The Stata Journal

The *Stata Journal* publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go "beyond the Stata manual" in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those who teach, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

The *Stata Journal* is indexed and abstracted by *CompuMath Citation Index*, *Current Contents/Social and Behavioral Sciences*, *RePEc: Research Papers in Economics*, *Science Citation Index Expanded* (also known as *SciSearch*, *Scopus*, and *Social Sciences Citation Index*.

For more information on the *Stata Journal*, including information for authors, see the webpage

http://www.stata-journal.com

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates listed below include both a printed and an electronic copy unless otherwise mentioned.

U.S. and Canada		Elsewhere	
Printed & electronic		**Printed & electronic**	
1-year subscription	$ 98	1-year subscription	$138
2-year subscription	$165	2-year subscription	$245
3-year subscription	$225	3-year subscription	$345
1-year student subscription	$ 75	1-year student subscription	$ 99
1-year university library subscription	$125	1-year university library subscription	$165
2-year university library subscription	$215	2-year university library subscription	$295
3-year university library subscription	$315	3-year university library subscription	$435
1-year institutional subscription	$245	1-year institutional subscription	$285
2-year institutional subscription	$445	2-year institutional subscription	$525
3-year institutional subscription	$645	3-year institutional subscription	$765
Electronic only		**Electronic only**	
1-year subscription	$ 75	1-year subscription	$ 75
2-year subscription	$125	2-year subscription	$125
3-year subscription	$165	3-year subscription	$165
1-year student subscription	$ 45	1-year student subscription	$ 45

Back issues of the *Stata Journal* may be ordered online at

http://www.stata.com/bookstore/sjj.html

Individual articles three or more years old may be accessed online without charge. More recent articles may be ordered online.

http://www.stata-journal.com/archives.html

The *Stata Journal* is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the *Stata Journal*, StataCorp, 4905 Lakeway Drive, College Station, TX 77845, USA, or emailed to sj@stata.com.

The *Stata Journal* (ISSN 1536-867X) is a publication of Stata Press. Stata, Stata, Stata Press, Mata, Mata, and NetCourse are registered trademarks of StataCorp LP.

The Stata Journal (2013)
13, Number 3, pp. 588–602

Distribution-free estimation of heteroskedastic binary response models in Stata

Jason R. Blevins
Ohio State University
Columbus, OH
blevins.141@osu.edu

Shakeeb Khan
Duke University
Durham, NC
shakeebk@econ.duke.edu

Abstract. In this article, we consider two recently proposed semiparametric estimators for distribution-free binary response models under a conditional median restriction. We show that these estimators can be implemented in Stata by using the `nl` command through simple modifications to the nonlinear least-squares probit criterion function. We then introduce `dfbr`, a new Stata command that implements these estimators, and provide several examples of its usage. Although it is straightforward to carry out the estimation with `nl`, the `dfbr` implementation uses Mata for improved performance and robustness.

Keywords: st0310, dfbr, binary response, heteroskedasticity, nonlinear least squares, semiparametric estimation, sieve estimation

1 Introduction

In this article, we consider the Stata implementations of two recently proposed semiparametric estimators for distribution-free binary response models of the form

$$y_i = 1\left(x_i'\beta + \varepsilon_i > 0\right) \tag{1}$$

where $y_i \in (0, 1)$ is an observed response variable, x_i is a vector of k observed covariates, ε_i is an unobserved disturbance term, and β is an unknown vector of parameters of interest. Our goal is to estimate β given a random sample of observations $(y_i, x_i)_{i=1}^n$.

Following Manski (1975, 1985) and Horowitz (1992), we impose only a relatively weak conditional median independence condition:

$$\operatorname{med}(\varepsilon_i \mid x_i) = 0$$

More formally, we assume that the distribution of ε_i conditional on x_i almost surely has median 0. Such a restriction ensures point identification of β while allowing for general forms of heteroskedasticity (for example, random coefficients). Thus the estimators we propose are semiparametric.

Alternatively, parametric methods specify the distribution of ε_i up to a finite vector of parameters and typically assume this distribution is independent of x_i. Under such an assumption, one can estimate β using maximum likelihood. However, if the distribution of ε_i is misspecified or heteroskedastic, then the maximum likelihood estimator is generally inconsistent (Yatchew and Griliches 1985). Semiparametric or "distribution-free"

methods avoid these issues by estimating β without making a particular parametric assumption about the distribution of ε_i.

