The Stata Journal (2025)
25, Number 1, pp. 97–135 DOI: 10.1177/1536867X251322964

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

Abstract. Linear panel models and the "event-study plots" that often accompany them are popular tools for learning about policy effects. We introduce the `xtevent` package, which enables the construction of event-study plots following the suggestions in Freyaldenhoven et al. (Forthcoming, *Visualization, identification, and estimation in the linear panel event-study design* [Cambridge University Press]). The package implements various procedures to estimate the underlying policy effects and allows for nonbinary policy variables and estimation adjusting for preevent trends.

Keywords: st0767, xtevent, xteventplot, xteventtest, get_unit_time_effects, linear panel-data models, two-way fixed-effects regression, pretrends, event study

1 Introduction

In this article, we introduce the `xtevent` package, which enables the estimation of linear panel models with dynamic policy effects under various identifying assumptions. It further enables the construction of the corresponding event-study plots following the suggestions in Freyaldenhoven et al. (Forthcoming).

We are interested in learning the dynamic effect of a scalar policy z_{it} on some outcome y_{it} in an observational panel of units $i \in \{1, \dots, N\}$ observed in a sequence of periods $t \in \{1, \dots, T\}$. We consider the following model:

$$y_{it} = \alpha_i + \gamma_t + \mathbf{q}'_{it}\boldsymbol{\psi} + \sum_{m=-G}^{M} \beta_m z_{i,t-m} + C_{it} + \varepsilon_{it} \tag{1}$$

Here α_i denotes a unit fixed effect, γ_t a time fixed effect, and $\mathbf{q}_{it}$ a vector of controls with conformable coefficients $\boldsymbol{\psi}$. The scalar C_{it} denotes a (potentially unobserved) confound that may be correlated with the policy, and the scalar ε_{it} represents an unobserved shock uncorrelated with the policy. The parameters $\{\beta_m\}_{m=-G}^{M}$ encapsulate the dynamic effects of the policy. Specifically, the outcome at time t can be directly affected by the policy variable's value at most $M \geq 0$ periods before t and at most $G \geq 0$ periods after t.

Typical event-study plots used to visualize the dynamic effects of the policy rely on the following variation of (1) (see Freyaldenhoven et al. [Forthcoming]):

$$\begin{aligned} y_{it} = & \sum_{k=-G-L_G}^{M+L_M-1} \delta_k \Delta z_{i,t-k} + \delta_{M+L_M} z_{i,t-M-L_M} + \delta_{-G-L_G-1}(1 - z_{i,t+G+L_G}) \\ & + \alpha_i + \gamma_t + \mathbf{q}'_{it}\boldsymbol{\psi} + C_{it} + \varepsilon_{it} \end{aligned} \tag{2}$$

Δ denotes the first-difference operator. In (2), the parameters $\{\delta_k\}_{k=-G-L_G-1}^{k=M+L_M}$ measure the cumulative effect of the policy at different horizons (Schmidheiny and Siegloch 2023). The corresponding event-study plot then depicts estimates of the cumulative treatment effects at different horizons k. Thus, the x axis corresponds to different values of k, and the y axis corresponds to estimates of policy effects $\left\{\widehat{\delta}_k\right\}_{k=-G-L_G-1}^{k=M+L_M}$. We refer to k as event time, to the vector $\boldsymbol{\delta} = (\delta_{-G-L_G-1}, \dots, \delta_{M+L_M})'$ as the event-time path of the outcome, and to its estimated counterpart $\widehat{\boldsymbol{\delta}}$ as the estimated event-time path.

To permit the visualization of overidentifying information, (2) includes the estimated cumulative effects of the policy at horizons outside the range of horizons over which the policy is thought to affect the outcome. For example, it is common to rule out effects of the policy at time t on the outcome in periods before t ($G = 0$). By including L_G additional periods in (2), we allow a visualization of preevent trends ("pretrends") that are generally inconsistent with the model in (1) (Freyaldenhoven, Hansen, and Shapiro 2019). Similarly, the estimating equation in (2) permits visualizing the estimated cumulative effect for an additional L_M periods after the cumulative treatment effect is assumed to be constant in (1).

