Appendix for "xtevent: Estimation and Visualization in the Linear Panel Event-Study Design"

Simon Freyaldenhoven
Federal Reserve Bank of Philadelphia
Philadelphia, PA
simon.freyaldenhoven@phil.frb.org

Christian B. Hansen
University of Chicago
Chicago, IL
chansen1@chicagobooth.edu

Jorge Pérez Pérez
Banco de México
Mexico City, Mexico
jorgepp@banxico.org.mx

Jesse M. Shapiro
Harvard University and NBER
Cambridge, MA
jesse_shapiro@fas.harvard.edu

Constantino Carreto
Banco de México
Mexico City, Mexico
constantino.carreto@banxico.org.mx

A Details on trend extrapolation

The default method to extrapolate a linear trend uses GMM and is implemented as follows. Let $T_G \leq L_G$ be the number of periods prior to G used to estimate the trend parameters and $T_M \leq M$ be the number of "post-event" periods where the trend is

active. We assume $f_k \neq 0$ for $k \in [-G - T_G, T_M]$ and 0 otherwise. We then have moments given by

$$\widehat{\delta}_k - \phi' f_k = 0$$

for $k = -G - T_G, ..., -G - 1$. Let $\widehat{\delta}_{T_G}$ be the T_G-vector collecting $\widehat{\delta}_k$. Let H_{T_G} be the $T_G \times \dim(\phi)$ matrix whose j^{th} row is f_k' for $k = j - 1 - G - T_G$.

A minimum distance estimator $\widehat{\phi}$ of ϕ solves

$$\widehat{\phi} = \arg\min_{\phi} \widehat{h}(\phi)' \widehat{W} \widehat{h}(\phi)$$
$$\widehat{h}(\phi) = \widehat{\delta}_{T_G} - H_{T_G}\phi.$$

Solving the FOC gives

$$\begin{aligned} 0 &= -H_L' \widehat{W}(\widehat{\delta}_{T_G} - H_L\widehat{\phi}) \\ \widehat{\phi} &= (H_{T_G}' \widehat{W} H_{T_G})^{-1} H_{T_G}' \widehat{W}\widehat{\delta} \end{aligned}$$

Under suitable regularity conditions, we have that

$$\sqrt{n}\begin{pmatrix} \widehat{\phi} - \phi_0 \\ \widehat{\delta}_{T_G} - \delta_{L0} \end{pmatrix} \to N\left(0, \begin{bmatrix} \Lambda_{T_G}\Omega_{T_G}\Lambda_{T_G}' & \Lambda_{T_G}\Omega_{T_G} \\ \Omega_{T_G}\Lambda_{T_G}' & \Omega_{T_G} \end{bmatrix}\right)$$

where

$$\Lambda_{T_G} = (H_{T_G}' W H_{T_G})^{-1} H_{T_G}' W$$

and Ω_{T_G} is the asymptotic variance of $\widehat{\delta}_{T_G}$. The feasible efficient weighting matrix is $\widehat{W} = \widehat{\Omega}_{T_G}^{-1} \to \Omega_{T_G}^{-1}$, and with $W = \Omega_{T_G}^{-1}$ we have that $\Lambda_{T_G}\Omega_{T_G}\Lambda_{T_G}' = (H_{T_G}'\Omega_{T_G}^{-1}H_{T_G})^{-1}$.

A.1 Estimation and inference on adjusted event-time path

Now let $\widehat{\delta}$ be the vector containing the entire estimated event-time path, so $\dim(\widehat{\delta}) = \dim(\delta) = M + L_M + G + L_G + 2$. Let H be the $\dim(\delta) \times \dim(\phi)$ matrix whose j^{th} row is f_k' for $k = j - 2 - G - L_G$. Given the estimate $\widehat{\phi}$ we obtain the plugin estimate $\widehat{\delta}^*$ of the adjusted event-time path by

