The Stata Journal (2019)
19, Number 2, pp. 335–374 DOI: 10.1177/1536867X19854016

xtspj: A command for split-panel jackknife estimation

Yutao Sun
Northeast Normal University
School of Economics
Changchun, China
and Erasmus University Rotterdam
Rotterdam, The Netherlands
sunyt100@nenu.edu.cn

Geert Dhaene
KU Leuven
Department of Economics
Leuven, Belgium
geert.dhaene@kuleuven.be

Abstract. In this article, we present a new command, `xtspj`, that corrects for incidental parameter bias in panel-data models with fixed effects. The correction removes the first-order bias term of the maximum likelihood estimate using the split-panel jackknife method. Two variants are implemented: the jackknifed maximum-likelihood estimate and the jackknifed log-likelihood function (with corresponding maximizer). The model may be nonlinear or dynamic, and the covariates may be predetermined instead of strictly exogenous. `xtspj` implements the split-panel jackknife for fixed-effects versions of linear, probit, logit, Poisson, exponential, gamma, Weibull, and negbin2 regressions. It also accommodates other models if the user specifies the log-likelihood function (and, possibly but not necessarily, the score function and the Hessian). `xtspj` is fast and memory efficient, and it allows large datasets. The data may be unbalanced. `xtspj` can also be used to compute uncorrected maximum-likelihood estimates of fixed-effects models for which no other `xt` (see [XT] **xt**) command exists.

Keywords: st0557, xtspj, split-panel jackknife, incidental parameter problem, maximum likelihood

1 Introduction

Panel-data analysis is an important tool in economic studies. In many panel-data applications, each cross-sectional unit or "individual" is allowed to have a latent unit-specific characteristic, or individual effect, that may be correlated with the covariates and hence must be controlled for. A standard tool to control for unobserved individual effects in panel data is the fixed-effects model, in which a separate parameter, or fixed effect, is included for each of the N individuals in the dataset. Recent empirical studies using fixed-effects models are, for example, Egger and Staub (2016), Griffin and Maturana (2016), and Milner et al. (2016). However, it is well known that the presence of fixed effects may introduce bias into the maximum likelihood (ML) estimator of the common parameters (the parameters assumed to be shared by all individuals, such as regression coefficients) when the number of time periods, T, is small. Specifically, the large-N, fixed-T probability limit of the ML estimator may be inconsistent. This is known as the incidental parameter problem (IPP) of Neyman and Scott (1948). Lancaster (2000) provides a survey on the IPP. Some models that are subject to the IPP are the probit model (for example, Fernández-Val [2009]), the logit model (for example, Katz [2001]), and the tobit model (for example, Greene [2004]). But even in the linear model with fixed effects, there is an IPP for the slope coefficients unless the regressors are strictly exogenous (Chudik, Pesaran, and Yang 2018).

To describe the IPP, let us consider an outcome variable Y_{it}, where $i = 1, \ldots, N$ and $t = 1, \ldots, T$. Suppose Y_{it}, given $\mathbf{X}_{it}$, has a density function $f(Y_{it}|\mathbf{X}_{it}; \boldsymbol{\theta}, \alpha_i)$, where $\boldsymbol{\theta}$ is the common parameter vector, α_i is the scalar fixed-effects parameter associated with the ith individual, and $\mathbf{X}_{it}$ is a vector of observed covariates. The ML estimator $\widehat{\alpha}_i$ of α_i is to be obtained only from T observations. Therefore, when T is fixed, $\widehat{\alpha}_i$ remains random even as $N \to \infty$. This randomness enters the log likelihood and, in many models, causes the ML estimator $\widehat{\boldsymbol{\theta}}$ of $\boldsymbol{\theta}$ to converge to an incorrect probability limit $\boldsymbol{\theta}_T \neq \boldsymbol{\theta}_0$, where $\boldsymbol{\theta}_0$ is the true value of $\boldsymbol{\theta}$. For a fixed T, no consistent alternative estimator of $\boldsymbol{\theta}_0$ can generally be found because $\boldsymbol{\theta}_0$ may not be point identified; see, for example, Honoré and Tamer (2006) and Chamberlain (2010). When both $N, T \to \infty$, the randomness in $\widehat{\alpha}_i$ vanishes and $\widehat{\boldsymbol{\theta}}$ converges to $\boldsymbol{\theta}_0$ in general. However, when N/T converges to a positive constant as $N, T \to \infty$, the asymptotic distribution of $\sqrt{NT}(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0)$ is often not centered at zero, invalidating inference and confidence intervals based on standard ML theory; see, for example, Hahn and Kuersteiner (2002) and Hahn and Newey (2004).

Researchers have been attempting to find solutions to the IPP. For instance, Cox and Reid (1987), Woutersen (2001), and Lancaster (2002) show that a parameter transformation that orthogonalizes $\boldsymbol{\theta}$ and the α_i may help to resolve the IPP or at least mitigate it. Andersen (1970) and Chamberlain (1980) show that in certain models, the maximizer of a conditional likelihood function, given sufficient statistics for the α_i, is a large-N, fixed-T consistent estimator; the logit model is a well-known example. However, these methods are not general, because there is no guarantee that, in a given model, an orthogonalizing parameter transformation exists nor that a conditional likelihood exists.

In search for methods that apply more generally, researchers have been trying to correct for the bias of $\widehat{\boldsymbol{\theta}}$ caused by the fixed effects. These methods are approximate in nature. The idea is to remove the first-order term of a large-T approximation of the bias of $\widehat{\boldsymbol{\theta}}$. Hahn and Newey (2004) and Hahn and Kuersteiner (2011) develop formulas of the first-order bias of $\widehat{\boldsymbol{\theta}}$. This formula, evaluated at ML estimates of $\boldsymbol{\theta}$ and the α_i, is subtracted from $\widehat{\boldsymbol{\theta}}$ to give a first-order bias-corrected estimate, that is, an estimate with first-order bias equal to zero. Fernández-Val and Weidner (2016) provide the first-order bias formula for the case where both individual effects and time effects are present. Alternatively, the log-likelihood function or, equivalently, the score function may be treated as the object that has to be bias corrected. For instance, Arellano and Bonhomme (2009) and Arellano and Hahn (2016) discuss first-order modified log-likelihood functions as an alternative to the usual log-likelihood function. Maximization of a modified log-likelihood function gives a first-order bias-corrected estimate. Li, Lindsay, and Waterman (2003) consider first-order modified score functions. Solving a modified score equation leads to a first-order bias-corrected estimate.

Approximate bias corrections can also be carried out without explicit bias formulas, for example, by using the jackknife. The jackknife is due to Quenouille (1949, 1956). In the context of the IPP, Hahn and Newey (2004) introduce the delete-one panel jackknife for first-order bias correction in static panel models. For dynamic models, Dhaene and Jochmans (2015) propose the split-panel jackknife (SPJ). The panel is split into two half-panels along the time dimension, and ML is applied to each half-panel separately, giving $\widehat{\boldsymbol{\theta}}_1$ and $\widehat{\boldsymbol{\theta}}_2$, say. The SPJ estimate, then, is defined as $2\widehat{\boldsymbol{\theta}} - (\widehat{\boldsymbol{\theta}}_1 + \widehat{\boldsymbol{\theta}}_2)/2$ and is a first-order bias-corrected estimate of $\boldsymbol{\theta}$. The same procedure can also be applied to the (concentrated) log likelihood to give a jackknifed log likelihood, whose maximizer is a first-order bias-corrected estimate of $\boldsymbol{\theta}$.

In this article, we implement the SPJ in Stata. Our new command, `xtspj`, takes as input a log likelihood written in terms of one or more linear indices (equations in the Stata language), either user-specified or preprogrammed. `xtspj` currently comes with preprogrammed fixed-effects log likelihoods of the (Gaussian) linear, probit, logit, Poisson, exponential, gamma, Weibull, and negbin2 regression models. The covariates are allowed to contain lagged values of the dependent variable. Hence, autoregressive versions of these models can be fit without user input. As main output, `xtspj` produces the SPJ estimates (in one of the two possible variants) or, if desired, the uncorrected ML estimates. The maximization routine used by `xtspj` exploits the sparsity of the Hessian of the log likelihood, so it is fast and memory efficient.

`xtspj` allows the panel dataset to be unbalanced. As discussed in Dhaene and Jochmans (2015), the jackknifed log-likelihood variant of the SPJ naturally allows for unbalanced data, so `xtspj` implements this variant as described there. It is less straightforward to define the jackknifed estimator variant for unbalanced data. `xtspj` defines and implements this variant via an estimator that is close to the ML estimator: the weighted average of the ML estimates associated with the balanced panel components that jointly form the full (unbalanced) panel, with weights taken in proportion to the balanced panel component sample sizes. Each of the balanced panel estimates can be jackknifed separately, and the resulting estimates can be averaged using the same weights. We conduct a simulation study of this variant of the SPJ when the data are unbalanced, showing that it works well. It should be mentioned that Chudik, Pesaran, and Yang (2018) introduce another variant of the SPJ for unbalanced data. They split the full (unbalanced) panel into two unbalanced half-panels such that the observations for each cross-sectional unit i are equally split over the two half-panels, and then form the SPJ estimate in the same way as in the balanced case.

`xtspj` is a complementary tool to the commands `probitfe` and `logitfe` recently developed by Cruz-Gonzalez, Fernández-Val, and Weidner (2017). These commands implement analytical and jackknife bias corrections—including the SPJ—for common parameter and average marginal effect estimates in probit and logit models with individual effects, time effects, or both. In comparison, `xtspj` implements the SPJ for the common parameters in models with individual effects only (if desired, time effects can be included as common parameters). On the other hand, `xtspj` is set up for a more general class of models, including user-written models. `xtspj` is also faster and can handle very large datasets.

This article is organized as follows. Section 2 introduces the use of `xtspj` together with the preprogrammed models and presents the syntax, output, and examples. Section 3 discusses the extension of the SPJ to deal with unbalanced data. Section 4 presents simulation results for unbalanced data. Section 5 provides details on how to use `xtspj` with a user-written model; the probit model serves as an example. Section 6 discusses details of the algorithms underlying `xtspj`. Section 7 concludes. Three appendixes contain technical details and the source code for the evaluation of the preprogrammed log likelihoods. The full source code of the preprogrammed log likelihoods, the code for the simulations, and additional simulation results are available in an online supplement available at https://www.researchgate.net/publication/328837218.

2 The command: Basics

This section presents the basic syntax and use of `xtspj`. For `xtspj` with user-written models, see section 5. Importantly, `xtspj` also computes the uncorrected ML estimate, so it may as well be used for fast and memory-efficient computation of ML estimates of fixed-effects models for which a suitable **xt** (see [XT] **xt**) command does not exist.

First, recall the definition of the SPJ for balanced data (see section 3 for unbalanced data). Consider a panel dataset of observations Y_{it} with $i = 1, \ldots, N$ indexing the individuals and $t = 1, \ldots, T$ indexing time, and suppose T is even (see Dhaene and Jochmans [2015] for the case with T odd). Let $\widehat{\boldsymbol{\theta}}$ be the ML estimate of $\boldsymbol{\theta}$ computed from the full panel, and let $\widehat{\boldsymbol{\theta}}_1$ and $\widehat{\boldsymbol{\theta}}_2$ be the ML estimates computed from, respectively, the first half-panel, where $t = 1, \ldots, T/2$, and the second half-panel, where $t = T/2 + 1, \ldots, T$. The SPJ estimator is defined as

$$\widetilde{\boldsymbol{\theta}} = 2\widehat{\boldsymbol{\theta}} - \frac{1}{2}\left(\widehat{\boldsymbol{\theta}}_1 + \widehat{\boldsymbol{\theta}}_2\right)$$

The SPJ log likelihood is defined similarly. Let $\widehat{l}(\boldsymbol{\theta})$ be the concentrated log-likelihood function computed from the full panel (by concentrating out the α_i), and let $\widehat{l}_1(\boldsymbol{\theta})$ and $\widehat{l}_2(\boldsymbol{\theta})$ be the concentrated log-likelihood functions computed from, respectively, the first and the second half-panels. Then, the SPJ log likelihood is

$$\dot{l}\left(\boldsymbol{\theta}\right) = 2\widehat{l}(\boldsymbol{\theta}) - \frac{1}{2}\left\{\widehat{l}_1\left(\boldsymbol{\theta}\right) + \widehat{l}_2\left(\boldsymbol{\theta}\right)\right\}$$

and the corresponding jackknife estimator is

$$\dot{\boldsymbol{\theta}} = \arg\max_{\theta} \dot{l}\left(\boldsymbol{\theta}\right)$$

Both $\widetilde{\boldsymbol{\theta}}$ and $\dot{\boldsymbol{\theta}}$ are first-order bias-corrected estimators. For a comparison of the two variants, Dhaene and Jochmans (2015) provide extensive discussion and simulations.

2.1 Syntax

`xtspj` *depvar* [*indepvars*] [*if*] [*in*], `model(`*string*`)` `method(`*string*`)` [`level(`*#*`)` `ltol(`*#*`)` `ptol(`*#*`)` `maxiter(`*#*`)` `diagnosis` `verbose` `alpha(`*newvar*`)`]

2.2 Description

`xtspj` implements the first-order SPJ (also termed the half-panel jackknife) for possibly nonlinear models with fixed effects. `xtspj` accepts balanced and unbalanced datasets, with one restriction: for every cross-sectional unit i, there must be no gaps in the time series. Hence, missing data are allowed only at the beginning and end of the observation period. *depvar* is the regressand in the model, and *indepvars* is an optional list of regressors that may contain lagged values of *depvar*. The data must be `xtset` (see [XT] **xtset**) with both *panelvar* and *timevar*. `xtspj` accepts *fvvarlist* (see [U] **11.4.3 Factor variables**) (for example, `i.x`) and *tsvarlist* (see [U] **11.4.4 Time-series varlists**) (for example, `L.x`) in *indepvars* but not in *depvar*. Prior to estimation, the *indepvars* are checked for multicollinearity. When *indepvars* contains lagged values of *depvar* up to p lags, the model becomes autoregressive of order p and is fit conditionally on the first p observations. The current version of `xtspj` maximizes the objective function using the Newton–Raphson algorithm.

2.3 Options

`model(`*string*`)` specifies the type of regression model to be fit: probit (`probit`), logit (`logit`), linear (`linear`), negative binomial (`negbin`), Poisson (`poisson`), exponential (`exponential`), gamma (`gamma`), Weibull (`weibull`), or some other user-written model. For user-written models, see section 5. The linear model can also be fit by specifying `model(regress)`; see section 2.5. `model()` is required.

`method(`*string*`)` takes `none`, `like`, or `parm` for, respectively, no correction ($\widehat{\boldsymbol{\theta}}$), the SPJ based on the jackknifed log likelihood ($\dot{\boldsymbol{\theta}}$), and the SPJ based on the jackknifed estimator ($\widetilde{\boldsymbol{\theta}}$). `method()` is required.