While (2) allows for general (for example, nonbinary) policy variables z_{it}, it is instructive to consider the particular case of *staggered adoption*, by which we mean that the policy is binary, all units begin without the policy, and once a given unit adopts the policy, it is never reversed. Then $\Delta z_{i,t-k}$ is an indicator for whether unit i adopted the policy exactly k periods before period t; $z_{i,t-M-L_M}$ is an indicator for whether unit i adopted at least $M + L_M$ periods before period t; and $(1 - z_{i,t+G+L_G})$ is an indicator for whether unit i will adopt more than $G + L_G$ periods after period t.

Our package complements many other recent contributions to estimation and visualization of panel event studies in Stata, such as `csdid` (Rios-Avila, Sant'Anna, and Callaway 2021), `did_imputation` (Borusyak 2023), `didmultiplegt` (de Chaisemartin, D'Haultfœuille, and Guyonvarch 2019), `did2s` (Butts 2021), `eventdd` (Clarke and Tapia-Schythe 2021), `eventstudyinteract` (Sun 2021), `jwdid` (Rios-Avila 2022), `lpdid` (Busch and Girardi 2023), `staggered` (Caceres Bravo 2023), and `wooldid` (Hegland 2023), as well as the official `xthdidregress` command recently introduced in Stata 18. Many of these focus on the case of staggered adoption, and by default, they require the user to specify a unit-specific treatment period, a time relative to treatment, or an indicator for observations subject to treatment. By contrast, our package allows for general policy variables z_{it}, such as continuous variables, allowing both estimation and visualization in a wide range of settings outside staggered adoption. We note that our package can be used in the staggered-adoption case by setting z_{it} equal to a unit-period-specific indicator for periods after treatment.

Our package also allows for estimation with preevent trends using approaches based on trend extrapolation (Dobkin et al. 2018) or proxy variables (Freyaldenhoven, Hansen, and Shapiro 2019). Moreover, our package includes tools to enhance event-study plots and ease their interpretation, such as calculation of uniform confidence bands and plotting of plausible confound trajectories consistent with the estimated event-time path.

There are advantages to implementations designed to ensure desirable econometric properties in more specialized settings such as staggered adoption. Our package incorporates one such procedure as an option if the policy indeed follows staggered adoption.

In the following section, we briefly discuss several estimation strategies for (2), provide more details on constructing the corresponding event-study plots, and introduce some additional features of the `xtevent` package. Then, in section 3, we give a more detailed description of the syntax and options for the `xtevent` package. In section 4, we illustrate usage of the package in simulated data from Freyaldenhoven et al. (Forthcoming) and by estimating the effect of a tax reform using data from Martínez (2022a). An appendix in the online supplementary material includes additional details on the implementation and functionality of the package.

2 Methods

2.1 Estimation strategies

In general, identification of the parameters $\boldsymbol{\delta}$ will require some form of restriction on how observable and latent variables relate to the confound C_{it} and the policy z_{it}. The appropriate restriction will depend on the economic setting and typically cannot be learned from the data. In turn, the choice of restriction will determine what type of estimator is appropriate to estimate $\boldsymbol{\delta}$ (see Freyaldenhoven et al. [Forthcoming] for a more detailed discussion). The package `xtevent` comprises the following estimators:

Two-way fixed-effects estimator. If $C_{it} = 0$, (2) may be estimated by ordinary least squares (OLS) using a standard two-way fixed-effects estimator. With only one group of fixed effects, `xtevent` uses `areg` for estimation. `xtevent` further allows for estimation using `xtreg` or the `reghdfe` command (Guimarães and Portugal 2010; Correia 2016b, 2014) to allow for multiple or high-dimensional fixed effects.

Controlling for unit-specific trends. If $C_{it} = \boldsymbol{\lambda}_i' \mathbf{f}(t)$, where $\mathbf{f}(\cdot)$ is a known low-dimensional set of basis functions [for example, $\mathbf{f}(t) = t$], then (2) may be estimated by including unit-specific time trends. These can be included in the regression using factor variables (for example, `i.crosssectionid#c.time`) or absorbing them using `reghdfe`.