$$\widehat{\delta}^* = \widehat{\delta} - H\widehat{\phi}.$$

Let $\Lambda = \begin{bmatrix} \mathbf{0} & \Lambda_{T_G} & \mathbf{0} \end{bmatrix}$, with $\mathbf{0}$ conformable matrices of 0s ($\dim(\phi) \times 1$ and $\dim(\phi) \times (\dim(\delta) - L_G)$, respectively), $\widehat{\Lambda}$ be its sample analogue, and I be a $\dim(\delta) \times \dim(\delta)$ identity matrix. Hence $\widehat{\delta}^* = \widehat{\delta} - H\widehat{\phi} = \left(I - H\widehat{\Lambda}\right)\widehat{\delta}$ and it follows that (again under suitable conditions)

$$\sqrt{n}\left(\widehat{\delta}^* - \delta_0^*\right) \to N\left(0, \Omega - H\Lambda\Omega - \Omega\Lambda' H' + H\Lambda\Omega\Lambda' H'\right)$$

where Ω is the asymptotic variance matrix of $\hat{\delta}$ and

$$\delta_{0,k}^* = \begin{cases} 0, & k < -G \\ \sum_{m=-G}^{k} \beta_m, & -G \leq k \leq M \\ \sum_{m=-G}^{M} \beta_m, & k > M. \end{cases}$$

Hypothesis testing for pre-trends and dynamics leveling off can now proceed as in the TWFE case, replacing $\widehat{\delta}$ with $\widehat{\delta}^*$.

A.2 Covariance of adjusted event-time path and coefficient on controls

For some purposes we may be interested in testing hypotheses jointly on (δ_0^*, ψ_0). Since $\widehat{\delta}^* = (I - H\widehat{\Lambda})\widehat{\delta}$, we have

$$\sqrt{n}\begin{pmatrix} \widehat{\delta}^* - \delta_0^* \\ \widehat{\psi} - \psi_0 \end{pmatrix} = \begin{pmatrix} I - H\widehat{\Lambda} & 0 \\ 0 & I \end{pmatrix} \sqrt{n}\begin{pmatrix} \widehat{\delta} - \delta_0 \\ \widehat{\psi} - \psi_0 \end{pmatrix} \to \begin{pmatrix} I - H\Lambda & 0 \\ 0 & I \end{pmatrix} N(0, V),$$

with 0 conformable matrices of zeros ($\dim(\delta) \times \dim(\psi)$ for the upper right and $\dim(\psi) \times \dim(\delta)$ for the lower left) and I conformable identity matrices ($\dim(\delta)$ for the upper left and $\dim(\psi)$ for the lower right). Finally, let Ω_ψ denote the $\dim(\psi) \times \dim(\psi)$ asymptotic variance of $\widehat{\psi}$ and $\Omega_{\delta,\psi}$ denote the $\dim(\psi) \times \dim(\delta)$ asymptotic covariance

between $\widehat{\delta}, \widehat{\psi}$. We can express

$$V = \begin{pmatrix} \Omega & \Omega'_{\delta,\psi} \\ \Omega_{\delta,\psi} & \Omega_{\psi} \end{pmatrix}$$

and the asymptotic variance of $\left(\widehat{\delta}^*, \widehat{\psi}\right)$ is

$$\begin{pmatrix} I - H\Lambda & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \Omega & \Omega'_{\delta,\psi} \\ \Omega_{\delta,\psi} & \Omega_{\psi} \end{pmatrix} \begin{pmatrix} I - \Lambda' H' & 0 \\ 0 & I \end{pmatrix} = \begin{pmatrix} (I - H\Lambda)\Omega(I - \Lambda' H') & (I - H\Lambda)\Omega'_{\delta,\psi} \\ \Omega_{\delta,\psi}(I - \Lambda' H') & \Omega_{\psi} \end{pmatrix}.$$

B Sup-t confidence bands

We use sup-t bands for uniform inference (see Freyberger and Rai (2018), Montiel Olea and Plagborg-Møller (2019), and the references therein for additional background). These bands are constructed by adding (subtracting) a constant times the vector of standard errors of $\hat{\delta}$ from $\hat{\delta}$, such that the simultaneous confidence band at each coefficient δ_k is equal to

$$\hat{B}_k(\alpha) \equiv \left[\hat{\delta}_k - c_\alpha \hat{\sigma}_k, \hat{\delta}_k + c_\alpha \hat{\sigma}_k\right]$$

for a chosen significance level α, where c_α denotes the corresponding sup-t critical value.

