000	0.000	0.000	0.000
swpo	mean	-0.001	0.000	0.001	-0.000
swpo	SD	0.047	0.057	0.032	0.037
swpo	SE	0.046	0.055	0.032	0.036
bicnaive	true	0.000	0.000	0.000	0.000
bicnaive	mean	0.031	0.032	0.014	0.012
bicnaive	SD	0.046	0.046	0.032	0.032
bicnaive	SE	0.033	0.033	0.024	0.024
cvnaive	true	0.000	0.000	0.000	0.000
cvnaive	mean	0.007	0.013	0.006	0.007
cvnaive	SD	0.041	0.043	0.027	0.028
cvnaive	SE	0.037	0.038	0.025	0.025
pluginnaive	true	0.000	0.000	0.000	0.000
pluginnaive	mean	0.046	0.051	0.022	0.023
pluginnaive	SD	0.062	0.062	0.041	0.039
pluginnaive	SE	0.033	0.033	0.024	0.023
swbicnaive	true	0.000	0.000	0.000	0.000
swbicnaive	mean	0.016	0.019	0.007	0.006
swbicnaive	SD	0.046	0.047	0.029	0.030
swbicnaive	SE	0.034	0.034	0.024	0.024

5.3.2 Summary results for parm design

Design parm, summary for d1					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	true	0.120	0.120	0.100	0.100
bic	mean	0.117	0.118	0.099	0.100
bic	SD	0.040	0.041	0.027	0.028
bic	SE	0.049	0.049	0.028	0.029
cv	true	0.120	0.120	0.100	0.100
cv	mean	0.126	0.127	0.102	0.104
cv	SD	0.036	0.037	0.026	0.026
cv	SE	0.031	0.032	0.023	0.024
plugin	true	0.120	0.120	0.100	0.100
plugin	mean	0.116	0.115	0.100	0.101
plugin	SD	0.042	0.042	0.024	0.024
plugin	SE	0.054	0.055	0.023	0.024
swpo	true	0.120	0.120	0.100	0.100
swpo	mean	0.113	0.115	0.100	0.100
swpo	SD	0.049	0.056	0.029	0.033
swpo	SE	0.064	0.068	0.030	0.034
bicnaive	true	0.120	0.120	0.100	0.100
bicnaive	mean	0.203	0.229	0.122	0.128
bicnaive	SD	0.047	0.046	0.028	0.028
bicnaive	SE	0.024	0.023	0.022	0.022
cvnaive	true	0.120	0.120	0.100	0.100
cvnaive	mean	0.192	0.233	0.114	0.123
cvnaive	SD	0.070	0.069	0.027	0.028
cvnaive	SE	0.028	0.026	0.024	0.024
pluginnaive	true	0.120	0.120	0.100	0.100
pluginnaive	mean	0.299	0.297	0.297	0.298
pluginnaive	SD	0.039	0.040	0.028	0.027
pluginnaive	SE	0.020	0.020	0.017	0.017
swbicnaive	true	0.120	0.120	0.100	0.100
swbicnaive	mean	0.161	0.168	0.114	0.118
swbicnaive	SD	0.041	0.043	0.026	0.027
swbicnaive	SE	0.025	0.025	0.022	0.022
true	true	0.120	0.120	0.100	0.100
true	mean	0.120	0.119	0.099	0.101
true	SD	0.027	0.027	0.022	0.022
true	SE	0.027	0.027	0.022	0.022