Controlling for event-time trends. If C_{it} can be written as

$$C_{it} = \widetilde{\alpha}_i + \widetilde{\gamma}_t + \mathbf{q}_{it}' \widetilde{\boldsymbol{\psi}} + \sum_m \boldsymbol{\phi}' \mathbf{f}(m) z_{i,t-m} \tag{3}$$

for a known set of basis functions $\mathbf{f}(\cdot)$ and unknown parameters $\widetilde{\alpha}_i, \widetilde{\gamma}_t$, and $\widetilde{\boldsymbol{\psi}}$, then (2) may be estimated by including the appropriate terms from (3) directly in a regression model or by generalized method of moments (GMM) in a second step following estimation of (2) via two-way fixed effects.

Intuitively, suppose that a trend in event time can approximate the confound. In that case, we can learn about the trend in periods where the policy is inactive and extrapolate it to later periods. The differences between the outcome variable and the extrapolated trend are then informative of the policy effects (Dobkin et al. 2018). For example, consider a staggered-adoption setting where the confound follows a linear trend in time; this trend starts three periods before the policy activates and continues for three periods afterward. We can represent this situation by taking $\mathbf{f}(m) = 1$ if $m \in [-3, 3]$ and $\mathbf{f}(m) = 0$ otherwise. In this case, we can extrapolate the trend to postadoption periods and subtract it to account for the confound.

We allow z_{it} to be continuous, and adoption may not be staggered. If (3) holds, then the estimand of a standard two-way fixed-effect estimator of (2) is given by

$$d_k = \begin{cases} \boldsymbol{\phi}' \mathbf{f}_k, & \text{if } k < -G \\ \delta_k + \boldsymbol{\phi}' \mathbf{f}_k, & \text{if } -G \le k \le M \\ \delta_M + \boldsymbol{\phi}' \mathbf{f}_k, & \text{otherwise} \end{cases}$$

where $\mathbf{f}_k = \sum_{m=-\infty}^{k} \mathbf{f}(m)$.

Given estimates of d_k, $\widehat{d}_k$, we can recover the trend parameters $\boldsymbol{\phi}$ from the estimates $\widehat{d}_k$ in the L_G unaffected periods. Let $T_G \le L_G$ be the number of periods prior to G used to estimate the trend parameters and $T_M \le M$ be the number of "postevent" periods where the trend is active. We assume $\mathbf{f}_k \ne 0$ for $k \in [-G - T_G, T_M]$ and 0 otherwise. We can recover the trend parameters by using the T_G moment conditions $\widehat{d}_k - \boldsymbol{\phi}' \mathbf{f}_k = 0$ for $k = -G - T_G, \ldots, -G - 1$. We can then calculate an adjusted estimated event-time path, $\widehat{\boldsymbol{\delta}}$, that accounts for the confound by subtracting $\boldsymbol{\phi}' \mathbf{f}_k$ from the unadjusted coefficients d_k for $k \in [-G - T_G, T_M]$. Appendix A in the online supplementary material provides further details about this estimator.

Instrumental-variables (IV) estimation with multiple proxies. If multiple additional variables are available that may serve as proxies for the confound C_{it}, such that

$$x_{it} = \alpha_i^x + \gamma_t^x + \boldsymbol{\phi}^x \mathbf{q}_{it} + \Xi^x C_{it} + u_{it} \tag{4}$$

with unknown parameters α_i^x, γ_t^x, and Ξ^x and an unobserved vector u_{it} (which is uncorrelated across proxies), then `xtevent` permits estimating (2) by two-stage least squares, including one of the proxies in (2), and using the other proxy variable as an excluded instrument (Freyaldenhoven, Hansen, and Shapiro 2019).

IV estimation with single proxy. If only a single additional variable is available that may serve as proxy for the confound C_{it}, with the error in (4) conditionally mean independent of the policy, then `xtevent` permits estimating (2) using a two-stage least-squares estimator, instrumenting for the proxy with leads of the policy variable (Freyaldenhoven, Hansen, and Shapiro 2019).

In principle, all leads of the policy variable further out than G are potential instruments. To select a default choice, `xtevent` estimates (2) via two-way fixed effects, with the proxy as the outcome variable. The default excluded instrument for estimation of (2) is then the variable with the largest absolute t statistic among the leads $\{\Delta z_{i,t+k}\}_{k=G+1}^{G+L_G}$ and $z_{i,t+G+L_G}$.

2.2 Event-study plots

The auxiliary command `xteventplot` includes functionality to visualize the event-study plots based on any of the estimators from the previous section. It further includes the enhancements to these plots suggested in Freyaldenhoven et al. (Forthcoming).