To compute c_α, we use a simple plug-in estimator (Montiel Olea and Plagborg-Møller (2019)).

1. Draw N i.i.d vectors $\hat{V}^{(\ell)}$, $\ell = 1, \ldots, N$ of dimension $K = \dim(\hat{\delta})$ from a multivariate normal with mean $\mathbf{0}_K$ and variance $\hat{\Omega}$ given by the estimated variance of $\hat{\delta}$.

2. For each replication $\ell = 1, \ldots, N$, let $t_\ell = \max_{k=1,\ldots,K} \left|\hat{\Omega}_{kk}^{-1/2} \hat{V}_k^{(\ell)}\right|$.

3. Set $c_\alpha = Q_{1-\alpha}(t_\ell)$, $\ell = 1, \ldots, N$, where Q is the quantile function.

C Details for least wiggly path

C.1 The least wiggly path proposal

We denote the dimension of δ as $K \equiv G + M + L_G + L_M + 2$. For v, a finite-dimensional coefficient vector, and k, an integer, define the polynomial term

$$\delta_k^*(v) = \sum_{j=1}^{\dim(v)} v_j \left(\frac{k - s_1}{s_2}\right)^{j-1},$$

where v_j denotes the j^{th} element of coefficient vector v and $\dim(v)$ denotes the dimension of this vector. s_1 and s_2 denote constants that shift and scale the event time (range of the polynomial). We set $s_1 = -G - L_G - 1$ and $s_2 = M + L_M + G + L_G + 2$. Let $\delta^*(v)$ collect the elements $\delta_k^*(v)$ for $-G - L_G - 1 \leq k \leq M + L_M$, so that $\delta^*(v)$ reflects a polynomial path in event time with coefficients v.

`xtevent` plots the least "wiggly" confound whose path is contained in the Wald region $CR(\delta)$ for the event-time path of the outcome. Specifically, it plots $\delta^*(v^*)$, where

$$p^* = \min\{\dim(v) : \delta^*(v) \in CR(\delta)\} \text{ and} \tag{1}$$

$$v^* = \arg\min_v \{v_{p^*}^2 : \dim(v) = p^*, \delta^*(v) \in CR(\delta)\}. \tag{2}$$

Intuitively, the Wald confidence region represents the set of event time paths for which a joint F-test of the observed point estimates is not rejected. Since this region is an ellipsis, there is no straightforward graphical illustration of this region in an event plot.

To plot the least wiggly path, we solve a two-part problem. In (1), we find the smallest order p^* such that a polynomial of order p^* is entirely contained in the Wald region $CR(\delta)$. In (2), we then choose the polynomial with the lowest coefficient on the highest order term of that polynomial.

In practice we normalize the event path, such that $\delta_k = 0$ for at least one k (e.g. usually at $k = -1$). We will use $\mathcal{N}$ to denote the set of size $|\mathcal{N}|$ that collects all

normalized coefficients, such that $\delta_k^*(v) = 0$ for $k \in \mathcal{N}$. Throughout, we only consider the case $|\mathcal{N}| \in \{1, 2\}$, i.e., we allow for at most two normalizations.

C.2 Implementation

C.2.1 Finding p^*

We start with the problem of finding p^* in (1). We define Σ as the covariance matrix of $\widehat{\delta}$ with added zeros in the rows and columns corresponding to the normalized coefficients.

Since p^* is generally small, it is feasible to solve (1) iteratively as follows:

Algorithm 1 Finding p^*

$p^* \leftarrow 0$
feasible $\leftarrow 0$
while feasible $= 0$ **do**
  $p^* \leftarrow p^* + 1$
  feasible $\leftarrow$ SolutionInWaldRegion$(\widehat{\delta}, p^*, \alpha)$
end while

function SolutionInWaldRegion$(\widehat{\delta}, p^*, \alpha)$
  $W^* = \min_{v:dim(v)=p^*} [\delta^*(v) - \widehat{\delta}]' \Sigma^{-1} [\delta^*(v) - \widehat{\delta}]$ s.t. $\delta_{k_n}^*(v) = 0$ for $n \in \mathcal{N}$
    ▷ Σ^{-1} denotes the generalized inverse.
  return 1$(W^* \leq c^{1-\alpha})$
    ▷ $c^{1-\alpha}$ is the $1-\alpha$ quantile of a random variable $\tau \sim \chi^2(K - |\mathcal{N}|)$.
end function

Note that p^* is less than $K = \dim(\delta)$ by construction, and thus the loop in Algorithm 1 is (at least theoretically) guaranteed to converge after at most K rounds. To ensure numerical stability, we restrict p^* to be less than or equal to ten in our implementation (with a user option to reduce the upper bound further). If $p^* > 10$, we conclude "no smooth path exists."