Design parm, summary for d2					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	true	0.060	0.060	0.050	0.050
bic	mean	0.058	0.058	0.050	0.049
bic	SD	0.041	0.042	0.027	0.027
bic	SE	0.050	0.049	0.028	0.029
cv	true	0.060	0.060	0.050	0.050
cv	mean	0.067	0.068	0.053	0.053
cv	SD	0.036	0.037	0.026	0.026
cv	SE	0.032	0.033	0.023	0.024
plugin	true	0.060	0.060	0.050	0.050
plugin	mean	0.055	0.056	0.051	0.050
plugin	SD	0.044	0.043	0.024	0.026
plugin	SE	0.056	0.056	0.023	0.024
swpo	true	0.060	0.060	0.050	0.050
swpo	mean	0.053	0.055	0.050	0.050
swpo	SD	0.049	0.060	0.029	0.033
swpo	SE	0.065	0.069	0.030	0.034
bicnaive	true	0.060	0.060	0.050	0.050
bicnaive	mean	0.144	0.173	0.074	0.077
bicnaive	SD	0.048	0.048	0.028	0.029
bicnaive	SE	0.024	0.024	0.022	0.022
cvnaive	true	0.060	0.060	0.050	0.050
cvnaive	mean	0.134	0.178	0.065	0.071
cvnaive	SD	0.073	0.071	0.027	0.028
cvnaive	SE	0.028	0.026	0.024	0.024
pluginnaive	true	0.060	0.060	0.050	0.050
pluginnaive	mean	0.244	0.245	0.254	0.254
pluginnaive	SD	0.039	0.040	0.028	0.027
pluginnaive	SE	0.020	0.020	0.017	0.017
swbicnaive	true	0.060	0.060	0.050	0.050
swbicnaive	mean	0.101	0.111	0.065	0.066
swbicnaive	SD	0.042	0.044	0.027	0.027
swbicnaive	SE	0.025	0.025	0.022	0.022
true	true	0.060	0.060	0.050	0.050
true	mean	0.060	0.060	0.050	0.050
true	SD	0.027	0.028	0.023	0.023
true	SE	0.027	0.027	0.022	0.023

Design parm, summary for d3					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	true	0.000	0.000	0.000	0.000
bic	mean	-0.004	0.000	-0.001	0.001
bic	SD	0.041	0.041	0.027	0.028
bic	SE	0.051	0.050	0.028	0.029
cv	true	0.000	0.000	0.000	0.000
cv	mean	0.007	0.011	0.003	0.005
cv	SD	0.036	0.037	0.026	0.026
cv	SE	0.032	0.033	0.024	0.024
plugin	true	0.000	0.000	0.000	0.000
plugin	mean	-0.006	-0.003	0.000	0.001
plugin	SD	0.048	0.046	0.025	0.025
plugin	SE	0.057	0.057	0.024	0.025
swpo	true	0.000	0.000	0.000	0.000
swpo	mean	-0.009	-0.002	-0.001	0.002
swpo	SD	0.050	0.057	0.030	0.033
swpo	SE	0.066	0.069	0.030	0.034
bicnaive	true	0.000	0.000	0.000	0.000
bicnaive	mean	0.087	0.119	0.023	0.029
bicnaive	SD	0.051	0.047	0.029	0.029
bicnaive	SE	0.025	0.024	0.022	0.022
cvnaive	true	0.000	0.000	0.000	0.000
cvnaive	mean	0.076	0.124	0.014	0.023
cvnaive	SD	0.073	0.071	0.028	0.029
cvnaive	SE	0.028	0.026	0.024	0.025
pluginnaive	true	0.000	0.000	0.000	0.000
pluginnaive	mean	0.192	0.193	0.210	0.209
pluginnaive	SD	0.039	0.039	0.028	0.028
pluginnaive	SE	0.020	0.020	0.017	0.017
swbicnaive	true	0.000	0.000	0.000	0.000
swbicnaive	mean	0.042	0.053	0.014	0.018
swbicnaive	SD	0.043	0.044	0.027	0.027
swbicnaive	SE	0.026	0.025	0.022	0.022