Normalization. Because the policy variables in (2) are collinear, a normalization is required to identify the event-time path $\{\delta_k\}_{k=-G-L_G-1}^{k=M+L_M}$ (Schmidheiny and Siegloch 2023; Freyaldenhoven et al. Forthcoming). `xtevent` normalizes $\delta_{-1} = 0$ by default. In the case of staggered adoption, this normalization implies that the plotted coefficients can be interpreted as estimated effects relative to the period before policy enactment.

Outcome variable level. To ease the interpretation of the estimated policy effects, `xtevent` includes a parenthetical label for the normalized coefficient that reflects the mean of the dependent variable. For instance, in the case of staggered adoption, under our default normalization, the label corresponds to the sample mean of y_{it} one period before adoption. More generally, the label corresponds to the value

$$\frac{\sum_{(i,t):\Delta z_{i,t-k^\star} \neq 0} y_{it}}{|(i,t) : \Delta z_{i,t-k^\star} \neq 0|}$$

where $k^\star$ corresponds to the normalized event-time coefficient $\delta_{k^\star}$.

Uniform inference. In addition to standard pointwise confidence intervals for the coefficients δ_k, `xtevent` allows plots of uniform sup-t confidence bands (Freyberger and Rai 2018; Montiel Olea and Plagborg-Møller 2019). Including these bands allows for visual tests of hypotheses about the entire coefficient path instead of just single coefficients. We provide details about the calculation of sup-t confidence bands in appendix B in the online supplementary material.

Overidentification and testing. The estimating equation in (2) includes L_G additional periods before policy adoption to visualize potential pretrends. Evidence of such pretrends is, in practice, often seen as evidence for the presence of a confound that invalidates the research design (Freyaldenhoven, Hansen, and Shapiro 2019). The estimating equation in (2) also includes L_M additional periods after the policy effects end to assess whether the dynamic effects have leveled off after the M postulated periods for which the policy has a direct effect. `xtevent` displays the p-values of Wald tests for "pretrends" ($\delta_k = 0$ for $-G - L_G - 1 \leq k < -G$) and for dynamic effects "leveling off" ($\delta_M = \delta_{M+k}$ for $0 < k \leq L_M$). The auxiliary command `xteventtest` allows for testing additional hypotheses, such as hypotheses about cumulative effects, whether effects are constant, and whether the effects follow linear trends.

Least wiggly path of confounds consistent with the estimates. To help visualize whether a confound can plausibly explain all the event-time dynamics of the outcome variable, `xtevent` allows for adding a representation of the "most plausible" confound trajectory consistent with the absence of policy effects. Our choice of "most plausible" confound is the least "wiggly" polynomial in event time that passes through the Wald confidence region of the event-time path. The idea is that if a "smooth" path exists, a confound could plausibly explain the entire event-time path of the outcome, even absent any policy effects. On the other hand, if no "smooth" path exists, a confound might not plausibly explain the entire event-time path of the outcome, so the policy does affect the outcome. We describe the computation of the least wiggly path in detail in appendix C in the online supplementary material.

Overlay plots. `xteventplot` allows event-study plots with overlays in different estimation scenarios. For estimation with event-time trends, `xteventplot` creates plots overlaying the trend. For IV estimation with proxies, `xteventplot` allows overlaying the dynamics implied by the proxy variable. `xteventplot` also allows overlays of a constant-effects model to assess whether the policy effects are constant over time.

2.3 Additional features

The package includes the following additional capabilities.

Imputation of missing values in the policy variable and its leads and lags. Because (1) includes leads and lags of the policy variable z_{it} and its first difference, the estimation sample may be smaller than the entire sample available (Schmidheiny and Siegloch 2023). In general, if the outcome variable is observed for $t \in \{\underline{t}, \ldots, \overline{t}\}$, we need to observe the policy variable z_{it} from $\underline{t} - G - L_G$ to $\overline{t} + M + L_M - 1$ to avoid dropping observations from the estimation sample. In a typical setup, this may imply that we must restrict the estimation window to calculate the necessary leads and lags of z_{it}. However, the user may have additional information that allows imputation of the policy variable.