The focus of this article is on the Stata implementation of the sieve nonlinear least-squares (SNLLS) estimator of Khan (2013) and the local nonlinear least-squares (LNLLS) estimator of Blevins and Khan (2013). These estimators have the advantage of consistently estimating the parameters of the potentially heteroskedastic binary choice model above while remaining computationally tractable enough that end users can easily carry out estimation with built-in Stata commands. We focus here on the implementation of these methods and refer the interested reader to the articles cited above for further results and technical details.

This article proceeds as follows. Section 2 briefly reviews Stata's nonlinear least-squares (NLLS) estimation framework and, as a motivating example, first reviews the NLLS probit estimator for a parametric version of the model above with $\varepsilon_i \sim N(0,1)$. Sections 3 and 4 describe, respectively, the LNLLS estimator of Blevins and Khan (2013) and the SNLLS estimator of Khan (2013). We show that both of these estimators can be easily implemented using Stata's `nl` command through simple modifications to the standard NLLS probit regression function. Finally, section 5 describes `dfbr`, a new Stata command that implements these estimators by using high-performance Mata code with analytic derivatives, and then provides several examples of its usage.

2 Nonlinear least-squares estimation in Stata

Stata's `nl` command provides an interface for fitting an arbitrary nonlinear parametric regression function $f(x,\theta) = E(y|x)$ by using least squares. There are three ways to provide the regression function to `nl`: by interactively using a substitutable expression, a substitutable expression program, or a function evaluator program. We focus here on the first approach—using substitutable expressions—because it is straightforward to implement for most simple models, including the ones we discuss in the following sections. See [R] **nl** for further details regarding Stata's NLLS capabilities.

As an example, consider the standard probit regression model

$$E(y_i|x_i) = \Phi(x_i'\beta) \tag{2}$$

where β is a vector of parameters of interest and Φ is the cumulative distribution function (c.d.f.) of the standard normal distribution. This is precisely the model in (1) when $\varepsilon_i \sim N(0,1)$. Given a sample of size n, $(y_i, x_i)_{i=1}^n$, the NLLS estimator $\widehat{\beta}$ of β is defined as a vector that satisfies $Q_n(\widehat{\beta}) = \min_{\beta \in B} Q_n(\beta)$, where the criterion function is

$$Q_n(\beta) = \frac{1}{n}\sum_{i=1}^{n} \left\{ y_i - \Phi\left(x_i'\beta\right) \right\}^2 \tag{3}$$

and where B is the parameter space.

Recall that the standard parametric probit model requires a scale normalization: the scale of β and the variance of ε_i, denoted σ^2, cannot be separately identified. To see

this, note that for any scalar $\alpha > 0$, the model with coefficients $\alpha\beta$ and variance $\alpha^2\sigma^2$ is observationally equivalent because $\Phi\{x'(\alpha\beta)/(\alpha\sigma)\} = \Phi\left(x'\beta/\sigma\right)$. We have imposed the scale normalization in (2) and (3) by setting $\sigma = 1$ and using the standard normal c.d.f., so we can identify and estimate all three slope coefficients. This is the usual scale normalization for the probit model, but an alternative would be to normalize one of the coefficients, say, $\beta_2 = 1$, and then estimate σ.

To make the example more concrete, we will suppose that we have a binary dependent variable y and two independent variables x_1 and x_2, and that the corresponding variables in our Stata dataset are named `y`, `x1`, and `x2`. We wish to estimate the intercept β_0 and the two slope coefficients β_1 and β_2, which we shall denote by `b0`, `b1`, and `b2` in Stata. To fit the model by using the `nl` command, we can express the regression function in (2) as a substitutable expression:

```
nl (y = normal({b0} + {b1}*x1 + {b2}*x2))
```

The expression in parentheses following `y =` is the regression function, and the parameters to estimate appear in braces. Here the function `normal()` evaluates the c.d.f. of the standard normal distribution (see [D] **functions**). Issuing the above command estimates β_0, β_1, and β_2 by minimizing the sum of squared residuals for this model:

$$\sum_{i=1}^{n} \left\{y_i - \Phi\left(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i}\right)\right\}^2$$

In the following sections, we describe two new estimators whose objective functions are also of the NLLS form and, therefore, can be implemented in Stata by using the `nl` command in a similar way. Note, however, that these estimators are for the more general model described in the introduction, which does not require a specific parametric assumption on the error term and which allows for general forms of heteroskedasticity. This differs from the parametric probit model in the example above, where the error term is homoskedastic and normally distributed.

