The Stata Journal (2020)
20, Number 4, pp. 976–998 DOI: 10.1177/1536867X20976339

Fitting partially linear functional-coefficient panel-data models with Stata

Kerui Du
School of Management
Xiamen University
Xiamen, China
kerrydu@xmu.edu.cn

Yonghui Zhang
School of Economics
Renmin University of China
Beijing, China
yonghui.zhang@hotmail.com

Qiankun Zhou
Department of Economics
Louisiana State University
Baton Rouge, LA
qzhou@lsu.edu

Abstract. In this article, we describe the implementation of fitting partially linear functional-coefficient panel models with fixed effects proposed by An, Hsiao, and Li [2016, Semiparametric estimation of partially linear varying coefficient panel data models in *Essays in Honor of Aman Ullah* (*Advances in Econometrics, Volume 36*)] and Zhang and Zhou (Forthcoming, *Econometric Reviews*). Three new commands `xtplfc`, `ivxtplfc`, and `xtdplfc` are introduced and illustrated through Monte Carlo simulations to exemplify the effectiveness of these estimators.

Keywords: st0624, xtplfc, ivxtplfc, xtdplfc, functional coefficients, panel data, fixed effects, endogenous variables, sieve, spline

1 Introduction

A partially linear functional-coefficient regression model allows for linearity in some regressors and nonlinearity in other regressors, where the effects of these covariates on the dependent variable vary according to a set of low-dimensional variables nonparametrically (Cai et al. 2017), thereby showing distinct advantages in capturing nonlinearity and heterogeneity. Through the seminal work of Chen and Tsay (1993), the functional-coefficient models have drawn much attention in the literature. To name a few, Cai, Fan, and Yao (2000), Cai, Fan, and Li (2000), and Cai, Li, and Park (2009) study functional-coefficient models under the time-series framework. Huang, Wu, and Zhou (2004) fit a functional-coefficient panel-data model without fixed effects via the series method. Cai and Li (2008) study functional-coefficient dynamic panel-data models without fixed effects based on the kernel method. Sun, Carroll, and Li (2009) consider functional-coefficient panel-data models with fixed effects that are removed via the least-square dummy variable approach. Alternatively, An, Hsiao, and Li (2016) deal with the fixed effects via the first time difference and fit the models using the series method. Zhang and Zhou (Forthcoming) propose to use a sieve two-step least square (2SLS) procedure to fit functional-coefficient panel dynamic models with fixed effects and develop a model specification test for the constancy of slopes.

In empirical studies, functional-coefficient models have been widely used. For example, they are applied to explore whether working experience matters to the impact of education on wage (Su, Murtazashvili, and Ullah 2013; Cai et al. 2017); to analyze the heterogeneous effect of foreign direct investment on economic growth (Delgado, Mc-

Cloud, and Kumbhakar 2014; Cai et al. 2017); to examine the nonlinear relationship between income level and democracy (Lundberg, Huynh, and Jacho-Chávez 2017; Zhang and Zhou Forthcoming); to compare the returns to scale of the U.S. commercial banks across different regimes (Feng et al. 2017); and to investigate the role of marketization in China's energy rebound effect (Li, Liu, and Du 2019).

The objective of this article is to present new commands to fit partially linear functional-coefficient panel-data models with fixed effects. The remainder of the article is organized as follows. Section 2 briefly describes the model and the estimation procedure. Sections 3–5 explain the syntax and options of the new commands. Section 6 provides some Monte Carlo simulations. Section 7 concludes the article.

2 The model

Consider a partially linear functional-coefficient panel-data model of the form

$$Y_{it} = \mathbf{Z}'_{it} g(\mathbf{U}_{it}) + \mathbf{X}'_{it}\boldsymbol{\beta} + a_i + \varepsilon_{it} \qquad i = 1, \ldots, N, \quad t = 1, \ldots, T \tag{1}$$

where the subscript i and t denote individual i and time t, respectively; Y_{it} is the scalar dependent variable; $\mathbf{U}_{it} = (U_{1,it}, \ldots, U_{l,it})'$ is a vector of continuous variables; $\mathbf{Z}_{it} = (Z_{1,it}, \ldots, Z_{l,it})'$ is a vector of covariates with functional coefficients $\mathbf{g}(\mathbf{U}_{it}) = \{g_1(U_{1,it}), \ldots, g_l(U_{l,it})\}'$; $\mathbf{X}_{it}$ is a $k \times 1$ vector of covariates with constant slopes $\boldsymbol{\beta}$; and a_i is the individual fixed effects that might be correlated with $\mathbf{Z}_{it}$, $\mathbf{U}_{it}$, and $\mathbf{X}_{it}$. ε_{it} represents the idiosyncratic error. Moreover, part of the elements in $\mathbf{Z}_{it}$ and $\mathbf{X}_{it}$ are allowed to be endogenous variables that are correlated with ε_{it}, and they could also include lagged dependent variables as in Zhang and Zhou (Forthcoming).[1] In the latter case, (1) becomes a dynamic panel model with functional coefficients.

Recently, An, Hsiao, and Li (2016) and Zhang and Zhou (Forthcoming) propose using the series method to fit (1). The estimation procedure is sketched as follows.

First, one can use a linear combination of sieve basis functions to approximate the unknown functional coefficients in (1). Let $\mathbf{h}^p(\cdot) = \{h_{p,1}(\cdot), \ldots, h_{p,L_p}(\cdot)\}'$ be a $L_p \times 1$ sequence of basis functions where the number of sieve basis functions $L_p \equiv L_{NL}$ increases as either N or T increases. We have $g_p(\cdot) \approx \mathbf{h}^p(\cdot)'\boldsymbol{\gamma}_p$ for $p = 1, \ldots, l$, where $\boldsymbol{\gamma}_p = (\gamma_{p1}, \ldots, \gamma_{pLp})$. Then, (1) can be rewritten as

$$Y_{it} = \mathbf{H}'_{it}\boldsymbol{\Gamma} + \mathbf{X}'_{it}\boldsymbol{\beta} + a_i + v_{it} \tag{2}$$

where $\mathbf{H}_{it} \equiv \{Z_{1,it}\mathbf{h}^1(U_{1,it})', \ldots, Z_{l,it}\mathbf{h}^l(U_{l,it})'\}'$, $\boldsymbol{\Gamma} \equiv (\boldsymbol{\gamma}'_1, \ldots, \boldsymbol{\gamma}'_l)'$, and $v_{it} = \varepsilon_{it} + r_{it}$,

$$r_{it} = \mathbf{Z}'_{it}\mathbf{g}(\mathbf{U}_{it}) - \mathbf{H}'_{it}\boldsymbol{\Gamma}$$

denoting the sieve approximation error that becomes asymptotic negligible as $L_p \to \infty$ for $p = 1, \ldots, l$ when $(N, T) \to \infty$.

1. In this article, we restrict that all elements in $\mathbf{U}_{it}$ are exogenous. Thus, for the case of dynamic panel-data models, the first lagged dependent variable should not enter $\mathbf{U}_{it}$.