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

`ltol(`*#*`)` sets the tolerance level for changes in the objective function value. When the difference between the objective function values in the current and the previous iteration, divided by the absolute value of the current objective function value, is nonnegative and less than `ltol(`*#*`)`, the algorithm stops and reports that convergence has been achieved. The default is `ltol(1e-4)`.

`ptol(`*#*`)` sets the tolerance level for changes in the parameter values. The algorithm stops and reports convergence when the change in the parameter is less than `ptol(`*#*`)`. The change is computed as the absolute difference between $\boldsymbol{\theta}$ in the current and the previous iteration. When $\boldsymbol{\theta}$ is a vector, the maximum element of the vector of absolute differences is taken. The default is `ptol(1e-4)`.

`maxiter(`$\#$`)` sets the maximum number of iterations the algorithm is allowed to use. The default is `maxiter(100)`.

`diagnosis` specifies that a simple diagnostic algorithm be invoked when the Newton–Raphson algorithm gives an updated parameter vector that does not improve the objective function value. This diagnostic algorithm is slow and disabled by default. Our recommendation is to activate `diagnosis` (together with `verbose`) in case of nonconvergence problems.

`verbose` specifies whether the iteration log of the maximization (that is, the objective function values) and extra notes (for example, about omitted individuals) should be displayed on the screen. When `method(parm)` is requested, the iteration log can be lengthy. It is disabled by default.

`alpha(`*newvar*`)` specifies the name of the variable to be created, if needed, to store the estimates $\widehat{\alpha}_i$ of the fixed effects α_i (the same value $\widehat{\alpha}_i$ for all T observations corresponding to i). When `method(none)` is requested, $\widehat{\alpha}_i$ is the ML estimate of α_i. When `method(like)` or `method(parm)` is requested, $\widehat{\alpha}_i$ is the ML estimate of α_i with $\boldsymbol{\theta}$ held fixed in the likelihood function at $\dot{\boldsymbol{\theta}}$ or $\widetilde{\boldsymbol{\theta}}$, respectively.

2.4 Stored results

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

Scalars	
`e(N)`	number of observations
`e(converged)`	contains `1` if the maximization converged
`e(empty)`	contains `1` if there are no covariates in the model
`e(ll)`	log-likelihood value (only for `method(none)` or `method(like)` when `model(regress)` is not specified)
Macros	
`e(cmd)`	`xtspj`
`e(cmdline)`	command as typed
`e(title)`	title of table of results
`e(vce)`	`oim` or *vcetype* specified in `vce()` (for `model(regress)` only)
`e(vcetype)`	title used to label Std. Err. (for `model(regress)` only)
`e(properties)`	`b V`
`e(model)`	value of `model()`
`e(method)`	value of `method()`
`e(depvar)`	the regressands or the list of regressands (when a user-written model with several *eq* was specified)
Matrices	
`e(b)`	coefficient vector
`e(V)`	covariance matrix
Functions	
`e(sample)`	marks estimation sample

2.5 Remarks

The covariance matrix of the estimated $\boldsymbol{\theta}$ is computed as follows. When no correction is requested (`method(none)`), the usual `oim` covariance matrix based on the Hessian of the concentrated log likelihood is computed—see [R] ***vce_option***. When `method(like)` or `method(parm)` is requested, the covariance matrix is obtained from the same Hessian but now evaluated at the corresponding estimate, $\dot{\boldsymbol{\theta}}$ or $\widetilde{\boldsymbol{\theta}}$. This requires maximizing the uncorrected log likelihood with respect to the fixed-effects parameters while $\boldsymbol{\theta}$ is kept fixed at $\dot{\boldsymbol{\theta}}$ or $\widetilde{\boldsymbol{\theta}}$. For this maximization, the options `ltol()`, `ptol()`, and `maxiter()` are also effective.

When `diagnosis` is turned on, a diagnostic algorithm is invoked every time the current objective function value in the Newton–Raphson algorithm is less than that in the previous iteration (see also Baldick [2006, 403]). Given the gradient vector and Hessian matrix obtained from the previous iteration, the algorithm successively reduces the step size, used to update the parameter vector, from `1` to `0.5`, `0.5^2`, ..., possibly down to a minimum value of `0.5^10`. The objective function is evaluated at the parameter vector updated with the reduced step size. When the objective function value improves on that from the previous iteration, the diagnostic algorithm stops, and maximization continues from the currently updated parameter vector. Otherwise, the step size is further reduced unless the minimum step size is already reached, in which case the diagnostic algorithm reports that no improvement could be achieved by reducing the step size, the algorithm stops, and maximization continues from the parameter vector obtained before the diagnostic algorithm was called.

With regard to `ereturn`, `e(ll)` is a stored result only when `method(none)` or `method(like)` is requested.

The standard behavior of `xtspj` is to compute the estimates using the Newton–Raphson algorithm to maximize the associated objective functions. This also applies to `model(linear)`. However, for the linear model, the ML and the two jackknife estimates are also available in closed form. For the ML and jackknifed ML estimates, this is straightforward: ML is just ordinary least squares, and the jackknifed ML estimate is just a linear combination of ordinary least-squares estimates. For the jackknife estimate based on the jackknifed log likelihood, $\dot{\boldsymbol{\theta}}$, the closed-form expression (accommodating unbalanced data) is derived in appendix A. As an alternative to `model(linear)`, `model(regress)` computes the estimates in the linear model directly from the closed-form expressions and therefore is much faster.

For the probit and logit models, a check is performed for each individual i in each of the half-panels to ensure that Y_{it} varies across t. This is to drop individuals for whom the regressand is perfectly predicted by setting α_i equal to $-\infty$ or $+\infty$. These individuals are not informative about $\boldsymbol{\theta}$ in the corresponding half-panel and are excluded from the estimation altogether. A similar check is performed in, for example, `probit` (see [R] **probit**). Similarly, for the Poisson and negative binomial regression models, individuals for which the regressand is zero for all t are uninformative and are therefore dropped. For the exponential, gamma, and Weibull regression models, individuals for which the regressand is nonpositive are dropped, similarly to `streg` (see [ST] **streg**).

2.6 Example

To illustrate `xtspj`, we use a synthetic dataset on whether workers complain to managers at fast-food restaurants. The data setup is as follows:

```
. webuse chicken
. generate worker=.
  (output omitted)
. by restaurant: replace worker=_n
  (output omitted)
. xtset restaurant worker
  (output omitted)
. label define sex 0 "female" 1 "male"
. label values gender sex
```

Uncorrected fixed-effects probit ML

```
. xtspj complain age i.gender, model(probit) method(none)
xtspj - probit - ML                             Number of obs     =      1,858

    complain       Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]

         age   -.0506949   .0154649    -3.28   0.001    -.0810055   -.0203844

      gender
      female           0  (empty)
        male     .394683   .0726083     5.44   0.000     .2523734    .5369927
```

The model is specified with *fvvarlist* (see [U] **11.4.3 Factor variables**) in the list of regressors, the output table is standard, and `i0.gender`, with value `female`, is omitted because of multicollinearity. The omitted variable is listed in the output table but is assigned a zero coefficient. `Number of obs` reports the effective number of cross-sectional and time-series observations used in the estimation, after excluding uninformative individuals (restaurants, in this example). When the option `verbose` is specified, the list of excluded individuals and the list of omitted regressors are reported. The title of the table, `xtspj - probit - ML`, lists the name of the command, the name of the model, and the estimation method.

Jackknifed log likelihood

```
. xtspj complain age i.gender, model(probit) method(like)
xtspj - probit - Jackknifed log-likelihood      Number of obs     =        524

    complain       Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]

         age   -.0448833   .0272204    -1.65   0.099    -.0982343    .0084677

      gender
     female           0  (empty)
       male    .5723342   .1271278     4.50   0.000     .3231684       .8215
```

Number of obs is much smaller here because any individual i is kept only if it is informative in each of the subpanels used to form the jackknife estimate. Hence, individuals that are informative in the full panel but not in one or more subpanels are excluded. Likewise, the multicollinearity check is performed at the level of each subpanel and, hence, it may occur that additional regressors have to be omitted (relative to ML) because they do not pass the multicollinearity check in each subpanel. Clearly, when the number of observations or the list of regressors is different between the ML and the jackknife estimates, it becomes harder to compare the estimates.

Jackknifed ML

```
. xtspj complain age i.gender, model(probit) method(parm)
xtspj - probit - Jackknifed ML                  Number of obs     =        524

    complain       Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]

         age   -.0483349   .0272095    -1.78   0.076    -.1016645    .0049947

      gender
     female           0  (empty)
       male    .5138605   .1266843     4.06   0.000     .2655638    .7621572
```

Number of obs and the set of omitted regressors are the same as for method(like). This is because method(like) and method(parm) use the same multicollinearity and informativeness checks. The estimates, however, are different.

3 Unbalanced panels and the SPJ

3.1 Unbalanced panels and the ML estimator

For simplicity of the analysis of unbalanced data, it is assumed that missingness is at random and that every individual i contributes only consecutive data (that is, there are no gaps). Then, an unbalanced panel can always be viewed as the union of J balanced panels, indexed by $j = 1, \ldots, J$. The jth balanced panel has N_j individuals, each with

T_j consecutive periods of data. Consider now an asymptotic setup where $N_j, T_j \to \infty$ and $N_j/T_j \to \kappa_j$ with $0 < \kappa_j < \infty$ for all j, and let J be fixed. First, the ML estimator obtained from the unbalanced panel is briefly studied, and a closely related estimator is suggested that can easily be jackknifed.

For the (i,t)th observation of the jth balanced panel, denote the outcome variable as Y_{it}^j and the regressors as $\mathbf{X}_{it}^j$. The conditional density of Y_{it}^j, given $\mathbf{X}_{it}^j$, is $f(Y_{it}^j|\mathbf{X}_{it}^j;\boldsymbol{\theta},\alpha_i^j)$, where α_i^j is the fixed-effects parameter. Let

$$\widehat{\boldsymbol{\theta}}^j = \arg\max_{\theta} \widehat{l}^j(\boldsymbol{\theta}), \qquad \widehat{l}^j(\boldsymbol{\theta}) = \frac{1}{N_jT_j}\sum_{i=1}^{N_j}\sum_{t=1}^{T_j}\log f\left\{Y_{it}^j|\mathbf{X}_{it}^j;\boldsymbol{\theta},\widehat{\alpha}_i^j(\boldsymbol{\theta})\right\}$$

$$\widehat{\alpha}_i^j(\boldsymbol{\theta}) = \arg\max_{\alpha_i^j}\frac{1}{T_j}\sum_{t=1}^{T_j}\log f\left(Y_{it}^j;|\mathbf{X}_{it}^j\boldsymbol{\theta},\alpha_i^j\right)$$

The jth log likelihood, $\widehat{l}^j(\boldsymbol{\theta})$, is normalized by the number of observations in the jth balanced panel, so $\widehat{l}^j(\boldsymbol{\theta}) = O_p(1)$. The log likelihood corresponding to the full (that is, unbalanced) panel, normalized by the number of observations, is

$$\begin{aligned}\widehat{l}(\boldsymbol{\theta}) &= \frac{1}{\sum_{j=1}^J N_jT_j}\sum_{j=1}^{J}\sum_{i=1}^{N_j}\sum_{t=1}^{T_j}\log f\left\{Y_{it}^j|\mathbf{X}_{it}^j;\boldsymbol{\theta},\widehat{\alpha}_i^j(\boldsymbol{\theta})\right\}\\ &= \sum_j w_j\widehat{l}^j(\boldsymbol{\theta})\end{aligned}$$

where $w_j = N_jT_j/\sum_j N_jT_j$ is the weight of the jth balanced panel. Hence, the ML estimator is

$$\widehat{\boldsymbol{\theta}} = \arg\max_{\theta}\sum_j w_j\widehat{l}^j(\boldsymbol{\theta})$$

Consider the weighted average of the $\widehat{\boldsymbol{\theta}}^j$,

$$\widehat{\boldsymbol{\theta}}_w = \sum_j w_j\widehat{\boldsymbol{\theta}}^j$$

and call it the weighted-average ML estimator. The ML and weighted-average ML estimators are closely related, as the expressions show. It may be conjectured that they are asymptotically equivalent under weak conditions. We will not study the precise conditions here, because they are not the focus of this article. We will, however, conduct a small-scale simulation study in section 4, showing that $\widehat{\boldsymbol{\theta}}$ and $\widehat{\boldsymbol{\theta}}_w$ are close to each other. Our proposal is to jackknife $\widehat{\boldsymbol{\theta}}$ by jackknifing $\widehat{\boldsymbol{\theta}}_w$, as discussed below in section 3.2.

As an example, the remainder of this subsection examines the many-normal-means IPP of Neyman and Scott (1948) under data unbalancedness. The observations Y_{it} follow the normal distribution $\mathcal{N}(\alpha_i,\theta_0)$, with common variance θ_0 and different means α_i; there are no covariates. When the panel is balanced, the (normalized) log-likelihood function is

$$\frac{1}{NT}\sum_{i,t}\log f\left(Y_{it};\theta,\alpha_i\right)=-\frac{1}{2}\log\left(2\pi\right)-\frac{1}{2}\log\theta-\frac{1}{NT}\sum_{i,t}\frac{\left(Y_{it}-\alpha_i\right)^2}{2\theta}$$

Plugging in $\overline{Y}_i=\sum_t Y_{it}/T=\widehat{\alpha}_i(\theta)$ for α_i gives the concentrated log likelihood,

$$\widehat{l}\left(\theta\right)=-\frac{1}{2}\log\left(2\pi\right)-\frac{1}{2}\log\theta-\frac{1}{NT}\sum_{i,t}\frac{\left(Y_{it}-\overline{Y}_i\right)^2}{2\theta}$$

and, on maximizing,

$$\widehat{\theta}=\frac{1}{NT}\sum_{i,t}\left(Y_{it}-\overline{Y}_i\right)^2$$

It follows easily that, as $N\to\infty$ with T fixed,

$$\widehat{\theta}\to_p\theta_0-\frac{\theta_0}{T}$$

that is, $\widehat{\theta}$ is inconsistent and biased toward zero. All of this has been well known since Neyman and Scott (1948).