To find W^* in practice, we use the first-order conditions of the minimization of the Wald statistic subject to the constraints on the normalized coefficients.

To do this, we write the least wiggly path polynomial in matrix notation as $\delta^*(v) = \underset{K\times p^* p^*\times 1}{F}\ v$, where $F_{kj} = \left(\frac{k-s_1}{s_2}\right)^{j-1}$ for $k = 1, \ldots, K$. The rows of F collect the polynomial terms for a given (shifted) event time k, and the vector v collects the polynomial coefficients. The problem for finding W^* can be rewritten as:

$$\min_v \left[Fv - \widehat{\delta}\right]' \Sigma^{-1} \left[Fv - \widehat{\delta}\right] \text{s.t. } \delta_k^*(v) = 0 \text{ for } k \in \mathcal{N}. \tag{3}$$

From the Lagrangian, the first-order conditions are:

$$F'\Sigma^{-1}Fv = F'\Sigma^{-1}\widehat{\delta} + \frac{1}{2}\lambda A'_{norm}$$
$$A_{norm}v = 0$$

Here, A_{norm} is the matrix with the rows of F corresponding to the normalized coefficients. Algebra then shows that $v(\lambda) = (F'\Sigma^{-1}F)^{-1}[F'\Sigma^{-1}\widehat{\delta} + \frac{1}{2}\lambda A'_{norm}]$, and plugging this back into the second first order condition above yields

$$\lambda = -2[A_{norm}(F'\Sigma^{-1}F)^{-1}A'_{norm}]^{-1}A_{norm}(F'\Sigma^{-1}F)^{-1}F'\Sigma^{-1}\widehat{\delta}.$$

Thus, the solution for v is given by

$$\tilde{v} = (F'\Sigma^{-1}F)^{-1}$$
$$\left[F'\Sigma^{-1}\widehat{\delta} - A'_{norm}[A_{norm}(F'\Sigma^{-1}F)^{-1}A'_{norm}]^{-1}A_{norm}(F'\Sigma^{-1}F)^{-1}F'\Sigma^{-1}\widehat{\delta}\right].$$

We can write the solution for v as a matrix product. Let $XX \equiv \begin{bmatrix} 2F'\Sigma^{-1}F & A'_{norm} \\ A_{norm} & \underset{|\mathcal{N}|\times|\mathcal{N}|}{\mathbf{0}} \end{bmatrix}$ and $Xy \equiv \begin{bmatrix} 2F'\Sigma^{-1}\widehat{\delta} \\ \underset{|\mathcal{N}|\times|\mathcal{N}|}{\mathbf{0}} \end{bmatrix}$. Then the solution for v is the vector with the first K rows of $\tilde{v} = (XX)^{-1}Xy$.

C.2.2 Finding the optimal path given p^*

Once we have found a solution to (1) using Algorithm 1, the next step is to find the polynomial with the lowest coefficient on the p^* term that is still contained in

the Wald region (see equation 2). First note that by construction $v^2_{p^*} \neq 0$ (If not, Algorithm 1 would have found a solution at $p^* - 1$). v^* can then be found through a simple constrained minimization on the vector v (of dimension p^*):

$$v^* = \arg\min_v v^2_{p^*} \tag{4}$$

$$\text{such that } [\delta^*(v) - \widehat{\delta}]\Sigma^{-1}[\delta^*(v) - \widehat{\delta}] \leq c^{1-\alpha}\} \tag{5}$$

$$\text{and } \delta^*_k(v) = 0 \text{ for } k \in \mathcal{N}, \tag{6}$$

with Σ and $c^{1-\alpha}$ defined as above.

