The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–15

posw: A Stata Command for the stepwise Neyman-orthogonal estimator

David M. Drukker
Sam Houston State University

Di Liu
Stata

Abstract. Inference for structural and treatment parameters while having high-dimensional covariates in the model is increasingly common. The Neyman-orthogonal (NO) estimators in Belloni, Chernozhukov, and Wei (2016) produce valid inferences for the parameters of interest while using GLM lasso methods to select the covariates. Drukker and Liu (2022b) extended the estimators in Belloni, Chernozhukov, and Wei (2016) by using a BIC-stepwise method and a testing-stepwise method as the covariate-selector. Drukker and Liu (2022b) found a family of data-generating processes for which the NO estimator based on BIC-stepwise produces much more reliable inferences than the lasso-based NO estimator. This paper describes the implementation of **posw**: a Stata command for the stepwise-based Neyman-orthogonal estimator for the high-dimensional linear, logit, and Poisson models.

Keywords: **posw**, high-dimensional model, partialing-out, stepwise, Neyman-orthogonal, generalized linear model, post-selection inference

1 Introduction

Many researchers face a situation in which they want to make inferences about a few coefficients on variables of interest while having more control variables than they could include in the model for the sample size at hand. This situation is a high-dimensional model (HDM), and the sparse modeling approach is frequently applied. The sparse modeling approach has three key features. First, the sparse modeling approach assumes that the model is "sparse". The model is sparse when the number of potential controls that must be included in the model is small relative to the sample size. Second, the sparse modeling approach uses a covariate-selection method to choose which of the many potential controls must be included. Third, the sparse modeling approach uses an estimator for the parameters of interest that is robust to the inevitable mistakes made in the covariate-selection step. As discussed in Chernozhukov, Hansen, and Spindler (2015), these robust estimators are known as Neyman-orthogonal (NO) estimators because they extend a technique derived in Neyman (1959) and Neyman (1979) and they are robust to the covariate-selection mistakes made by high-quality covariate-selection techniques.[1]

Why covariate-selection methods inevitably make mistakes deserves an explanation. Covariate selection has a long and somewhat controversial history in statistics and

1. A high-quality covariate selection technique selects the required covariates at a fast enough rate, see Chernozhukov, Hansen, and Spindler (2015).

st0001

econometrics. When all of the nonzero coefficients are large enough in magnitude and the model is sparse, some covariate-selection methods will include the required covariates with probability approaching one as the sample size increases, under some regularity conditions. These regularity conditions require a minimum ability of the statistical model to approximate the true model and place limits on the possible joint distribution of the covariates. The assumption that the nonzero coefficients are large enough in magnitude is known as a "beta-min" assumption, and it is the crucial assumption. It is now commonly accepted that the beta-min assumption is way too strong to be a part of a realistic approach to estimation and inference.

While Leeb and Pötscher (2006), Leeb and Pötscher (2008), and Pötscher and Leeb (2009) contain formal results and intuition based on uniform versus point-wise results, there is a simple thought experiment that captures why the beta-min assumption is too strong. It is frequently the case in applied studies that several coefficients have p-values that are just a little too large to reject the null hypothesis that their true values are zero. These covariates are on the border of being included or not in the model. The beta-min assumption would require that the true values of the coefficients on these not-included covariates are either zero or very close to zero, where the definition of very close is a function of the sample size. Dropping the beta-min assumption allows for a more realistic scenario in which these coefficients have nonzero values that are just a little too small in magnitude to warrant including the covariates in the model, according to the covariate-selection method.

Dropping the beta-min assumption implies that even the best possible covariate-selection methods will omit some covariates whose coefficients are small in magnitude. When we accept that covariate-selection methods make mistakes, we must use an estimation technique that is robust to these mistakes in covariate selection. Naive estimators that simply include the selected covariates in a model are not robust and do not provide reliable inferences. In repeated samples, the random inclusion or exclusion of covariates with small coefficients causes the distribution of the naive estimators to be multimodal. Using a normal distribution to approximate this multimodal distribution produces unreliable results in theory and practice. Again, see Leeb and Pötscher (2006), Leeb and Pötscher (2008), and Pötscher and Leeb (2009) for details.