`xtevent` allows for the following imputation schemes:

1. If the policy variable is declared to follow staggered adoption, `xtevent` can automatically impute the following:

   a. Any missing values in the policy variable *outside* the observed data range, assuming no policy changes outside the sample period. For example, observing $z_{jt} = 1$ under staggered adoption implies that $z_{js} = 1$ for $s > t$. We provide an example of this functionality in section 4.1.

   b. Missing values of the policy variable *inside* the observed data range. For example, observing $z_{jt} = 1$ and $z_{j,t+2} = 1$ under staggered adoption implies that $z_{j,t+1} = 1$. We provide examples of this functionality in appendix D in the online supplementary material.

2. Even absent staggered adoption, if the policy variable is declared not to change values outside the observed sample, `xtevent` can automatically impute z_{it} outside the observed sample. For example, if the sample starts at $\underline{t}$ and $z_{1\underline{t}} = 0$ for unit 1, we set $z_{it} = 0$ for $t \in \{\underline{t} - G - L_G, \ldots, \underline{t} - 1\}$.

Estimation with repeated cross-sectional data. Some setups involve repeated cross-sectional data instead of panel data, with treatment varying at a higher level of aggregation. For example, we may have a series of repeated cross-sections of individuals for each US state s and a state-level policy z_{st}. In this environment, estimating (2) with individual fixed effects is unfeasible, but `xtevent` allows estimation of a related model with fixed effects by state,

$$\begin{aligned} y_{it} = & \sum_{k=-G-l_G}^{M+L_M-1} \delta_k \Delta z_{s(i),t-k} + \delta_{M+L_M} z_{s(i),t-M-L_M} + \delta_{-G-L_G-1}(1 - z_{s(i),t+G+L_G}) \\ & + \alpha_{s(i)} + \gamma_t + \mathbf{q}'_{it}\boldsymbol{\psi} + C_{s(i)t} + \varepsilon_{it} \end{aligned}$$

where the parameters $\alpha_{s(i)}$ are fixed effects corresponding to state s, where unit i belongs.

An alternative (Amemiya 1978; Hansen 2007) is to regress y_{it} on a set of state-time indicators, plus any control variables $\mathbf{q}_{it}$ that vary at the individual level, and then estimate (2) with the estimated state-time effects as dependent variables, including state fixed effects, time fixed effects, and controls that vary at the state level. The auxiliary command `get_unit_time_effects` facilitates this approach. We provide an example of this command's usage in appendix E in the online supplementary material.

Heterogeneous treatment effects in staggered adoption settings. The model in (1) assumes that the causal effect of the policy is homogeneous over units i. Recent literature has highlighted that if treatment effects are heterogeneous by treatment time, then the effects estimated with (1) may not be properly weighted averages of the cohort-level treatment effects (Athey and Imbens 2022; Callaway and Sant'Anna 2021; Goodman-Bacon 2021; Sun and Abraham 2021). Sun and Abraham (2021) propose estimating event studies for each treated cohort separately, comparing each one with an untreated cohort and then averaging the effects, weighting by the percentage of treated units in each cohort to arrive at a treatment effect on the treated. For the two-way fixed-effects case and under staggered adoption, `xtevent` allows for estimation of cohort-specific effects, in which case it reports an average weighted by the number of treated observations in each cohort, which is an estimate of a weighted average treatment effect on the treated under assumptions discussed in Sun and Abraham (2021). We provide an example of estimation with heterogeneous treatment effects in section 4 and provide details about the estimation in appendix F in the online supplementary material.

3 The xtevent package

The `xtevent` package includes the commands `xtevent` for estimation, `xteventplot` for visualization, and `xteventtest` for postestimation hypothesis testing. It also includes `get_unit_time_effects`, an auxiliary command to use in combination with `xtevent` in repeated cross-section settings. This section describes the syntax and options of these commands.

3.1 The xtevent command

The `xtevent` command has the following syntax:

```
xtevent depvar [indepvars] [if] [in] [weight], policyvar(policyvar)
   [panelvar(varname) timevar(varname) window(windowspec) pre(integer)
   post(integer) overidpre(integer) overidpost(integer) static
   impute(type[, saveimp]) norm(integer) diffavg savek(stub[, subopt])
   kvars(stub) reghdfe addabsorb(varlist) plot nofe note additional_options
   proxy(varlist) proxyiv(proxyiv_spec) trend(#[, subopt])
   cohort(cohort_spec[, subopt])
   control_cohort(control_cohort_spec[, subopt]) sunabraham repeatedcs]
```

`aweight`s, `fweight`s, and `pweight`s are allowed; see [U] **11.1.6 weight**.