5.3.3 Summary results for alt design

Design alt, summary for d1					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	true	0.400	0.400	0.240	0.240
bic	mean	0.455	0.521	0.265	0.279
bic	SD	0.056	0.068	0.032	0.037
bic	SE	0.048	0.048	0.033	0.035
cv	true	0.400	0.400	0.240	0.240
cv	mean	0.490	0.557	0.261	0.291
cv	SD	0.083	0.061	0.037	0.055
cv	SE	0.023	0.028	0.027	0.029
plugin	true	0.400	0.400	0.240	0.240
plugin	mean	0.591	0.597	0.367	0.369
plugin	SD	0.052	0.049	0.030	0.029
plugin	SE	0.047	0.045	0.027	0.027
swpo	true	0.400	0.400	0.240	0.240
swpo	mean	0.408	0.426	0.246	0.249
swpo	SD	0.044	0.065	0.030	0.035
swpo	SE	0.043	0.059	0.030	0.034
bicnaive	true	0.400	0.400	0.240	0.240
bicnaive	mean	0.541	0.587	0.352	0.369
bicnaive	SD	0.064	0.048	0.040	0.031
bicnaive	SE	0.025	0.023	0.022	0.022
cvnaive	true	0.400	0.400	0.240	0.240
cvnaive	mean	0.582	0.599	0.330	0.366
cvnaive	SD	0.072	0.052	0.059	0.039
cvnaive	SE	0.022	0.020	0.024	0.022
pluginnaive	true	0.400	0.400	0.240	0.240
pluginnaive	mean	0.606	0.603	0.384	0.383
pluginnaive	SD	0.059	0.056	0.026	0.027
pluginnaive	SE	0.019	0.019	0.021	0.021
swbicnaive	true	0.400	0.400	0.240	0.240
swbicnaive	mean	0.429	0.476	0.273	0.284
swbicnaive	SD	0.060	0.088	0.046	0.051
swbicnaive	SE	0.025	0.025	0.024	0.024
true	true	0.400	0.400	0.240	0.240
true	mean	0.401	0.399	0.240	0.240
true	SD	0.024	0.025	0.025	0.025
true	SE	0.024	0.024	0.025	0.025

Design alt, summary for d2					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	true	0.200	0.200	0.120	0.120
bic	mean	0.252	0.322	0.146	0.159
bic	SD	0.055	0.071	0.032	0.036
bic	SE	0.049	0.049	0.033	0.035
cv	true	0.200	0.200	0.120	0.120
cv	mean	0.292	0.373	0.138	0.160
cv	SD	0.091	0.066	0.037	0.051
cv	SE	0.024	0.029	0.029	0.031
plugin	true	0.200	0.200	0.120	0.120
plugin	mean	0.411	0.416	0.256	0.259
plugin	SD	0.052	0.049	0.030	0.029
plugin	SE	0.047	0.046	0.027	0.027
swpo	true	0.200	0.200	0.120	0.120
swpo	mean	0.206	0.224	0.127	0.130
swpo	SD	0.045	0.064	0.032	0.034
swpo	SE	0.043	0.059	0.031	0.034
bicnaive	true	0.200	0.200	0.120	0.120
bicnaive	mean	0.351	0.406	0.240	0.258
bicnaive	SD	0.067	0.047	0.042	0.032
bicnaive	SE	0.025	0.023	0.023	0.022
cvnaive	true	0.200	0.200	0.120	0.120
cvnaive	mean	0.403	0.425	0.216	0.256
cvnaive	SD	0.078	0.051	0.062	0.041
cvnaive	SE	0.022	0.021	0.025	0.023
pluginnaive	true	0.200	0.200	0.120	0.120
pluginnaive	mean	0.431	0.431	0.275	0.275
pluginnaive	SD	0.057	0.053	0.027	0.026
pluginnaive	SE	0.019	0.019	0.022	0.022
swbicnaive	true	0.200	0.200	0.120	0.120
swbicnaive	mean	0.230	0.283	0.155	0.168
swbicnaive	SD	0.064	0.093	0.048	0.054
swbicnaive	SE	0.026	0.026	0.025	0.025
true	true	0.200	0.200	0.120	0.120
true	mean	0.199	0.201	0.120	0.121
true	SD	0.025	0.025	0.026	0.025
true	SE	0.025	0.026	0.026	0.026