Now consider the unbalanced case. For the jth balanced panel, the concentrated log likelihood and its maximizer are

$$\widehat{l^j}\left(\theta\right)=-\frac{1}{2}\log\left(2\pi\right)-\frac{1}{2}\log\theta-\frac{1}{N_jT_j}\sum_{i=1}^{N_j}\sum_{t=1}^{T_j}\frac{\left(Y_{it}^j-\overline{Y}_i^j\right)^2}{2\theta}$$

$$\widehat{\theta^j}=\frac{1}{N_jT_j}\sum_{i=1}^{N_j}\sum_{t=1}^{T_j}\left(Y_{it}^j-\overline{Y}_i^j\right)^2$$

where $\overline{Y}_i^j=\sum_{t=1}^{T_j}Y_{it}^j/T_j=\widehat{\alpha}_i^j(\theta)$ is the ML estimator of α_i^j. The concentrated log likelihood for the full panel is

$$\begin{aligned}\widehat{l}\left(\theta\right)&=\sum_j w_j\widehat{l^j}\left(\theta\right)\\&=-\frac{1}{2}\log\left(2\pi\right)-\frac{1}{2}\log\theta-\sum_j w_j\frac{1}{N_jT_j}\sum_{i=1}^{N_j}\sum_{t=1}^{T_j}\frac{\left(Y_{it}^j-\overline{Y}_i^j\right)^2}{2\theta}\end{aligned}$$

with maximizer

$$\begin{aligned}\widehat{\theta}&=\sum_j w_j\frac{1}{N_jT_j}\sum_{i=1}^{N_j}\sum_{t=1}^{T_j}\left(Y_{it}^j-\widehat{\alpha}_i^j\right)^2\\&=\sum_j w_j\widehat{\theta^j}\\&=\widehat{\theta}_w\end{aligned}$$

Thus, the ML estimator, $\widehat{\theta}$, and the weighted-average ML estimator, $\widehat{\theta}_w$, are identical. Furthermore, when $N_j \to \infty$ and T_j is fixed for all j,

$$\widehat{\theta} \to_p \theta_0 - \sum_j w_j \frac{\theta_0}{T_j}$$

When $T_j \to \infty$ for all j,

$$\sum_j w_j \frac{\theta_0}{T_j} = \sum_j w_j O\left(T_j^{-1}\right)$$

Even when J is allowed to grow with the sample size, $\widehat{\theta}$ will typically be consistent because

$$\sum_j w_j O\left(T_j^{-1}\right) = \sum_j \frac{N_j T_j}{\sum_j N_j T_j} O\left(T_j^{-1}\right) = \frac{\sum_j N_j}{\sum_j N_j T_j} O\left(1\right)$$

where $\sum_j N_j / \sum_j N_j T_j \to 0$ under weak conditions on the sequences (N_j, T_j), $j = 1, \ldots, J$, as J grows. For the asymptotic distribution of $\sqrt{\sum_j N_j T_j}(\widehat{\theta} - \theta_0)$ to be centered at 0, it can easily be shown that

$$\frac{\sum_j N_j}{\sqrt{\sum_j N_j T_j}} = o\left(1\right) \tag{1}$$

is necessary and sufficient. When the panel is balanced, that is, $J = 1$, $N_1 = N$, and $T_1 = T$, (1) is equivalent to $N/T = o(1)$, which is well known as necessary and sufficient for $\sqrt{NT}(\widehat{\theta} - \theta_0)$ to be centered at 0; see, for example, Hahn and Newey (2004).

3.2 Unbalanced panels and the SPJ

Our proposal is to jackknife $\widehat{\boldsymbol{\theta}}$ by jackknifing the closely related estimator $\widehat{\boldsymbol{\theta}}_w$. Given that $\widehat{\boldsymbol{\theta}}_w$ is a weighted average of ML estimates defined by balanced panels, $\widehat{\boldsymbol{\theta}}_w$ can be jackknifed by parts, that is, by jackknifing each $\widehat{\boldsymbol{\theta}}^j$ in the usual way and then forming a weighted average of the jackknifed $\widehat{\boldsymbol{\theta}}^j$ using weights w_j.

Consider panel j. To allow T_j to be even or odd, we need to introduce a little more notation. When T_j is even, the jackknife splits panel j into two subpanels with $T_j/2$ time periods each. Let

$$S_1^j = \{1, \ldots, T_j/2\} \qquad S_2^j = \{T_j/2 + 1, \ldots, T_j\}$$

The SPJ computes, for $S = S_1^j, S_2^j$,

$$\widehat{\boldsymbol{\theta}}_S^j = \arg\max_{\theta} \widehat{l}_S^j(\boldsymbol{\theta}) \qquad \widehat{l}_S^j(\boldsymbol{\theta}) = \frac{1}{N_j\,|S|} \sum_{i=1}^{N_j} \sum_{t \in S} \log f\left\{Y_{it}^j | \mathbf{X}_{it}^j; \boldsymbol{\theta}, \widehat{\alpha}_{iS}^j\left(\boldsymbol{\theta}\right)\right\}$$

$$\widehat{\alpha}_{iS}^j\left(\boldsymbol{\theta}\right) = \arg\max_{\alpha_i^j} \frac{1}{|S|} \sum_{t \in S} \log f\left(Y_{it}^j | \mathbf{X}_{it}^j; \boldsymbol{\theta}, \alpha_i^j\right)$$

where $|S| = T_j/2$ denotes the number of elements in S. Letting $\widehat{\boldsymbol{\theta}}^j_k = \widehat{\boldsymbol{\theta}}^j_{S^j_k}$ and $\widehat{l}^j_k = \widehat{l}^j_{S^j_k}$ for $k = 1, 2$, the jackknifed versions of $\widehat{\boldsymbol{\theta}}^j$ and $\widehat{l}^j(\boldsymbol{\theta})$ are, as before,

$$\widetilde{\boldsymbol{\theta}}^j = 2\widehat{\boldsymbol{\theta}}^j - \frac{1}{2}\left(\widehat{\boldsymbol{\theta}}^j_1 + \widehat{\boldsymbol{\theta}}^j_2\right)$$
$$\dot{l}^j(\boldsymbol{\theta}) = 2\widehat{l}^j(\boldsymbol{\theta}) - \frac{1}{2}\left\{\widehat{l}^j_1(\boldsymbol{\theta}) + \widehat{l}^j_2(\boldsymbol{\theta})\right\}$$

When T_j is odd, the situation is slightly different because there are two ways of splitting panel j into almost equal subpanels. Let

$$S^j_{11} = \{1, \ldots, \lceil T_j/2 \rceil\} \qquad S^j_{12} = \{\lceil T_j/2 \rceil + 1, \ldots, T_j\}$$
$$S^j_{21} = \{1, \ldots, \lfloor T_j/2 \rfloor\} \qquad S^j_{22} = \{\lfloor T_j/2 \rfloor + 1, \ldots, T_j\}$$

where $\lceil T_j/2 \rceil$ is the least integer τ satisfying $\tau \geq T_j/2$ and $\lfloor T_j/2 \rfloor$ is the greatest integer τ satisfying $\tau \leq T_j/2$. (Note that $S^j_{11} = S^j_{21}$ and $S^j_{12} = S^j_{22}$ when T_j is even.) The jackknifed versions of $\widehat{\boldsymbol{\theta}}^j$ and $\widehat{l}^j(\boldsymbol{\theta})$ are then defined as

$$\widetilde{\boldsymbol{\theta}}^j = 2\widehat{\boldsymbol{\theta}}^j - \frac{1}{2}\left(\overline{\boldsymbol{\theta}}^j_1 + \overline{\boldsymbol{\theta}}^j_2\right) \qquad \overline{\boldsymbol{\theta}}^j_k = \frac{|S^j_{k1}|}{T_j}\widehat{\boldsymbol{\theta}}^j_{S^j_{k1}} + \frac{|S^j_{k2}|}{T_j}\widehat{\boldsymbol{\theta}}^j_{S^j_{k2}} \qquad k = 1, 2$$
$$\dot{l}^j(\boldsymbol{\theta}) = 2\widehat{l}^j(\boldsymbol{\theta}) - \frac{1}{2}\left\{\overline{l}^j_1(\boldsymbol{\theta}) + \overline{l}^j_2(\boldsymbol{\theta})\right\} \quad \overline{l}^j_k(\boldsymbol{\theta}) = \frac{|S^j_{k1}|}{T_j}\widehat{l}^j_{S^j_{k1}}(\boldsymbol{\theta}) + \frac{|S^j_{k2}|}{T_j}\widehat{l}^j_{S^j_{k2}}(\boldsymbol{\theta}) \quad k = 1, 2$$

see Dhaene and Jochmans (2015). Given $\widetilde{\boldsymbol{\theta}}^j$ and $\dot{l}^j(\boldsymbol{\theta})$ for each panel j, the SPJ versions of $\widehat{\boldsymbol{\theta}}_w$ and $\widehat{l}(\boldsymbol{\theta})$ are

$$\widetilde{\boldsymbol{\theta}} = \sum_j w_j \widetilde{\boldsymbol{\theta}}^j \qquad \dot{l}(\boldsymbol{\theta}) = \sum_j w_j \dot{l}^j(\boldsymbol{\theta})$$

The estimator $\widetilde{\boldsymbol{\theta}}$ is new, while $\dot{\boldsymbol{\theta}} = \arg\max_\theta \dot{l}(\boldsymbol{\theta})$ was proposed in Dhaene and Jochmans (2015). The two estimators, $\widetilde{\boldsymbol{\theta}}$ and $\dot{\boldsymbol{\theta}}$, remove the leading bias term of $\widehat{\boldsymbol{\theta}}_w$ and $\widehat{\boldsymbol{\theta}}$, respectively. They are consistent, have the same large-N, large-T asymptotic variance as $\widehat{\boldsymbol{\theta}}_w$ and $\widehat{\boldsymbol{\theta}}$, and their limit distributions are correctly centered. That is, under our asymptotic setup, with $M = \sum_j N_j T_j$ denoting the total number of observations, the limit distributions of $\sqrt{M}(\widetilde{\boldsymbol{\theta}} - \boldsymbol{\theta}_0)$ and $\sqrt{M}(\dot{\boldsymbol{\theta}} - \boldsymbol{\theta}_0)$ are normal and centered at zero, while those of $\sqrt{M}(\widehat{\boldsymbol{\theta}}_w - \boldsymbol{\theta}_0)$ and $\sqrt{M}(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0)$ are normal but not generally centered at zero.

As mentioned in the introduction, Chudik, Pesaran, and Yang (2018) propose an alternative splitting scheme for unbalanced panels. We give a brief comparison here. For simplicity, suppose T_j is even for all $j = 1, \ldots, J$. The SPJ estimate of Chudik, Pesaran, and Yang (2018) is

$$\widetilde{\boldsymbol{\theta}}_{\mathrm{CPY}} = 2\widehat{\boldsymbol{\theta}} - \frac{1}{2}\left(\widehat{\boldsymbol{\theta}}_1 + \widehat{\boldsymbol{\theta}}_2\right)$$

where $\widehat{\boldsymbol{\theta}}$ is the usual ML estimate computed from the full panel and $\widehat{\boldsymbol{\theta}}_k$ $(k = 1, 2)$ is the ML estimate computed from the kth unbalanced half-panel, with observations

$$\begin{aligned} &\{(i,t)\,|i = 1,\dots,N_j; t = 1,\dots,T_j/2; j = 1,\dots,J\}, &&\text{for } k = 1\\ &\{(i,t)\,|i = 1,\dots,N_j; t = T_j/2+1,\dots,T_j; j = 1,\dots,J\}, &&\text{for } k = 2 \end{aligned}$$

The estimator $\widetilde{\boldsymbol{\theta}}$ replaces $\widehat{\boldsymbol{\theta}}$ with $\widehat{\boldsymbol{\theta}}_w = \sum_j w_j \widehat{\boldsymbol{\theta}}^j$ and $\widehat{\boldsymbol{\theta}}_k$ with $\sum_j w_j \widehat{\boldsymbol{\theta}}^j_k$ $(k = 1, 2)$. The latter is a weighted average, over j, of the ML estimate computed from the kth half of the jth balanced panel, with observations

$$\begin{aligned} &\{(i,t)\,|i = 1,\dots,N_j; t = 1,\dots,T_j/2\}, &&\text{for } k = 1\\ &\{(i,t)\,|i = 1,\dots,N_j; t = T_j/2+1,\dots,T_j\}, &&\text{for } k = 2 \end{aligned}$$

An advantage of $\widetilde{\boldsymbol{\theta}}_{\text{CPY}}$ is that it does not rely on the conjectured asymptotic equivalence of $\widehat{\boldsymbol{\theta}}_w$ and $\widehat{\boldsymbol{\theta}}$, while $\widetilde{\boldsymbol{\theta}}$ does. On the other hand, Chudik, Pesaran, and Yang (2018) showed the bias-reduction property of $\widetilde{\boldsymbol{\theta}}_{\text{CPY}}$ in the unbalanced case only for dynamic linear models, whereas $\widetilde{\boldsymbol{\theta}}$ and $\dot{\boldsymbol{\theta}}$ are generally bias reducing in nonlinear models as well (relative to $\widehat{\boldsymbol{\theta}}_w$ and $\widehat{\boldsymbol{\theta}}$, respectively).

3.3 Practical considerations

For some panels j, T_j may be too small for $\widetilde{\boldsymbol{\theta}}^j$ or $\dot{l}^j(\boldsymbol{\theta})$ to be defined. In that case, the individuals in those panels have to be excluded from the estimation. `xtspj` checks only if $T_j \geq 2$. This is a necessary condition for splitting the panel. All individuals with fewer than two periods of data are automatically excluded. When $T_j = 2$, the half-panel will contain only one time period, which in most models is insufficient to estimate $\boldsymbol{\theta}$. For example, the linear fixed-effects model requires at least two time periods in each subpanel, so the theoretical minimum for T_j is 4. Because this is a model-specific issue, the user must determine if T_j is large enough, and, if it is not, the user must exclude the corresponding individuals from the estimation, similarly to the `::Check()` function discussed below in section 5.3. Furthermore, one may also choose to set the minimum for T_j above the theoretical minimum for at least two reasons. First, the panels with small T_j typically contribute more higher-order bias to the SPJ than those with greater T_j. Therefore, excluding panels with small T_j often reduces the bias but at the expense of increased variance. Second, for panels with very small T_j, the half-panel concentrated log likelihoods may be nearly flat, potentially causing numerical problems for computing the jackknife estimates.

`xtspj` does not allow gaps in the data. In particular, for every individual, missing data are allowed only at the beginning of the observation period, at the end, or both at the beginning and the end. In certain situations, this is a realistic assumption, but it is often unrealistic. However, this assumption is without loss of generality when the observations are assumed to be independent across time (conditional on covariates) because then we may simply reassign the observations to different time periods. For dynamic models, the situation is more complicated, and the assumption of consecutive data was primarily made to accommodate dynamic models without difficulty in the analysis. When the model is dynamic and there are gaps in the data, a simple solution (though with loss of efficiency) is to redefine an "individual" as a patch of consecutive observations. For example, an individual i with two patches of consecutive observations, separated by a gap, is replaced by two new individuals, one for each patch. Then, the analysis proceeds as in the case with consecutive data. The efficiency loss of this solution is twofold: more fixed effects have to be estimated, and more initial observations are lost because of conditioning on them. Avoiding the efficiency loss appears difficult in general, although in linear time-series models it may be possible to extend Wincek and Reinsel (1986) to the SPJ.

4 Simulations for unbalanced panels

This section presents a simulation study of the effect of moderate data unbalancedness on the performance of the SPJ. The results are based on 5,000 Monte Carlo replications. We set $N = 500$ throughout. The code used for the simulations is available in the online supplement.