Then, taking the first time difference of (2) to eliminate the fixed effects yields

$$\Delta Y_{it} = \Delta \mathbf{H}_{it}'\boldsymbol{\Gamma} + \Delta \mathbf{X}_{it}'\boldsymbol{\beta} + \Delta v_{it} \tag{3}$$

where Δ represents a first-difference operator; that is, $\Delta A_t \equiv A_t - A_{t-1}$.

If all the variables are exogenous, (3) can be estimated through the least-squares method as in An, Hsiao, and Li (2016),

$$\left(\widehat{\boldsymbol{\Gamma}}', \widehat{\boldsymbol{\beta}}'\right)' = \left(\Delta\widetilde{\mathbf{X}}'\Delta\widetilde{\mathbf{X}}\right)^{-}\left(\Delta\widetilde{\mathbf{X}}'\Delta\mathbf{Y}\right)$$

where $\widetilde{\mathbf{X}} = (\mathbf{X}, \mathbf{H})$; $\mathbf{X} = (\mathbf{X}_1', \ldots, \mathbf{X}_N')'$, $\mathbf{X}_i = (X_{i1}, \ldots, X_{iT})'$; $\mathbf{H} = (\mathbf{H}_1, \ldots, \mathbf{H}_N)'$, $\mathbf{H}_i = (H_{i1}, \ldots, H_{iT})'$; and $^{-}$ denotes the generalized inverse.

If part of the variables in $\mathbf{Z}_{it}$ and $\mathbf{X}_{it}$ are endogenous, (3) can be estimated via the 2SLS method as in Zhang and Zhou (Forthcoming). Suppose we have a $d \times 1$ vector of instrumental variables (IVs) with $d \geq (\sum_{p=1}^{l} L_p + k)$. The 2SLS estimator is given by

$$\left(\widehat{\boldsymbol{\Gamma}}', \widehat{\boldsymbol{\beta}}'\right)' = \left(\Delta\widetilde{\mathbf{X}}'\mathbf{P}_W\Delta\widetilde{\mathbf{X}}\right)^{-}\left(\Delta\widetilde{\mathbf{X}}'\mathbf{P}_W\Delta\mathbf{Y}\right)$$

where $\mathbf{P}_W = \mathbf{W}(\mathbf{W}'\mathbf{W})^{-}\mathbf{W}'$ is a projection matrix with $\mathbf{W}$ being the IVs matrix.

Once $\widehat{\boldsymbol{\Gamma}}$ is obtained, the functional coefficients $\mathbf{g}(\mathbf{U}_{it})$ can be estimated by

$$\widehat{g}_p(U_{p,it}) = \mathbf{h}^p(U_{p,it})'\widehat{\boldsymbol{\gamma}_p} \qquad p = 1, \ldots, l$$

Under certain regular assumptions, Zhang and Zhou (Forthcoming) establish the consistency and asymptotic normality of the above estimators when sample size N and T go to infinity jointly or only N tends to infinity.

In practice, there are several sieve methods to approximate the unknown functions. We follow Libois and Verardi (2013) to use the B-splines. More technical details on the B-splines can be found in Newson (2000b).

3 The xtplfc command

`xtplfc` fits An, Hsiao, and Li's (2016) partially linear functional-coefficient panel-data models with exogenous variables.

3.1 Syntax

`xtplfc` *varlist*`,` `zvars(`*varlist*`)` `uvars(`*varlist*`)` `generate(`*prefix*`)` [`te`
`power(`*numlist*`)` `nknots(`*numlist*`)` `quantile` `maxnknots(`*numlist*`)`
`minnknots(`*numlist*`)` `grid(`*string*`)` `pctile(`#`)` `brep(`#`)` `wild` `predict(`*prspec*`)`
`nodots` `level(`#`)` `fast` `tenfoldcv`]

3.2 Options

`zvars(`*varlist*`)` specifies variables that have functional coefficients. `zvars()` is required.

`uvars(`*varlist*`)` specifies (continuous) variables that enter the functional coefficients interacted with variables in order specified by `zvars()`. `uvars()` is required.

`generate(`*prefix*`)` specifies a prefix for the variable names to store fitted values of functional coefficients. `generate()` is required.

`te` specifies to include time fixed effects.

`power(`*numlist*`)` (nonnegative integers) specifies the power (or degree) of the splines in order specified by `uvars()`. The default is `power(3)`.

`nknots(`*numlist*`)` specifies the number of knots used for the spline interpolation in order specified by `uvars()`. The default is `nknots(2)`.

`quantile` specifies creating knots based on empirical quantiles. By default, the knots are generated by the rule of equal space.

`maxnknots(`*numlist*`)` specifies the maximum number of knots used for conducting least-squares cross-validation (LSCV). If present, LSCV is used to determine the optimal number of knots. In our practice, we perform the leave-one-out cross-validation (CV) across the *panelvar*. That is to say, we leave one individual (with all observations during the sample period) out each time.

`minnknots(`*numlist*`)` specifies the minimum number of knots used for performing LSCV. The default is `minnknots(2)`.

`grid(`*string*`)` specifies a prefix for the names to store the grid points of the variable specified by `uvars(`*varlist*`)`. If present, the functional coefficients are estimated over the grid points. By default, they are estimated over the observations.

`pctile(`#`)` specifies the domain of the generating grid points. It can be used only when `grid()` is specified. The default is `pctile(0)`.

`brep(`#`)` specifies the number of bootstrap replications. The default is `brep(200)`. We recommend that you select the number of replications.

`wild` specifies using the wild bootstrap. By default, residual bootstrap with the option `cluster(`*panelvar*`)` is performed.

`predict(`*prspec*`)` stores predicted values of the conditional mean and fixed effects using variable names specified in *prspec*. Specifically, one uses `predict(`*varlist* | *stub*`*` [`, replace noai`]`)`. The option takes a variable list or *stub*. The first variable name corresponds to the predicted conditional mean. The second name corresponds to fixed effects. When `replace` is used, variables with the names in *varlist* or *stub*`*` are replaced by those in the new computation. If `noai` is specified, only a variable for the mean is created.

`nodots` suppresses the iteration dots.

`level(#)` sets the confidence level. The default is `level(95)`.

`fast` speeds up using Mata functions.

`tenfoldcv` specifies using tenfold CV instead of LSCV. It is done by dividing the sample into 10 pieces and conducting LSCV across these 10 pieces. Specifically, given the number of knots, leave one piece out, run the regression using the left pieces, and predict the dependent variable for the remaining piece.