Design alt, summary for d3					
	n	500	500	1000	1000
method	stat p	250	500	500	1000
bic	true	0.000	0.000	0.000	0.000
bic	mean	0.052	0.121	0.025	0.038
bic	SD	0.059	0.075	0.033	0.035
bic	SE	0.050	0.050	0.034	0.036
cv	true	0.000	0.000	0.000	0.000
cv	mean	0.093	0.186	0.013	0.034
cv	SD	0.094	0.073	0.036	0.048
cv	SE	0.026	0.030	0.030	0.033
plugin	true	0.000	0.000	0.000	0.000
plugin	mean	0.231	0.235	0.146	0.149
plugin	SD	0.054	0.050	0.031	0.031
plugin	SE	0.047	0.046	0.028	0.027
swpo	true	0.000	0.000	0.000	0.000
swpo	mean	0.006	0.022	0.004	0.009
swpo	SD	0.045	0.065	0.032	0.035
swpo	SE	0.044	0.060	0.031	0.034
bicnaive	true	0.000	0.000	0.000	0.000
bicnaive	mean	0.166	0.225	0.127	0.147
bicnaive	SD	0.070	0.047	0.043	0.032
bicnaive	SE	0.026	0.024	0.023	0.023
cvnaive	true	0.000	0.000	0.000	0.000
cvnaive	mean	0.226	0.251	0.100	0.145
cvnaive	SD	0.083	0.051	0.067	0.043
cvnaive	SE	0.023	0.021	0.025	0.023
pluginnaive	true	0.000	0.000	0.000	0.000
pluginnaive	mean	0.261	0.261	0.165	0.166
pluginnaive	SD	0.055	0.052	0.027	0.026
pluginnaive	SE	0.020	0.020	0.022	0.022
swbicnaive	true	0.000	0.000	0.000	0.000
swbicnaive	mean	0.032	0.087	0.037	0.050
swbicnaive	SD	0.065	0.096	0.050	0.056
swbicnaive	SE	0.028	0.027	0.026	0.025

A Formulas for $Q()$

Here are the formulas for $Q()$

- For linear models,

$$Q(y_i, \mathbf{w}_i\boldsymbol{\delta}') = (y_i - \mathbf{w}_i\boldsymbol{\delta}')^2$$

- For Poisson models

$$Q(y_i, \mathbf{w}_i\boldsymbol{\delta}') = -[y_i\mathbf{w}_{i,t}\boldsymbol{\delta}' - \exp(\mathbf{w}_i\boldsymbol{\delta}') - \ln(y_i!)]$$

- For logit models

$$Q(y_i, \mathbf{w}_i\boldsymbol{\delta}') = \ln[1 + \exp(\mathbf{w}_i\boldsymbol{\delta}')] - y_i(\mathbf{w}_i\boldsymbol{\delta})$$

References

Allcott, H. and J. B. Kessler (2019): "The welfare effects of nudges: A case study of energy use social comparisons," American Economic Journal: Applied Economics, 11, 236–76.

Banerjee, A., A. G. Chandrasekhar, E. Duflo, and M. O. Jackson (2019): "Using gossips to spread information: Theory and evidence from two randomized controlled trials," The Review of Economic Studies, 86, 2453–2490.

Belloni, A., D. Chen, V. Chernozhukov, and C. Hansen (2012): "Sparse models and methods for optimal instruments with an application to eminent domain," Econometrica, 80, 2369–2429.

Belloni, A. and V. Chernozhukov (2011): "1-penalized quantile regression in high-dimensional sparse models," The Annals of Statistics, 39, 82–130.