We first present simulation results suggesting that, for unbalanced data, $\widehat{\theta}$ and $\widehat{\theta}_w$ are asymptotically equivalent (note that they are identical for balanced data). We simulated data from the probit model, $Y_{it} = 1(X_{it}\theta_0 + \alpha_i + \varepsilon_{it} \geq 0)$, where $\varepsilon_{it} \sim \mathcal{N}(0, 1)$, with $\theta_0 = 0.5$ and (X_{it}, α_i) as specified below. We first generated balanced panels with $T = 8, \dots, 18$ and then introduced unbalancedness by letting a fraction $r > 0$ of the individuals have $T - 4$ observations instead of T. For example, when $T = 8$ and $r = 0.25$, the unbalanced panel consists of $J = 2$ balanced panels, one with $N_1 = 125$ and $T_1 = 4$, and the other with $N_2 = 375$ and $T_2 = 8$. We generated unbalanced panels with $r = 0.25, 0.5, 0.75$ (for balanced panels, $r = 0$). The covariate was generated as $X_{it} \sim \mathcal{N}(\alpha_i, 1)$, with $\alpha_i = 0$ (design 1) or $\alpha_i \sim \mathcal{N}(0, 1/2)$ (design 2). Design 1 is the case where, in fact, pooled estimation (that is, without fixed effects) is consistent, while design 2 calls for fixed-effects estimation because the α_i are correlated with the regressor. Figures 1 and 2 present the bias and root mean squared error (RMSE) of $\widehat{\theta}$ and $\widehat{\theta}_w$ in the unbalanced designs.[1] The first point to note is that there is an IPP even when there are no fixed effects (design 1). That is, fitting a fixed-effects model already induces an IPP even when the true data-generating process has no individual effects. In both designs, $\widehat{\theta}$ and $\widehat{\theta}_w$ are close to each other and so are their biases and RMSEs. The differences are most pronounced when $T \leq 10$ and $r = 0.5$, and in those cases, $\widehat{\theta}_w$ tends to perform slightly better than $\widehat{\theta}$ in terms of RMSE. As T increases, the difference disappears as expected.

1. Throughout, the Monte Carlo results are conditional on the existence of all estimates. That is, the results are based on the replications for which all subpanel estimates needed for the ML, weighted ML, and jackknife estimates effectively exist. As soon as one subpanel estimate does not exist—because there are no informative individuals—the replication is discarded. The fraction of discarded replications was almost always 0 or close to 0 (less than 2%). In a few cases—those with $T = 8, 9$ and $r = 0.25$, where the jackknife uses subpanels of size $T = 2$ and $N = 125$—around 15% of the replications had to be discarded.

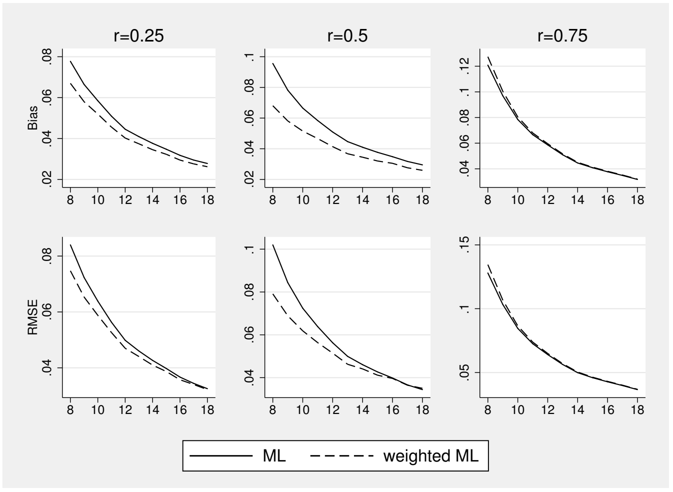

Figure 1. ML and weighted-average ML: probit model, design 1. Model: $Y_{it} = 1(X_{it}\theta_0 + \alpha_i + \varepsilon_{it} \geq 0)$, $\varepsilon_{it} \sim \mathcal{N}(0,1)$. Data generated with $\theta_0 = 0.5$, $X_{it} \sim \mathcal{N}(\alpha_i, 1)$, $\alpha_i = 0$ (design 1), $N = 500$, T on x axis, and rN individuals with $T-4$ instead of T. ML $= \widehat{\theta}$, weighted ML $= \widehat{\theta}_w$.

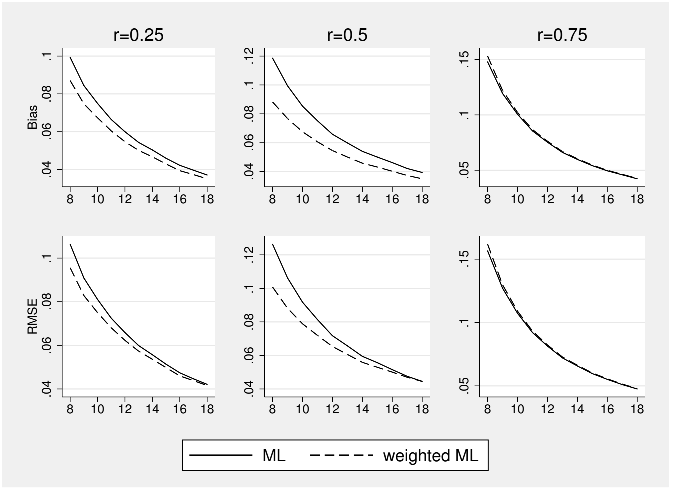

Figure 2. ML and weighted-average ML: probit model, design 2. Model: $Y_{it} = 1(X_{it}\theta_0 + \alpha_i + \varepsilon_{it} \geq 0)$, $\varepsilon_{it} \sim \mathcal{N}(0,1)$. Data generated with $\theta_0 = 0.5$, $X_{it} \sim \mathcal{N}(\alpha_i, 1)$, $\alpha_i \sim \mathcal{N}(0, 1/2)$ (design 2), $N = 500$, T on x axis, and rN individuals with $T-4$ instead of T. ML $= \widehat{\theta}$, weighted ML $= \widehat{\theta}_w$.

Next, we study the effect of data unbalancedness on the ML estimate, $\widehat{\theta}$, and on the SPJ estimates, $\widetilde{\theta}$ and $\dot{\theta}$. We consider three models: the static probit, the static logit, and the stationary Gaussian first-order autoregressive [AR(1)] model. We set $T = 4, \ldots, 18$ in the balanced panels, which serve as a benchmark of comparison with unbalanced panels. For $T \geq 8$, we generated unbalanced panels as above, with $r = 0.25, 0.5, 0.75$.

Figures 3–6 present the results for $\theta_0 = 0.5, 1$. The bias of the ML estimator is uniformly upward both in design 1 and design 2. Both variants of the jackknife yield significant improvements over ML in terms of bias and RMSE. The variant based on the jackknifed log likelihood tends to perform best. The bias of $\widetilde{\theta}$ is uniformly downward in the designs studied, and the bias of $\dot{\theta}$ is uniformly upward. Although this seems to suggest a general pattern, the pattern found here is specific to the chosen design. The effect of data unbalancedness shows a clear picture. For any T, as r increases from 0 to 0.75, an increasing fraction of the panel has only $T-4$ observations, and the incidental parameter bias of all three estimators increases, as expected. Note further that T with $r = 0$ is equivalent to $T+4$ with $r = 1$, so one can see the gradual effect of successively adding 4 time periods of data for a rotating group of 125 individuals from a balanced panel with $T = 4$ up to one with $T = 18$. In line with the theory, the bias changes smoothly for all three estimators in most design points and shows that the jackknife is also bias reducing for unbalanced data.

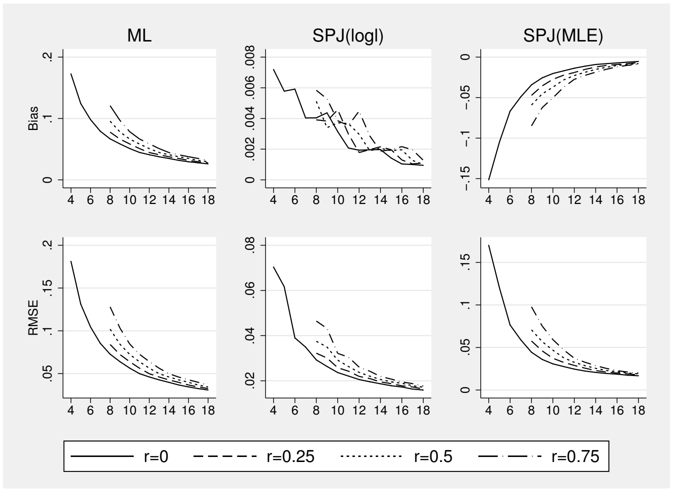

Figure 3. ML and SPJ: probit model, $\theta_0 = 0.5$, design 1. Model: $Y_{it} = 1(X_{it}\theta_0 + \alpha_i + \varepsilon_{it} \geq 0)$, $\varepsilon_{it} \sim \mathcal{N}(0,1)$. Data generated with $\theta_0 = 0.5$, $X_{it} \sim \mathcal{N}(\alpha_i, 1)$, $\alpha_i = 0$ (design 1), $N = 500$, T on x axis, and rN individuals with $T-4$ instead of T. ML $= \widehat{\theta}$, SPJ(logl) $= \dot{\theta}$, SPJ(MLE) $= \widetilde{\theta}$. Maximum likelihood estimation = MLE.

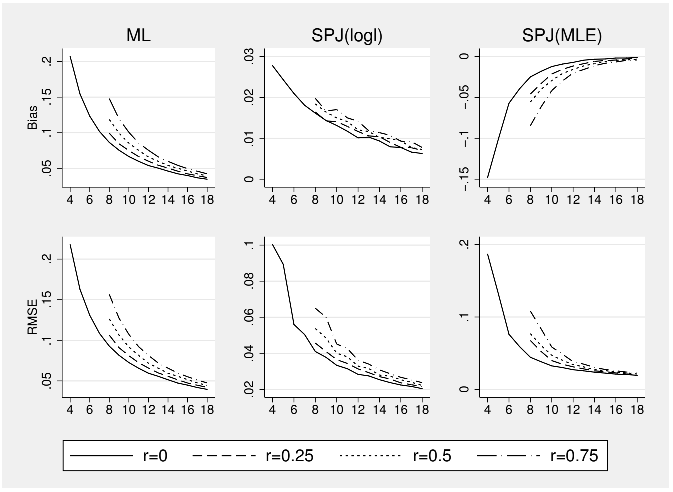

Figure 4. ML and SPJ: probit model, $\theta_0 = 0.5$, design 2. Model: $Y_{it} = 1(X_{it}\theta_0 + \alpha_i + \varepsilon_{it} \geq 0)$, $\varepsilon_{it} \sim \mathcal{N}(0,1)$. Data generated with $\theta_0 = 0.5$, $X_{it} \sim \mathcal{N}(\alpha_i, 1)$, $\alpha_i \sim \mathcal{N}(0, 1/2)$ (design 2), $N = 500$, T on x axis, and rN individuals with $T-4$ instead of T. ML $= \widehat{\theta}$, SPJ(logl) $= \dot{\theta}$, SPJ(MLE) $= \widetilde{\theta}$.

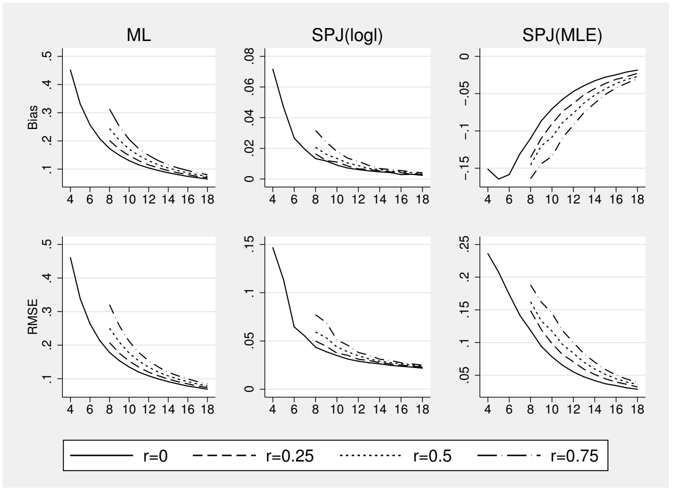

Figure 5. ML and SPJ: probit model, $\theta_0 = 1$, design 1. Model: $Y_{it} = 1(X_{it}\theta_0 + \alpha_i + \varepsilon_{it} \geq 0)$, $\varepsilon_{it} \sim \mathcal{N}(0,1)$. Data generated with $\theta_0 = 1$, $X_{it} \sim \mathcal{N}(\alpha_i, 1)$, $\alpha_i = 0$ (design 1), $N = 500$, T on x axis, and rN individuals with $T-4$ instead of T. ML $= \widehat{\theta}$, SPJ(logl) $= \dot{\theta}$, SPJ(MLE) $= \widetilde{\theta}$.

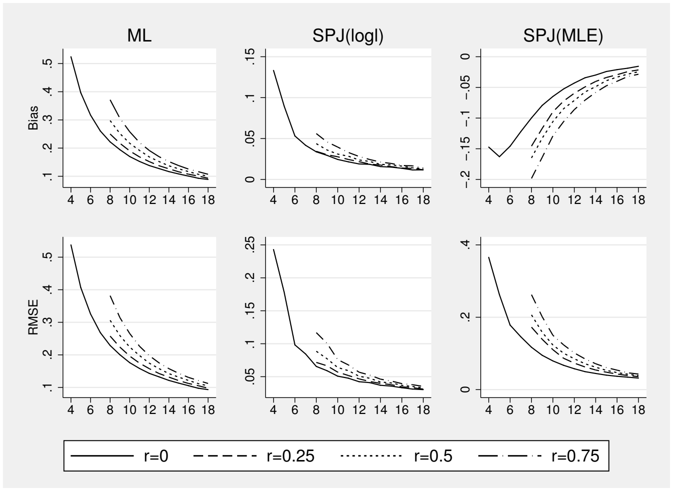

Figure 6. ML and SPJ: probit model, $\theta_0 = 1$, design 2. Model: $Y_{it} = 1(X_{it}\theta_0 + \alpha_i + \varepsilon_{it} \geq 0)$, $\varepsilon_{it} \sim \mathcal{N}(0,1)$. Data generated with $\theta_0 = 1$, $X_{it} \sim \mathcal{N}(\alpha_i, 1)$, $\alpha_i \sim \mathcal{N}(0, 1/2)$ (design 2), $N = 500$, T on x axis, and rN individuals with $T-4$ instead of T. ML $= \widehat{\theta}$, SPJ(logl) $= \dot{\theta}$, SPJ(MLE) $= \widetilde{\theta}$.

For the static logit model, the data were generated as in the probit model above, except that here ε_{it} is standard logistically distributed and, in accordance with the scale of ε_{it}, the covariate was generated as $X_{it} \sim \mathcal{N}(\alpha_i, \pi^2/3)$, with $\alpha_i = 0$ (design 1) or $\alpha_i \sim \mathcal{N}(0, \pi^2/6)$ (design 2). The results for the logit model are available in the online supplement. They are similar to those for the probit model: $\widehat{\theta}$ is upward biased; $\widetilde{\theta}$ and $\dot{\theta}$ are far less biased and have smaller RMSE; the bias of all three estimators changes smoothly with the degree of data unbalancedness; the jackknife is bias reducing for unbalanced data; and, again, the jackknife based on the jackknifed log likelihood performs best.