3 The LNLLS estimator

The LNLLS estimator developed by Blevins and Khan (2013) is defined as a vector $\widehat{\beta}$ that satisfies $Q_n(\widehat{\beta}) = \min_{\beta \in \Theta \times 1} Q_n(\beta)$, where

$$Q_n(\beta) = \frac{1}{n} \sum_{i=1}^{n} \left\{ y_i - F\left(\frac{x_i'\beta}{h_n}\right)\right\}^2$$

Here F is a nonlinear regression function, such as a c.d.f., that we will specify below, and h_n is a sequence of positive numbers such that $h_n \to 0$ as $n \to \infty$, similar to a bandwidth sequence used in nonparametric kernel estimation. We adopt the standard semiparametric scale normalization (Horowitz 1992), normalizing the kth element of β to 1 so that $\widehat{\beta} = (\widehat{\theta}', 1)'$. We denote this normalization by using $\Theta \times 1$ as the parameter space.

When we choose F to be Φ, the standard normal c.d.f., then aside from scaling the index $x_i'\beta$ by the bandwidth and normalizing the coefficient on x_k, we use an objective function identical to that of the NLLS probit estimator in (3). Thus to implement the estimator in Stata, we simply need to normalize one component of β and divide the index by h_n.

Using the two-regressor example from before, we will suppose there are $n = 1000$ observations. Then we can fit the model by using the bandwidth $h_n = n^{-1/3} = 0.1$ as follows,

```
nl (y = normal(({b0} + {b1}*x1 + x2) / 0.1))
```

where we have used the normal c.d.f. as the regression function. We normalized the coefficient on x_2 by simply omitting this parameter from the substitutable expression, effectively setting it to 1 and leaving the coefficient on x_1 and the intercept as the only parameters.

Blevins and Khan (2013) show that while the estimator above is consistent, the rate of convergence is only $n^{1/3}$ because the bias converges at the rate h_n, in contrast to the rate h_n^2 for estimators such as the smoothed maximum score estimator.[1] They propose two methods for reducing the order of the bias and consequently improving the rate of convergence to $n^{2/5}$.

The first method is to use a different regression function,

$$F(u) = (1/2 - \alpha_F - \beta_F) + 2\alpha_F\Phi(u) + 2\beta_F\Phi(\sqrt{2}u) \tag{4}$$

where $\Phi(\cdot)$ is the standard normal c.d.f., $\alpha_F = -1/2\left(1 - \sqrt{2} + \sqrt{3}\right)\beta_F$, and $\beta_F \neq 0$. This function was chosen so that a particular term in the asymptotic bias of the estimator equals 0, something that cannot be achieved when $F(\cdot)$ is a c.d.f. In this case, the bandwidth sequence should be proportional to $n^{-1/5}$ to achieve the fastest rate of convergence.

As with the NLLS probit objective function, this function can be expressed entirely using Stata's built-in `normal()` function, for example,

```
local h = _N^(-1/5)
local index "({b0} + {b1}*x1 + x2) / `h´"
local beta = 1.0
local alpha = -0.5 * (1 - sqrt(2) + sqrt(3))*`beta´
local const = 0.5 - `alpha´ - `beta´
nl (y = `const´ + 2*`alpha´*normal(`index´) + 2*`beta´*normal(sqrt(2)*`index´))
```

The second proposed bias-reduction method is to define a jackknife version of the estimator,

$$\widehat{\theta}_{\text{jk}} = w_1\widehat{\theta}_1 + w_2\widehat{\theta}_2$$

where θ_1 and θ_2 are two LNLLS estimators using the normal c.d.f. and bandwidths $h_{1n} = \kappa_1 n^{-1/5}$ and $h_{2n} = \kappa_2 n^{-1/5}$, respectively, and where w_1 and w_2 are weights.

1. Note that although the estimator is defined by an NLLS criterion, the assumptions are quite different, so the estimator does not have the same limiting distribution as the standard NLLS estimator.

The weights and bandwidth constants must satisfy the constraints $w_1 + w_2 = 1$ and $w_1\kappa_1 + w_2\kappa_2 = 0$. The optimal choice of these values is discussed in Blevins and Khan (2013). Note that obtaining the two estimates is no more difficult than obtaining the NLLS probit estimate from before and that constructing the final weighted sum can be accomplished with basic Stata macro programming.

Although here we have emphasized that both estimators can be implemented in Stata manually if needed, the `dfbr` command we introduce below automates the process of obtaining both estimators described above. For the NLLS estimator using the regression function in (4), `dfbr` will automatically estimate the feasible optimal bandwidth sequence, so the user does not have to actually choose the bandwidth. For the jackknife NLLS estimator, the jackknife weights and constants are selected according to the rule of thumb provided by Blevins and Khan (2013). Thus in both cases, the user simply needs to provide the dependent and independent variables.

