posis: Stata command for the sure-independence-screening Neyman-orthogonal estimator

David M. Drukker
Sam Houston State University

Di Liu
Stata

Abstract. Inference for structural parameters in a high-dimensional model has become increasingly popular. Belloni et al. (2016) proposed a Lasso-based Neyman-orthogonal estimator that produces valid inference for the coefficients of interests in the generalized linear model. Drukker and Liu (2022) extends their estimator by using a BIC-stepwise-based Neyman-orthogonal estimator and the simulations show the advantage of using BIC-based stepwise as the covariate-selection technique. However, the BIC-stepwise-based NO estimator becomes computational infeasible when there are much more control variables. To overcome this computational bottleneck, Drukker and Liu (2022) proposes to combine the sure-independence-screening technique with BIC-based-stepwise in order to improve the computational speed while maintaining a similar or better statistical performance. This paper presents the implementation of `posis`: a Stata command for a iterative-sure-independence-screening-based Neyman orthogonal estimator for the high-dimensional linear, logit, and Poisson models.

Keywords: **posis**, **isis**, high-dimensional model, partialing-out, sure independence screening, Neyman-orthogonal, generalized linear model, post-selection inference

1 Introduction

The growing use of flexbile functional forms and the availability of many covariates in "big data" environments have created an abundance of data in which researchers now have access to hundreds or thousands of covariates. This abundance of data allows researchers to build more more flexible models that better approximate reality. But this abundance of data also forces us to perform covaraite selection, because including all the available covariates is not feasible. Covariate selection raises a theoretical challenge and a computational challenge. The theoretical challenge is how to find estimators that provide valid inference after performing covariate selection. The computational challenge is how to quickly perform covariate selection when there are many covariates.

There are two key ingredients to solving the theoretical challenge: the sparsity assumption and the Neyman orthogonality property. The sparsity assumption requires that only a few of the many available covariates need to included in the model. Sparsity means covariate selection is part of the solution. But covariate selection cannot be performed without error unless the unrealistic "beta-min" condidion is imposed. The

beta-min condition requires that the coefficients in the model are either large in magnitude or exactly zero. When this unrealistic assumption is made covariate selection can be performed without error, in large samples. Without a beta-min condition, selection make some small mistakes in repeated samples. See Leeb and Pötscher (2006), Leeb and Pötscher (2008), and Pötscher and Leeb (2009) for more details. We do not impose impose a beta-min condition, so we need an estimator that is robust to mistakes made by first-stage covariate selection method.

Estimators that are Neyman orthogonal (NO) are robust to the errors made in the first-stage covariate selection. NO estimators extend a technique derived in Neyman (1959) and Neyman (1979). Belloni, Chen, Chernozhukov, and Hansen (2012), Belloni, Chernozhukov, and Hansen (2014), Chernozhukov, Hansen, and Spindler (2015), and Belloni, Chernozhukov, and Wei (2016) developed NO estimators that are robust to covariate-selection mistakes.

Models with many covariates that potentially might need to be included are called high-dimensional models. High-dimensional models that are subject to a sparsity constraint are called sparse high-dimensional models. High-dimensional models allow the number of potential covariates p to be greater than the sample size N. Ultra-high-dimensional models allow p to be much larger N. Ultra-high-dimensional models that obey a sparsity condition are called sparse ultra-high dimensional models. The methods and software discussed in the paper are useful for the case in which p is greater than N or p is much greater than N.

Drukker and Liu (2022) studied the finite-sample performance of some NO estimators for a sparse high-dimensional generalized linear model (GLM). The NO estimators differed in their covariate-selection techniques, which included the Lasso, and BIC stepwise. Both the Lasso-based NO estimator and BIC-stepwise-based NO estimator peformed well for some of the data-generating processes (DGPs). But Drukker and Liu (2022) found a family of DGPs for which the Lasso-based NO estimator produced unreliable estimates, while the BIC-stepwise-based NO estimator performed well. The price of the better statistical performance came at the cost of dramatically increased computational time. The BIC-stepwise-based NO estimator takes much longer to compute than the Lasso NO estimator.

To solve this computational challenge, we combined the technique of sure-independence-screening (SIS) with the NO estimator so that it can handle many more potential covariates. Fan and Lv (2008a) proposed the sure independence screening method for linear, sparse ultra-high-dimensional models. Fan et al. (2009a) and Fan and Song (2010a) extended this method to the generalized linear model. The SIS technique first prescreens the potential controls to dramatically reduce the number of potential controls from the original high-dimensional set to a much smaller set. Then a Lasso or BIC-based stepwise method is used to select from the smaller set.

To improve the selection accuracy, Fan and Lv (2008b) also proposed the iterative sure independence screening (ISIS). This procedure uses the joint covariate information

while keeping the computation fast. Fan et al. (2009b) and Fan and Song (2010b) extended the ISIS framework to the generalized linear model.

Drukker and Liu (2022) provides simulation evidence that the ISIS-Lasso estimator and the ISIS-BIC-based-stepwise NO estimator perform as well as their counterparts without using ISIS while dramatically reducing the computational time. Note that ISIS reduces the computation time and the increases the selection accuracy when p is greater than N. The methods and the software discussed in the article improve the computation and the statistical results when p is greater than N.