Belloni, Chen, Chernozhukov, and Hansen (2012), Belloni, Chernozhukov, and Hansen (2014), Chernozhukov, Hansen, and Spindler (2015), and Belloni, Chernozhukov, and Wei (2016) pioneered NO estimators that are robust the mistakes in covariate selection made by high-quality covariate-selection methods. These theoretical studies formally demonstrated that NO estimators based on lasso methods with a feasible version of the optimal lasso tuning parameters produce valid inferences for the coefficients of interest. The feasible version of the optimal lasso tuning parameters is known as the plugin method. Drukker and Liu (2022b) and Drukker and Liu (2022a) extended the feasible version to the GLM case.

Drukker and Liu (2022b) studied the finite-sample performance of the Lasso-based NO estimator for the high-dimensional generalized linear model (GLM) using different methods to select the lasso tuning parameters. The simulation results in Drukker and

Liu (2022b) suggest both advantages and disadvantages of using the lasso as a covariate-selection method in a NO estimator. The main advantage of the lasso-based NO estimator is speed. If the tuning parameters are appropriately chosen, the lasso-based NO estimator can relatively quickly provide valid inferences in the presence of small coefficients and many potential covariates. The main disadvantage of the lasso-based NO estimator is that the simulations in Drukker and Liu (2022b) reveal a problematic family of data-generating processes (DGPs) for which the lasso-based NO estimator fails, regardless of the choice of the tuning parameter. The problematic family of DGPs has coefficients that alternate in sign. Coefficients that alternate in sign are commonly observed in models that use powers and interaction terms among covariates to approximate nonlinear functional forms.

To accommodate this family of DGPs, Drukker and Liu (2022b) extended the lasso-based NO estimator in Belloni, Chernozhukov, and Wei (2016) to BIC-stepwise-based NO estimator for the high-dimensional GLM model. The simulation results in Drukker and Liu (2022b) show that the BIC-stepwise-based NO estimator performs well on the designs for which the lasso-based NO estimator failed.

The price of the increased performance was computation time. The BIC-stepwise-based NO estimator is much slower than the lasso-based NO estimators [2]. In practical terms, the BIC-stepwise-based NO estimators become computationally infeasible for huge numbers of potential covariates, which the lasso-based NO estimators can handle.

The good performance of the BIC-stepwise-based NO estimator is not without some theoretical support. Kozbur (2020) presents conditions in which a testing-stepwise-based NO estimator will produce valid inference. Ironically, the testing-stepwise-based NO estimator did not perform well in the problematic DGP in Drukker and Liu (2022b). One possible reason is that the significance level is essentially an unoptimized tuning parameter in the covariate selection method. Personally, we recommend using the BIC-based stepwise estimator instead of the testing-based stepwise estimator, but both are available in `posw`.

We organize this paper as follows. Section 2 describes the high-dimensional GLM model, the BIC-stepwise-based NO estimators, and the testing-stepwise-based NO estimators. Section 3 documents the syntax and options for the Stata command **posw**. Section 4 shows a numerical example of using **posw**. Section 5 presents the simulation results. Finally, section 6 concludes.

2 stepwise-based NO estimator for parameters in HDM

2.1 High-dimensional models

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

2. For example, in a dataset with 1000 observations and 100 controls, the BIC-based `popoisson` takes 0.5 seconds while `posw` takes 24 seconds.

$$\mathbf{E}[y_i|\mathbf{d}_i, \mathbf{x}_i] = G(\mathbf{d}_i\boldsymbol{\alpha}_0' + \mathbf{x}_i\boldsymbol{\beta}_0') \tag{1}$$

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 function for logit models, or when $G()$ is the exponential function for (quasi) 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 the sparsity assumption mentioned in the introduction.

The goal is to obtain reliable estimation and inference for $\boldsymbol{\alpha}_0$. The covariates in $\mathbf{d}_i$ must be specified a-priori, and their number is assumed to be small relative to n. Any or all of the coefficients in $\boldsymbol{\alpha}_0$ can be zero. Specifying a covariate to be of interest does not imply that it has a nonzero effect.

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 the 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 produce a root-n consistent and asymptotically normal estimator for $\boldsymbol{\alpha}_0$. Belloni, Chen, Chernozhukov, and Hansen (2012), Chernozhukov, Hansen, and Spindler (2015), and Belloni, Chernozhukov, and Wei (2016) provide formal statements and analyses of how to allow for and how to bind the approximation error.