First, if $p^* \leq |\mathcal{N}|$, the constraint in (6) implies that $v^* = 0$ and we are done. Next, if $p^* > |\mathcal{N}|$, we note that v^* will always be on the boundary of the Wald region. Thus, the constraint in (5) will always be binding, and we can substitute both constraints directly to solve for v^*. In particular, given a set $\mathcal{N}$ of normalized coefficients and the constraint in (5), we can solve for some of the other coefficients. If $p^* > |\mathcal{N}| + 1$, we use an unconstrained optimization to solve for the remaining ones after that.

Specifically, partition the matrices A_{norm} and F into three parts as follows

$$A_{norm} = \underset{|\mathcal{N}|\times(p^*-|\mathcal{N}|-1)}{\left[A_b\right.}, \underset{|\mathcal{N}|\times|\mathcal{N}|}{A_1}, \underset{|\mathcal{N}|\times 1}{\left.A_2\right]}$$

$$F = \underset{K\times(p^*-|\mathcal{N}|-1)}{\left[F_b\right.}, \underset{K\times|\mathcal{N}|}{F_1}, \underset{K\times 1}{\left.F_2\right]},$$

with the vector v partioned accordingly into $v = [v_b; v_1; v_2]$. We will solve for the coefficients v_1 and v_2 using the constraints in (6) and (5) respectively, and then solve for the coefficients v_b by unconstrained minimization. To do so, first note that, because A_{norm} contains the rows of F associated with the normalized coefficients,

$$A_{norm}v = A_b v_b + A_1 v_1 + A_2 v_2 = 0 \text{ and thus } v_1 = {A_1}^{-1}(-A_b v_b - A_2 v_2). \tag{7}$$

It follows that

$$\delta^*(v) = Fv = F_b v_b + F_1 v_1 + F_2 v_2 = F_b v_b - F_1\left[{A_1}^{-1}(A_b v_b + A_2 v_2)\right] + F_2 v_2,$$

and the constraint in (5) becomes

$$
\begin{aligned}
0 =& \left([(F_b - F_1 {A_1}^{-1} A_b) v_b - \widehat{\delta}]' \Sigma^{-1} [(F_b - F_1 {A_1}^{-1} A_b) v_b - \widehat{\delta}] - c^{1-\alpha} \right) \\
&+ 2 \left([(F_b - F_1 {A_1}^{-1} A_b) v_b - \widehat{\delta}]' \Sigma^{-1} [(F_2 - F_1 {A_1}^{-1} A_2) \right) v_2 \\
&+ v_2' \left([(F_2 - F_1 {A_1}^{-1} A_2)]' \Sigma^{-1} [(F_2 - F_1 {A_1}^{-1} A_2)] \right) v_2.
\end{aligned}
$$

This is a quadratic expression for (the scalar) v_2 in terms of v_b and, defining the scalars d_0, d_1 and d_2 as

$$
\begin{aligned}
d_0 &= \left[(F_2 - F_1 {A_1}^{-1} A_2) \right]' \Sigma^{-1} \left[(F_2 - F_1 {A_1}^{-1} A_2) \right], \\
d_1(v_b) &= 2 \left([(F_b - F_1 {A_1}^{-1} A_b) v_b - \widehat{\delta}]' \Sigma^{-1} [(F_2 - F_1 {A_1}^{-1} A_2) \right), \text{and} \\
d_2(v_b) &= \left([(F_b - F_1 {A_1}^{-1} A_b) v_b - \widehat{\delta}]' \Sigma^{-1} [(F_b - F_1 {A_1}^{-1} A_b) v_b - \widehat{\delta}] - c^{1-\alpha} \right)
\end{aligned}
$$

simplifies to $d_0 v_2^2 + d_1(v_b) v_2 + d_2(v_b) = 0$.

Using the quadratic formula, we can then solve for v_2 by solving the minimization problem,

$$
v_2(v_b) = \frac{-d_1(v_b) \pm \sqrt{d_1(v_b)^2 - 4 d_0 d_2(v_b)}}{2 d_0}. \tag{8}
$$

Note that, by definition, $v_2 = v_{p^*}$.