A final but important reason for providing a dedicated command for these estimators is that although the point estimates reported by `nl` for these estimators are correct, the reported standard errors are not. The point estimates are correct because our estimators are indeed defined by NLLS criteria. On the other hand, the standard errors reported by `nl` are based on the limiting distribution of the NLLS estimator, which is derived under the conditional mean independence assumption $E(\varepsilon_i \mid x_i) = 0$. The assumptions underlying our estimators are different, and our estimators perform smoothing and scaling, so the asymptotic properties are different.

The asymptotic variance–covariance matrices for the estimators described involve unknown density functions that would need to be estimated nonparametrically, so `dfbr` instead reports bootstrap estimates of the standard errors. Although we implement this internally in Mata,[2] this could also be achieved using Stata's `bootstrap` prefix in conjunction with `nl` as in the following example:

```
bootstrap, rep(1001): nl (y = normal(({b0} + {b1}*x1 + x2) / 0.1))
```

4 The SNLLS estimator

Although the objective function for the SNLLS estimator introduced by Khan (2013) is slightly more complex, it is still ultimately a variation on the NLLS probit objective function in (3), and so it is straightforward to obtain estimates by using `nl`. Specifically, the estimator is defined by minimization of the criterion function

$$Q_n(\theta, \ell) = \frac{1}{n}\sum_{i=1}^{n} \left(y_i - \Phi\left[x_i'\beta \times \exp\left\{\ell(x_i)\right\}\right]\right)^2$$

where ℓ is a scaling function—an infinite-dimensional unknown—and $\beta = (\theta', 1)'$ is a finite-dimensional vector of parameters.

2. Specifically, we use the `mm_bs` bootstrap function from the `moremata` package (Jann 2005).

To use NLLS, we introduce a finite-dimensional approximation of ℓ by using a linear-in-parameters sieve estimator. Let $b_{0j}(x_i)$ denote a sequence of known basis functions for $j = 1, \ldots, \kappa_n$ for some integer κ_n, and let $b^{\kappa_n}(x_i) = \{b_{01}(x_i), \ldots, b_{0\kappa_n}(x_i)\}'$. The function $g(x_i) \equiv \exp\{\ell(x_i)\}$ in the above objective function can be approximated by $g_n(x_i) = \exp\{b^{\kappa_n}(x_i)'\gamma_n\}$, where γ_n is a vector of parameters of length κ_n. Let $\alpha_n \equiv (\theta, g_n) \in \mathcal{A}_n$, where $\mathcal{A}_n$ is the sieve space. The estimator can be defined as a vector $\widehat{\alpha}_n \in \mathcal{A}_n$, which minimizes the objective function

$$Q_n(\alpha) = \frac{1}{n}\sum_{i=1}^{n}\left[y_i - \Phi\left\{x_i'\beta \times g_n(x_i)\right\}\right]^2$$

where, as before, $\beta = (\theta', 1)'$.

Under the conditions of Khan (2013), if the number of basis functions κ_n approaches infinity, but slower than n, then this estimator is consistent and asymptotically normal. As is the case with many related semiparametric estimators, the rate of convergence depends on the smoothness of certain unknown functions. In this case, when $\Phi\left[x_i'\beta \times \exp\{\ell(x_i)\}\right]$ has p continuous derivatives and some additional regularity conditions are satisfied, the rate of convergence is $n^{p/(2p+1)}$. For example, when $p = 2$, this rate simplifies to $n^{2/5}$.

The SNLLS estimator has the advantage that choice probabilities and regression coefficients are estimated simultaneously. That is, once $\widehat{\alpha}_n = (\widehat{\theta}, \widehat{g}_n)$ is obtained, choice probabilities $\widehat{P}_i$ can be estimated by substituting these estimates into the regression function as follows:

$$\widehat{P}_i = \Phi\left\{x_i'\widehat{\beta} \times \widehat{g}_n(x_i)\right\}$$

To illustrate the Stata implementation of this estimator, we consider a simple model with two regressors, x_1 and x_2. We approximate the scaling function by using powers of the independent variables and interaction terms up to second order as basis functions:

$$g_n(x_i) = \exp(\gamma_0 + \gamma_1 x_1 + \gamma_2 x_2 + \gamma_3 x_1 x_2 + \gamma_4 x_1^2 + \gamma_5 x_2^2)$$

To fit the model by using `nl`, we construct the corresponding substitutable expression:

```
nl (y = normal(({b0} + {b1}*x1 + x2) * exp({g0} + {g1}*x1      ///
         + {g2}*x2 + {g3}*x1*x2 + {g4}*x1*x1 + {g5}*x2*x2)))
```

Again we have normalized the coefficient on x_2 by omitting the corresponding parameter.