3.3 Stored results

`xtplfc` stores the following in `e()`:

Scalars

`e(N)`	number of individuals
`e(df_m)`	model degrees of freedom
`e(df_r)`	residual degrees of freedom
`e(r2)`	within R^2
`e(r2_a)`	adjusted within R^2
`e(rmse)`	root mean squared error
`e(mss)`	model sum of squares
`e(rss)`	residual sum of squares

Macros

`e(cmd)`	`xtplfc`
`e(depvar)`	name of dependent variable
`e(title)`	title in estimation output
`e(estfun)`	variables storing the estimated functional coefficients
`e(properties)`	`b V`
`e(vcetype)`	type of variance–covariance
`e(model)`	`Fixed-effect Series Semiparametric Estimation`
`e(k#)`	list of knots for the #th function

Matrices

`e(b)`	coefficient vector in the linear part
`e(V)`	variance–covariance matrix of the estimators in the linear part
`e(bs)`	coefficient vector in the approximating model
`e(Vs)`	variance–covariance matrix of the estimators in the approximating model
`e(nknots)`	number of knots
`e(power)`	power (or degree) of splines

3.4 Dependency of xtplfc

`xtplfc` depends on the `moremata` (Jann 2005) and `bspline` (Newson 2000a) packages.

4 The ivxtplfc command

ivxtplfc fits Zhang and Zhou's (Forthcoming) partially linear functional-coefficient panel-data models with endogenous variables using the sieve 2SLS method.

4.1 Syntax

ivxtplfc *varlist*, zvars(*varlist*) uvars(*varlist*) generate(*prefix*) [endox(*varlist*) endozflag(*numlist*) ivx(*varlist*) ivz(*varlist*, uflag(*numlist*) [ivtype(*numlist*)]) te power(*numlist*) nknots(*numlist*) quantile maxnknots(*numlist*) minnknots(*numlist*) grid(*string*) pctile(#) brep(#) wild predict(*prspec*) nodots level(#) fast tenfoldcv]

4.2 Options

zvars(*varlist*) specifies variables that have functional coefficients. zvars() is required.

uvars(*varlist*) specifies (continuous) variables that enter the functional coefficients interacted with variables in order specified by zvars(). uvars() is required.

generate(*prefix*) specifies a prefix for the names to store fitted values of functional coefficients. generate() is required.

endox(*varlist*) specifies endogenous variables that enter the model linearly.

endozflag(*numlist*) specifies the orders of variables in zvars(*varlist*) that are endogenous variables. For example, endozflag(1 3) indicates that the first and third variables specified in zvars(*varlist*) are endogenous.

ivx(*varlist*) specifies IVs that enter the model linearly.

ivz(*varlist*, uflag(*numlist*) [ivtype(*numlist*)]) specify IVs entering the model nonlinearly that interact with the functions specified by the orders in uflag(*numlist*). Optionally, one may specify the type of nonlinear IVs to be constructed. ivtype(*numlist*) means using the #th lag of the basis functions, and the final IVs are formed from ivz() $\times L\#.S(u)$ [where $S(u)$ are basis functions of u]. By default, the first lag of the basis functions is used.

te specifies to include time fixed effects.

power(*numlist*) (nonnegative integers) specifies the power (or degree) of the splines in order specified by uvars(). The default is power(3).

nknots(*numlist*) specifies the number of knots used for the spline interpolation in order specified by uvars(). The default is nknots(2).

quantile specifies creating knots based on empirical quantiles. By default, the knots are generated by the rule of equal space.

`maxnknots(`*numlist*`)` specifies the maximum number of knots used for conducting LSCV. If present, LSCV is used to determine the optimal number of knots. In our practice, we perform the leave-one-out CV across the *panelvar*. That is to say, we leave one individual (with all observations during the sample period) out each time.

`minnknots(`*numlist*`)` specifies the minimum number of knots used for performing LSCV. The default is `minnknots(2)`.

`grid(`*string*`)` specifies a prefix for the names to store the grid points of the variable specified by `uvars()`. If present, the functional coefficients are estimated over the grid points. By default, they are estimated over the observations.

`pctile(#)` specifies the domain of the generating grid points. It can be used only when `grid()` is specified. The default is `pctile(0)`.

`brep(#)` specifies the number of bootstrap replications. The default is `brep(200)`. We recommend that you select the number of replications.

`wild` specifies using the wild bootstrap. By default, residual bootstrap with the option `cluster(`*panelvar*`)` is performed.

`predict(`*prspec*`)` stores predicted values of the conditional mean and fixed effects using variable names specified in *prspec*. Specifically, one uses `predict(`*varlist* | *stub*`*` [`, replace noai`]`)`. The option takes a variable list or *stub*. The first variable name corresponds to the predicted conditional mean. The second name corresponds to fixed effects. When `replace` is used, variables with the names in *varlist* or *stub*`*` are replaced by those in the new computation. If `noai` is specified, only a variable for the mean is created.

`nodots` suppresses the iteration dots.

`level(#)` sets the confidence level. The default is `level(95)`.

`fast` speeds up using Mata functions.

`tenfoldcv` specifies using tenfold CV instead of LSCV.

4.3 Stored results

`ivxtplfc` stores the following in `e()`:

Scalars

`e(N)`	number of individuals
`e(df_m)`	model degrees of freedom
`e(df_r)`	residual degrees of freedom
`e(r2)`	within R^2
`e(r2_a)`	adjusted within R^2
`e(rmse)`	root mean squared error
`e(mss)`	model sum of squares
`e(rss)`	residual sum of squares

Macros

`e(cmd)`	`ivxtplfc`
`e(depvar)`	name of dependent variable
`e(title)`	title in estimation output
`e(ivlist)`	IVs used for estimation
`e(vcetype)`	type of variance–covariance
`e(estfun)`	variables storing the estimated functional coefficients
`e(properties)`	`b V`
`e(model)`	`Fixed-effect Sieve 2SLS Estimation`
`e(k#)`	list of knots for the #th function

Matrices

`e(b)`	coefficient vector in the linear part
`e(V)`	variance–covariance matrix of the estimators in the linear part
`e(bs)`	coefficient vector in the approximating model
`e(Vs)`	variance–covariance matrix of the estimators in the approximating model
`e(knots)`	number of knots
`e(power)`	power (or degree) of splines

Functions

`e(sample)`	marks estimation sample

4.4 Dependency of ivxtplfc

`ivxtplfc` depends on the `moremata` (Jann 2005) and `bspline` (Newson 2000a) packages.

5 The xtdplfc command

`xtdplfc` fits Zhang and Zhou's (Forthcoming) partially linear functional-coefficient panel dynamic data models using the sieve 2SLS method.

5.1 Syntax

`xtdplfc` *varlist*`,` `uvars(`*varlist*`)` `generate(`*prefix*`)` [`zvars(`*varlist*`)` `lags(`#`)`
`lagyinz(`*numlist*`)` `endox(`*varlist*`)` `endozflag(`*numlist*`)` `ivx(`*varlist*`)`
`ivz(`*varlist*`,` `uflag(`*numlist*`)` [`ivtype(`*numlist*`)`]`)` `onlyivxz` `ivtype(`*numlist*`)`
`te` `power(`*numlist*`)` `nknots(`*numlist*`)` `quantile` `maxnknots(`*numlist*`)`
`minnknots(`*numlist*`)` `grid(`*string*`)` `pctile(`#`)` `brep(`#`)` `wild` `predict(`*prspec*`)`
`nodots` `level(`#`)` `fast` `tenfoldcv`]

5.2 Options

`uvars(`*varlist*`)` specifies (continuous) variables that enter the functional coefficients interacted with variables in order specified by `zvars()`. `uvars()` is required.