Further, if $p^* = |\mathcal{N}| + 1$, v_b is empty and thus v_2 does not depend on v_b. (8) results in two solutions, v_2^+ and v_1^-, corresponding to the sign ambiguity in (8). We choose the solution v_2^* with the smaller absolute value.

If $p^* > |\mathcal{N}|+1$, the constrained optimization in (4)-(6) is equivalent to the following:

$$
v_2^2 = \min_{v_b} \min_{\{+,-\}} \left(\frac{-d_1(v_b) \pm \sqrt{d_1(v_b)^2 - 4 d_0 d_2(v_b)}}{2 d_0} \right)^2, \tag{9}
$$

where the inner minimization is over the sign in the quadratic formula.

At this point, we have both v_2^* and v_b^*. Recovering v_1^* using (7), we obtain $v^* =$

$[v_b^*, v_1^*, v_2^*]$.

D Policy variable imputation

Panel event study estimation requires assumptions about the behavior of the policy variable outside the observed time range. In section 4.1 of the article, we estimated panel event studies assuming no unobserved changes in the policy variable outside the estimation period. This imputation scheme is implemented using the `impute(nuchange)` option.

`xtevent` allows for other schemes to impute the policy variable. For example, `xtevent` can assume that the policy variable follows staggered adoption, using the `impute(stag)` option. It can also impute missing values of the policy variable inside the observed date range using the `impute(instag)` option.

To illustrate these options, we use the simulated data example of section 4 of the article and show the implied event-time dummies under the different imputation schemes. For the example, we add some missing values to unit 19. Then, we differentiate the policy variable. `xtevent` uses leads and lags of the differentiated policy variable to generate the event-time dummies, following equation (2).

First, we ask `xtevent` to generate the event-time dummies without any imputation and specify the option `savek(stub, noestimate)` to save them without estimating the model.

```
. use simulation_data_dynamic.dta, clear
. qui xtset id t
. qui replace z=. if id==19 & (t==35 | t>=39)
. qui gen z_d=d.z
. qui xtevent y x, panelvar(id) timevar(t) policyvar(z)
>  window(5) savek(v, noestimate)
```

The event-time dummies with a "v" prefix and ending in `m#` or `p#` correspond to leads and lags of the differentiated policy variable, as described in section 3 of the article. Now, we display these event-time dummies for unit 19 in some periods.

```
. list id t z z_d v_eq_m6 -v_eq_m1 if id==19 & t>=29, ///
> separator(4) noobs
```

id	t	z	z_d	v_eq_m6	v_eq_m5	v_eq_m4	v_eq_m3	v_eq_m2	v_eq_m1

19	29	0	0	1	0	0	0	0	0
19	30	0	0	.	.	0	0	0	0
19	31	0	0	1	.	.	0	0	0
19	32	0	0	0	1	.	.	0	0
19	33	0	0	0	0	1	.	.	0
19	34	0	0	.	.	0	1	.	.
19	35	.	.	.	.	.	0	1	.
19	36	0	.	.	.	.	.	0	1
19	37	1	1	.	.	.	.	.	0
19	38	1	0	.	.	.	.	.	.
19	39	.	.	.	.	.	.	.	.
19	40	.	.	.	.	.	.	.	.

Notice that the event-time dummies have missing values at the bottom of the table because we have not made any assumptions about the policy variable outside the observed time range. Besides, notice that the event-time dummies have some missing values inside the observed time range due to the missing value in the policy variable in period 35. From equation (2), this latter missing value translates into two inner missing values in the event-time dummies and one missing value in the case of the left endpoint.

To impute the policy variable under staggered adoption, we use the `impute(stag)` option. `xtevent` verifies that the policy variable follows staggered adoption. If so, `xtevent` imputes the policy variable outside the observed time range. Then, it uses the imputed policy variable to generate the event-time dummies and endpoints. We add the suboption `saveimp` to save the imputed policy variable as `z_imputed`. We also differentiate the new imputed policy variable to see how its leads and lags translate to new event-time dummies.

```
. cap drop v*

. qui xtevent y x, panelvar(id) timevar(t) policyvar(z) ///
> window(5) savek(v, noestimate) impute(stag, saveimp)

. qui gen z_imputed_d=d.z_imputed
```