In this paper, we describe two Stata commands: 1) `isis` implements the covariate selector that combines a version of ISIS with Lasso or BIC stepwise techniques for the linear, logit, and Poisson models; 2) `posis` implements the ISIS-based NO estimator for the linear, logit, and Poisson models. This paper can also be seen as a follow-up paper to Drukker and Liu, in which we described `posw` that implements the BIC-stepwise NO estimator without using the ISIS techniques.

We organize this paper as follows. Section 2 describes the algorithms for the iterative version of sure-independence-screening and the SIS-based NO estimator. Section 3 provides details on the syntax of `isis` and `posis`. Section 4 illustrates the use of `isis` and `posis` through some numerical examples. Section 5 presents simulation results. Finally, section 6 concludes.

2 Algorithms

2.1 High-dimensional GLM

Consider the high-dimensional generalized linear model (GLM)

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

where

- y_i is the outcome variable.
- $\mathbf{d}_i$ is a low-dimensional vector of predefined variables of interest. We want to make inferences about the corresponding coefficients $\boldsymbol{\alpha}$.
- $\mathbf{x}_i$ is a ultra-high-dimensional, or high-dimensional, vector of controls. $\mathbf{x}_i$ has dimension $1 \times p$ and $\boldsymbol{\beta}$ is the corresponding coefficient vector. The methods discussed below are useful when $p > N/\ln(N)$.
- $G()$ is the link function, which is the identity function, the exponential function, or the logit function for the linear model, Poisson/Expoential mean model, and the logit model, respectively.

We assume $\boldsymbol{\beta}$ is sparse, so only s^* of the elements in $\boldsymbol{\beta}$ are nonzero. (s^* is a small number, where small is relative to the sample size.) Of course, we do not know which s^* elements in $\boldsymbol{\beta}$ have nonzero coefficients, so covariate selection is required.

In this section, we describe the algorithm of ISIS when using Lasso or BIC stepwise as a covariate-selection technique for the linear, logit, and Poisson models. We also present the algorithm for the ISIS-based NO estimator for coeficients of interest in a sparse high-dimensional GLM model. Although we only use ISIS in practice , the original SIS can help us understand the intuition. Section 2.2 presents the algorithm for the original SIS. Section 2.3 describes the ISIS algorithm for the GLM model. Section 2.4 shows the algorithm of ISIS-based NO estimator.

2.2 Original sure independence screening

When we talk about covariate selection, there is no need to distinguish between the variables of interest and the controls. Thus we use a compact notation.

$$E(y_i|\mathbf{w}) = G(\mathbf{w}_i\boldsymbol{\theta}') \tag{2}$$

Where $\mathbf{w}_i$ is an ultra-high-dimensional vector or a high-dimensional vector, of potential controls and $\boldsymbol{\theta}$ is the corresponding coefficient vector.

The original SIS algorithm consists of two steps. First, pick q variables by ranking the marginal maximum likelihood estimates (MMLEs). Second, use Lasso or BIC-based stepwise to select variables among the q variables picked in the first step.

The MMLE for θ_j, the jth element in $\boldsymbol{\theta}$, is the solution of the component-wise maximum likelihood estimation.

$$(\hat{\theta}_j^M, \hat{\theta}_0^M) = \arg\max_{\theta_0,\theta_j} \sum_{i=1}^{N} Q(y_i, \theta_0 + w_{i,j}\theta_j) \tag{3}$$

For each j in $\{1, \ldots, p\}$, we compute the MMLE and then rank them by their magintudes. Only the top q variables are selected. Denote $\widehat{\mathcal{A}}_1$ as the index of the selected variables. $\widehat{\mathcal{A}}_1$ is defined as

$$\widehat{\mathcal{A}}_1 = \{1 \leq j \leq p : |\hat{\theta}_j^M| \geq \delta_n\} \tag{4}$$

where the threshold δ_n is set such that only q variables are selected. In theory, q is $O(\lfloor N/\log N \rfloor)$, step (a) in algorithm provides the value of q for each model. This screening step effectively reduces the model size from p to q, a number much below N.

Let $\mathcal{M}_*$ be the true model. Under some regularity regularity conditions that restrict the correlations between the w_j variables and that enforce a sparsity constraint, then the probability that $\widehat{\mathcal{A}}_1$ contains the true model $\mathcal{M}_*$ converges to 1. Precisely, given a sequence of δ_n,

$$P(\mathcal{M}_* \subset \widehat{\mathcal{A}}_1) \to 1$$

$\widehat{\mathcal{A}}_1$ may also contain many unimportant variables, so the second step is to use a covariate-selection technique to reduce the model further. For example, we can use Lasso with penalty parameters selected by the plugin method to select variables among the q variables screened in the first step.

Algorithm 1 provides details of the implementation of the original SIS. All the control variables are assumed to be normalized to have a mean of 0 and a standard deviation of 1.

Algorithm 1: Original SIS

1. Set initial conditions.

   a. Set q, the screening model size. By default, the value of q depends on the model. In particular,

q	Model
$\lfloor N/log(N) \rfloor$	linear
$\lfloor N/(4log(N)) \rfloor$	logit
$\lfloor N/(2log(N)) \rfloor$	Poisson

   b. Set variable selection technique. It can be Lasso or BIC-based stepwise.