Belloni, A., V. Chernozhukov, and C. Hansen (2014): "Inference on treatment effects after selection among high-dimensional controls," The Review of Economic Studies, 81, 608–650.

Belloni, A., V. Chernozhukov, and Y. Wei (2016): "Post-selection inference for generalized linear models with many controls," Journal of Business & Economic Statistics, 34, 606–619.

Bennedsen, M., M. Tsoutsoura, and D. Wolfenzon (2019): "Drivers of effort: Evidence from employee absenteeism," Journal of Financial Economics, 133, 658–684.

Bickel, P., C. Klaassen, Y. Ritov, and J. Wellner (1998): Efficient and Adaptive Estimation for Semiparametric Models, Springer New York.

Bickel, P. J., Y. Ritov, and A. B. Tsybakov (2009): "Simultaneous analysis of Lasso and Dantzig selector," The Annals of Statistics, 37, 1705–1732.

Cattaneo, M. D., M. Jansson, and X. Ma (2018a): "Two-step estimation and inference with possibly many included covariates," The Review of Economic Studies, 86, 1095–1122.

Cattaneo, M. D., M. Jansson, and W. K. Newey (2018b): "Inference in Linear Regression Models with Many Covariates and Heteroskedasticity," Journal of the American Statistical Association, 113, 1350–1361.

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

Chernozhukov, V., C. Hansen, and M. Spindler (2015): "Valid Post-Selection and Post-Regularization Inference: An Elementary, General Approach," Annual Review of Economics, 7, 649–688.

Chetverikov, D., Z. Liao, and V. Chernozhukov (2020): "On Cross-Validated Lasso in High Dimensions," https://arxiv.org/pdf/1605.02214.pdf.

Friedman, J., T. Hastie, H. Höfling, and R. Tibshirani (2007): "Pathwise coordinate optimization," The Annals of Applied Statistics, 1, 302–332.

Friedman, J., T. Hastie, and R. Tibshirani (2010): "Regularization Paths for Generalized Linear Models via Coordinate Descent," Journal of statistical software, 33, 1–22.

Hastie, T., R. Tibshirani, and M. Wainwright (2015): Statistical Learning with Sparsity: The Lasso and Generalizations, Boca Rotaon: CRC Press.

Jing, B.-Y., Q.-M. Shao, and Q. Wang (2003): "Self-normalized Cramér-type large deviations for independent random variables," The Annals of probability, 31, 2167–2215.

Kallestrup-Lamb, M., A. B. Kock, and J. T. Kristensen (2016): "Lassoing the determinants of retirement," Econometric Reviews, 35, 1522–1561.

Kozbur, D. (2020): "Testing-Based Forward Model Selection," https://arxiv.org/pdf/1512.02666.pdf.

Leeb, H. and B. M. Pötscher (2006): "Can one estimate the conditional distribution of post-model-selection estimators?" The Annals of Statistics, 34, 2554–2591.

——— (2008): "Sparse estimators and the oracle property, or the return of Hodges estimator," Journal of Econometrics, 142, 201–211.

Paravisini, D., V. Rappoport, P. Schnabl, and D. Wolfenzon (2014): "Dissecting the effect of credit supply on trade: Evidence from matched credit-export data," The Review of Economic Studies, 82, 333–359.

Peña, V. H., T. L. Lai, and Q.-M. Shao (2009): Self-normalized processes: Limit theory and Statistical Applications, Berlin: Springer-Verlag.

Pötscher, B. M. and H. Leeb (2009): "On the distribution of penalized maximum likelihood estimators: The LASSO, SCAD, and thresholding," Journal of Multivariate Analysis, 100, 2065–2082.

Tsiatis, A. A. (2006): Semiparametric theory and missing data, New York: Springer Verlag.

Zhang, Y., R. Li, and C.-L. Tsai (2010): "Regularization parameter selections via generalized information criterion," Journal of the American Statistical Association, 105, 312–323.