3.1.1 Options

3.1.2 Main

`policyvar(`*policyvar*`)` specifies the policy variable of interest. `policyvar()` is required.

`panelvar(`*varname*`)` specifies the cross-sectional identifier variable that identifies the panels. `panelvar()` is required if the data have not been previously `xtset`; see [XT] **xtset**.

`timevar(`*varname*`)` specifies the time variable. `timevar()` is required if the data have not been previously `xtset`; see [XT] **`xtset`**.

`window(`*windowspec*`)` specifies the window around the policy change event to estimate dynamic effects.

`window(`k`)` with a single positive integer $k > 0$ uses a symmetric window of k periods around the event. For example, if `window(2)`, there will be five coefficients in the window $(-2, -1, 0, 1, 2)$ and two endpoints: -3 and $+3$.

`window(`k_1 k_2`)` with two distinct integers $k_1 \leq 0$ and $k_2 \geq 0$ uses an asymmetric window with k_1 periods before the event and k_2 periods after the event. For example, with `window(-1 2)`, there will be four coefficients in the window $(-1, 0, 1, 2)$ and two endpoints: -2 and $+3$.

`window(max)` uses the largest possible window with the minimum and maximum event times in the estimation sample, accounting for the endpoints. `window(max)` is allowed only if the policy follows staggered adoption; it requires `impute(stag)` or `impute(instag)` to be specified (see below).

`window(balanced)` uses the largest possible window with the minimum and maximum event times in the estimation sample for which all cross-sectional units have data. `window(balanced)` is allowed only if the policy follows staggered adoption and requires `impute(stag)` or `impute(instag)` to be specified (see below).

`window()` is required unless `static` is specified or if the estimation window is specified using options `pre()`, `post()`, `overidpre()`, and `overidpost()` (see below).

`pre()`, `post()`, `overidpre()`, and `overidpost()` is an alternative way to specify the estimation window:

`pre()` is the number of preevent periods where anticipation effects are allowed. With `window()`, `pre()` is `0`.

`post()` is the number of postevent periods where policy effects are allowed. With `window()`, `post()` is the number of periods after the event (not including the period for the event, for example, event time = 0), except the last two periods (assigned to `overidpost()` for the leveling-off test).

`overidpre()` is the number of preevent periods for an overidentification test of pretrends. With `window()`, `overidpre()` is the number of periods before the event.

`overidpost()` is the number of postevent periods for an overidentification test of effects leveling off. With `window()`, `overidpost()` is `2`.

Only one of `window()` or `pre()`, `post()`, `overidpre()`, and `overidpost()` can be declared.

`static` fits a static panel-data model and does not generate or plot event-time dummies. `static` is not allowed with `window()`, `pre()`, `post()`, `overidpre()`, `overidpost()`, or `diffavg`.

`impute(`*type*[`, saveimp`]`)` imputes leads, lags, and missing values in `policyvar()` and uses this new variable as *policyvar*. *type* determines the imputation rule. The suboption `saveimp` adds the new variable to the dataset as *policyvar*`_imputed`. The following imputation types are available:

`impute(nuchange)` imputes missing values in `policyvar()` according to *no unobserved change*: it assumes that for each unit, 1) in periods before the first observed value, the policy value is the same as the first observed value; and 2) in periods after the last observed value, the policy value is the same as the last observed value.

`impute(stag)` applies *no unobserved change* if `policyvar()` satisfies staggered-adoption assumptions for all units: 1) `policyvar()` must be binary, and 2) once `policyvar()` reaches the adopted-policy state, it never reverts to the unadopted-policy state. See Freyaldenhoven et al. (Forthcoming) for a detailed explanation of the staggered-adoption case.

`impute(instag)` applies `impute(stag)` and additionally imputes missing values inside the observed data range: a missing value or a group of them will be imputed only if they are both preceded and followed by the unadopted-policy state or by the adopted-policy state. See appendix D in the online supplementary material for a detailed example of the `impute()` option.

`norm(`*integer*`)` specifies the event-time coefficient to be normalized to 0. The default is `norm(-1)`.