Figures 7–8 report the results for the stationary Gaussian AR(1) model, with data generated as

$$Y_{i0} \sim \mathcal{N}\left(\frac{\alpha_i}{1-\theta_0}, \frac{1}{1-\theta_0^2}\right); \qquad Y_{it} = \theta_0 Y_{it-1} + \alpha_i + \varepsilon_{it}, \quad t = 1, \ldots T$$

where $\varepsilon_{it} \sim \mathcal{N}(0,1)$, $\alpha_i \sim \mathcal{N}(0, 1/2)$ (design 2), and $\theta_0 = 0.5, -0.5$. The simulation results are qualitatively similar to those in the logit and probit models. Here, however, the jackknifed estimator, $\widetilde{\theta}$, performs better than $\dot{\theta}$ in terms of bias and RMSE. This is in line with the simulation results in Dhaene and Jochmans (2015). Furthermore, the bias is not symmetric in θ_0 around 0, in line with the analysis of Nickell (1981). The corresponding results for the case $\alpha_i = 0$ (design 1) are available in the online supplement. Apart from Monte Carlo error, these results are identical to those for design 2, confirming that all three estimators are invariant with respect to the α_i.

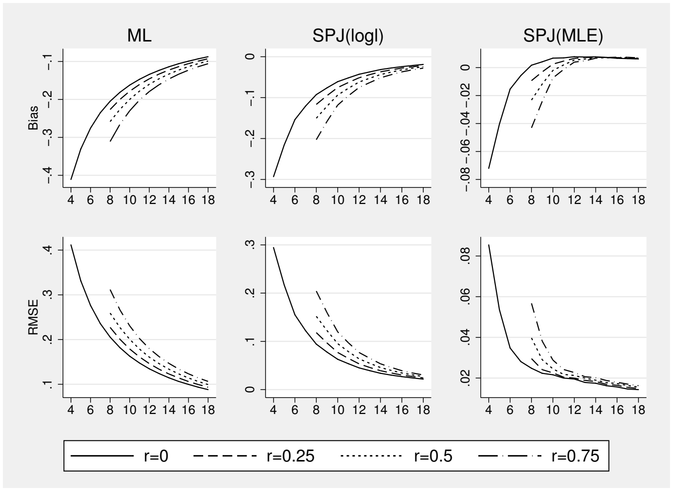

Figure 7. ML and SPJ: Gaussian AR(1) model, $\theta_0 = 0.5$, design 2. Model: $Y_{it} = \theta_0 Y_{it-1} + \alpha_i + \varepsilon_{it}$, $\varepsilon_{it} \sim \mathcal{N}(0,1)$. Data generated with $\theta_0 = 0.5$, stationary Y_{i0}, $\alpha_i \sim \mathcal{N}(0,1/2)$ (design 2), $N = 500$, T on x axis, and rN individuals with $T-4$ instead of T. ML $= \widehat{\theta}$, SPJ(logl) $= \dot{\theta}$, SPJ(MLE) $= \widetilde{\theta}$.

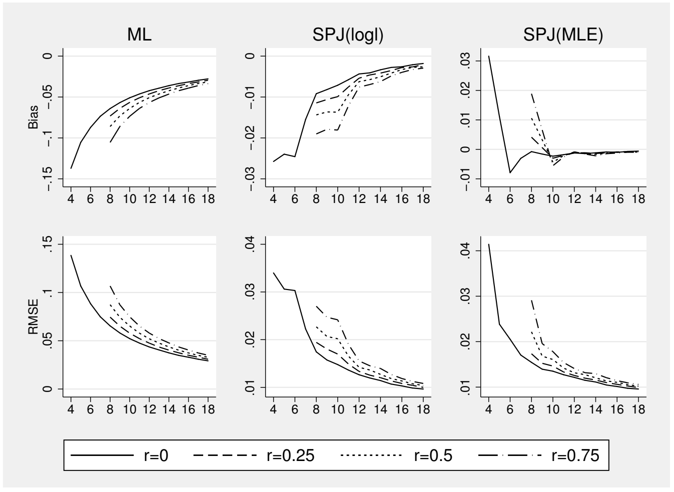

Figure 8. ML and SPJ: Gaussian AR(1) model, $\theta_0 = -0.5$, design 2. Model: $Y_{it} = \theta_0 Y_{it-1} + \alpha_i + \varepsilon_{it}$, $\varepsilon_{it} \sim \mathcal{N}(0,1)$. Data generated with $\theta_0 = -0.5$, stationary Y_{i0}, $\alpha_i \sim \mathcal{N}(0,1/2)$ (design 2), $N = 500$, T on x axis, and rN individuals with $T-4$ instead of T. ML $= \widehat{\theta}$, SPJ(logl) $= \dot{\theta}$, SPJ(MLE) $= \widetilde{\theta}$.

To conclude, the simulation results confirm that $\widehat{\theta}$ can be jackknifed by jackknifing $\widehat{\theta}_w$. The biases of $\widehat{\theta}$, $\widetilde{\theta}$, and $\dot{\theta}$ are roughly equal to the corresponding weighted averages, using weights w_j, of the biases associated with estimates computed from balanced panels. For example, when $T_1 = 6$ and $T_2 = 10$, the bias of $\dot{\theta}$ is roughly the weighted average of the bias when $T = 6$ and the bias when $T = 10$. The bottom line is that, at least for moderately unbalanced panels, data unbalancedness poses no problem for the SPJ.

5 User-written models

`xtspj` can also be used with a user-written model, provided that the density function is of the form

$$f(\mathbf{Y}_{it}|\mathbf{X}_{it};\boldsymbol{\theta},\alpha_i) = f(\mathbf{Y}_{it}|\mu_{1,it},\ldots,\mu_{M,it}) \tag{2}$$

with

$$\begin{aligned} \mu_{1,it} &= X'_{1,it}\theta_1 + \alpha_i \\ \mu_{m,it} &= X'_{m,it}\theta_m \qquad \text{for } m = 2,\ldots,M \end{aligned}$$

for an arbitrary number $M \geq 1$ of linear indices $\mu_{m,it}$ (equations in the Stata language) depending on covariates $X_{m,it}$. The regressand, $\mathbf{Y}_{it}$, can be a vector, at most of dimension M. The first linear index, $\mu_{1,it}$, contains an additive fixed effect. Nevertheless, this design is general in that when the fixed effects enter nonadditively, one simply needs to keep the first linear index free of covariates. One can also set $X_{m,it} = 1$ for one or several $m > 1$; then the corresponding $\mu_{m,it} = \theta_m$ are simply parameters, entering the model as specified by f.

5.1 Syntax

```
xtspj eq [eq ...] [if] [in], model(string) method(string) [level(#)
   ltol(#) ptol(#) maxiter(#) diagnosis verbose alpha(newvar)]
```

5.2 Description

The rules for `xtspj` with a user-written model are the same as in section 2 with two exceptions. First, each `eq` (`equation`) specifies a linear index, $\mu_{m,it}$. The specification of an `eq` is similar to `ml model` (see [R] **ml**), that is,

```
([eqname:] [depvar] = [indepvars][, noconstant] )
```

where `eqname` is an optional `equation` name, *depvar* and *indepvars* are specified as usual, and `noconstant` is optional and specifies that the corresponding equation has no constant term. Here the equal sign (`=`) is compulsory, even if the equation is completely empty in other parts of the specification; that is, `eq` must be, in the simplest form, `(=)`.

In the first `eq`, which is always without constant term because of the presence of fixed effects, `noconstant` is implicitly assumed without specifying it. When a factor variable appears among the *indepvars* in more than one `eq`, it must appear with a different name in each `eq`. Hence, one has to generate a duplicate of the variable underlying the factor variable for each additional `eq` where the factor variable appears. Second, `model(`*string*`)` contains the name of the user-written model, which has to be programmed according to the rules given below in section 5.3. The `ereturn` stored results are also as in section 2 except that `e(depvar)` now is the list of regressands.

5.3 User-written log-likelihood evaluation

The function f in (2) has to be specified as a Mata class containing two member functions, `::Evaluate()` and `::Check()` (see [M-2] **class**). As examples, the Mata classes for the preprogrammed models are given in appendix B, and the do-files are available in the online supplement. The template for the Mata class is as follows.

Template code block

```
 1  mata
 2  class xtspj<YourModel> extends xtspjModel {
 3      public:  void Evaluate(), Check()
 4  }
 5  void function xtspj<YourModel>::Evaluate(real matrix Y, real matrix XB,
 6      real colvector LogLikelihood, real matrix Gradient,
 7      pointer(real colvector) matrix Hessian) {
 8          // compute log likelihood, gradient, Hessian
 9  }
10  void function xtspj<YourModel>::Check(real matrix Data,
11      real scalar Keep) {
12          // check data
13  }
14  end
```

The user needs to supply the relevant Mata code in the lines `8` and `12` as discussed below and change "`<YourModel>`" (lines `2`, `5`, `10`) to a desired name, for example, "`MyModel`". When `xtspj` is executed, this name must be specified in the option `model(`*string*`)`, for example, `model(MyModel)`, so that `xtspj` calls the community-contributed specification of f.

void function ::Evaluate()

The function `::Evaluate()` needs to be supplied (see [M-2] **declarations**). It computes the model's log likelihood, gradient, and Hessian for a given set of observations (i, t), $t \in S$. The function is called repeatedly and separately for each i and various sets S. In balanced panels with even T, for example, S is $\{1, \ldots, T\}$ or $\{1, \ldots, T/2\}$ or $\{T/2 + 1, \ldots, T\}$. The arguments of `::Evaluate()` are as follows:

`real matrix Y`, the matrix of regressands. The kth column of `Y` corresponds to the kth regressand, defined by an `eq`, and each row corresponds to an observation (i,t), $t \in S$.

`real matrix XB`, the matrix of linear indices. The mth column of `XB` corresponds to the mth linear index, defined by an `eq`, and each row corresponds to an observation (i,t), $t \in S$.

`real colvector LogLikelihood`, the column vector of log-likelihood values. Each row contains the log-likelihood value corresponding to an observation (i,t), $t \in S$, evaluated at the corresponding row of `XB` and `Y`. Thus, `LogLikelihood` has elements

$$\log f(\mathbf{Y}_{it}|\mu_{1,it},\ldots,\mu_{M,it}), \qquad t \in S$$

`real matrix Gradient`, the matrix of scores. The mth column of `Gradient` corresponds to the log-likelihood derivative with respect to the mth linear index, evaluated at `XB` and `Y`. Thus, the mth column of `Gradient` has elements

$$\frac{\partial \log f(\mathbf{Y}_{it}|\mu_{1,it},\ldots,\mu_{M,it})}{\partial \mu_{m,it}}, \qquad t \in S$$

`pointer(real colvector) matrix Hessian`, the pointer matrix pointing to the Hessian (see [M-2] **pointers**). It must be symmetric (see [M-5] **makesymmetric( )**). The (m,n)th element of `Hessian` is a pointer to the column vector of second-order log-likelihood derivatives with respect to the mth and nth linear indices, evaluated at `XB` and `Y`. Thus, the (m,n)th element of `Hessian` points to the elements

$$\frac{\partial^2 \log f(\mathbf{Y}_{it}|\mu_{1,it},\ldots,\mu_{M,it})}{\partial \mu_{m,it}\partial \mu_{n,it}}, \qquad t \in S$$

The arguments `real matrix Gradient` and `pointer(real colvector) matrix Hessian` are optional. If they are not changed inside `::Evaluate()`, `xtspj` uses a numerical differentiation algorithm to compute the derivatives. See section 6.1 for details.

The following Mata code gives `::Evaluate()` for the probit model:

Code block 1

```
void function xtspjprobit::Evaluate(real matrix Y,
    real matrix XB, real colvector LogLikelihood,
     real matrix Gradient, pointer(real colvector) matrix Hessian) {
        q=(Y:*2:-1)
        LogLikelihood=lnnormal(q:*XB)
        Gradient=q:*(normalden(q:*XB):/normal(q:*XB))
        Hessian=&(-Gradient:*(XB+Gradient))
}
```

Compare code block 1 with the corresponding portion of the template code block: "`probit`" replaces "`<YourModel>`", and `LogLikelihood`, `Gradient`, and `Hessian` are computed in lines `5` to `7`. Lines `6` and `7` can be omitted; then `Gradient` and `Hessian` are computed by numerical differentiation.

Mata function arguments are passed by reference (see [M-2] **declarations**, remark 8), so the changes made inside the function-to-function arguments remain also after the function execution has terminated. Hence, the values computed and stored in `LogLikelihood`, `Gradient`, and `Hessian` are preserved after the function call and are passed to `xtspj`.

When a model contains more than one `equation` ($M > 1$), the variables `Gradient` and `Hessian` may be preallocated to reduce the computation time. This can be done by inserting the code

```
Gradient=J(rows(Data),cols(Data),.)
Hessian=J(cols(Data),cols(Data),NULL)
```

between lines `3` and `4` of code block 1 (see [M-5] **J()**, [M-5] **rows()**, and remark 8 of [M-2] **pointers** for `NULL`).

`Hessian` is computed as a matrix of pointers. In line `7` of code block 1, the operator `&()` creates a pointer to the value of the expression in parentheses, that is, a pointer to the value of `-Gradient:*(XB+Gradient)`. There is a subtle difference between a pointer to a variable and a pointer to the value of an expression (see [M-2] **pointers**, remarks 2 and 3). When pointing to a variable, this may cause mistakes when that variable is reused. For example, the code

```
1 | H=...
2 | Hessian[1,1]=&H
3 | H=...
4 | Hessian[1,2]=&H
```

will not produce the intended result, because line `3` alters `H` and thus also `Hessian[1,1]`. It is on the safe side to use different variable names instead, as in

```
1 | H11=...
2 | Hessian[1,1]=&H11
3 | H12=...
4 | Hessian[1,2]=&H12
```

void function ::Check()

Prior to estimation, `xtspj` checks that for each individual, there are at least two time periods of data so that, when the panel is split, each subpanel has at least one time period per individual. Individuals that do not meet this condition are excluded from the estimation. The function `::Check()` serves as an additional, model-specific data check and, if necessary, serves to additionally exclude individuals that are uninformative about $\boldsymbol{\theta}$ in some subpanel or subpanels. For example, in the linear model, at least two time periods are required in each subpanel. Another example is that, in binary choice models, $\mathbf{Y}_{it}$ must vary across t in each subpanel. If such data checks are not needed, one may simply replace line `12` of the template code block with "`Keep=1`".

The function `::Check()` is called prior to estimation and checks a given set of observations (i, t), $t \in S$. Similar to `::Evaluate()`, `::Check()` is called separately for each i and various sets S that correspond to the subpanels. Only if i passes the data check imposed by `::Check()` for all relevant sets S (which results in "`Keep=1`" for all S) is i included in the dataset to be used for estimation. The arguments of `::Check()` are as follows:

`real matrix Data`, the data to be checked. The columns of `Data` correspond to the regressands of the model, followed by the regressors, in the same order as specified by the equations. The rows correspond to the observations (i, t), $t \in S$.