`generate(`*prefix*`)` specifies a prefix for the names to store fitted values of functional coefficients. `generate()` is required.

`zvars(`*varlist*`)` specifies variables that have functional coefficients.

`lags(`#`)` specifies using # lags of the dependent variable as covariates. The default is `lags(1)`.

`lagyinz(`*numlist*`)` specifies lags of the dependent variable that have functional coefficients. When this option is used, the specified lags of the dependent variable are automatically added in front of variables in `zvars()`.

`endox(`*varlist*`)` specifies endogenous variables that enter the model linearly.

`endozflag(`*numlist*`)` specifies the orders of variables in `zvars(`*varlist*`)` that are endogenous variables. For example, `endozflag(1 3)` indicates that the first and third variables specified in `zvars(`*varlist*`)` are endogenous.

`ivx(`*varlist*`)` specifies IVs that enter the model linearly.

`ivz(`*varlist*`,` `uflag(`*numlist*`)` [`ivtype(`*numlist*`)`]`)` specify IVs entering the model nonlinearly that interact with the functions specified by the orders in `uflag(`*numlist*`)`. Optionally, one may specify the type of nonlinear IVs to be constructed. `ivtype(`*numlist*`)` means using the #th lag of the basis functions, and the final IVs are formed from `ivz()` $\times L\#.S(u)$ [where $S(u)$ are basis functions of u]. By default, the first lag of the basis functions is used.

`onlyivxz` uses only instruments specified by `ivx()` and `ivz()`. By default, additional instruments are automatically constructed using lags of the dependent variables, variables specified by `endox()` and `endozflag()`, and the generating splines.

`ivtype(`*numlist*`)` specifies the lag of the basis functions to be used for constructing the IVs. Suppose Z is an endogenous variable interacting with $g(U)$; $S(U)$ are basis functions for $g(U)$. `ivtype(1)` indicates constructing IVs from $L2.Z \times L.S(U)$.

`te` specifies to include time fixed effects.

`power(`*numlist*`)` (nonnegative integers) specifies the power (or degree) of the splines in order specified by `uvars()`. The default is `power(3)`.

`nknots(`*numlist*`)` specifies the number of knots used for the spline interpolation in order specified by `uvars()`. The default is `nknots(2)`.

`quantile` specifies creating knots based on empirical quantiles. By default, the knots are generated by the rule of equal space.

`maxnknots(`*numlist*`)` specifies the maximum number of knots used for conducting LSCV. If present, LSCV is used to determine the optimal number of knots. In our practice, we perform the leave-one-out CV across the *panelvar*. That is to say, we leave one individual (with all observations during the sample period) out each time.

`minnknots(`*numlist*`)` specifies the minimum number of knots used for performing LSCV. The default is `minnknots(2)`.

`grid(`*string*`)` specifies the name for storing the grid points of the variable specified by `uvars()`. If present, the functional coefficients are estimated over the grid points. By default, they are estimated over the observations.

`pctile(`#`)` specifies the domain of the generating grid points. It can be used only when `grid()` is specified. The default is `pctile(0)`.

`brep(`#`)` specifies the number of bootstrap replications. The default is `brep(200)`. We recommend that you select the number of replications.

`wild` specifies using the wild bootstrap. By default, residual bootstrap with the option `cluster(`*panelvar*`)` is performed.

`predict(`*prspec*`)` stores predicted values of the conditional mean and fixed effects using variable names specified in *prspec*. Specifically, one uses `predict(`*varlist* | *stub*`*` [`, replace noai`]`)`. The option takes a variable list or *stub*. The first variable name corresponds to the predicted conditional mean. The second name corresponds to fixed effects. When `replace` is used, variables with the names in *varlist* or *stub*`*` are replaced by those in the new computation. If `noai` is specified, only a variable for the mean is created.

`nodots` suppresses the iteration dots.

`level(#)` sets the confidence level. The default is `level(95)`.

`fast` speeds up using Mata functions.

`tenfoldcv` specifies using tenfold CV instead of LSCV.

5.3 Stored results

`xtdplfc` stores the following in `e()`:

Scalars

`e(N)`	number of individuals
`e(df_m)`	model degrees of freedom
`e(df_r)`	residual degrees of freedom
`e(r2)`	within R^2
`e(r2_a)`	adjusted within R^2
`e(rmse)`	root mean squared error
`e(mss)`	model sum of squares
`e(rss)`	residual sum of squares

Macros

`e(cmd)`	`xtdplfc`
`e(depvar)`	name of dependent variable
`e(title)`	title in estimation output
`e(ivlist)`	IVs used for estimation
`e(vcetype)`	type of variance–covariance
`e(estfun)`	variables storing the estimated functional coefficients
`e(properties)`	`b V`
`e(model)`	`Fixed-effect Sieve 2SLS Estimation`
`e(k#)`	list of knots for the #th function

Matrices

`e(b)`	coefficient vector in the linear part
`e(V)`	variance–covariance matrix of the estimators in the linear part
`e(bs)`	coefficient vector in the approximating model
`e(Vs)`	variance–covariance matrix of the estimators in the approximating model
`e(knots)`	number of knots
`e(power)`	power (or degree) of splines

Functions

`e(sample)`	marks estimation sample

5.4 Dependency of xtdplfc

`xtdplfc` depends on the `moremata` (Jann 2005) and `bspline` (Newson 2000a) packages.

6 Monte Carlo simulation

In this section, we investigate the finite sample performance of estimators discussed above through Monte Carlo simulations. For all data-generating processes (DGPs) to be considered, we set up a standard fixed-effects panel comprising 50 individuals over 40 time periods; that is, $N = 50$, $T = 40$. We carry out 500 replications.