`diffavg` calculates the difference in averages between the postevent estimated coefficients and the preevent estimated coefficients. It also calculates its standard error with `lincom`. `diffavg` is not allowed with `static`.

`savek(`*stub*[`,` *subopt*]`)` saves variables for time-to-event, event-time, trend, and interaction variables. Event-time dummies are stored as *stub*`_eq_m`# for the dummy variable # periods before the policy change and *stub*`_eq_p`# for the dummy variable # periods after the policy change. The dummy variable for the policy change time is *stub*`_eq_p0`. Event time is stored as *stub*`_evtime`. The trend is stored as *stub*`_trend`. For estimation with the Sun and Abraham (2021) method, such that `cohort()` and `control_cohort()` or `sunabraham` is active, the interaction variables are stored as *stub*`_m`#`_c`# or *stub*`_p`#`_c`#, where `c`# indicates the cohort. The following suboptions can be specified:

`noestimate` saves variables for event-time dummies, event time, and trends without fitting the model. This option is helpful if users want to customize their regressions and plots.

`saveinteract` saves interaction variables if `cohort()` and `control_cohort()` or `sunabraham` is specified. `noestimate` and `saveinteract` cannot be specified simultaneously.

`replace` replaces variables for time-to-event, event-time, trend, and interaction variables starting with *stub*.

`kvars(`*stub*`)` uses previously used event-time dummies saved with prefix *stub*. This can be used to speed up estimation.

`reghdfe` uses the command `reghdfe` for estimation instead of `areg`, `ivregress`, and `xtivreg`. `reghdfe` is useful for large datasets. By default, it absorbs the panel fixed effects and the time fixed effects. For OLS estimation, the `reghdfe` option requires `reghdfe` and `ftools` (Correia 2016a) to be installed. For IV estimation, it also requires `ivreghdfe` (Correia 2018) and `ivreg2` (Baum, Schaffer, and Stillman 2002) to be installed. Note that standard errors may differ and singleton clusters may be dropped using `reghdfe`. See Correia (2016b).

`addabsorb(`*varlist*`)` specifies additional fixed effects to be absorbed when using `reghdfe`. By default, `xtevent` includes time and unit fixed effects. `addabsorb()` requires `reghdfe`.

`plot` displays a default event-study plot with standard confidence intervals and sup-t confidence bands (Montiel Olea and Plagborg-Møller 2019). Additional options are available with the postestimation command `xteventplot`.

`nofe` excludes panel fixed effects.

`note` excludes time fixed effects.

additional_options are additional options to be passed to the estimation command. When `proxy()` is specified, these options are passed to `ivregress`. When `reghdfe` is specified, these options are passed to `reghdfe`. Otherwise, they are passed to `areg` or to `regress` if `nofe` is specified. This option is useful for calculating clustered standard errors or changing regression reporting.

3.1.3 IV estimation with proxy variables (Freyaldenhoven et al. Forthcoming)

`proxy(`*varlist*`)` specifies proxy variables for the confound to be included. `proxy()` is not allowed with `cohort()`, `control_cohort()`, or `sunabraham`.

`proxyiv(`*proxyiv_spec*`)` specifies instruments for the proxy variable for the policy. It admits three syntaxes to use either leads of the policy variable or additional variables as instruments. `proxyiv()` is not allowed with `cohort()`, `control_cohort()`, or `sunabraham`.

`proxyiv(select)` selects the lead with the strongest first stage among all possible leads of the differenced policy variable to be used as an instrument. It is the default for the one proxy, one instrument case, and it is available only in this case.

`proxyiv(`*numlist*`)` specifies a *numlist* with the leads of the differenced policy variable as instruments. For example, `proxyiv(1 2)` specifies that the two first leads of the difference of the policy variable will be used as instruments.

`proxyiv(`*varlist*`)` specifies a *varlist* with the additional variables to be used as instruments.

3.1.4 Controlling for event-time trends

`trend(`#[`,` *subopt*]`)` extrapolates a linear trend using the periods from period # before the policy change to one period before the policy change, as in Dobkin et al. (2018). For example, `trend(-3)` uses the coefficients on event times -3, -2, and -1 to estimate the trend. The estimated effect of the policy is the deviation from the extrapolated linear trend. # must be less than -1. `trend()` is available only when the normalized coefficient is -1 and `pre(0)`. The following can be passed as suboptions:

`method(`*string*`)` sets the method to estimate the linear trend. It can be OLS (`ols`) or GMM (`gmm`). `method(ols)` omits the event-time dummies from `trend()` to -1 and adds a linear trend (`_ttrend`) to the regression. `method(gmm)` uses the GMM to compute the trend for the event-time dummy coefficients. The default is `method(gmm)`.