`real scalar Keep`, the main output. `Keep` is set to `1` if `Data` passes the check and set to `0` otherwise.

The following Mata code gives `::Check()` for the probit model:

```
1 | void function xtspjprobit::Check(real matrix Data, real scalar Keep) {
2 |     if (sum(Data[.,1])==0 || sum(Data[.,1])==rows(Data)) {
3 |         Keep=0
4 |     } else {
5 |         Keep=1
6 |     }
7 | }
```

Here line `2` tests if `Y`, the first column of `Data`, varies. When the elements of `Y` are all `0` or all `1`, the expression

```
sum(Data[.,1])==0 || sum(Data[.,1])==rows(Data)
```

is evaluated as `true` and `Keep` is set to `0` (see [M-5] **sum()** and [M-2] **op_logical**). Otherwise, `Y` varies and `Keep` is set to `1`.

5.4 Example

Here we compute the jackknifed ML estimate with the user-written model `PROBIT`, specified in the Mata part, for the same setup as in section 2.6. Note that the model name, `PROBIT`, is capitalized so that it does not conflict with the preprogrammed model named `probit`. The output table is identical to the corresponding output table in section 2.6 except for the model name, `PROBIT`.

```
. // data setup omitted; see section 2.6
. mata
------------------------------------------------- mata (type end to exit) ------
: class xtspjPROBIT extends xtspjModel {
>         public: void Evaluate(), Check()
> }
: void function xtspjPROBIT::Evaluate(real matrix Y, real matrix XB,
>         real colvector LogLikelihood, real matrix Gradient,
>         pointer(real colvector) matrix Hessian) {
>                 q=(Y:*2:-1)
>                 LogLikelihood=lnnormal(q:*XB)
>                 Gradient=q:*(normalden(q:*XB):/normal(q:*XB))
>                 Hessian=&(-Gradient:*(XB+Gradient))
> }
: void function xtspjPROBIT::Check(real matrix Data, real scalar Keep) {
>         if (sum(Data[.,1])==0 || sum(Data[.,1])==rows(Data)
>                 || rows(Data)<=1) {Keep=0;}
>         else {Keep=1;}
> }
: end
--------------------------------------------------------------------------------

. xtspj (complain = age i.gender), model(PROBIT) method(parm)
xtspj - PROBIT - Jackknifed ML                  Number of obs     =        524

------------------------------------------------------------------------------
    complain |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |  -.0483349   .0272095    -1.78   0.076    -.1016645    .0049947
             |
      gender |
     female  |          0  (empty)
       male  |   .5138605   .1266843     4.06   0.000     .2655638    .7621572
------------------------------------------------------------------------------
```

6 Algorithms

6.1 Details on numerical differentiation

Numerical differentiation is a well-established technique for the approximation of the derivatives of a function when analytical differentiation is difficult or infeasible. The subject is treated extensively in, for example, Burden, Faires, and Burden (2016), Griewank and Walther (2008), and Press et al. (2007, chap. 5.7). In Mata, [M-5] **deriv( )** computes numerical derivatives. However, `xtspj` does not use [M-5] **deriv( )**, because it is not optimized for the present setup. To see why, we briefly review the algorithm implemented by [M-5] **deriv( )** and the implied time cost. Consider a generic log-likelihood function

$$L(\theta, \boldsymbol{\alpha}) = L(\theta, \alpha_1, \ldots, \alpha_N) = \sum_{i,t} \log f(\mathbf{Y}_{it} | \mathbf{X}_{it}; \theta, \alpha_i)$$

where θ is a scalar common parameter and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)'$ is the vector of fixed-effects parameters. [M-5] **deriv( )** approximates the first derivative of $L(\theta, \alpha_1, \ldots, \alpha_N)$,

$$\left(\frac{\partial L(\theta, \boldsymbol{\alpha})}{\partial \theta}, \frac{\partial L(\theta, \boldsymbol{\alpha})}{\partial \alpha'}\right) = \left(\frac{\partial L(\theta, \boldsymbol{\alpha})}{\partial \theta}, \frac{\partial L(\theta, \boldsymbol{\alpha})}{\partial \alpha_1}, \ldots, \frac{\partial L(\theta, \boldsymbol{\alpha})}{\partial \alpha_N}\right) \tag{3}$$

as

$$\begin{aligned}
\frac{\partial L(\theta, \boldsymbol{\alpha})}{\partial \theta} &= \frac{L(\theta + h, \boldsymbol{\alpha}) - L(\theta - h, \boldsymbol{\alpha})}{2h} \\
\frac{\partial L(\theta, \boldsymbol{\alpha})}{\partial \alpha_i} &= \frac{L(\theta, \alpha_1, \ldots, \alpha_i + h, \ldots, \alpha_N) - L(\theta, \alpha_1, \ldots, \alpha_i - h, \ldots, \alpha_N)}{2h}, \; i = 1, \ldots, N
\end{aligned}$$

for some small h. This is the standard formula discussed in many textbooks, including those cited above. (Higher-order techniques also exist; see, for example, Fornberg [1988].) Now, assuming the time cost of evaluating $L(\theta, \boldsymbol{\alpha})$ is $O(1)$, the time cost of approximating (3) as above is $O(2N + 2)$ or, when $\boldsymbol{\theta}$ is a vector of P elements, $O(2N + 2P)$. That is, it takes $2(N + P)$ times as much time to compute the first derivative as to compute the value of the log-likelihood function. When the second derivative is approximated in the same way, the time cost becomes even much higher. In typical microeconomic panel datasets, N is large and T is small, so this algorithm is computationally very slow.

`xtspj` implements an alternative algorithm for numerical differentiation, exploiting the property that the model may be expressed in terms of linear indices. Given (2), the log-likelihood function may be written as

$$L(\mu_1, \ldots, \mu_M) = \sum_{i,t} \log f(\mathbf{Y}_{it} | \mu_{1,it}, \ldots, \mu_{M,it})$$

where $\boldsymbol{\mu}_m$ is the $NT \times 1$ vector collecting all the elements $\mu_{m,it}$. Correspondingly, let $\mathbf{X}_m$ be the $NT \times P_m$ matrix collecting all the values of the regressors that appear in the mth linear index. Observing that

$$\frac{\partial \boldsymbol{\mu}_m}{\partial \boldsymbol{\theta}'_m} = \mathbf{X}_m$$

for every m, we have

$$\frac{\partial L(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_M)}{\partial \boldsymbol{\theta}'_m} = \frac{\partial L(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_M)}{\partial \boldsymbol{\mu}'_m} \frac{\partial \boldsymbol{\mu}_m}{\partial \boldsymbol{\theta}'_m} = \frac{\partial L(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_M)}{\partial \boldsymbol{\mu}'_m} \mathbf{X}_m$$

Here we only have to approximate $\partial L(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_M)/\partial \boldsymbol{\mu}_m$ as

$$\frac{\partial L(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_M)}{\partial \boldsymbol{\mu}_m} = \frac{L(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m + h, \ldots, \boldsymbol{\mu}_M) - L(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m - h, \ldots, \boldsymbol{\mu}_M)}{2h}$$

which requires just two evaluations of the log-likelihood function. For the fixed-effects parameter α_i, we have

$$\frac{\partial L(\boldsymbol{\theta}, \boldsymbol{\alpha})}{\partial \alpha_i} = \sum_t \frac{\partial \log f(\mathbf{Y}_{it} | \mu_{1,it}, \ldots, \mu_{M,it})}{\partial \mu_{1,it}}$$

which is linked to objects already computed because

$$\frac{\partial L\left(\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_M\right)}{\partial \boldsymbol{\mu}_1}=\sum_{i,t}\frac{\partial \log f\left(\mathbf{Y}_{it}|\mu_{1,it},\ldots,\mu_{M,it}\right)}{\partial \mu_{1,it}}$$

For the second derivative, a similar computation follows easily.

The algorithm just described is efficient in that the time cost is only slightly higher than $O(2M)$, so the time spent obtaining the first derivative is roughly the same as that of evaluating the log-likelihood function $2M$ times. In typical settings, M is much less than $N+P$. For example, in the probit model, $M=1$. In the linear model, $M=2$ (there is a constant linear index for the error variance, say, $\mu_{2,it}=\theta_2=\sigma^2$). In addition, for some models, the time cost of evaluating the derivatives analytically exceeds the time cost of computing them numerically using the proposed algorithm.

`xtspj` currently sets h equal to `epsilon(1)^0.25`, where `epsilon(1)` is the machine precision (see [M-5] **epsilon( )**).

6.2 Details on the Newton–Raphson algorithm

The Newton–Raphson algorithm is a common technique for numerical optimization of functions that are twice differentiable. Its properties, potential problems, and implementation details can be found in, for example, Süli and Mayers (2003, chap. 4), Bonnans et al. (2006, chap. 4), and Press et al. (2007, chap. 9). In Stata and Mata, `ml`, `optimize()`, and `moptimize()` (see [R] **ml**, [M-5] **optimize( )**, and [M-5] **moptimize( )**) are numerical optimizers. In the context of ML estimation of fixed-effects models, the problem with these optimizers, including the standard way of implementing the Newton–Raphson algorithm, is that they do not exploit the sparsity of the Hessian matrix, which makes them slow and memory inefficient. This section reviews these problems and discusses how `xtspj` implements the Newton–Raphson algorithm more efficiently.

Consider first how one may fit a fixed-effects model by ML in Stata when no `xt` command is available. As an example, suppose one wishes to fit a fixed-effects probit model, given a panel dataset with a binary regressand `Y`, a regressor `X`, and a group variable `i` indexing the individuals (`i` is the *panelvar* in `xtset`). Then, one would run

```
quietly tabulate i, generate(group)
probit Y X group*, nocons
```

This code is conveniently short but can be slow. As an example, we ran it with Stata/MP2 15.1 on an Intel i7-6820HQ computer for balanced datasets generated with $X_{it},\alpha_i\sim\mathcal{N}(0,1)$, $T=10$, $\theta_0=0.5$, and various values of N. As shown in table 1, the time cost of `probit` grows rapidly with N, while that of `xtspj` is roughly proportional to N.

Table 1. Time cost of `probit` versus `xtspj`

	Time using `probit`		Time using `xtspj`	
N	Mean	Max	Mean	Max
10	0.03	0.05	0.04	0.06
100	0.15	0.17	0.06	0.08
500	4.26	4.51	0.16	0.18
1000	32.95	33.83	0.29	0.32
2000	277.03	281.55	0.51	0.54

NOTES: Time is reported in seconds. Mean and max are obtained by running `probit` and `xtspj` 10 times on the same dataset, generated with X_{it} and α_i drawn from $\mathcal{N}(0,1)$, $T=10$, and $\theta_0 = 0.5$.

The number of parameters in fixed-effects models is $N + \dim \boldsymbol{\theta}$, so when N is large, standard optimization methods become very slow. The SPJ adds to the challenge because the model has to be fit several times or, in the case of the jackknifed log likelihood, one has to optimize simultaneously over several sets of fixed-effects parameters. The number of fixed-effects parameters that implicitly enter $\dot{l}(\boldsymbol{\theta})$ varies between $3N$ (when T_j is even for all i) and $5N$ (when T_j is odd for all i). In short, when N is large, the optimization problem is high dimensional. Standard optimizers that use second derivatives, such as the Newton–Raphson algorithm, become slow in high dimensions because the Hessian is high dimensional. Here, however, the Hessian is also sparse. In particular, the matrix of second derivatives of the original (unconcentrated) log likelihood with respect to all fixed-effects parameters is a diagonal matrix. This is also true, as shown in appendix C, for the Hessian of the unconcentrated objective function underlying $\dot{l}(\boldsymbol{\theta})$, which depends on multiple sets of fixed-effects parameters. The sparsity of the Hessian simplifies its inversion and the updating step of the Newton–Raphson algorithm, as detailed below.

Let $\boldsymbol{\theta}$ be a $P \times 1$ column vector, and consider an objective function of the form

$$\dot{l}\left(\boldsymbol{\theta}, \boldsymbol{\phi}\right) = \sum_j w_j \dot{l}_j \left(\boldsymbol{\theta}, \boldsymbol{\alpha}^j, \boldsymbol{\gamma}^j, \boldsymbol{\delta}^j\right)$$

$$\boldsymbol{\phi} = \left(\alpha_1^1, \ldots, \alpha_{N_1}^1, \gamma_1^1, \ldots, \gamma_{N_1}^1, \delta_1^1, \ldots, \delta_{N_1}^1, \ldots, \alpha_1^J, \ldots, \alpha_{N_J}^J, \gamma_1^J, \ldots, \gamma_{N_J}^J, \delta_1^J, \ldots, \delta_{N_J}^J\right)'$$

where w_j is the weight associated with the jth balanced panel and $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\delta}$ are vectors of fixed-effects parameters associated with, respectively, all time periods, the first half, and the second half of the time periods (assuming T_j is even for all j). Partition the gradient and the Hessian as

$$g\left(\boldsymbol{\theta}, \boldsymbol{\phi}\right) = \begin{pmatrix} g_\theta\left(\boldsymbol{\theta}, \boldsymbol{\phi}\right) \\ g_\phi\left(\boldsymbol{\theta}, \boldsymbol{\phi}\right) \end{pmatrix}, \qquad H\left(\boldsymbol{\theta}, \boldsymbol{\phi}\right) = \begin{pmatrix} H_{\theta\theta}\left(\boldsymbol{\theta}, \boldsymbol{\phi}\right) & H_{\theta\boldsymbol{\phi}}\left(\boldsymbol{\theta}, \boldsymbol{\phi}\right) \\ H_{\phi\theta}\left(\boldsymbol{\theta}, \boldsymbol{\phi}\right) & H_{\phi\phi}\left(\boldsymbol{\theta}, \boldsymbol{\phi}\right) \end{pmatrix}$$

Given the parameter vector $(\boldsymbol{\theta}^{(k)\prime}, \boldsymbol{\phi}^{(k)\prime})'$ at the kth iteration, the Newton–Raphson algorithm obtains the updated parameter vector as

$$\begin{pmatrix} \boldsymbol{\theta}^{(k+1)} \\ \boldsymbol{\phi}^{(k+1)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\theta}^{(k)} \\ \boldsymbol{\phi}^{(k)} \end{pmatrix} + u\left(\boldsymbol{\theta}^{(k)}, \boldsymbol{\phi}^{(k)}\right)$$

where

$$u(\boldsymbol{\theta}, \boldsymbol{\phi}) = -H^{-1}(\boldsymbol{\theta}, \boldsymbol{\phi})\, g(\boldsymbol{\theta}, \boldsymbol{\phi})$$