Algorithm 1 gives the naive estimator for $\boldsymbol{\alpha}_0$ estimator discussed in the Introduction. 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 that they 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 unacceptably poor results in finite samples.

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 $\boldsymbol{\alpha}_0$.
```

Instead of a naive estimator, **posw** implements NO estimators that were explicitly designed to provide valid inference for $\boldsymbol{\alpha}$ when some of the model's coefficients are small in magnitude. See Belloni, Chen, Chernozhukov, and Hansen (2012), Chernozhukov, Hansen, and Spindler (2015), and Belloni, Chernozhukov, and Wei (2016) for formal results. Instead of naively using the covariates selected in a model of y on $\mathbf{d}$ and $\mathbf{x}$, NO estimators use moment conditions that are robust to the inevitable mistakes that covariate-selection methods make. The NO estimators use multiple covariate-selection steps to form a moment condition for $\boldsymbol{\alpha}$ that is orthogonal to the first-stage selection. This process is an extension of the technique discussed in Neyman (1959) and Neyman (1979), hence the NO moniker.

The NO estimators can be implemented using different covariate-selection techniques. One popular choice is to use the lasso. Belloni, Chernozhukov, and Wei (2016) derives the Lasso-based NO estimator for the high-dimensional GLM. It uses a particular version of the lasso that selects the tuning parameters using a plugin method. In practice, the NO estimator's performance critically depends on the choice of tuning parameter selection method.

Drukker and Liu (2022b) studied the finite sample behavior of the lasso-based NO estimator for the HDM using different tuning parameter selection methods. The simulation results reveal that a family of DGPs for which the lasso-based NO estimators provide poor inferential results, but that BIC-stepwise-based NO estimator provide reliable inferential results.

Kozbur (2020) presents formal results for testing-stepwise selection for linear models. These results show that testing-stepwise-based NO estimators will perform well in the large sample under the conditions described in the paper. Given these formal results, it is a little surprising that testing-stepwise-based NO estimators did not perform well for the problematic DGP in Drukker and Liu (2022b). We conjecture that the poor performance of testing-stepwise-based NO on the problematic DGP was due to the choice of the significance level. In practice, we recommend the BIC-stepwise selection because it avoids choosing the significance level and how well it has performed in our simulations.

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

2.2 Algorithms

The BIC-stepwise algorithm used by `posw` is algorithm 3 in Drukker and Liu (2022b). The testing-based-stepwise algorithm used by `posw` is algorithm 4 in Drukker and Liu (2022b).

Belloni, Chernozhukov, and Wei (2016) derived lasso-based NO estimators for the GLM model. We implement versions of NO estimators that use BIC-stepwise or testing-stepwise for covariate selection. Algorithm 2 provides the details about these versions of the Belloni, Chernozhukov, and Wei (2016) NO estimator. Algorithm 2 generalizes algorithm 6 in Drukker and Liu (2022b) from the Poisson-regression case to the GLM-regression case.

Algorithm 2: Stepwise-based NO GLM estimation

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

    - For the BIC-stepwise NO estimator, we find the subset of the $\mathbf{x}$ that BIC stepwise includes.
    - For the testing-stepwise NO estimator, we find the subset of the $\mathbf{x}$ that testing stepwise includes.

2. Use the unpenalized QML GLM regression estimator to estimate the coefficients $\widetilde{\boldsymbol{\alpha}}$ and $\widetilde{\boldsymbol{\beta}}$ in a GLM 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 $G(\cdot)$. Let $\sigma_i^2 = \widehat{Var}(\mathbf{y}_i|\mathbf{d}_i, \mathbf{x}_i)$. Let $f_i = \omega_i/\sigma_i$.

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

    - The BIC-stepwise NO estimator uses a weighted BIC-stepwise for covariate selection.
    - The testing-stepwise NO estimator uses a weighted testing-stepwise for covariate selection.

6. For each $j \in \{1, \dots, J\}$, run a linear, ordinary least squares regression of the jth variable in $\mathbf{d}$ on $\check{\mathbf{x}}_j$ with observation-level weights f_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 of instrumental variables $\mathbf{z} = (\widetilde{d}_1, \dots, \widetilde{d}_J)$ and $\mathbf{z}_i$ be the ith observation on this vector of instrumental variables. Note that $\mathbf{z}_i = (z_{1,i}, \dots, z_{J,i}) = (\widetilde{d}_{1,i}, \dots, \widetilde{d}_{J,i})$.

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

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

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

3 Syntax of posw

posw has the following syntax.

`posw` *depvar* *varsofinterest* [*if*] [*in*], `controls(`*varlist*`)`
    `model(linear|logit|poisson)` [`method(bic|test) alpha(#)`]

where *varofinterest* is a *varlist*.

3.1 Options

`controls(`*varlist*`)` specifies the set of control variables, which control for omitted variables. Control variables are also known as confounding variables. `posw` uses the forward stepwise to select the control variables for each of *depvar* and *varsofinterest*. `controls()` is required.

`model(linear|logit|poisson)` specifies the model for the outcome variable *depvar*. It can be one of `linear`, `logit`, or `poisson` model. `model()` is required.

`method(bic|test)` specifies the method used in stepwise covariate selection. It can be one of `bic` and `test`. Specifying `bic` implies using the BIC-based stepwise. Specifying `test` implies using the testing-based stepwise. The default is `bic`.

`alpha(#)` specifies the level of significance for the testing-based stepwise. The default is 0.05.

3.2 Stored results

posw stores the following results in `e()`.

Scalars	
`e(N)`	number of observations
`e(k_controls)`	number of controls
`e(k_controls_sel)`	number of selected controls
`e(k_varsofinterest)`	number of variables of interest
Macros	
`e(cmd)`	posw
`e(varsofinterest)`	variables of interest
`e(depvar)`	dependent variable
`e(controls_sel)`	selected control variables
`e(controls)`	control variables
`e(model)`	type of model
`e(title)`	title in estimation output
`e(vcetype)`	robust
`e(vce)`	Robust
`e(properties)`	`b V`
Matrices	
`e(b)`	coefficient vector
`e(V)`	variance-covariance matrix of the estimators
Functions	
`e(sample)`	marks estimation sample

4 A numerical example

We now illustrate the use of **posw** through an empirical example. We have an extract of data in (Sunyer et al. 2017) to measure the effect of air pollution level on the student's response time. The model is

$$\texttt{react}_i = \texttt{no2\_class}_i \alpha + \mathbf{x}_i \beta' + \epsilon_i$$

where $\texttt{react}_i$ is the response time of child i on a test, $\texttt{no2\_class}_i$ is the pollution level in the school attended by child i, $\mathbf{x}_i$ is a high-dimensional control to be included in the model, and ϵ_i is the disturbance term.

To start, we bring the data `breathe` into memory.