We first consider the static panel-data model as follows (DGP1):

$$\begin{aligned} Y_{it} &= X_{1,it} - X_{2,it} + g(X_{3,it})Z_{it} + a_i + \varepsilon_{it} \\ g(X_{3,it}) &= X_{3,it} + 2(X_{3,it})^2 - 0.25(X_{3,it})^3 \end{aligned}$$

Following Libois and Verardi (2013), we generate $X_{1,it}$, $X_{2,it}$, $X_{3,it}$, and a_i via a two-step procedure[2] as follows:

$$\begin{aligned} X_{1,it} &= X^e_{1,it} \\ X_{2,it} &= (X^f_{2,i} + X^e_{2,it})/\sqrt{2} \\ X_{3,it} &= (X^f_{3,i} + X^e_{3,it})/\sqrt{2} \end{aligned}$$

We draw $X^f_{2,it}$, $X^f_{3,it}$, and a_i from the multivariate normal distribution with mean $\boldsymbol{\mu} = (0, 0, 0)$ and covariance matrix

$$\begin{array}{c} \\ X^f_{2,it} \\ X^f_{3,it} \\ a_i \end{array} \begin{array}{c} \begin{array}{ccc} X^f_{2,it} & X^f_{3,it} & a_i \end{array} \\ \begin{pmatrix} 1 & & \\ 0 & 1 & \\ 0.42 & 0.85 & 1 \end{pmatrix} \end{array}$$

Similarly, $X^e_{1,it}$, $X^e_{2,it}$, and $X^e_{3,it}$ are drawn from the multivariate normal distribution with mean $\boldsymbol{\mu} = (0, 0, 0)$ and covariance matrix

$$\begin{array}{c} \\ X^e_{1,it} \\ X^e_{2,it} \\ X^e_{3,it} \end{array} \begin{array}{c} \begin{array}{ccc} X^e_{1,it} & X^e_{2,it} & X^e_{3,it} \end{array} \\ \begin{pmatrix} 1 & & \\ 0.2 & 1 & \\ 0.8 & 0 & 1 \end{pmatrix} \end{array}$$

For comparison, we consider the following three regression models.

- Model 1: `xtplfc`, considering that $X_{1,it}$ and $X_{2,it}$ enter the model linearly, whereas Z_{it} is included with a functional coefficient of $X_{3,it}$.
- Model 2: `xtreg`, regressing Y_{it} on $X_{1,it}$, $X_{2,it}$, $X_{3,it}$, and Z_{it} with fixed effects.
- Model 3: `xtreg`, regressing Y_{it} on $X_{1,it}$, $X_{2,it}$, $X_{3,it}$, Z_{it}, and $c.X_{3,it}\#c.Z_{it}$ with fixed effects.

2. The generating covariates consist of two components. The first one is generated for each individual and then duplicated T times, indicating that they are fixed for each individual. The second one is a random realization for each time.

The simulation codes for DGP 1 are presented as follows:[3]

```
. set obs 50
number of observations (_N) was 0, now 50
. set seed 123456
. matrix C = (1,0,0.42\0,1,0.85\0.42,0.85,1)
. drawnorm x2f x3f d, corr(C)
. generate id = _n
. expand 40
(1,950 observations created)
. bysort id: generate t = _n
. xtset id t
       panel variable:  id (strongly balanced)
        time variable:  t, 1 to 40
                delta:  1 unit
. generate y = 0
. matrix D = (1,0.2,0.8\0.2,1,0\0.8,0,1)
. drawnorm x1 x2e x3e, corr(D)
. generate x2 = (x2f+x2e)/sqrt(2)
. generate x3 = (x3f+x3e)/sqrt(2)
. generate z=rnormal()
. generate gf=1*x3+ 2*x3^2 - 0.25*(x3)^3
```

3. To run the simulation code, one should install Baum and Azevedo's (2001) `outtable` package and the UCLA Statistical Consulting Group's `graph2tex` (Statistical Consulting Group 2017) package in advance. `outtable` can be installed with `ssc install outtable`. `graph2tex` can be installed with `net install graph2tex, from(https://stats.idre.ucla.edu/stat/stata/ado/analysis)`.

```
. forvalues j=1/500 {
  2.     cap drop e
  3.     cap drop *_sd
  4.     quietly drawnorm e
  5.     quietly replace y = x1 - x2 + z*gf + d + e
  6.     quietly xtplfc y x1 x2, z(z) u(x3) maxnk(20) generate(fit`j´) fast
> brep(500)
  7.     matrix B=e(b)
  8.     matrix B=B[1,1..2]
  9.     matrix B1=(nullmat(B1)\B)
 10.     matrix V=e(Vs)
 11.     matrix V=vecdiag(V)
 12.     matrix V=V[1,1..2]
 13.     matrix V1=(nullmat(V1)\V)
 14.     quietly xtreg y x1 x2 x3 z, fe
 15.     matrix B=e(b)
 16.     matrix B=B[1,1..2]
 17.     matrix B2=(nullmat(B2)\B)
 18.     matrix V=e(V)
 19.     matrix V=vecdiag(V)
 20.     matrix V=V[1,1..2]
 21.     matrix V2=(nullmat(V2)\V)
 22.     quietly xtreg y x1 x2 x3 z c.x3#c.z, fe
 23.     matrix B=e(b)
 24.     matrix B=B[1,1..2]
 25.     matrix B3=(nullmat(B3)\B)
 26.     matrix V=e(V)
 27.     matrix V=vecdiag(V)
 28.     matrix V=V[1,1..2]
 29.     matrix V3=(nullmat(V3)\V)
 30. }
. * Figure 1
. egen av_fit = rowmean(fit*)
. egen sd_fit = rowsd(fit*)
. generate c = invnormal(1 - (100 - 95) / 200)
. generate low = av_fit - c * sd_fit
. generate up = av_fit + c * sd_fit
. twoway (rarea low up x3, sort(x3) color(gs7))
>    (line av_fit x3, sort(x3) color(black) lpattern(solid))
>    (line gf x3, color(gs10) sort lpattern(longdash)),
>    legend(label(1 confidence interval at 95%) label(3 DGP) label(2 average fit)
>    cols(3) order(3 1 2)) xtitle("X3", height(5)) ytitle("g(X3)", height(5))
>    scheme(sj)
. graph2tex, epsfile(fig1) caption(Average fit of g(X3) across replications)
> label(fig1)
% exported graph to fig1.eps
% We can see in Figure \ref{fig:fig1} that
\begin{figure}[h]
\begin{centering}
  \includegraphics[height=3in]{fig1}
  \caption{Average fit of g(X3) across replications}
  \label{fig:fig1}
\end{centering}
\end{figure}
```

```
. * Table 1
. clear
. set obs 500
number of observations (_N) was 0, now 500
. matrix res1=J(3,8,.)
. forvalues k=1/3 {
  2.     quietly svmat B`k´
  3.     summarize B`k´1, meanonly
  4.     matrix res1[`k´,1]=r(mean)-1
  5.     summarize B`k´2, meanonly
  6.     matrix res1[`k´,2]=r(mean)+1
  7.     quietly svmat V`k´
  8.     quietly replace V`k´1=sqrt(V`k´1)
  9.     quietly generate B`k´1_lb=B`k´1-invnormal(0.975)*V`k´1
 10.     quietly generate B`k´1_ub=B`k´1+invnormal(0.975)*V`k´1
 11.     quietly generate CIlen`k´1=B`k´1_ub-B`k´1_lb
 12.     quietly summarize CIlen`k´1, detail
 13.     matrix res1[`k´,5]=r(p50)
 14.     quietly replace V`k´2=sqrt(V`k´2)
 15.     quietly generate B`k´2_lb=B`k´2-invnormal(0.975)*V`k´2
 16.     quietly generate B`k´2_ub=B`k´2+invnormal(0.975)*V`k´2
 17.     quietly generate CIlen`k´2=B`k´2_ub-B`k´2_lb
 18.     quietly summarize CIlen`k´2, detail
 19.     matrix res1[`k´,6]=r(p50)
 20.     quietly generate cov`k´1=(B`k´1_lb<=1 & B`k´1_ub>=1)
 21.     summarize cov`k´1, meanonly
 22.     matrix res1[`k´,7]=r(mean)
 23.     quietly generate cov`k´2=(B`k´2_lb<=-1 & B`k´2_ub>=-1)
 24.     summarize cov`k´2, meanonly
 25.     matrix res1[`k´,8]=r(mean)
 26.     quietly replace B`k´1=(B`k´1-1)^2
 27.     summarize B`k´1, meanonly
 28.     matrix res1[`k´,3]=r(mean)
 29.     quietly replace B`k´2=(B`k´2+1)^2
 30.     summarize B`k´2, meanonly
 31.     matrix res1[`k´,4]=r(mean)
 32. }
. outtable using res1, mat(res1) format(%9.4f) replace
```