When $\dim \boldsymbol{\phi}$ is large, computing $u(\boldsymbol{\theta}, \boldsymbol{\phi})$ by Gaussian elimination is slow and inaccurate. As suggested by Hall (1978) and Chamberlain (1980), the computation of $u(\boldsymbol{\theta}, \boldsymbol{\phi})$ can be carried out more efficiently using a block inversion algorithm. Because $H_{\phi\phi}(\boldsymbol{\theta}, \boldsymbol{\phi})$ is diagonal, only its diagonal needs to be stored, and it is straightforward and fast to compute

$$\begin{aligned} h^{-1}_{\phi\phi}(\boldsymbol{\theta}, \boldsymbol{\phi}) &= \operatorname{diag}\left\{H^{-1}_{\phi\phi}(\boldsymbol{\theta}, \boldsymbol{\phi})\right\} \\ A(\boldsymbol{\theta}, \boldsymbol{\phi}) &= H_{\phi\theta}(\boldsymbol{\theta}, \boldsymbol{\phi}) \odot \left\{h^{-1}_{\phi\phi}(\boldsymbol{\theta}, \boldsymbol{\phi})\,\mathbf{1}'_P\right\} \\ B(\boldsymbol{\theta}, \boldsymbol{\phi}) &= H_{\theta\theta}(\boldsymbol{\theta}, \boldsymbol{\phi}) - H_{\theta\phi}(\boldsymbol{\theta}, \boldsymbol{\phi})\, A(\boldsymbol{\theta}, \boldsymbol{\phi}) \end{aligned}$$

where $\odot$ denotes the Hadamard product, $\mathbf{1}_P$ is the $P \times 1$ vector of ones, and $\operatorname{diag}(\mathbf{V})$ is the column vector formed by the diagonal elements of $\mathbf{V}$. Then

$$u(\boldsymbol{\theta}, \boldsymbol{\phi}) = \begin{pmatrix} u_\theta(\boldsymbol{\theta}, \boldsymbol{\phi}) \\ u_\phi(\boldsymbol{\theta}, \boldsymbol{\phi}) \end{pmatrix}$$

is given by

$$\begin{aligned} u_\theta(\boldsymbol{\theta}, \boldsymbol{\phi}) &= -B^{-1}(\boldsymbol{\theta}, \boldsymbol{\phi})\left\{g_\theta(\boldsymbol{\theta}, \boldsymbol{\phi}) - A'(\boldsymbol{\theta}, \boldsymbol{\phi})\, g_\phi(\boldsymbol{\theta}, \boldsymbol{\phi})\right\} \\ u_\phi(\boldsymbol{\theta}, \boldsymbol{\phi}) &= -g_\phi(\boldsymbol{\theta}, \boldsymbol{\phi}) \odot h^{-1}_{\phi\phi}(\boldsymbol{\theta}, \boldsymbol{\phi}) - A(\boldsymbol{\theta}, \boldsymbol{\phi})\, u_\theta(\boldsymbol{\theta}, \boldsymbol{\phi}) \end{aligned}$$

This way of computing the updated parameter vector is memory efficient and dramatically speeds up the Newton–Raphson algorithm.

7 Conclusion

`xtspj` implements the SPJ of Dhaene and Jochmans (2015) in Stata using code written in Mata. The command allows for unbalanced panels. It implements two variants of the SPJ: the jackknifed ML estimator and the jackknifed (concentrated) log likelihood. The model is generically defined through linear indices. The log-likelihood gradient and Hessian may be supplied by the user or computed numerically by `xtspj`. The command is much faster than standard Stata code because it is written in Mata and the maximization of the log likelihood is carried out by a tailored Newton–Raphson algorithm that exploits the sparsity of the Hessian.

The current version of `xtspj` is preliminary in several respects. For instance, unlike other Stata routines such as `optimize()`, `moptimize()`, and `ml`, `xtspj` does not support the Nelder–Mead algorithm (see, for example, Dennis and Woods [1987]) and does not allow constraints in the maximization. Further, `xtspj` provides a bias correction for fixed effects but not for time dummies (although the model may include time effects, treated as common parameters). Recently, Fernández-Val and Weidner (2016) extended the SPJ to models with both fixed and time effects, in an asymptotic setting where N/T converges to a positive constant. Also, `xtspj` is restricted to first-order bias correction, whereas higher-order versions of the SPJ exist as well (see Dhaene and Jochmans [2015]). Finally, `xtspj` focuses on estimating $\boldsymbol{\theta}$—the common parameter. There are, however, many other estimands of interest, for example, average marginal (or partial) effects or other average quantities that depend on $\boldsymbol{\theta}$, the fixed effects, and the covariate values. The ML estimates of such quantities are also biased because of the estimation of the fixed effects and may be bias corrected by the SPJ.

In applied work with panel data, most panel datasets are unbalanced. On the other hand, most theoretical work on panel data assumes balanced data. Here a version of the SPJ for unbalanced data is proposed, and its properties are investigated. We find that mild degrees of data unbalancedness pose no problem for the SPJ. The properties of the estimators under more extreme forms of data unbalancedness, for example, when the vast majority of individuals have only a few data periods remain to be investigated.

8 Acknowledgments

We are grateful to a referee for valuable comments and suggestions. Financial support from the Flemish Science Foundation grant G.0505.11 and from the Fundamental Research Funds for the Central Universities (grant 2412018QD026) are gratefully acknowledged.

9 References

Andersen, E. B. 1970. Asymptotic properties of conditional maximum-likelihood estimators. *Journal of the Royal Statistical Society, Series B* 32: 283–301.

Arellano, M., and S. Bonhomme. 2009. Robust priors in nonlinear panel data models. *Econometrica* 77: 489–536.

Arellano, M., and J. Hahn. 2016. A likelihood-based approximate solution to the incidental parameter problem in dynamic nonlinear models with multiple effects. *Global Economic Review* 45: 251–274.

Baldick, R. 2006. *Applied Optimization: Formulation and Algorithms for Engineering Systems*. Cambridge: Cambridge University Press.

Bonnans, J.-F., J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal. 2006. *Numerical Optimization: Theoretical and Practical Aspects*. 2nd ed. New York: Springer.

Burden, R. L., D. J. Faires, and A. M. Burden. 2016. *Numerical Analysis*. 10th ed. Boston: Cengage Learning.

Chamberlain, G. 1980. Analysis of covariance with qualitative data. *Review of Economic Studies* 47: 225–238.

———. 2010. Binary response models for panel data: Identification and information. *Econometrica* 78: 159–168.

Chudik, A., M. H. Pesaran, and J.-C. Yang. 2018. Half-panel jackknife fixed-effects estimation of linear panels with weakly exogenous regressors. *Journal of Applied Econometrics* 33: 816–836.

Cox, D. R., and N. Reid. 1987. Parameter orthogonality and approximate conditional inference. *Journal of the Royal Statistical Society, Series B* 49: 1–39.

Cruz-Gonzalez, M., I. Fernández-Val, and M. Weidner. 2017. Bias corrections for probit and logit models with two-way fixed effects. *Stata Journal* 17: 517–545.

Dennis, J. E., Jr., and D. J. Woods. 1987. Optimization on microcomputers: The Nelder-Mead simplex algorithm. In *New Computing Environments: Microcomputers in Large-Scale Computing*, ed. A. Wouk, 116–122. Philadelphia: Society for Industrial and Applied Mathematics.

Dhaene, G., and K. Jochmans. 2015. Split-panel jackknife estimation of fixed-effect models. *Review of Economic Studies* 82: 991–1030.

Egger, P. H., and K. E. Staub. 2016. GLM estimation of trade gravity models with fixed effects. *Empirical Economics* 50: 137–175.

Fernández-Val, I. 2009. Fixed effects estimation of structural parameters and marginal effects in panel probit models. *Journal of Econometrics* 150: 71–85.

Fernández-Val, I., and M. Weidner. 2016. Individual and time effects in nonlinear panel models with large N, T. *Journal of Econometrics* 192: 291–312.

Fornberg, B. 1988. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of Computation* 51: 699–706.

Greene, W. 2004. Fixed effects and bias due to the incidental parameters problem in the tobit model. *Econometric Reviews* 23: 125–147.

Griewank, A., and A. Walther. 2008. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics.

Griffin, J. M., and G. Maturana. 2016. Who facilitated misreporting in securitized loans? *Review of Financial Studies* 29: 384–419.

Hahn, J., and G. Kuersteiner. 2002. Asymptotically unbiased inference for a dynamic panel model with fixed effects when both n and T are large. *Econometrica* 70: 1639–1657.

———. 2011. Bias reduction for dynamic nonlinear panel models with fixed effects. *Econometric Theory* 27: 1152–1191.

Hahn, J., and W. Newey. 2004. Jackknife and analytical bias reduction for nonlinear panel models. *Econometrica* 72: 1295–1319.

Hall, B. H. 1978. A general framework for the time series-cross section estimation. *Annales de l'Inséé* 30/31: 177–202.

Honoré, B. E., and E. Tamer. 2006. Bounds on parameters in panel dynamic discrete choice models. *Econometrica* 74: 611–629.

Katz, E. 2001. Bias in conditional and unconditional fixed effects logit estimation. *Political Analysis* 9: 379–384.

Lancaster, T. 2000. The incidental parameter problem since 1948. *Journal of Econometrics* 95: 391–413.

———. 2002. Orthogonal parameters and panel data. *Review of Economic Studies* 69: 647–666.

Li, H., B. G. Lindsay, and R. P. Waterman. 2003. Efficiency of projected score methods in rectangular array asymptotics. *Journal of the Royal Statistical Society, Series B* 65: 191–208.

Milner, A., L. Krnjacki, P. Butterworth, and A. D. LaMontagne. 2016. The role of social support in protecting mental health when employed and unemployed: A longitudinal fixed-effects analysis using 12 annual waves of the HILDA cohort. *Social Science and Medicine* 153: 20–26.

Neyman, J., and E. Scott. 1948. Consistent estimates based on partially consistent observations. *Econometrica* 16: 1–32.

Nickell, S. 1981. Biases in dynamic models with fixed effects. *Econometrica* 49: 1417–1426.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing*. 3rd ed. Cambridge: Cambridge University Press.

Quenouille, M. H. 1949. Approximate tests of correlation in time-series 3. In *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 45, 483–484. Cambridge: Cambridge University Press.

———. 1956. Notes on bias in estimation. *Biometrika* 43: 353–360.

Süli, E., and D. F. Mayers. 2003. *An Introduction to Numerical Analysis*. Cambridge: Cambridge University Press.

Wincek, M. A., and G. C. Reinsel. 1986. An exact maximum likelihood estimation procedure for regression-ARMA time series models with possibly nonconsecutive data. *Journal of the Royal Statistical Society, Series B* 48: 303–313.

Woutersen, T. 2001. Robustness against incidental parameters and mixing distributions. University of Western Ontario, Department of Economics, Working Paper No. 2001-10. https://ir.lib.uwo.ca/cgi/viewcontent.cgi?article=1382&context=economicsresrpt.

About the authors

Yutao Sun is a postdoctoral researcher at Erasmus University Rotterdam, holding an affiliation also in Northeast Normal University. His research interests include panel-data econometrics and maximum likelihood methods.

Geert Dhaene is a professor at KU Leuven. His research interests focus on theoretical and applied econometrics, with a focus on panel-data methods.

A Jackknifed log-likelihood estimates for the linear model

The model is

$$y_{it} = \alpha_i + \mathbf{X}_{it}'\boldsymbol{\beta} + \varepsilon_{it}, \qquad \varepsilon_{it} \sim \mathcal{N}\left(0, \sigma^2\right)$$

Suppose, initially, that the panel is balanced. Define the demeaned variables

$$\widetilde{y}_{it} = y_{it} - \frac{1}{T}\sum_t y_{it}, \qquad \widetilde{\mathbf{X}}_{it} = \mathbf{X}_{it} - \frac{1}{T}\sum_t \mathbf{X}_{it}$$

The ML estimate is

$$\widehat{\boldsymbol{\beta}} = \mathbf{A}^{-1}\mathbf{b}, \qquad \widehat{\sigma}^2 = \text{MSR}\left(\widehat{\boldsymbol{\beta}}\right)$$

where

$$\mathbf{A} = \frac{1}{NT}\sum_{i,t}\widetilde{\mathbf{X}}_{it}\widetilde{\mathbf{X}}_{it}', \qquad \mathbf{b} = \frac{1}{NT}\sum_{i,t}\widetilde{\mathbf{X}}_{it}\widetilde{y}_{it}, \qquad \text{MSR}(\boldsymbol{\beta}) = \frac{1}{NT}\sum_{i,t}\left(\widetilde{y}_{it} - \widetilde{\mathbf{X}}_{it}'\boldsymbol{\beta}\right)^2$$

With the mean squared residual (MSR) written as

$$\text{MSR}(\boldsymbol{\beta}) = \widehat{\sigma}^2 + \left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right)'\mathbf{A}\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right)$$

the concentrated log likelihood (normalized by number of observations) is

$$l\left(\boldsymbol{\beta}, \sigma^2\right) = -\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma^2 - \frac{\widehat{\sigma}^2 + \left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right)'\mathbf{A}\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right)}{2\sigma^2}$$

Suppose, initially, that T is even. For the subpanels 1 and 2, define, respectively,

$$\begin{array}{lllll} \widetilde{y}_{it1}, \widetilde{\mathbf{X}}_{it1}, & \mathbf{A}_1, \mathbf{b}_1, & SSR_1(\boldsymbol{\beta}), & \widehat{\boldsymbol{\beta}}_1, \widehat{\sigma}_1^2, & l_1\left(\boldsymbol{\beta}, \sigma^2\right) \\ \widetilde{y}_{it2}, \widetilde{\mathbf{X}}_{it2}, & \mathbf{A}_2, \mathbf{b}_2, & SSR_2(\boldsymbol{\beta}), & \widehat{\boldsymbol{\beta}}_2, \widehat{\sigma}_2^2, & l_2\left(\boldsymbol{\beta}, \sigma^2\right) \end{array}$$

by analogy to the earlier definitions. The jackknifed concentrated log likelihood is

$$\dot{l}\left(\boldsymbol{\beta},\sigma^2\right) = -\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma^2 - \frac{\widehat{\sigma}^2 + \left(\boldsymbol{\beta}-\widehat{\boldsymbol{\beta}}\right)'\mathbf{A}\left(\boldsymbol{\beta}-\widehat{\boldsymbol{\beta}}\right)}{\sigma^2} + \sum_{r=1,2}\frac{\widehat{\sigma}_r^2 + \left(\boldsymbol{\beta}-\widehat{\boldsymbol{\beta}}_r\right)'\mathbf{A}_r\left(\boldsymbol{\beta}-\widehat{\boldsymbol{\beta}}_r\right)}{4\sigma^2}$$

Hence, $\dot{\boldsymbol{\beta}}$ follows on minimizing the quadratic form

$$2\left(\boldsymbol{\beta}-\widehat{\boldsymbol{\beta}}\right)'\mathbf{A}\left(\boldsymbol{\beta}-\widehat{\boldsymbol{\beta}}\right) - \frac{1}{2}\sum_{r=1,2}\left(\boldsymbol{\beta}-\widehat{\boldsymbol{\beta}}_r\right)'\mathbf{A}_r\left(\boldsymbol{\beta}-\widehat{\boldsymbol{\beta}}_r\right)$$