```
. use https://www.stata-press.com/data/r16/breathe, clear
(Nitrogen dioxide and attention)
```

Next, we need to define the control variables **x**. It would be tedious to separate the factor variables from the continuous variables manually. Instead, we can use Stata's variable management tools `vl`. Here, we directly call a do-file from the Stata website to save some typing.

```
. quietly do https://www.stata-press.com/data/r16/no2
```

The purpose of this do file is to define a global macro `$fc` for the factor control variables and a global macro `$cc` for the continuous control variables. We can display their content.

```
. display "$cc"
no2_home age age0 sev_home green_home noise_school sev_school precip siblings_o
> ld siblings_young
. display "$fc"
sex grade overweight lbweight breastfeed msmoke meducation feducation
```

Now, we can define the control variables as raw variables and the full second-order interaction among them. The control variables are stored in global macro `$controls` for later use.

```
. global controls (c.($cc) i.($fc))##(c.($cc) i.($fc))
```

Finally, we can fit our model using **posw**. We specify option `controls()` for the control variables and the option `model()` for the linear model.

```
. posw react no2_class, controls($controls) model(linear)
select controls for react using stepwise bic
select controls for no2_class using stepwise bic
Partialing-out stepwise bic          Number of obs               =      1,036
```

```
                                       Number of controls           =        516
                                       Number of selected controls  =         16
                                       Wald chi2(1)                 =      20.97
Model: linear                          Prob > chi2                  =     0.0000