5 The dfbr command

The new `dfbr` command implements each of the estimators described above: the SNLLS estimator of Khan (2013) and both variants of the LNLLS estimator of Blevins and Khan (2013). Rather than constructing substitutable expressions for the modified NLLS probit objective functions and calling Stata's built-in `nl` command, we instead implement the estimators by using the lower-level Mata language. This allows us to use Mata's

`optimize` framework and to provide analytic derivatives during optimization for improved performance and accuracy.

The SNLLS estimator is the default method, but this choice may be made explicit by using the `sieve` option. The user may supply a set of basis variables, such as polynomial terms of the independent variables, by using the `basis()` option. If no basis elements are provided, then the given independent variables and a constant are used.

The LNLLS estimator may be selected using the `local` option. By default, the regression function in (4) is used, and `dfbr` will automatically calculate the feasible optimal bandwidth. Alternatively, the user may override this choice by supplying a custom bandwidth in the `bandwidth()` option.

To select the jackknife LNLLS estimator with the normal c.d.f. as the regression function, the user must provide both the `local` and the `normal` options. The jackknife weights and bandwidth constants are chosen automatically and need not be provided; however, custom bandwidth constants κ_1 and κ_2 can be provided using the `k1()` and `k2()` options, with the corresponding weights being calculated to satisfy the constraints.

For all three estimators, bootstrap-estimated standard errors are reported by default. Both the number of replications and the random-number generator seed can be specified. In all cases, the coefficient on the last independent variable is normalized to 1.

The formal syntax is given below along with a detailed description of each of the options and return values. Some examples are then provided to illustrate the usage.

5.1 Syntax

SNLLS estimation (default)

`dfbr` *depvar indepvars* [*if*] [*in*] [`, sieve basis(`*basis_vars*`) noconstant brep(#) seed(#) level(#) nmiter(#) nmdelta(#)`]

LNLLS estimation

`dfbr` *depvar indepvars* [*if*] [*in*] `, local` [`normal bandwidth(#) k1(#) k2(#) noconstant brep(#) seed(#) level(#) nmiter(#) nmdelta(#)`]

5.2 Options

The `dfbr` command accepts several options, which are listed below. First, the options specific to either SNLLS or LNLLS are listed, followed by the options common to both estimators.

SNLLS

`sieve` specifies to use the SNLLS estimator (the default).

`basis(`*basis_vars*`)` provides a list of variables to use in the linear-in-parameters sieve approximation of the scaling function. An intercept term is automatically included in the scale equation and need not be specified along with the other variables. If this option is omitted, then a constant and the provided independent variables are used.

LNLLS

`local` specifies to use the LNLLS estimator, using the alternative nonlinear regression function by default. `local` is required.

`normal` uses the jackknife LNLLS estimator with the standard normal c.d.f. as the nonlinear regression function. The rule-of-thumb jackknife weights and rate constants described in Blevins and Khan (2013) are used.

`bandwidth(#)` specifies the bandwidth. If this option is omitted, the feasible optimal bandwidth will be used, following a procedure analogous to that of Horowitz (1992). This option has no effect if `normal` is specified.

`k1(#)` overrides the first bandwidth constant for the jackknife estimator and must be specified along with `k2(#)`.

`k2(#)` overrides the second bandwidth constant for the jackknife estimator and must be specified along with `k1(#)`.

Common options

`noconstant` suppresses the constant term (intercept) in the linear index.

`brep(#)` specifies the number of bootstrap replications used to estimate standard errors. If standard errors are not needed, specify `brep(0)` to skip the bootstrap step entirely and report only the estimated coefficients. Corresponding to Stata's `bootstrap`, the default value is `brep(50)`, but this may be too low for many applications.

`seed(#)` sets the seed of the random-number generator used for bootstrap replications. This is useful for generating reproducible results.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [U] **20.7 Specifying the width of confidence intervals**.

`nmiter(#)` sets the number of initial Nelder–Mead iterations. See [M-5] **optimize( )** for additional details.

`nmdelta(#)` sets the step sizes for constructing the initial Nelder–Mead simplex. See [M-5] **optimize( )** for additional details.

5.3 Stored results

The `dfbr` command stores the results below in `e()` upon completion. After sieve estimation, only the index coefficients are stored in `e(b)` with estimated variance–covariance matrix `e(V)`, but if the estimated sieve parameters are required, the vector of all parameters is stored in `e(alpha)` with corresponding variance–covariance `e(V_alpha)`.