2. For $j \in \{1, \ldots, p\}$, compute the marginal maiximum likihood esimator (MMLE) $\hat{\theta}_j^M$ as

$$(\hat{\theta}_j^M, \hat{\alpha}^M) = \arg\max_{\alpha,\theta_j} \sum_{i=1}^{N} Q(y_i, \alpha + w_{i,j}\theta_j)$$

3. Let $\widehat{\mathcal{A}}_1$ be the set of indicies of the q largest-in-magnitude estimated coeffcients. Formally, let $\tilde{\theta}^M$ contain the values of $|\hat{\theta}_j^M|$, sorted from largest to smallest. Let $\delta = \tilde{\theta}_q^M$, which the q(th) largest $|\hat{\theta}_j^M|$.

   Denote $\widehat{\mathcal{A}}_1$ as the index of the selected variables.

$$\widehat{\mathcal{A}}_1 = \{1 \leq j \leq p : |\hat{\theta}_j^M| \geq \delta\}$$

4. Use the specified covariate selection technique to select variables among the q variables in $\widehat{\mathcal{A}}_1$. The final selected variables are denoted as $\widehat{\mathcal{M}}_1$.

2.3 Iterative sure independence screening

The original SIS procedure fails (1) if there is a variable that is marginally unrelated but jointly related to the outcome, or (2) if there is a variable that is jointly unrelated to the outcome but has a higher marginal correlation with the outcome than some other variables that belong in the model. In the first scenario, an important variable will be omitted. In the second case, the selected covariates will include unimportant variables and will exclude some important features from the model.

To overcome the limitation of the original SIS, Fan and Lv (2008b) also proposed the ISIS algorithm. It uses joint covariate information, while keeping the computations fast. Fan et al. (2009b) extends it to the GLM model.

In order to use the joint covariate information, Fan et al. (2009b) introduces the screening framework using the conditional marginal maximum likelihood estimates (CMMLEs). Let $\widehat{\mathcal{M}}_1$ be the index of the selected model in the original SIS and let j be an index not in $\widehat{\mathcal{M}}_1$, then the CMMLE for θ_j conditional on $\widehat{\mathcal{M}}_1$ is defined as the solution to the following optimization problem.

$$(\tilde{\theta}_j^M, \tilde{\theta}_0^M, \tilde{\boldsymbol{\theta}}_{\widehat{\mathcal{M}}_1}^M) = \underset{\theta_0, \boldsymbol{\theta}_{\widehat{\mathcal{M}}}, \theta_j}{\arg\max} \sum_{i=1}^{N} Q(y_i, \theta_0 + \mathbf{w}_{i,\widehat{\mathcal{M}}_1} \boldsymbol{\theta}'_{\widehat{\mathcal{M}}_1} + w_{i,j}\theta_j) \tag{5}$$

For each $j \notin \widehat{\mathcal{M}}_1$, we compute their CMMLE estimate and rank them by their magnitude. Only $k_2 = q - |\widehat{\mathcal{M}}_1|$ variables are selected and denote $\widehat{\mathcal{A}}_2$ as the index of selected variables. $\widehat{\mathcal{A}}_2$ is defined as

$$\widehat{\mathcal{A}}_2 = \{j \notin \widehat{\mathcal{M}}_1 : |\tilde{\theta}_j^M| \geq \delta_n\} \tag{6}$$

Again, δ_n is set such that only k_2 variables are picked.

Then, we use a covariate selection technique such as Lasso to perform covariate selection among all the variables selected in $\widehat{\mathcal{M}}_1$ and $\widehat{\mathcal{A}}_2$. Namely, we select among the covariates in $\widehat{\mathcal{M}}_1 \cup \widehat{\mathcal{A}}_2$. We can understand this step in two ways. First, $\widehat{\mathcal{A}}_2$ can be seen as the important variables omitted in $\widehat{\mathcal{M}}_1$, and then it is natural to merge it with $\widehat{\mathcal{M}}_1$. Second, the covariate selection is necessary to delete any unimportant variables in $\widehat{\mathcal{M}}_1$ and $\widehat{\mathcal{A}}_1$. Denote the selected covariates in this step as $\widehat{\mathcal{M}}_2$.

Based on $\widehat{\mathcal{M}}_2$, we can repeat the above procedure until the covariate selection converges or the maximum number of iterations is attained. Algorithm 2 describes the details of the iterative SIS procedure. All the control variables are assumed to be normalized to have a mean of 0 and a standard deviation of 1.

Algorithm 2: Iterative SIS

1. Set initial conditions.

   a. Set q, the screening size. (See Algorithm (1) for details)

   b. Set l_{max}, the maximum number of iteration

   c. Set $\widehat{\mathcal{M}}_0$, the initial set of selected variables, to be an empty set

   d. Set the selection covariate technique. It can be Lasso, or BIC-based stepwise.