Below, we compare the original policy variable, the imputed policy variable, the differentiated imputed policy variable, some event-time dummies, and the left endpoint generated using the imputed policy variable. First, the policy variable has been imputed in the observed time range. Nonetheless, the imputation also assumes that

the policy variable in periods after $t = 40$ would have the same value as the one in that last period. This imputation can be seen in the event-time dummies, which now have zeros corresponding to leads of the differentiated policy variable in periods after 40.

```
. list id t z z_imputed z_imputed_d v_eq_m6 -v_eq_m3 if id==19 & t>=29, ///
> separator(4) noobs ab(11)
```

id	t	z	z_imputed	z_imputed_d	v_eq_m6	v_eq_m5	v_eq_m4	v_eq_m3
19	29	0	0	0	1	0	0	0
19	30	0	0	0	.	.	0	0
19	31	0	0	0	1	.	.	0
19	32	0	0	0	0	1	.	.
19	33	0	0	0	0	0	1	.
19	34	0	0	0	0	0	0	1
19	35	.	.	.	0	0	0	0
19	36	0	0	.	0	0	0	0
19	37	1	1	1	0	0	0	0
19	38	1	1	0	0	0	0	0
19	39	.	1	0	0	0	0	0
19	40	.	1	0	0	0	0	0

We now ask `xtevent` to impute the policy variable using the `impute(instag)` option. This imputation scheme lets us impute missing values in the policy variable outside and inside the observed time range. As described in section 3 of the article, the `impute(instag)` option implements the `impute(stag)` option, but it also imputes missing values inside the observed time range in cases where it is possible to assume some value based on the policy values in surrounding periods. As in the previous example, we also generate the differentiated imputed policy variable for comparison.

```
. cap drop v* z_imputed z_imputed_d

. qui xtevent y x, panelvar(id) timevar(t) policyvar(z) ///
> window(5) savek(v, noestimate) impute(instag, saveimp)

. qui gen z_imputed_d=d.z_imputed
```

Below, we compare the original policy variable, the imputed policy variable, the differentiated imputed policy variable, some event-time dummies, and the left endpoint generated with the imputed policy variable. First, the imputed policy variable

does not have missing values inside or outside the event-time range. As in the example using `impute(stag)`, the event-time dummies have zeros corresponding to leads of the differentiated policy variable in periods greater than 40. Additionally, now the event-time dummies do not have missing values inside the event-time range.

```
. list id t z z_imputed z_imputed_d v_eq_m6 -v_eq_m3 if id==19 & t>=29, ///
> separator(4) noobs ab(11)
```

id	t	z	z_imputed	z_imputed_d	v_eq_m6	v_eq_m5	v_eq_m4	v_eq_m3
19	29	0	0	0	1	0	0	0
19	30	0	0	0	1	0	0	0
19	31	0	0	0	1	0	0	0
19	32	0	0	0	0	1	0	0
19	33	0	0	0	0	0	1	0
19	34	0	0	0	0	0	0	1
19	35	.	0	0	0	0	0	0
19	36	0	0	0	0	0	0	0
19	37	1	1	1	0	0	0	0
19	38	1	1	0	0	0	0	0
19	39	.	1	0	0	0	0	0
19	40	.	1	0	0	0	0	0

E Estimation in repeated cross-sectional datasets

`xtevent` allows estimation with repeated cross-sectional datasets when `policyvar` varies at the group level, and `panelvar` identifies the groups. For instance, `panelvar` could indicate states at which `policyvar` changes, while the observations in the dataset could be individuals in each state. `xtevent` allows estimations in these settings directly with the `repeatedcs` option. It also allows for using the two step procedure described in Hansen (2007). To use the latter method, the user should first use the `get_unit_time_effects` command to estimate unit-time effects and then use these estimations as input for `xtevent`.