Following Burton et al. (2006) and White (2010), we report the bias, mean square error (MSE), median width of 95% confidence interval, and coverage of 95% confidence interval of the estimated coefficients associated with $X_{1,it}$ and $X_{2,it}$ in table 1. We find that the fixed-effects sieve estimator outperforms the fixed-effects estimator in terms of both bias and MSE. As expected, the bias is relatively large in models 2 and 3 because they suffer from endogeneity due to omitting the nonlinear relationship of $X_{3,it}$ and Z_{it}. In terms of the 95% confidence interval, the fixed-effects sieve estimator gives rise to a relatively small interval width. Moreover, coverage probabilities of 95% confidence interval generated by the fixed-effects sieve estimator are quite close to the nominal value (0.95). The average fit of $g(X_{it})$ with the corresponding 95% confidence band in the simulations is presented in figure 1. We see that the average fit is very close to the true function $g(X_{it})$ and that the 95% confidence band is relatively small except for the edges.

Table 1. Simulation results for the parametric parts in DGP 1

	Bias		MSE		95% CI width		Coverage	
	$X_{1,it}$	$X_{2,it}$	$X_{1,it}$	$X_{2,it}$	$X_{1,it}$	$X_{2,it}$	$X_{1,it}$	$X_{2,it}$
Model 1	0.0016	0.0026	0.0007	0.0014	0.1084	0.1559	0.9540	0.9560
Model 2	−0.0547	−0.0769	0.0045	0.0071	0.5667	0.4891	1.0000	1.0000
Model 3	−0.0830	-0.0605	0.0084	0.0048	0.5595	0.4828	1.0000	1.0000

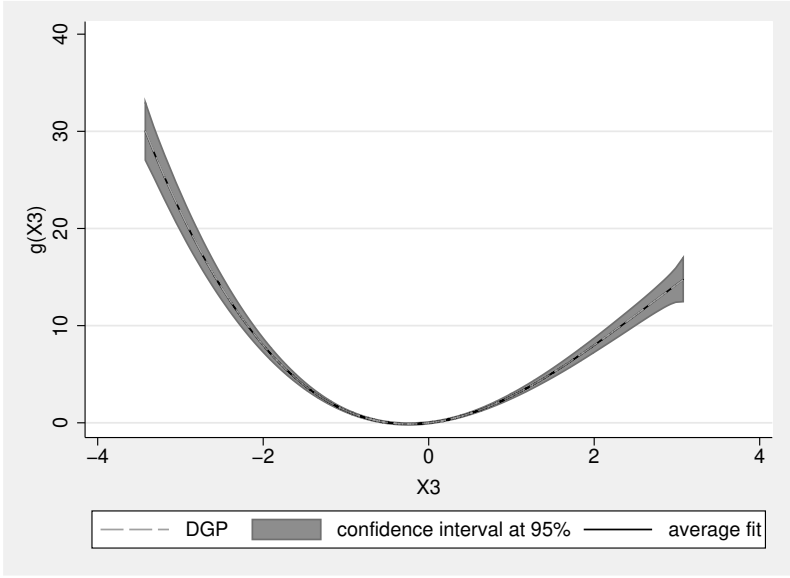

Figure 1. Average fit of $g(X_3)$ across replications

Next, we consider the case of dynamic panel data (DGP2):

$$\begin{aligned} Y_{it} &= g(U_{it})Y_{it-1} + 0.3Y_{it-2} + 0.3X_{it} + a_i + \varepsilon_{it} \\ g(U_{it}) &= \sin\left(\frac{\pi}{3}U_{it}\right) \\ X_{it} &= W_{it} + 0.5a_i \end{aligned}$$

We assume ε_{it} are independent and identically distributed (IID) $N(0,1)$ across both i and t and a_i are IID $N(0,1)$. W_{it} and U_{it} are drawn from IID $U(0,10)$ and IID $U(-9,9)$, respectively.

We compare the performance of `ivxtplfc` and `ivregress 2sls` via the following models:

- Model 4: `ivxtplfc`, considering that X_{it} and Y_{it-2} enter the model linearly, whereas Y_{it-1} is included with a functional coefficient of U_{it}.
- Model 5: `ivregress 2sls`, taking first difference to remove fixed effects and regressing ΔY_{it} on ΔY_{it-1}, ΔY_{it-2}, and ΔX_{it}.
- Model 6: `ivregress 2sls`, taking first difference to remove fixed effects and regressing ΔY_{it} on ΔY_{it-1}, ΔY_{it-2}, ΔX_{it}, and ΔU_{it}.