2. For l from 1 to l_{max}, do the following loop.

   a. For each $j \notin \widehat{\mathcal{M}}_{l-1}$, compute the CMMLEs $\hat{\theta}_j^M$ conditional on the set of covariates defined in $\widehat{\mathcal{M}}_{l-1}$

   $$(\tilde{\theta}_j^M, \tilde{\alpha}_0^M, \tilde{\boldsymbol{\theta}}_{\widehat{\mathcal{M}}_{l-1}}^M) = \underset{\alpha, \boldsymbol{\theta}_{\widehat{\mathcal{M}}}, \theta_j}{\arg\max} \sum_{i=1}^{N} Q(y_i, \alpha + \mathbf{w}_{i,\widehat{\mathcal{M}}_{l-1}} \boldsymbol{\theta}'_{\widehat{\mathcal{M}}_{l-1}} + w_{i,j}\theta_j)$$

   b. Let $\widehat{\mathcal{A}}_l$ be the set of indicies of the k_l largest-in-magnitude estimated coeffcients, where

   $$k_l = \begin{cases} \lfloor \frac{2q}{3} \rfloor & \text{if } l = 1 \\ q - |\widehat{\mathcal{M}}_{l-1}| & \text{if } l > 1 \end{cases}$$

   Formally, let $\check{\theta}^M$ be $p - |\widehat{\mathcal{M}}_{l-1}|$ vector containing the values of $|\tilde{\theta}_j^M|$ $(j \notin \widehat{\mathcal{M}}_{l-1})$, sorted from largest to smallest. Let $\delta = \check{\theta}_{k_l}^M$, which is the k_l(th) largest $|\tilde{\theta}_j^M|$.

   Denote the selected variables as $\widehat{\mathcal{A}}_l$.

   $$\widehat{\mathcal{A}}_l = \{j \notin \widehat{\mathcal{M}}_{l-1} : |\tilde{\theta}_j^M| \geq \delta\}$$

   c. For all the variables in $\widehat{\mathcal{A}}_l \cup \widehat{\mathcal{M}}_{l-1}$, apply the specified covariate-selection technique. Denote the selected variable as $\widehat{\mathcal{M}}_l$.

   d. Exit the loop if $|\widehat{\mathcal{M}}_l| = q$, or $\widehat{\mathcal{M}}_l = \widehat{\mathcal{M}}_j$ for some $j < l$, or $l = l_{max}$.

3. The selected covariates are $\widehat{\mathcal{M}}_l$.

2.4 Sure-independence-screening-based NO GLM estimaion

Belloni, Chernozhukov, and Wei (2016) derived lasso-based NO estimators for the GLM model. We implement the versions of NO estimators that use the iterative sure independence screening for covariate selection. Algorithm 3 provides the details about these versions of the Belloni, Chernozhukov, and Wei (2016) NO estimator. For inference, we need to distinguish the variables of interests and controls. Thus, we come back to use the notation in Equation (1).

Algorithm 3: ISIS-based NO GLM estimation

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

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 $j \in \{1, \ldots, J\}$, use the ISIS for 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.

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 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 $\mathbf{z} = (\widetilde{d}_1, \ldots, \widetilde{d}_J)$ 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 Syntax

We document the syntax and options for **isis** and **posis** as follows.

3.1 Syntax of isis

`isis` *depvar controls* [*if*] [*in*] [`weight`] , `model`*(model_spec)*
[`method(`*method_spec*`)` `always(`*varlist*`)` `maxiter(#)`]

where *depvar* is a *varname*, *controls* is a *varlist*, and only `fw` and `iw` are allowed as the weights.

Options for isis

controls specifies the set of control variables, which control for omitted variables. Control variables are also known as observed confounding variables and as covariates.

`model(`*model_spec*`)` specifies the model. *method_spec* is one of `linear`, `logit`, or `poisson`. Option `method()` is required.

`method(`*method_spec)* specifies the covariate selection technique to be used within sure independence screening. The default is `method(lasso, bic)`.

The syntax of *method_spec* is one of the following

method_spec	Description
`stepbic`	BIC-based stepwise
`lasso`, *lasso_spec*	lasso

lasso_spec specifies how to choose the tuning parameter in lasso. It is one of the following

lasso_spec	Description
`cv`	cross-validation
`plugin`	plug-in method
`adaptive`	adaptive lasso
`bic`	minimize BIC

`always(`*varlist*`)` specifies the variables will always be included in the model. The default is none.

`maxiter(#)` specifies the maximum number of iterations. The default is 5.

3.2 Syntax of posis

The syntax of **posis** is

`posis` *depvar varsofinterest* [*if*] [*in*] , `controls`*(varlist)* `model`*(model_spec)*
[`method(`*method_spec*`) maxiter(#)`]

where *depvar* is a *varname*, *varsofinterest* are variables for which coefficients and their standard errors are estimated.

Options for posis

`controls(`*varlist*`)` specifies the set of control variables, which control for omitted variables. Control variables are also known as observed confounding variables and as covariates. Option `controls()` is required.

`model(`*model_spec*`)` specifies the model. *method_spec* is one of `linear`, `logit`, or `poisson`. Option `method()` is required.

`method(`*method_spec)* specifies the variable selection technique to be used within sure independence screening. The default is `method(lasso, bic)`.

The syntax of *method_spec* is one of the following

method_spec	Description
`stepbic`	BIC-based stepwise
`lasso`, *lasso_spec*	lasso

lasso_spec specifies how to choose the tuning parameter in lasso. It is one of the following

lasso_spec	Description
`cv`	cross-validation
`plugin`	plug-in method
`adaptive`	adaptive lasso
`bic`	minimize BIC

`maxiter(#)` specifies the maximum number of iterations. The default is 5.