We illustrate the use of `get_unit_time_effects`. First, we create a variable `state` that represents groups where individuals receive the treatment in the same period. Then, we call `get_unit_time_effects`. It saves a `dta` file with the unit-time effects.

```
. gen state=eventtime

. xtset, clear
```

```
. get_unit_time_effects y x, panelvar(state) timevar(t)
> saving("effect_file.dta", replace)
  (output omitted)
file .outputanalysiseffect_file.dta saved
```

Then, we keep one observation per state-time in the repeated cross-sectional data and merge the dataset with the unit-time effects. Afterwards, we execute `xtevent`. Since we use a smaller dataset to estimate the event-study, this method can be faster than using the `repeatedcs` option.

```
. qui bysort state t (z): keep if _n==1

. keep state t z

. qui merge m:1 state t using effect_file.dta, nogen

. xtevent _unittimeeffects, panelvar(state) timevar(t) policyvar(z) window(5)
  (output omitted)
```

F Estimation with heterogenous treatment effects in staggered adoption settings

`xtevent` allows estimation permitting heterogeneous treatment effects in staggered adoption settings. Following the approach of Sun and Abraham (2021), we introduce a setting with heterogeneous treatment effects across treatment cohorts as follows: let $t^\star(i)$ be the cohort for unit i, namely, the period when unit i adopts the policy. Denote the effects of the policy on the outcome for cohort $t^\star$ as $\{\beta_{m,t^\star}\}_{m=-G}^{M}$. The equation for the outcome becomes:

$$y_{it} = \alpha_i + \gamma_t + q'_{it}\psi + \sum_{m=-G}^{M} \beta_{m,t^\star(i)} z_{i,t-m} + C_{it} + \varepsilon_{it}.$$

To estimate the event-study path of the outcome in this setting, `xtevent` estimates an extended version of equation (2) interacting the event-time dummies with cohort indicators as proposed in Freyaldenhoven et al. (Forthcoming):

$$y_{it} = \sum_{c} \sum_{k=-G-L_G}^{M+L_M-1} \left(\mathbf{1}\left\{t^\star(i) = c\right\}\right) \left(\delta_{k,c}\Delta z_{i,t-k} + \delta_{M+L_M,c} z_{i,t-M-L_M} + \delta_{-G-L_G-1,c}(1 - z_{i,t+G+L_G})\right)$$
$$+\alpha_i + \gamma_t + q'_{it}\psi + C_{it} + \varepsilon_{it}, \tag{10}$$

Using the two-way fixed effects estimator to estimate (10), this is equivalent to the estimator proposed in Sun and Abraham (2021). `xtevent` further allows to estimate (10) using any of the estimation strategies outlined in Section 2.1., except IV estimation.

`xtevent` estimates (10) on a sample defined by the `cohort` and `control_cohort` options. In staggered adoption settings, the option `cohort(create)` asks `xtevent` to automatically generate a categorical variable for treatment cohorts. `xtevent` sets the value of this categorical variable to the value of the time variable the first time a unit is treated. For units that are never treated, `xtevent` sets the value of this variable to missing. Similarly, the `control_cohort(create)` option asks `xtevent` to automatically generate an indicator for the control cohort. `xtevent` sets this indicator to 1 for never-treated units and zero otherwise.

If the cohort in `control_cohort` is a never-treated cohort, `xtevent` estimates equation (10) on the whole sample. Otherwise, `xtevent` estimates (10) on the subset of periods when observations in the control cohort have not yet been treated. By default, the estimation excludes always-treated cohorts.

After obtaining estimates $\hat{\delta}_{k,c}$ for $k = -G - L_G - 1, ..., M + L_M$ and for each cohort c, `xtevent` obtains the estimate of the average treatment effect at event-time k, $\hat{\delta}_k$, as an average of the $\hat{\delta}_{k,c}$ estimates weighting by the share of observations from cohort c in each relative time k (Sun and Abraham, 2021):

$$\hat{\delta}_k = \sum_c \delta_{k,c} \hat{Pr}\left\{t^\star(i) = c \mid t^\star(i) \in [-k, T-k]\right\}.$$

The variance of $\hat{\delta}_k$ is obtained using the formulas in Appendix C.1 of Sun and Abraham (2021).

To permit greater flexibility, `xtevent` also allows estimation of (10) outside stag-

gered adoption settings or in settings where the user wants to aggregate treatment cohorts for user-provided cohort and control cohort variables through the `cohort(variable varname, [,force])` and `control_cohort(variable varname, [,force])` options.