Note that $L2.Y_{it}$ (the second lag of Y_{it}) is used as the IV in `ivregress 2sls` and that the interaction of $L2.Y_{it}$ and the first lag of the basis functions of U_{it} is constructed as the IVs in `ivxtplfc`. The simulation codes for DGP 2 are presented as follows:

```
. clear all
. set obs 50
number of observations (_N) was 0, now 50
. set seed 789
. generate a=rnormal()
. generate id=_n
. expand 80
(3,950 observations created)
. bysort id: generate year=_n
. generate x=10*runiform()+0.5*a
. generate u=-9+18*runiform()
. generate gf=sin(_pi/3*u)
. generate y=0
. xtset id year
       panel variable:  id (strongly balanced)
        time variable:  year, 1 to 80
                delta:  1 unit
. mata: gfmat=J(2000,500,.)
```

```
. forvalues j=1/500 {
  2.     preserve
  3.     quietly replace y=a+0.3*x+0.3*L2.y+L1.y*gf+sqrt(2)*rnormal() if year>2
  4.     quietly drop if year<41
  5.     quietly generate L_y=L1.y
  6.     quietly generate L2_y=L2.y
  7.     quietly ivxtplfc y L2_y x, zvars(L_y) uvar(u) generate(g) endozflag(1)
>         maxnknots(20) ivz(L2_y,uflag(1)) fast brep(500)
  8.     quietly putmata gfhat=g_1, replace
  9.     mata: gfmat[.,`j´]=gfhat
 10.     matrix B=e(b)
 11.     matrix B=B[1,1..2]
 12.     matrix B1=(nullmat(B1)\B)
 13.     matrix V=e(Vs)
 14.     matrix V=vecdiag(V)
 15.     matrix V=V[1,1..2]
 16.     matrix V1=(nullmat(V1)\V)
 17.     quietly ivregress 2sls D.y D.L2.y D.x (D.L.y=L2.y), noconstant
 18.     matrix B=e(b)
 19.     matrix B=B[1,2..3]
 20.     matrix B2=(nullmat(B2)\B)
 21.     matrix V=e(V)
 22.     matrix V=vecdiag(V)
 23.     matrix V=V[1,1..2]
 24.     matrix V2=(nullmat(V2)\V)
 25.     quietly ivregress 2sls D.y D.L2.y D.x D.u (D.L.y=L2.y), noconstant
 26.     matrix B=e(b)
 27.     matrix B=B[1,2..3]
 28.     matrix B3=(nullmat(B3)\B)
 29.     matrix V=e(V)
 30.     matrix V=vecdiag(V)
 31.     matrix V=V[1,1..2]
 32.     matrix V3=(nullmat(V3)\V)
 33.     restore
 34. }
. * Figure 2
. quietly drop if year<41
. quietly getmata (gfs*)=gfmat
. egen av_fit = rowmean(gfs*)
. egen sd_fit = rowsd(gfs*)
. generate c = invnormal(1 - (100 - 95) / 200)
. generate low = av_fit - c * sd_fit
. generate up = av_fit + c * sd_fit
. twoway (rarea low up u, sort(u) color(gs7))
>    (line av_fit u, sort(u) color(black) lpattern(solid))
>    (line gf u, color(gs10) sort lpattern(longdash)),
>    legend(label(1 confidence interval at 95%) label(3 DGP) label(2 average fit)
>    cols(3) order(3 1 2)) xtitle("U", height(5)) ytitle("g(U)", height(5))
>    scheme(sj)
```

```
. graph2tex, epsfile(fig2) caption(Average fit of g(U) across replications)
> label(fig2)
% exported graph to fig2.eps
% We can see in Figure \ref{fig:fig2} that
\begin{figure}[h]
\begin{centering}
  \includegraphics[height=3in]{fig2}
  \caption{Average fit of g(U) across replications}
  \label{fig:fig2}
\end{centering}
\end{figure}
. * Table 2
. clear
. set obs 500
number of observations (_N) was 0, now 500
. matrix res2=J(3,8,.)
. forvalues k=1/3 {
  2.     quietly svmat B`k´
  3.     summarize B`k´1, meanonly
  4.     matrix res2[`k´,1]=r(mean)-0.3
  5.     summarize B`k´2, meanonly
  6.     matrix res2[`k´,2]=r(mean)-0.3
  7.     svmat V`k´
  8.     quietly replace V`k´1=sqrt(V`k´1)
  9.     quietly generate B`k´1_lb=B`k´1-invnormal(0.975)*V`k´1
 10.     quietly generate B`k´1_ub=B`k´1+invnormal(0.975)*V`k´1
 11.     quietly generate CIlen`k´1=B`k´1_ub-B`k´1_lb
 12.     quietly summarize CIlen`k´1, detail
 13.     matrix res2[`k´,5]=r(p50)
 14.     quietly replace V`k´2=sqrt(V`k´2)
 15.     quietly generate B`k´2_lb=B`k´2-invnormal(0.975)*V`k´2
 16.     quietly generate B`k´2_ub=B`k´2+invnormal(0.975)*V`k´2
 17.     quietly generate CIlen`k´2=B`k´2_ub-B`k´2_lb
 18.     quietly summarize CIlen`k´2, detail
 19.     matrix res2[`k´,6]=r(p50)
 20.     quietly generate cov`k´1=(B`k´1_lb<=0.3 & B`k´1_ub>=0.3)
 21.     summarize cov`k´1, meanonly
 22.     matrix res2[`k´,7]=r(mean)
 23.     quietly generate cov`k´2=(B`k´2_lb<=0.3 & B`k´2_ub>=0.3)
 24.     summarize cov`k´2, meanonly
 25.     matrix res2[`k´,8]=r(mean)
 26.     quietly replace B`k´1=(B`k´1-0.3)^2
 27.     summarize B`k´1, meanonly
 28.     matrix res2[`k´,3]=r(mean)
 29.     quietly replace B`k´2=(B`k´2-0.3)^2
 30.     summarize B`k´2, meanonly
 31.     matrix res2[`k´,4]=r(mean)
 32. }
. outtable using res2, mat(res2) format(%9.4f) replace
(note: file res2.tex not found)
```

Table 2 presents the bias, mean square error (MSE), median width of 95% confidence interval, and coverage of 95% confidence interval of the estimated coefficients associated with Y_{it-2} and X_{it}. Similar to DGP1, the simulation results show that the fixed-effects sieve 2SLS estimator performs much better than the fixed-effects 2SLS estimator in terms of bias, MSE, width of 95% confidence interval, and coverage of 95% confidence interval.

Figure 2 displays the average fit of $g(U_{it})$ with the corresponding 95% band in the simulations. Although $g(U_{it})$ is fluctuated greatly in model 4, the proposed method estimates it quite well.

Table 2. Simulation results for the parametric parts in DGP 2

	Bias		MSE		95% CI width		Coverage	
	Y_{it-2}	X_{it}	Y_{it-2}	X_{it}	Y_{it-2}	X_{it}	Y_{it-2}	X_{it}
Model 4	−0.0009	−0.0005	0.0002	0.0002	0.0542	0.0527	0.9360	0.9400
Model 5	0.0302	0.0357	0.0017	0.0018	0.3623	0.2555	1.0000	1.0000
Model 6	0.0266	0.0385	0.0015	0.0020	0.3603	0.2539	1.0000	1.0000

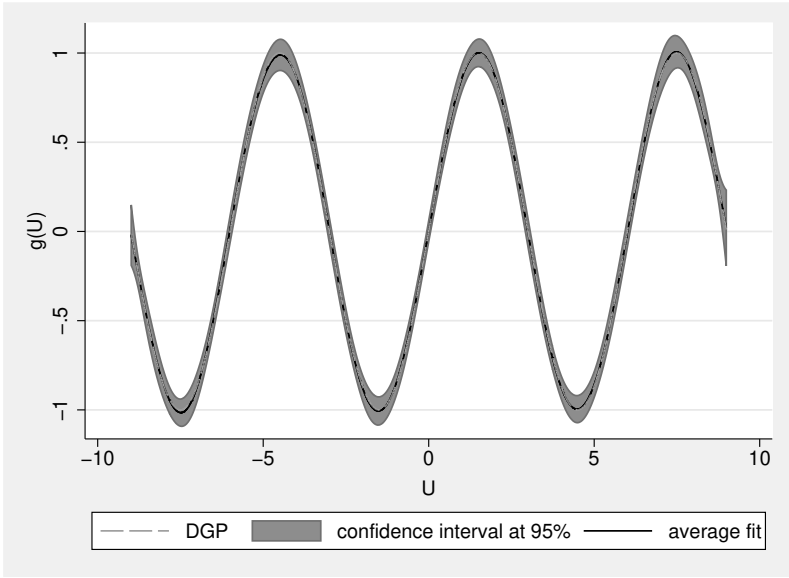

Figure 2. Average fit of $g(U)$ across replications

7 Conclusion

Popular in academic research, partially linear functional-coefficient models are flexible enough to accommodate the nonlinear structure and capture the heterogeneity over individuals and times. This article briefly introduced the new development of functional-coefficient panel-data models and provided three new commands for implementing estimation. Additionally, we illustrated the usefulness of our proposed commands via some simple simulations.