Scalars

`e(N)`	number of observations	`e(k2)`	jackknife bandwidth constant (`local normal` only)
`e(K)`	number of coefficients	`e(w1)`	jackknife weight (`local normal` only)
`e(brep)`	number of bootstrap replications	`e(w2)`	jackknife weight (`local normal` only)
`e(level)`	confidence level		
`e(h)`	bandwidth (`local` only)		
`e(k1)`	jackknife bandwidth constant (`local normal` only)		

Macros

`e(method)`	`sieve` or `local`	`e(vce)`	`bootstrap`
`e(cmdname)`	`dfbr`	`e(basis)`	sieve basis (`sieve` only)
`e(depvar)`	name of dependent variable	`e(properties)`	`b V`

Matrices

`e(b)`	coefficient vector	`e(V_alpha)`	variance–covariance matrix for $\widehat{\alpha}$ (`sieve` only)
`e(V)`	variance–covariance matrix of the estimators	`e(BS_alpha)`	bootstrap replicates for $\widehat{\alpha}$ (`sieve` only)
`e(BS)`	bootstrap replicates for $\widehat{\beta}$		
`e(start)`	optimization starting values		
`e(alpha)`	estimated parameters $\widehat{\alpha}$ (`sieve` only)		

Functions

`e(sample)`	marks estimation sample

5.4 Examples

We first generate a dataset containing a binary response variable that is generated from two independent variables and a heteroskedastic error term. We use the same dataset throughout the remaining examples. The dataset is generated using a fixed seed so that the results can be easily reproduced.

▷ Heteroskedastic binary response data

We generate a random sample of 2,000 observations with normally distributed regressors $x_1 \sim N(0,1)$ and $x_2 \sim N(1,1)$ and a uniformly distributed error term, normalized to have mean 0 and variance 1. We scale the errors by using the scaling function $\exp(x_1 \times |x_2|)$ to introduce (multiplicative) heteroskedasticity. We normalize the coefficient on x_2 to 1 in the data-generating process to make the true values and estimates comparable without scaling.

```
. set seed 2012111707
. set obs 2000
obs was 0, now 2000
. generate x1 = invnorm(uniform())
. generate x2 = 1 + invnorm(uniform())
. generate u = (sqrt(12)*uniform() - sqrt(12)/2) * exp(x1*abs(x2))
. generate y = (-0.1 + 0.3 * x1 + x2 - u) > 0
```

◁

▷ Basic SNLLS estimation

The simplest usage is to invoke `dfbr` with only the dependent and independent variables and no additional options:

```
. dfbr y x1 x2
Bootstrap replications (50)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
Sieve Nonlinear Least Squares (SNLLS)           Number of obs   =         2000
------------------------------------------------------------------------------
             |   Observed
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _cons |  -.1048974   .0683328    -1.54   0.125    -.2388271    .0290324
          x1 |   .2888343   .0410659     7.03   0.000     .2083466    .3693219
------------------------------------------------------------------------------
Coefficient on x2 normalized to 1.
Sieve basis: _cons x1 x2
```

◁

▷ SNLLS estimation with custom basis

To fit the model by using the sieve estimator with second-order polynomial terms, we can first generate the additional basis variables and then invoke `dfbr` with the `sieve` option:

```
. generate x1x2 = x1 * x2
. generate x1_2 = x1^2
. generate x2_2 = x2^2
. dfbr y x1 x2, sieve basis(x1 x2 x1x2 x1_2 x2_2)
Bootstrap replications (50)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
Sieve Nonlinear Least Squares (SNLLS)           Number of obs   =         2000
------------------------------------------------------------------------------
             |   Observed
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _cons |  -.1113518   .1437641    -0.77   0.439    -.3931243    .1704206
          x1 |   .3063337   .0875112     3.50   0.000     .1348149    .4778525
------------------------------------------------------------------------------
Coefficient on x2 normalized to 1.
Sieve basis: _cons x1 x2 x1x2 x1_2 x2_2
```

In Stata 11 and later, one can use factor-variable notation to automatically generate the basis terms without actually generating and storing any additional variables:

```
. dfbr y x1 x2, sieve basis((c.x1 c.x2)##(c.x1 c.x2))
Bootstrap replications (50)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
Sieve Nonlinear Least Squares (SNLLS)           Number of obs   =         2000
------------------------------------------------------------------------------
             |   Observed
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _cons |  -.1113518   .1467072    -0.76   0.448    -.3988926      .176189
          x1 |   .3063337   .0905896     3.38   0.001     .1287813     .4838861
------------------------------------------------------------------------------
Coefficient on x2 normalized to 1.
Sieve basis: _cons x1 x2 c.x1#c.x1 c.x1#c.x2 c.x2#c.x2
```

Here the expression `(c.x1 c.x2)##(c.x1 c.x2)` is equivalent to the manually generated basis `x1 x2 x1x2 x1_2 x2_2` from before. See [U] **11.4.3 Factor variables** for additional details on factor variables.