------------------------------------------------------------------------------
             |               Robust
       react |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
   no2_class |   2.493274     .54448     4.58   0.000     1.426113    3.560435
------------------------------------------------------------------------------
Note: Chi-squared test is a Wald test of the coefficients of the variables of
      interest jointly equal to zero.
```

The results imply that another microgram of NO2 per cubic meter increases the mean reaction time by 2.5 milliseconds. Only the coefficient on the covariate of interest is estimated. The coefficients on the control covariates are not estimated. The cost of using covariate-selection methods is that these estimators do not produce estimates for the coefficients on the control covariates. Remarkably, there are 516 control variables, and only 16 of them are selected.

5 Simulations

This section describes some Monte Carlo simulations that illustrate the good performance of the **posw** command in linear, logit, and Poisson/exponential-conditional-mean models. We note that the naive estimators perform unacceptably poorly on these designs. [3]

5.1 Designs

The design for the simulations reflects the structure of the high-dimensional GLM model. In each design, there are a few covariates whose coefficients have values that are large in magnitude, there are a few covariates whose coefficients have values that are small in magnitude, and there are many covariates whose coefficients are zero. Following Drukker and Liu (2022b), we specify the value of each small coefficient to be about two times its standard error in the true model for the sample size used in the simulations. We specify the value of each large coefficient to be about four times its standard error in the true model for the sample size used in the simulations. These values encode a representation of DGPs for which there is no beta-min condition. The small coefficients are close to being statistically significant while the large coefficients should be statistically significant in the vast majority of the repeated samples.

The DGPs for the three designs are given in equations (2)-(4).

3. For a more detailed simulation study on the comparison of the `posw` and `popoisson`, see Drukker and Liu (2022b).

$$\begin{aligned} \text{linear} \qquad & y_i = w_i + \epsilon_i & (2) \\ \text{logit} \qquad & y_i = w_i + \epsilon_i > 0 & (3) \\ \text{poisson} \qquad & y_i = exp(w_i)\epsilon_i & (4) \end{aligned}$$

For each design

$$w_i = \alpha_{big} d_{big,i} + \alpha_{small} d_{small,i} + \alpha_{zero} d_{zero,i} + \mathbf{x}_{big,i} \boldsymbol{\beta}'_{big} + \mathbf{x}_{small,i} \boldsymbol{\beta}'_{small}$$

where

- d_{big} is the covariate of interest whose coefficient, α_{big}, is large;
- d_{small} is the covariate of interest whose coefficient, α_{small}, is small;
- d_{zero} is the covariate of interest whose coefficient, α_{zero}, is zero;
- $\mathbf{x}_{big}$ is the vector of control covariates whose coefficients, $\boldsymbol{\beta}_{big}$, are large;
- $\mathbf{x}_{small}$ is the vector of control covariates whose coefficients, $\boldsymbol{\beta}_{small}$, are small;

A type of Toeplitz structure is frequently used to generate the covariates in high-dimensional models; see Belloni, Chen, Chernozhukov, and Hansen (2012) for an example. In these Toeplitz structures, each covariate has an index $j \in \{1, \ldots, p\}$, where p is the total number of covariates in the model. (This includes the covariates of interest and the control covariates.) Covariates with near-by indices are significantly correlated, but the amount of correlation decays as the distance between the indices increases. Each covariate was generated as

$$x_j = .3x_{j-1} + .2 * x_{j-4} + .2x_{j-8} + \eta$$

where $\eta = (\texttt{rchi2}(25) - 25)/\texttt{sqrt}(50))$, and `rchi2(a)` is the Stata function that generates draws from a χ^2 distribution with `a` degrees of freedom. The distribution for η has mean zero, variance one, and its higher moments are distinctly different than those from a normal distribution.