3.3 Stored results

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

Scalars

`e(N)`	number of observations
`e(screen_size)`	screen size
`e(iter)`	actual number of iterations
`e(maxiter)`	maximum number of iterations
`e(k_controls)`	number of controls
`e(k_controls_sel)`	number of selected controls

Macros

`e(cmd_extend)`	isis
`e(title)`	title in estimation output
`e(selopt)`	selection method suboptions
`e(selcmd)`	selection method command
`e(depvar)`	dependent variable
`e(model)`	type of model
`e(allvars_sel)`	name of the selected variables
`e(allvars)`	name of all the variables

Matrices

`e(b)`	coefficient vector

Functions

`e(sample)`	marks estimation sample

posis 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
`e(rank)`	rank of `e(V)`

Macros

`e(cmd)`	posis
`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 Numerical examples

We will illustrate the use of **isis** and **posis** through some examples. In these examples, we use the simulated data in `test.dta`, which has 500 observations. In test.dta, the

dependent variable is `y`, there are 3 variables of interest named `d1`, `d2`, and `d3`, and there are 597 covariates which are named as `x1 - x597`. When we combine the variables of interest with the covariates, there 600 variables. Note that there are more candidate variables than there are observations.

After reading in `test.dta` into Stata, we define some global macros to save some typing. We put the names the variables of interest into the global macro `dvars`. We put the names of the control variables into the global macro `controls`. And, finally, we put the names of the variables of interest and the names of the control variables into the global macro `allvars`.

```
. use test, clear
.
. global dvars d*
. global controls x*
. global allvars d* x*
```

Example 1: covariate selection with isis

In this example, we illustrate the use of **isis** for an outcome y that we would model using a linear regression. In this example, we use the default covariate-selection method, which is the BIC-lasso method. When discussing the covariate-selection techniques such as `lasso` or **isis**, there is no need to distinguish between the variables of interest and the controls. All the covariates are treated as potential variables to be chosen. Thus, we use the global macro `allvars` to specify the list of potential covariates. We specify option `model(linear)` to conduct a linear regression covariate selection. Finally, we specify option `method(stepbic)` to use the BIC-based stepwise method as the covariate selection technique.

```
. isis y $allvars, model(linear)
Iteration 1:
   CMMLE over 600 variables
   SIS picked 53 variables
   lasso selected 11 variables among 53 controls
Iteration 2:
   CMMLE over 589 variables
   SIS picked 69 variables
   lasso selected 11 variables among 80 controls
Covariate selection converged.
Iterative sure independence screening    Number of obs               =    500
                                         Max number of iterations    =      5
                                         Actual number of iterations =      2
                                         Screen size                 =     80
model : linear                           Number of controls          =    600
method: lasso bic                        Number of selected controls =     11
------------------------------------------------------------------------------
```

y	Coefficient
x4	.1752422
x3	.2264602
d1	.1980964
x10	.1969763
x2	.0653429
x5	.074989
x9	.0693233
x8	.0917194
x285	-.1052104
x428	-.1147566
x511	.1376979
_cons	-.0704456

As discussed in algorithm 2, each iteration of ISIS has three steps. First, in step 2a), it selects a set of variables to include in the CMMLE set. In the output above, each line of `CMMLE over # variables` says how many variables were searched over in the CMMLE step in that iteration. Second, in step 2b), the ISIS algorithm selects a subset of the CMMLE variables into an SIS set. The SIS set is denoted $\widehat{\mathcal{A}}_l$ in the algorithm and the `isis` output above reports the number of variables in $\widehat{\mathcal{A}}_l$ as the number of variables that `SIS picked`, for each iteration. Third, in step 2c), the ISIS algorithm uses the lasso or BIC stepwise to select a candidated set of covariates. For each iteration, the output above reports the number of variables selected by BIC lasso.

When the set of candidate covariates selected by the lasso or BIC stepwise does not change from one iteration to the next, the ISIS algorithm has converged. In this example, the ISIS algorithm converged after two iterations.

The output table reports estimated coefficients for each of the variables included in the converged set of covariates.

Here are some explanations about the output-header information.

- The "Number of selected controls" is 11, which implies that only 11 variables are chosen from all the 600 controls.
- The "Max number of iterations" is 5, the default number of maximum iterations. The "Actual number of iterations" is 2 because the covariate selection converged after 2 iterations.
- The "Screen size" is 80, which implies that the BIC lasso will only do covariate selection among 80 variables at the maximum. Compared with doing BIC lasso on the original 600 variables, this step dramatically decreases the computational burden. Note that the screen size is obtained via $\lfloor 500/\ln(500) \rfloor = 80$; see the table in step 1a) of algorithm 1.
- Unlike a traditional estimation command such as `regress`, **isis** does not report

standard errors or t-tests against zero because these inferential statistics are unreliable without making an unrealistic beta-min assumption.

To see the selected covariates, we can display the ereturn results `e(allvars_sel)`.

```
. di `"`e(allvars_sel)´"´
x4 x3 d1 x10 x2 x5 x9 x8 x285 x428 x511
```

The selected covariates can be used either for prediction or inference. For prediction, we can use the selected variables and their coefficients estimates to construct the out-of-sample prediction. For inference, we need to construct an NO estimator using the **isis** as the covariate-selection technique and make inference on the variables of interests, which is implemented in **posis**.

Example 2 illustrates an example of using **isis** for prediction. Example 3 illustrates an example of using **posis** for inference.