8 Acknowledgments

Kerui Du acknowledges financial support from the National Natural Science Foundation of China (No. 72074184 and 71603148). Yonghui Zhang acknowledges financial support from the National Natural Science Foundation of China (No. 71401166 and 71973141). Qiankun Zhou acknowledges financial support from the National Natural Science Foundation of China (No. 71431006). We are grateful to the anonymous reviewer for the helpful comments and suggestions which led to an improved version of this article.

9 Programs and supplemental materials

To install a snapshot of the corresponding software files as they existed at the time of publication of this article, type

```
. net sj 20-4
. net install st0624     (to install program files, if available)
. net get st0624         (to install ancillary files, if available)
```

10 References

An, Y., C. Hsiao, and D. Li. 2016. Semiparametric estimation of partially linear varying coefficient panel data models. In *Essays in Honor of Aman Ullah (Advances in Econometrics, Volume 36)*, ed. G. González-Rivera, R. C. Hill, and T.-H. Lee, 47–65. Bingley, UK: Emerald. https://doi.org/10.1108/S0731-905320160000036011.

Baum, C. F., and J. P. Azevedo. 2001. outtable: Stata module to write matrix to LaTeX table. Statistical Software Components S419501, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s419501.html.

Burton, A., D. G. Altman, P. Royston, and R. L. Holder. 2006. The design of simulation studies in medical statistics. *Statistics in Medicine* 25: 4279–4292. https://doi.org/10.1002/sim.2673.

Cai, Z., J. Fan, and R. Li. 2000. Efficient estimation and inferences for varying-coefficient models. *Journal of the American Statistical Association* 95: 888–902. https://doi.org/10.1080/01621459.2000.10474280.

Cai, Z., J. Fan, and Q. Yao. 2000. Functional-coefficient regression models for nonlinear time series. *Journal of the American Statistical Association* 95: 941–956. https://doi.org/10.1080/01621459.2000.10474284.

Cai, Z., Y. Fang, M. Lin, and J. Su. 2017. Inferences for a partially varying coefficient model with endogenous regressors. *Journal of Business & Economic Statistics* 37: 158–170. https://doi.org/10.1080/07350015.2017.1294079.

Cai, Z., and Q. Li. 2008. Nonparametric estimation of varying coefficient dynamic panel data models. *Econometric Theory* 24: 1321–1342. https://doi.org/10.1017/S0266466608080523.

Cai, Z., Q. Li, and J. Y. Park. 2009. Functional-coefficient models for nonstationary time series data. *Journal of Econometrics* 148: 101–113. https://doi.org/10.1016/j.jeconom.2008.10.003.

Chen, R., and R. S. Tsay. 1993. Functional-coefficient autoregressive models. *Journal of the American Statistical Association* 88: 298–308. https://doi.org/10.1080/01621459.1993.10594322.

Delgado, M. S., N. McCloud, and S. C. Kumbhakar. 2014. A generalized empirical model of corruption, foreign direct investment, and growth. *Journal of Macroeconomics* 42: 298–316. https://doi.org/10.1016/j.jmacro.2014.09.007.

Feng, G., J. Gao, B. Peng, and X. Zhang. 2017. A varying-coefficient panel data model with fixed effects: Theory and an application to US commercial banks. *Journal of Econometrics* 196: 68–82. https://doi.org/10.1016/j.jeconom.2016.09.011.

Huang, J. Z., C. O. Wu, and L. Zhou. 2004. Polynomial spline estimation and inference for varying coefficient models with longitudinal data. *Statistica Sinica* 14: 763–788.

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

Li, J., H. Liu, and K. Du. 2019. Does market-oriented reform increase energy rebound effect? Evidence from China's regional development. *China Economic Review* 56: 101304. https://doi.org/10.1016/j.chieco.2019.101304.

Libois, F., and V. Verardi. 2013. Semiparametric fixed-effects estimator. *Stata Journal* 13: 329–336. https://doi.org/10.1177/1536867X1301300207.

Lundberg, A. L., K. P. Huynh, and D. T. Jacho-Chávez. 2017. Income and democracy: A smooth varying coefficient redux. *Journal of Applied Econometrics* 32: 719–724. https://doi.org/10.1002/jae.2536.

Newson, R. B. 2000a. bspline: Stata modules to compute B-splines parameterized by their values at reference points. Statistical Software Components S411701, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s411701.html.

———. 2000b. sg151: B-splines and splines parameterized by their values at reference points on the x-axis. *Stata Technical Bulletin* 57: 20–27. Reprinted in *Stata Technical Bulletin Reprints*. Vol. 10, pp. 221–230. College Station, TX: Stata Press.

Statistical Consulting Group. 2017. graph2tex: Convert Stata graph to .eps with LaTeX code. UCLA: Statistical Consulting Group. https://stats.idre.ucla.edu/stat/stata/ado/analysis.

Su, L., I. Murtazashvili, and A. Ullah. 2013. Local linear GMM estimation of functional coefficient IV models with an application to estimating the rate of return to schooling. *Journal of Business & Economic Statistics* 31: 184–207. https://doi.org/10.1080/07350015.2012.754314.

Sun, Y., R. J. Carroll, and D. Li. 2009. Semiparametric estimation of fixed-effects panel data varying coefficient models. In *Nonparametric Econometric Methods*, ed. Q. Li and J. S. Racine, 101–129. Bingley, UK: Emerald. https://doi.org/10.1108/S0731-9053(2009)0000025022.

White, I. R. 2010. simsum: Analyses of simulation studies including Monte Carlo error. *Stata Journal* 10: 369–385. https://doi.org/10.1177/1536867X1001000305.

Zhang, Y., and Q. Zhou. Forthcoming. Partially linear functional-coefficient dynamic panel data models: Sieve estimation and specification testing. *Econometric Reviews*.

About the authors

Kerui Du is an associate professor at the School of Management, Xiamen University. His primary research interests are applied econometrics, energy, and environmental economics.

Yonghui Zhang (corresponding author) is an assistant professor of economics at the School of Economics, Renmin University of China. His primary research interests are both theoretical and applied econometrics, with a focus on nonparametric econometrics.

Qiankun Zhou is an associate professor of economics in the Department of Economics of Louisiana State University. His primary research interests are both theoretical and applied econometrics, with a focus on panel-data econometrics.