The solution is

$$\dot{\boldsymbol{\beta}} = \left\{2\mathbf{A} - (\mathbf{A}_1+\mathbf{A}_2)/2\right\}^{-1}\left\{2\mathbf{A}\widehat{\boldsymbol{\beta}} - \left(\mathbf{A}_1\widehat{\boldsymbol{\beta}}_1 + \mathbf{A}_2\widehat{\boldsymbol{\beta}}_2\right)/2\right\}$$

$$\dot{\sigma}^2 = 2\mathrm{MSR}\left(\dot{\boldsymbol{\beta}}\right) - \frac{\mathrm{MSR}_1\left(\dot{\boldsymbol{\beta}}\right) + \mathrm{MSR}_2\left(\dot{\boldsymbol{\beta}}\right)}{2}$$

When T is possibly odd,

$$\dot{l}\left(\boldsymbol{\beta},\sigma^2\right) = -\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma^2 - \frac{\widehat{\sigma}^2 + \left(\boldsymbol{\beta}-\widehat{\boldsymbol{\beta}}\right)'\mathbf{A}\left(\boldsymbol{\beta}-\widehat{\boldsymbol{\beta}}\right)}{\sigma^2} + \sum_{r,s=1,2}\omega_{rs}\frac{\widehat{\sigma}_{rs}^2 + \left(\boldsymbol{\beta}-\widehat{\boldsymbol{\beta}}_{rs}\right)'\mathbf{A}_{rs}\left(\boldsymbol{\beta}-\widehat{\boldsymbol{\beta}}_{rs}\right)}{4\sigma^2}$$

with $\mathbf{A}_{rs}, \widehat{\boldsymbol{\beta}}_{rs}, \widehat{\sigma}_{rs}^2$ corresponding to subpanel S_{rs} as defined in section 3.2, and

$$\omega_{rs} = \begin{cases} \frac{\lceil T/2\rceil}{T}, & \text{if } r = s \\ 1 - \frac{\lceil T/2\rceil}{T}, & \text{if } r \neq s \end{cases}$$

hence,

$$\dot{\boldsymbol{\beta}} = \left(2\mathbf{A} - \tfrac{1}{2}\textstyle\sum_{r,s}\omega_{rs}\mathbf{A}_{rs}\right)^{-1}\left(2\mathbf{A}\widehat{\boldsymbol{\beta}} - \tfrac{1}{2}\textstyle\sum_{r,s}\omega_{rs}\mathbf{A}_{rs}\widehat{\boldsymbol{\beta}}_{rs}\right)$$

$$\dot{\sigma}^2 = 2\mathrm{MSR}\left(\dot{\boldsymbol{\beta}}\right) - \tfrac{1}{2}\textstyle\sum_{r,s}\omega_{rs}\mathrm{MSR}_{rs}\left(\dot{\boldsymbol{\beta}}\right)$$

The generalization to the unbalanced case is as follows. When all T_j are even,

$$\dot{l}\left(\boldsymbol{\beta},\sigma^2\right) = \sum_j w_j \dot{l}^j\left(\boldsymbol{\beta},\sigma^2\right)$$

and

$$\dot{\boldsymbol{\beta}} = \left[\sum_j w_j \left\{2\mathbf{A}^j - \left(\mathbf{A}_1^j + \mathbf{A}_2^j\right)/2\right\}\right]^{-1} \sum_j w_j \left\{2\mathbf{A}^j \widehat{\boldsymbol{\beta}}^j - \left(\mathbf{A}_1^j \widehat{\boldsymbol{\beta}}_1^j + \mathbf{A}_2^j \widehat{\boldsymbol{\beta}}_2^j\right)/2\right\}$$

$$\dot{\sigma}^2 = \sum_j w_j \left(2\mathrm{MSR}^j\left(\dot{\boldsymbol{\beta}}\right) - \frac{\mathrm{MSR}_1^j\left(\dot{\boldsymbol{\beta}}\right) + \mathrm{MSR}_2^j\left(\dot{\boldsymbol{\beta}}\right)}{2}\right)$$

where the superscript j indicates the jth balanced subpanel; when some T_j are possibly odd,

$$\dot{\boldsymbol{\beta}} = \left\{\sum_j w_j \left(2\mathbf{A}^j - \tfrac{1}{2}\textstyle\sum_{r,s}\omega_{rs}^j \mathbf{A}_{rs}^j\right)\right\}^{-1} \sum_j w_j \left(2\mathbf{A}^j \widehat{\boldsymbol{\beta}}^j - \tfrac{1}{2}\textstyle\sum_{r,s}\omega_{rs}^j \mathbf{A}_{rs}^j \widehat{\boldsymbol{\beta}}_{rs}^j\right)$$

$$\dot{\sigma}^2 = \sum_j w_j \left\{2\mathrm{MSR}^j\left(\dot{\boldsymbol{\beta}}\right) - \tfrac{1}{2}\textstyle\sum_{r,s}\omega_{rs}^j \mathrm{MSR}_{rs}^j\left(\dot{\boldsymbol{\beta}}\right)\right\}$$

B Mata classes

Linear model

```
mata
class xtspjlinear extends xtspjModel {
    public: void Evaluate(), Check()
}
void function xtspjlinear::Evaluate(real matrix Y, real matrix XB,
    real colvector LogLikelihood, real matrix Gradient,
    pointer(real colvector) matrix Hessian) {
        YXB=(Y:-XB[.,1])
        S=exp(XB[.,2]) // variance
        LogLikelihood=lnnormalden(YXB,sqrt(S))
        Gradient=J(rows(Y),2,0)
        Hessian=J(2,2,NULL)
        Gradient[.,1]=YXB:/S
        Gradient[.,2]=(YXB:^2:/(2:*S):-1/2)
        Hessian[1,1]=&( -1:/S )
        Hessian[2,2]=&( (-YXB:^2:/S:^2:-1:/(2:*S)):+(-YXB:^2:/S:^1:-1/2) )
        Hessian[1,2]=&( (-YXB:/S) )
        Hessian[2,1]=Hessian[1,2]
}
void function xtspjlinear::Check(real matrix Data, real scalar Keep) {
    if (rows(Data)<=1) {Keep=0;}
    else {Keep=1;}
}
end
```

Probit model

```
mata
class xtspjprobit extends xtspjModel {
    public: void Evaluate(), Check()
}
void function xtspjprobit::Evaluate(real matrix Y, real matrix XB,
    real colvector LogLikelihood, real matrix Gradient,
    pointer(real colvector) matrix Hessian) {
        q=(Y:*2:-1)
        LogLikelihood=lnnormal(q:*XB)
        Gradient=q:*(normalden(q:*XB):/normal(q:*XB))
        Hessian=&(-Gradient:*(XB+Gradient))
}
void function xtspjprobit::Check(real matrix Data, real scalar Keep) {
    if (sum(Data[.,1])==0 || sum(Data[.,1])==rows(Data) || rows(Data)<=1) {Keep=0;}
    else {Keep=1;}
}
end
```

Logit model

```
mata
class xtspjlogit extends xtspjModel {
    public: void Evaluate(), Check()
}
void function xtspjlogit::Evaluate(real matrix Y, real matrix XB,
    real colvector LogLikelihood, real matrix Gradient,
    pointer(real colvector) matrix Hessian) {
        LogLikelihood=ln( (exp(-XB)):^(1:-Y) :/ (exp(-XB):+1) )
        Gradient=( Y:+Y:*exp(-XB):-1):/(1:+exp(-XB) )
        Hessian=&( -exp(-XB):/(1:+exp(-XB)):^2 )
}
void function xtspjlogit::Check(real matrix Data, real scalar Keep) {
    if (sum(Data[.,1])==0 || sum(Data[.,1])==rows(Data) || rows(Data)<=1) {Keep=0;}
    else {Keep=1;}
}
end
```

Poisson model

```
mata
class xtspjpoisson extends xtspjModel {
    public: void Evaluate(), Check()
}
void function xtspjpoisson::Evaluate(real matrix Y, real matrix XB,
    real colvector LogLikelihood, real matrix Gradient,
    pointer(real colvector) matrix Hessian) {
        LogLikelihood=Y:*ln(exp(XB))-exp(XB)-lnfactorial(Y)
        Gradient=Y-exp(XB)
        Hessian=&(-exp(XB))
}
void function xtspjpoisson::Check(real matrix Data, real scalar Keep) {
    if (allof(Data[.,1],0) || rows(Data)<=1) {Keep=0;}
    else {Keep=1;}
}
end
```

Exponential model

```
mata
class xtspjexponential extends xtspjModel {
    public: void Evaluate(), Check()
}
void function xtspjexponential::Evaluate(real matrix Y, real matrix XB,
    real colvector LogLikelihood, real matrix Gradient,
    pointer(real colvector) matrix Hessian) {
        LogLikelihood=ln(1/exp(XB))-Y:/exp(XB)
}
void function xtspjexponential::Check(real matrix Data, real scalar Keep) {
    if (sum(Data[.,1]:<=0)!=0 || rows(Data)<=1) {Keep=0;}
    else {Keep=1;}
}
end
```

Weibull model

```
mata
class xtspjweibull extends xtspjModel {
    public: void Evaluate(), Check()
}
void function xtspjweibull::Evaluate(real matrix Y, real matrix XB,
    real colvector LogLikelihood, real matrix Gradient,
    pointer(real colvector) matrix Hessian) {
        u=exp(XB[.,1])
        k=exp(XB[.,2])
        LogLikelihood=ln(k)+(k:-1):*ln(Y)+ln(u)-u:*Y:^k
}
void function xtspjweibull::Check(real matrix Data, real scalar Keep) {
    if (sum(Data[.,1]:<=0)!=0 || rows(Data)<=1) {Keep=0;}
    else {Keep=1;}
}
end
```

Gamma model

```
mata
class xtspjgamma extends xtspjModel {
    public: void Evaluate(), Check()
}
void function xtspjgamma::Evaluate(real matrix Y, real matrix XB,
    real colvector LogLikelihood, real matrix Gradient,
    pointer(real colvector) matrix Hessian) {
        u=exp(XB[.,1])
        k=XB[.,2]
        LogLikelihood=-lngamma(k)-k:*ln(u)+(k:-1):*ln(Y)-Y:/u
        dldXB_u=(-k:/u+Y:/u:^2):*u
        dldXB_k=-digamma(k)-ln(u)+ln(Y)
        Gradient=(dldXB_u,dldXB_k)
        dl2dXB2_u2=((k:/u:^2-2:*Y:/u:^3):*u+(-k:/u+Y:/u:^2)):*u
        dl2dXB2_uk=(-1:/u):*u
        dl2dXB2_k2=-trigamma(k)
        Hessian=(&dl2dXB2_u2,&dl2dXB2_uk&dl2dXB2_uk,&dl2dXB2_k2)
}
void function xtspjgamma::Check(real matrix Data, real scalar Keep) {
    if (sum(Data[.,1]:<=0)!=0 || rows(Data)<=1) {Keep=0;}
    else {Keep=1;}
}
end
```

Negbin2 model

```
mata
class xtspjnegbin extends xtspjModel {
    public: void Evaluate(), Check()
}
void function xtspjnegbin::Evaluate(real matrix Y, real matrix XB,
    real colvector LogLikelihood, real matrix Gradient,
    pointer(real colvector) matrix Hessian) {
        u=exp(XB[.,1])
        r=exp(-XB[.,2])
        LogLikelihood=lngamma(r+Y)-lngamma(Y:+1)
                -lngamma(r)-r:*ln(1:+u:/r)-Y:*ln(1:+r:/u)
        dldXB_u=-r:/u:*(u-Y):/(r+u):*u
        dldXB_r=1:/(r+u):*(Y-u+r:*ln(1:+u:/r)+u:*ln(1:+u:/r)
                -(r+u):*digamma(r+Y)+(r+u):*digamma(r)):*(r)
        Gradient=(dldXB_u,dldXB_r)
}
void function xtspjnegbin::Check(real matrix Data, real scalar Keep) {
    if (allof(Data[.,1],0) ||  rows(Data)<=1) {Keep=0;}
    else {Keep=1;}
}
end
```

C Sparsity of the Hessian

Assuming that T_j is even for all j, the unconcentrated version of $\dot{l}^j(\boldsymbol{\theta})$ is

$$\dot{l}^j\left(\boldsymbol{\theta}, \boldsymbol{\alpha}^j, \boldsymbol{\gamma}^j, \boldsymbol{\delta}^j\right) = 2l^j\left(\boldsymbol{\theta}, \boldsymbol{\alpha}^j\right) - \frac{1}{2}\left\{l_1^j\left(\boldsymbol{\theta}, \boldsymbol{\gamma}^j\right) + l_2^j\left(\boldsymbol{\theta}, \boldsymbol{\delta}^j\right)\right\}$$

where $\boldsymbol{\alpha}^j = (\alpha_1^j, \ldots, \alpha_{N_j}^j)$, $\boldsymbol{\gamma}^j = (\gamma_1^j, \ldots, \gamma_{N_j}^j)$, and $\boldsymbol{\delta}^j = (\delta_1^j, \ldots, \delta_{N_j}^j)$ are the three sets of fixed-effects parameters implicit in $\dot{l}^j(\boldsymbol{\theta})$, and

$$l^j\left(\boldsymbol{\theta}, \boldsymbol{\alpha}^j\right) = \frac{1}{N_j T_j}\sum_{i=1}^{N_j}\sum_{t=1}^{T_j}\log f\left(\mathbf{Y}_{it}^j|\mathbf{X}_{it}^j; \boldsymbol{\theta}, \alpha_i^j\right)$$

$$l_1^j\left(\boldsymbol{\theta}, \boldsymbol{\gamma}^j\right) = \frac{1}{N_j T_j/2}\sum_{i=1}^{N_j}\sum_{t=1}^{T_j/2}\log f\left(\mathbf{Y}_{it}^j|\mathbf{X}_{it}^j; \boldsymbol{\theta}, \gamma_i^j\right)$$

$$l_2^j\left(\boldsymbol{\theta}, \boldsymbol{\delta}^j\right) = \frac{1}{N_j T_j/2}\sum_{i=1}^{N_j}\sum_{t=T_j/2+1}^{T_j}\log f\left(\mathbf{Y}_{it}^j|\mathbf{X}_{it}^j; \boldsymbol{\theta}, \delta_i^j\right)$$

Here $l^j(\boldsymbol{\theta}, \boldsymbol{\alpha}^j)$, $l_1^j(\boldsymbol{\theta}, \boldsymbol{\gamma}^j)$, and $l_2^j(\boldsymbol{\theta}, \boldsymbol{\delta}^j)$ are unconcentrated log-likelihood functions. Thus, each has a sparse Hessian. It is also easy to see that

$$\frac{\partial^2 \dot{l}^j\left(\boldsymbol{\theta}, \boldsymbol{\alpha}^j, \boldsymbol{\gamma}^j, \boldsymbol{\delta}^j\right)}{\partial\eta_1\partial\eta_2} = 0$$

for every pair (η_1, η_2) of distinct elements of $(\boldsymbol{\alpha}^j, \boldsymbol{\gamma}^j, \boldsymbol{\delta}^j)$.