Example 2: prediction with isis

For prediction, we first need to split the original data into training and testing samples. The training data is used to perform covariate selection, while the testing data is used to evaluate the out-of-sample prediction performance based on the selected covariates. We use `splitsample` to split the data into two parts. 60% of the data is used as the training sample, while the remaining part is used as a testing sample. The new variable `gr` indicates the type of sample.

```
. splitsample, generate(group) split(0.6 0.4)
. label define lb  1 "Training" 2 "Testing"
. label values group lb
. tabulate group
```

group	Freq.	Percent	Cum.
Training	300	60.00	60.00
Testing	200	40.00	100.00
Total	500	100.00	

Next, we use **isis** to peform covariate selection in the training sample. Notice that we add "`gr == 1`" to restrict the estimation sample to be training data only. We also specify option `method(lasso, plugin)` to use lasso with plugin penalty parameter as a covariate-selection technique. The interpretation of the output is the same as in Example 1. For later reference, we store the estimation results as `isis`.

```
. isis y $allvars if group == 1, model(linear)
Iteration 1:
```

```
   CMMLE over 600 variables
   SIS picked 34 variables
   lasso selected 4 variables among 34 controls
Iteration 2:
   CMMLE over 596 variables
   SIS picked 48 variables
   lasso selected 7 variables among 52 controls
Iteration 3:
   CMMLE over 593 variables
   SIS picked 45 variables
   lasso selected 7 variables among 52 controls
Covariate selection converged.
Iterative sure independence screening    Number of obs               =    300
                                         Max number of iterations    =      5
                                         Actual number of iterations =      3
                                         Screen size                 =     52
model : linear                           Number of controls          =    600
method: lasso bic                        Number of selected controls =      7
```

y	Coefficient
x460	-.1841022
x8	.080793
x16	.0753784
x4	.212914
x3	.2167369
x10	.2431481
d1	.2126844
_cons	-.0549942

```
. estimate store isis
```

As a comparison with **isis** result, we also run a lasso with plugin penalty parameter. The result is stored as `lasso`.

```
. lasso linear y $allvars if group == 1, sel(plugin)
Computing plugin lambda ...
Iteration 1:     lambda = .2670831   no. of nonzero coef. =       4
Iteration 2:     lambda = .2670831   no. of nonzero coef. =       4
Lasso linear model                          No. of obs        =        300
                                            No. of covariates =        600
Selection: Plugin heteroskedastic
```

ID	Description	lambda	No. of nonzero coef.	In-sample R-squared	BIC
* 1	selected lambda	.2670831	4	0.2206	920.7542

```
* lambda selected by plugin formula assuming heteroskedastic errors.
. estimate store lasso
```

Based on the selected covariates, we can evaluate their out-of-sample prediction performance using `lassogof`.

```
. lassogof isis lasso, over(group)
Penalized coefficients
```

Name	group	MSE	R-squared	Obs
isis				
	Training	.911803	0.3798	300
	Testing	.8957799	0.4089	200
lasso				
	Training	1.145958	0.2206	300
	Testing	1.148989	0.2418	200

The R-squared or the MSE in the testing sample measures the out-of-sample prediction performance. The greater the R-squared (or the smaller the MSE), the better the prediction is. Clearly, the lasso-based sure independence screening is better than the straight lasso in this example. As a result, we can use the `isis` result to make the prediction.

Finally, we load a data `test_new.dta`, which does not contain the dependent variable. We use the results from **isis** to predict the outcome.

```
. use test_new, clear
. estimates restore isis
(results isis are active now)
. predict yhat
(option xb assumed; fitted values)
```

Example 3: inference with posis

We can use **isis** to select the covariates for predictive model, in which case we think of the naive estimator that uses the selected covariates as producing estimates that minimize a bias-variance trade off. But we can not do reliable inference because covariate selection techniques such as ISIS make unavoidable mistakes. If we want to make inferences on a subset of parameters in the model, we need to use **posis**, which uses the NO approach to construct an estimator that is robust to the covariate-selection mistakes.

For example, in the data `test.dta`, we want to make inference on the coefficients for `d1`, `d2`, and `d3` while treating all the other variables as potential controls. The global macro `dvars` defines the variables of interest. (We want to make inferences the coefficients on the variables in `dvars`.) The global macro `controls` defines the control variables. **posis** uses **isis** to select the relevant controls as part of the NO estimator given in algorithm 3.

Here is the **posis** output. We specify `controls($controls)` to use control variables defined in the global macro `controls`. We use `model(linear)` for the linear model. Finally, we specify `method(stepbic)` to use the BIC-stepwise ISIS as the covariate-selection technique.

```
. use test, clear
. posis y $dvars, controls($controls) model(linear) method(stepbic)
select controls for y using ISIS stepbic
select controls for d1 using ISIS stepbic
select controls for d2 using ISIS stepbic
select controls for d3 using ISIS stepbic
Partialing-out ISIS                Number of obs                 =        500
                                   Number of controls            =        597
                                   Number of selected controls   =         43
Method: stepbic                    Wald chi2(3)                  =      16.49
Model: linear                      Prob > chi2                   =     0.0009
```

y	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
d1	.1625513	.0425851	3.82	0.000	.0790859	.2460166
d2	.027177	.0403703	0.67	0.501	-.0519474	.1063014
d3	.0230802	.0443444	0.52	0.603	-.0638333	.1099937

```
Note: Chi-squared test is a Wald test of the coefficients of the variables of
      interest jointly equal to zero.
```

After we note that only 43 of the 597 controls were selected in the NO estimator, we can interpret the results as we would for `regress`. For example, the estimated coefficient on `d1` is 0.163, which implies that an increase of one unit in `d1` will increase the dependent variable by an estimated amount of 0.163. Standard inference tools such as `test` or `testnl` can also be used to test different hypotheses. For example, we use the `sotable`, discussed in Drukker (2022), to get an output table with p-values and a confidence band that accounts for the multiple comparisons made.

```
. sotable
Max-t results
       p-value = 0.000
Critical value = 2.385
```

y	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Band]
d1	.1625513	.0425851	3.817	0.000	.0609704	.2641322
d2	.027177	.0403703	0.673	0.875	-.0691208	.1234747
d3	.0230802	.0443444	0.520	0.937	-.0826972	.1288576

As expected, the non-zero p-values are much larger and the confidence intervals are much wider, after we account for comparing all three parameters with zero.