◁

▷ Basic LNLLS estimation with custom bootstrap replications

To fit the model by using the LNLLS estimator with the default bandwidth and report standard errors estimated using 200 bootstrap replications, type

```
. dfbr y x1 x2, local brep(200)
Bootstrap replications (200)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
..................................................   100
..................................................   150
..................................................   200
Local Nonlinear Least Squares (LNLLS)           Number of obs   =         2000
                                                Bandwidth       =  2.09358e-01
------------------------------------------------------------------------------
             |   Observed
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _cons |   .0601835   .1732894     0.35   0.728    -.2794576     .3998245
          x1 |   .4113817   .0912445     4.51   0.000     .2325459     .5902176
------------------------------------------------------------------------------
Coefficient on x2 normalized to 1.
```

◁

▷ LNLLS estimation with custom bandwidth

A custom bandwidth can be chosen with the `bandwidth()` option:

```
. dfbr y x1 x2, local bandwidth(0.1)
Bootstrap replications (50)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
Local Nonlinear Least Squares (LNLLS)           Number of obs   =         2000
                                                Bandwidth       =  1.00000e-01
------------------------------------------------------------------------------
           y |   Observed
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _cons |   .0321924   .1989456     0.16   0.871    -.3577338    .4221186
          x1 |   .3852976   .1454903     2.65   0.008     .1001417    .6704534
------------------------------------------------------------------------------
Coefficient on x2 normalized to 1.
```

◁

▷ Basic jackknife LNLLS estimation

To use the jackknife LNLLS estimator, which uses the normal c.d.f. as the regression function, invoke `dfbr` with the `local` and `normal` options:

```
. dfbr y x1 x2, local normal
Bootstrap replications (50)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
Local Nonlinear Least Squares (LNLLS)           Number of obs   =         2000
------------------------------------------------------------------------------
           y |   Observed
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _cons |   .3631714   .3749945     0.97   0.333    -.3718043    1.098147
          x1 |   .6163898     .21703     2.84   0.005     .1910188    1.041761
------------------------------------------------------------------------------
Coefficient on x2 normalized to 1.
```

◁

5.5 Monte Carlo evidence

This section provides some additional evidence on the finite-sample properties of the estimators beyond that provided by Khan (2013) and Blevins and Khan (2013). The results are based on replications of the model

$$y_i = 1\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i > 0\right)$$

where we normalize $\beta_2 = 1$ and choose $\beta_0 = -1$ and $\beta_1 = 2$. The covariates have distributions $x_{i1} \sim N(0,1)$ and $x_{i2} \sim N(1,1)$. We consider two specifications where the distribution of ε_i is independent of x_{i1} and x_{i2} and two specifications where there is multiplicative heteroskedasticity. For the independent specifications, we draw ε_i from

the uniform and standard normal distributions, respectively. For the heteroskedastic designs, we multiply each draw by the factor $\exp(x_{i1} \times |x_{i2}|)$ so that the variance of the error term depends on both x_{i1} and x_{i2}.

For each specification, we report results for 1,001 replications of sample size $n = 200$. Specifically, we report the mean bias and mean squared error (MSE) for both β_0 and β_1. Additionally, we report the coverage of the bootstrap confidence intervals for both parameters. We use 100 bootstrap replications (that is, `brep(100)`) to obtain a confidence interval for each estimate, and this is repeated for each of the 1,001 replications for each sample size.[3] The confidence intervals have 95% nominal coverage, so the fraction of replications where the confidence interval covers the true parameter values should be approximately 0.95. Other than increasing the number of bootstrap replications, we use the default options for each estimator. The results are reported in table 1.

Table 1. Monte Carlo results

	β_0			β_1		
Estimator	Bias	MSE	Coverage	Bias	MSE	Coverage
Homoskedastic normal						
LNLLS (F)	−0.0108	0.0002	0.9720	−0.0920	0.0090	0.9750
LNLLS (Φ)	0.0015	0.0001	0.9770	−0.1190	0.0150	0.9740
SNLLS	0.0010	0.0000	0.9720	−0.0310	0.0011	0.9590
Homoskedastic uniform						
LNLLS (F)	−0.0008	0.0001	0.9680	−0.1951	0.0392	0.9800
LNLLS (Φ)	0.0084	0.0003	0.9740	−0.2184	0.0493	0.9720
SNLLS	0.0025	0.0000	0.9670	−0.0466	0.0023	0.9600
Heteroskedastic normal						
LNLLS (F)	0.0238	0.0007	0.9670	−0.0372	0.0023	0.9740
LNLLS (Φ)	0.0309	0.0011	0.9700	−0.0386	0.0031	0.9700
SNLLS	0.0635	0.0041	0.9470	0.1211	0.0149	0.9411
Heteroskedastic uniform						
LNLLS (F)	0.0327	0.0012	0.9710	−0.1092	0.0140	0.9690
LNLLS (Φ)	0.0384	0.0018	0.9600	−0.1121	0.0160	0.9620
SNLLS	0.0878	0.0077	0.9391	0.1578	0.0252	0.9341