For each design, we generated $p = 100$ covariates. For each draw, we drew 120 covariates and discarded the first 20 to burn in the Toeplitz structure.

- Covariate 1 is d_{big}, the covariate of interest whose coefficient is large.
- Covariates 2-5 are $\mathbf{x}_{big}$, the control covariates whose coefficients are large.
- Covariate 6 is d_{small}, the covariate of interest whose coefficient is small.

- Covariates 7-10 are $\mathbf{x}_{small}$, the control covariates whose coefficients are small.
- Covariate 11 is d_{zero}, the covariate of interest whose coefficient is zero.
- Covariates 12-100 are $\mathbf{x}_{zero}$, the control covariates whose coefficients are zero.

Here are true values for the coefficients.

Table 1: True coefficient values

Model	Coefficient	Value
Linear	α_{big}	.12
Linear	α_{small}	.06
Linear	β_{big}	.12
Linear	β_{small}	.06
Logit	α_{big}	.32
Logit	α_{small}	.16
Logit	β_{big}	.32
Logit	β_{small}	.16
Poisson	α_{big}	.08
Poisson	α_{small}	.04
Poisson	β_{big}	.08
Poisson	β_{small}	.04

Here is how the error term was generated for each design. For the linear design, $\epsilon_i = (\texttt{rchi2}(25) - 25)/\texttt{sqrt}(50))$, where `rchi2(a)` is the Stata function that generates draws from a χ^2 distribution with `a` degrees of freedom. This distribution has mean zero, variance one, and its higher moments are distinctly different from those in a normal distribution.

For the logit design, $\epsilon_i = \texttt{rlogitic}()$, where `rlogistic()` generates draws from a standard logistic distribution.

For the Poisson design, $\epsilon_i = \texttt{rweibull}(2, \texttt{b})$, where `rweibull(c,b)` generate draws from a Weibull distribution with shape parameter c and scale paramter b. We set $b = 1/\texttt{exp}(\texttt{lngamma}(1 + 1/2))$ so that the mean of ϵ is 1. Note that we are drawing from a conditional exponential mean model, not from a Poisson model.

We ran 2,400 repetitions for each design. There are 2,400 repetitions because we used Stata's stream random numbers to parallelize the simulations over 40 cores, with 60 repetitions on each core. (See `help rngstream` for an introduction to stream random numbers.)

5.2 Results

Tables 2, 3, and 4 summarize the simulation results. In short, we see that the NO estimators in bic step and test step perform well and that the naive estimators in bic naive and test naive do not perform well.

Here is how each table is structured.

- Model specifies the DGP design.
- Estimator specifies the estimator used for that row's results.
- Mean specifies the sample mean of the estimates of that coefficient over the repetitions. The sample mean should be close to the true value given in table 1.
- SD specifies the sample standard deviation of the estimates of that coefficient over the repetitions.
- SE specifies the sample mean of the standard errors of that coefficient over the repetitions. The sample mean of the standard errors should be close to the sample standard deviation of the estimates.
- RR specifies the rejection rate of a test against the true null hypothesis. The significance level of each test was .05, so the RR should be close to .05.

Table 2: Result for large coefficient α_{big}

Model	Estimator	Mean	SD	SE	RR
linear	bic step	0.1172	0.0343	0.0336	0.060
linear	test step	0.1157	0.0343	0.0337	0.059
linear	bic naive	0.1381	0.0387	0.0316	0.146
linear	test naive	0.1546	0.0420	0.0311	0.280
linear	true	0.1198	0.0338	0.0332	0.055
logit	bic step	0.3260	0.0854	0.0824	0.054
logit	test step	0.3214	0.0863	0.0797	0.072
logit	bic naive	0.3649	0.0921	0.0791	0.128
logit	test naive	0.3980	0.0979	0.0769	0.240
logit	true	0.3255	0.0824	0.0820	0.045
poisson	bic step	0.0787	0.0192	0.0198	0.047
poisson	test step	0.0796	0.0194	0.0192	0.052
poisson	bic naive	0.1088	0.0246	0.0298	0.137
poisson	test naive	0.0963	0.0224	0.0176	0.218
poisson	true	0.0799	0.0185	0.0185	0.051

Table 3: Result for small coefficient α_{small}

Model	Estimator	Mean	SD	SE	RR
linear	bic step	0.0586	0.0339	0.0341	0.053
linear	test step	0.0583	0.0338	0.0341	0.052
linear	bic naive	0.0760	0.0360	0.0317	0.111