5 Simulations

We conducted a simulation study to measure the finite-sample statistical performance of **posis** and to measure the computational-speed gain of **posis** compared with other NO estimators that don't use an ISIS covariate-selection technique. As a benchmark, we also use a ISIS-based naive estimator to illustrate that the naive estimator cannot provide reliable inference while the partialling-out estimator can. Thus, we compare two versions of NO estimators in simulations:

- **naive_isis**: A naive estmiator that uses the BIC-stepwise ISIS technique as the covariate-selection method. This estimator is implemented as a straightforward `isis` wth options `method(stepbic)` and `always(dvars)`, where `dvars` are variables of interest that we want to make inference.
- **posis**: An NO estimator that uses the BIC-stepwise ISIS technique as the covariate-selection method; this estimator is implemented in `posis` with option `method(stepbic)`.
- **posw**: An NO estimator that uses a BIC-stepwise as the covariate-selection method, without the ISIS framework; this estimator is implemented in `posw` with option `method(bic)`.

5.1 Designs

We compare these two estimators in linear, logit, and Poisson models. The DGP for the three models can be summarized in the following steps.

1. The outcome variable y_i is generated as

$$\begin{aligned} \text{linear} \quad & y_i = w_i + \epsilon_i \\ \text{logit} \quad & y_i = w_i + \epsilon_i > 0 \\ \text{Poisson} \quad & y_i = \texttt{rpoisson}(\exp(w_i)) \end{aligned}$$

   where w_i is a linear combination of the variables of interest and the control variables (see step 2 for details), ϵ_i is an error term which varies across the models (see Step 3 for details), and `rpoisson(`μ`)` is a Stata function that generates draws from the Poisson distribution with mean μ.

2. The term w_i has a common structure as

$$w_i = d_{big,i}\alpha_{big} + d_{small,i}\alpha_{small} + d_{zero,i} * 0 + \mathbf{x}'_{big,i}\boldsymbol{\beta}_{big} + \mathbf{x}'_{small,i}\boldsymbol{\beta}_{small}$$

   where

   - $d_{big,i}$ is a variable of interest with a large coefficient α_{big}.
   - $d_{small,i}$ is a variable of interest with a small coefficient α_{small}.

- $d_{zero,i}$ is a variable of interest with coefficient 0.
- $\mathbf{x}_{big,i}$ is a vector of controls with large coefficient $\boldsymbol{\beta}_{big}$.
- $\mathbf{x}_{small,i}$ is a vector of controls with small coefficient $\boldsymbol{\beta}_{small}$.

The covariates (This includes the variables of interests and the control variables) are generated from a type of Toeplitz structure; see Belloni, Chen, Chernozhukov, and Hansen (2012) for an example. These Toeplitz structures allow the covariates to have different degrees of correlation depending on the variable indices. In particular, each covariate has an index $j \in \{1, \ldots, p\}$, where p is the total number of covariates in the model. Covariates with near-by indices values are more correlated than the covariates with far-away indices. Each covariate was generated as

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

For each draw, we drew p covariates and discarded the first 25 to burn in the Toeplitz structure. 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 from a normal distribution.

For each design, we generated p covariates.

- 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-p are $\mathbf{x}_{zero}$, the control covariates whose coefficients are zero.

The size of large coefficients is twice that of the small coefficients. Here are the true values of the coefficients.

Table 1: True values of coefficients

Model	Coefficient	value
linear	α_{big}	0.16
	α_{small}	0.08
	$\boldsymbol{\beta}_{big}$	0.16
	$\boldsymbol{\beta}_{small}$	0.08
logit	α_{big}	0.32
	α_{small}	0.16
	$\boldsymbol{\beta}_{big}$	0.32
	$\boldsymbol{\beta}_{small}$	0.16
Poisson	α_{big}	0.32
	α_{small}	0.16
	$\boldsymbol{\beta}_{big}$	0.32
	$\boldsymbol{\beta}_{small}$	0.16

The numbers of observations (N) and the number of covariates (p) for each design are defined as

Table 2: Values of N and p

Model	N	p
linear	1000	600
logit	2000	1000
Poisson	500	600

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

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

We ran 3,000 repetitions for each design.

5.2 Results

Tables 3–5 summarize the simulation results for `posis` and `posw` estimators. In short, both partialing-out estimators perform well in terms of consistency and the rejection rate while the naive estimator can not provide valid inference. However, `posis` is much faster than `posw`, see details in Table 6. [1]

1. The simulation times are average computation time over 3,000 repetitions.

For each model and estimator, the simulation results in Tables 3–5 are summarized as follows.

- Mean specifies the sample mean of the point estimate of the coefficient over the simulation repetitions. The value of the mean should be close to the true values in Table 1.

- SD specifies the standard deviation of the point estimates.