3. The results were qualitatively very similar for `brep(250)` and `brep(500)`.

5.6 Implementation details

We conclude with a few notes on specific implementation details. For each estimator, `dfbr` uses six starting values and returns the best estimate. Two starting values are the constant vectors of all 0s and all 1s. The remaining four are based on other, easier-to-calculate estimators: ordinary least squares, least absolute deviations, probit, and logit. These values are stored in `e(start)`.

The bootstrap standard errors and confidence intervals reported by `dfbr` are calculated in the same way as those produced by Stata's `bootstrap` command. That is, they are based on the variance matrix of the bootstrap replicates. In particular, the reported standard errors are square roots of the diagonal elements of the variance matrix, and the confidence intervals are based on a normal approximation (that is, using the standard errors and critical values of the standard normal distribution). The bootstrap replicates are stored in the `e(BS)` matrix to allow further processing, if desired.

For example, to convert the columns of the `e(BS)` matrix to variables in the current dataset named `coeff1`, `coeff2`, and so on, use the `svmat` command after executing `dfbr`:

```
. dfbr y x1 x2, local brep(500)
. matrix BS = e(BS)
. svmat BS, names(coeff)
. summarize coeff*
. correlate coeff*, covariance
```

For the LNLLS estimator, we first obtain an estimate $\widehat{\beta}^{(1)}$ by using the default bandwidth, $h^{(1)} = n^{-1/5}$. This estimate is then used to estimate the optimal bandwidth $h^{(2)}$, using a procedure analogous to that of Horowitz (1992). This procedure was also written in Mata for ease of implementation and for performance reasons. Finally, using the bandwidth $h^{(2)}$, we obtain the reported estimates $\widehat{\beta}^{(2)}$. This process can be skipped, and a custom bandwidth can be used instead by specifying the `bandwidth()` option.

By default, for each starting value, the program begins with at most $10k$ Nelder–Mead iterations, followed by a complete run of Broyden–Fletcher–Goldfarb–Shanno with analytic gradient and Hessian calculations. This procedure is more robust to poor starting values that might be in nonconcave regions of the objective function, while switching to a more accurate gradient-based method before reporting the final estimates. The maximum number of initial Nelder–Mead iterations can be adjusted using the `nmiter()` option, where using `nmiter(0)` skips this initial step completely. The initial Nelder–Mead simplex step sizes are set to a vector of 1s by default, but can be set to a vector equal to some constant `delta` by using the `nmdelta(delta)` option. The maximum number of Broyden–Fletcher–Goldfarb–Shanno iterations can be controlled by using `set maxiter`.

6 Acknowledgments

We are grateful to Jonah Gelbach, Phil Haile, and Jeff Wooldridge for useful comments that helped improve `dfbr`.

7 References

Blevins, J. R., and S. Khan. 2013. Local NLLS estimation of semiparametric binary choice models. *Econometrics Journal* 16: 135–160.

Horowitz, J. L. 1992. A smoothed maximum score estimator for the binary response model. *Econometrica* 60: 505–531.

Jann, B. 2005. moremata: Stata module (Mata) to provide various functions. Statistical Software Components S455001, Department of Economics, Boston College. http://ideas.repec.org/c/boc/bocode/s455001.html.

Khan, S. 2013. Distribution free estimation of heteroskedastic binary response models using probit/logit criterion functions. *Journal of Econometrics* 172: 168–182.

Manski, C. 1975. Maximum score estimation of the stochastic utility model of choice. *Journal of Econometrics* 3: 205–228.

Manski, C. F. 1985. Semiparametric analysis of discrete response: Asymptotic properties of the maximum score estimator. *Journal of Econometrics* 27: 313–333.

Yatchew, A., and Z. Griliches. 1985. Specification error in probit models. *Review of Economics and Statistics* 67: 134–139.

About the authors

Jason R. Blevins is an assistant professor of economics at The Ohio State University.

Shakeeb Khan is a professor of economics at Duke University.