linear	test naive	0.0868	0.0373	0.0314	0.188
linear	true	0.0602	0.0327	0.0331	0.050
logit	bic step	0.1640	0.0851	0.0837	0.055
logit	test step	0.1632	0.0844	0.0813	0.061
logit	bic naive	0.2005	0.0876	0.0780	0.112
logit	test naive	0.2232	0.0868	0.0761	0.165
logit	true	0.1643	0.0797	0.0809	0.052
poisson	bic step	0.0381	0.0196	0.0200	0.048
poisson	test step	0.0387	0.0193	0.0196	0.052
poisson	bic naive	0.0612	0.0211	0.0308	0.032
poisson	test naive	0.0530	0.0208	0.0176	0.168
poisson	true	0.0391	0.0186	0.0185	0.055

Table 4: Result for zero coefficient α_{zero}

Model	Estimator	Mean	SD	SE	RR
linear	bic step	0.0003	0.0341	0.0341	0.053
linear	test step	0.0015	0.0343	0.0341	0.055
linear	bic naive	0.0072	0.0338	0.0314	0.078
linear	test naive	0.0147	0.0353	0.0312	0.114
linear	true	0.0005	0.0320	0.0317	0.056
logit	bic step	0.0020	0.0841	0.0837	0.050
logit	test step	0.0051	0.0826	0.0815	0.054
logit	bic naive	0.0136	0.0802	0.0765	0.064
logit	test naive	0.0316	0.0813	0.0749	0.092
logit	true	-0.0002	0.0762	0.0773	0.040
poisson	bic step	0.0014	0.0197	0.0200	0.051
poisson	test step	0.0018	0.0197	0.0196	0.055
poisson	bic naive	0.0131	0.0204	0.0310	0.013
poisson	test naive	0.0070	0.0194	0.0175	0.100
poisson	true	0.0003	0.0177	0.0177	0.050

6 Conclusion

We motivated and described **posw**, a Stata command for the stepwise-based Neyman-orthogonal estimator in the linear, logit, and Poisson model. This command can be viewed as an alternative to the lasso-based NO estimators, which are implemented in official Stata commands `poregress`, `pologit`, and `popoisson`. Simulations in Drukker and Liu (2022b) show that the implemented BIC-stepwise-based NO can perform better than the Lasso-based NO estimators for a family of DGPs.

The main cost of using a stepwise-based NO estimator instead of a lasso-based NO estimator is an increase in computation time. Future development could speed up **posw** by using cluster-parallel computation or the the sure-independence-screening version of stepwise partialing-out estimator outlined in Drukker and Liu (2022b).

7 Acknowledgments

We thank the editor and an anonymoous referee for their comments and sugestions.

8 References

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(6): 2369–2429.

Belloni, A., and V. Chernozhukov. 2011. L1-penalized quantile regression in high-dimensional sparse models. *The Annals of Statistics* 39(1): 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(2): 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(4): 606–619.

Chernozhukov, V., C. Hansen, and M. Spindler. 2015. Valid Post-Selection and Post-Regularization Inference: An Elementary, General Approach. *Annual Review of Economics* 7(1): 649–688.

Drukker, D., and D. Liu. 2022a. A cluster plugin method for selecting the GLM lasso tuning parameters in models for unbalanced panel data. *Econometrics and Statistics* in press. https://doi.org/10.1016/j.ecosta.2022.02.006.

Drukker, D. M., and D. Liu. 2022b. Finite-sample results for lasso and stepwise Neyman-orthogonal Poisson estimators. *Econometric Reviews* 41: 1047–1076. https://doi.org/10.1080/07474938.2022.2091363.

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(5): 2554–2591.

———. 2008. Sparse estimators and the oracle property, or the return of Hodges' estimator. *Journal of Econometrics* 142(1): 201–211.

Neyman, J. 1959. Optimal asymptotic tests of composite statistical hypotheses. In *Probability and Statistics: The Harald Cramer Volume*, ed. U. Grenander, 213–234. New York: Wiley.

———. 1979. C(α) tests and their use. *Sankhya* 41: 1–21.

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(9): 2065–2082.

Sunyer, J., E. Suades-González, R. García-Esteban, I. Rivas, J. Pujol, M. Alvarez-Pedrerol, J. Forns, X. Querol, and X. Basagaña. 2017. Traffic-related Air Pollution and Attention in Primary School Children: Short-term Association. *Epidemiology* 28(2): 181–189.

About the authors

David M. Drukker is an Associate Professor in the Department of Economics and International Business of Sam Houston State University.

Di Liu is a Principal Econometrician in StataCorp in the United States.