- SE specifies the sample mean of the standard errors over the repetitions. The sample mean of standard errors should be close to the standard deviation of the point estimates.

- Rej. Rate specifies the rejection rate of a test against the null hypothesis that the point estimates of coefficient equal to its true value. The significance level of the test is 0.05, so the rejection rate should be close to 0.05.

Table 3: Results for the large coefficient α_{big}

		Mean	SD	SE	Rej. Rate
linear	naive_isis	0.1739	0.0396	0.0317	0.138
	posis	0.1522	0.0351	0.0336	0.069
	posw	0.1522	0.0351	0.0336	0.069
logit	naive_isis	0.3528	0.0675	0.0573	0.140
	posis	0.3330	0.0631	0.0602	0.064
	posw	0.3330	0.0631	0.0602	0.064
poisson	naive_isis	0.3234	0.0379	0.0305	0.117
	posis	0.3202	0.0601	0.0587	0.059
	posw	0.3194	0.0677	0.0666	0.057

Table 4: Results for the small coefficient α_{small}

		Mean	SD	SE	Rej. Rate
	naive_isis	0.0923	0.0376	0.0317	0.117
linear	posis	0.0747	0.0343	0.0341	0.051
	posw	0.0747	0.0343	0.0341	0.051
	naive_isis	0.1867	0.0655	0.0561	0.120
logit	posis	0.1659	0.0625	0.0608	0.058
	posw	0.1659	0.0625	0.0608	0.058
	naive_isis	0.1620	0.0385	0.0322	0.101
poisson	posis	0.1594	0.0653	0.0625	0.061
	posw	0.1595	0.0713	0.0696	0.061

Table 5: Results for the zero coefficient α_{zero}

		Mean	SD	SE	Rej. Rate
	naive_isis	0.0042	0.0335	0.0311	0.068
linear	posis	-0.0006	0.0336	0.0341	0.047
	posw	-0.0006	0.0336	0.0341	0.047
	naive_isis	0.0052	0.0591	0.0544	0.074
logit	posis	0.0011	0.0624	0.0607	0.058
	posw	0.0011	0.0624	0.0607	0.058
	naive_isis	0.0007	0.0380	0.0320	0.096
poisson	posis	-0.0013	0.0654	0.0631	0.055
	posw	-0.0008	0.0720	0.0698	0.055

Table 6: Results for computation time

		Timing (seconds)
	naive_isis	58.00
linear	posis	198.19
	posw	276.01
	naive_isis	183.67
logit	posis	630.26
	posw	1019.14
	naive_isis	144.67
poisson	posis	555.33
	posw	1524.12

6 Conclusion

We motivate and present Stata commands `isis` and `posis`. `isis` implements the iterative sure-independence screening (ISIS) combined with the Lasso or stepwise covariate-selection technique. `posis` implements a ISIS-based NO estimator for the inference in the high-dimensional linear, logit, and Poisson model. `posis` can be viewed as an extension of `posw`, which implements a NO estimator using straightforward stepwise methods. The simulations show that, compared with `posw`, `posis` can greatly improve the computational speed while maintaining similar or better statistical performance.

Appendix

The $Q()$ is defined as

- For linear models,
$$Q(y_i, \mathbf{x}_i\boldsymbol{\beta}') = \left(y_i - \mathbf{x}_i\boldsymbol{\beta}'\right)^2$$
- For Poisson models
$$Q(y_i, \mathbf{x}_i\boldsymbol{\beta}') = -[y_i\mathbf{x}_{i,t}\boldsymbol{\beta}' - \exp(\mathbf{x}_i\boldsymbol{\beta}') - \ln(y_i!)]$$
- For logit models
$$Q(y_i, \mathbf{x}_i\boldsymbol{\beta}') = \ln[1 + \exp(\mathbf{x}_i\boldsymbol{\beta}')] - y_i(\mathbf{x}_i\boldsymbol{\beta})$$

1 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., 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. `posw`: A Stata Command for the stepwise Neyman-orthogonal estimator. Accepted by The Stata Journal .

Drukker, D. M. 2022. Simultaneous tests and confidence bands for Stata estimation commands. Accepted by The Stata Journal .

Drukker, D. M., and D. Liu. 2022. Finite-sample results for lasso and stepwise Neyman-orthogonal Poisson estimators. Econometric Reviews 41(9): 1047–1076.

Fan, J., and J. Lv. 2008a. Sure independence screening for ultrahigh dimensional feature space. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 70(5): 849–911.

———. 2008b. Sure independence screening for ultrahigh dimensional feature space. Journal of the Royal Statistical Society. Series B: Statistical Methodology 70(5): 849–911. https://www.jstor.org/stable/20203862.

Fan, J., R. Samworth, and Y. Wu. 2009a. Ultrahigh dimensional feature selection: beyond the linear model. The Journal of Machine Learning Research 10: 2013–2038.

———. 2009b. Ultrahigh Dimensional Feature Selection: Beyond the Linear Model. Journal of Machine Learning Research 10: 2013–2038.

Fan, J., and R. Song. 2010a. Sure independence screening in generalized linear models with NP-dimensionality. The Annals of Statistics 38(6): 3567–3604.

———. 2010b. Sure independence screening in generalized linear models with NP-dimensionality. Annals of Statistics 38(6): 3567–3604.

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.

———. 1979. C(α) tests and their use .

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.

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.