Note that the coefficients for negative-event time will differ between `method(ols)` and `method(gmm)`. `method(ols)` omits the event-time coefficients used to calculate the trend, while `method(gmm)` expresses them as differences from the estimated linear trend.

`saveoverlay` saves estimations for the overlay plot produced by `xteventplot` with option `overlay(trend)`.

3.1.5 Heterogeneous treatment effects (Sun and Abraham 2021)

`cohort(`*cohort_spec*[`,` *subopt*]`)` specifies how to identify the treatment cohorts used for estimation of heterogeneous effects by cohort using the estimator from Sun and Abraham (2021). `cohort()` requires the command `avar` (Baum and Schaffer 2013). `cohort()` is not allowed with `proxy()` or `proxyiv()`.

`cohort(variable` *varname*[`, force`]`)` specifies that the categorical variable *varname* identify each treatment cohort. By default, `xtevent` checks for consistency of the cohort variable and the policy variable. `force` forces `xtevent` to skip this check. This can be useful when estimating heterogeneous treatment effects across groups not defined by treatment cohorts.

`cohort(create`[`, save replace`]`)` asks `xtevent` to create the categorical treatment cohort variable based on values of the policy variable. `save` adds the new cohort variable to the dataset as *policyvar*`_cohort`. `replace` replaces the cohort variable if it already exists. The automatic creation of the cohort variable is available only in the staggered-adoption case.

`control_cohort(`*control_cohort_spec*[`,` *subopt*]`)` specifies how to identify the control cohort used for estimation of heterogeneous effects by cohort using the estimator from Sun and Abraham (2021). `control_cohort()` requires `cohort()` to be specified. `control_cohort()` is not allowed with `proxy()` or `proxyiv()`.

`control_cohort(variable` *varname*[`, force`]`)` specifies that the binary variable *varname* identify the control cohort. By default, `xtevent` checks for consistency of the control cohort variable and the policy variable. `force` forces `xtevent` to skip this check. This can be useful when estimating heterogeneous treatment effects across groups not defined by treatment cohorts.

`control_cohort(create`[`, save replace`]`)` asks `xtevent` to create the binary control cohort variable based on the missing values of the cohort variable. `save` adds the new control cohort variable to the dataset as *policyvar*`_control_cohort`. `replace` replaces the variable if it already exists. `control_cohort(create)` is the default if `cohort(create)` is specified but `control_cohort()` is not specified.

`sunabraham` is a shorthand to specify estimation with heterogeneous treatment effects by cohort using the estimator from Sun and Abraham (2021). `sunabraham` is equivalent to `cohort(create)` and `control_cohort(create)`.

3.1.6 Estimation with repeated cross-sectional data

`repeatedcs` indicates that the dataset in memory is repeated cross-sectionally. In this case, `panelvar()` should indicate the groups at which `policyvar()` changes. For instance, `panelvar()` could indicate states at which `policyvar()` changes, while the observations in the dataset are individuals in each state. An alternative method to estimate the event study in a repeated cross-sectional dataset involves using `get_unit_time_effects` first and then `xtevent`. See the description of the `get_unit_time_effects` command below. For fixed-effects estimation, `repeatedcs` enables `reghdfe`.

3.1.7 Stored results

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

Scalars	
`e(lwindow)`	left endpoint for estimation window
`e(rwindow)`	right endpoint for estimation window
Macros	
`e(cmd)`	estimation command: can be `regress`, `areg`, `ivregress`, `xtivreg`, or `reghdfe`
`e(cmd2)`	`xtevent`
`e(depvar)`	dependent variable
`e(names)`	names of the variables for the event-time dummies
`e(y1)`	mean of dependent variable at event time $= -1$
`e(x1)`	mean of proxy variable at event time $= -1$ when only one proxy is specified
`e(trend)`	`trend` if estimation included extrapolation of a linear trend
`e(trendmethod)`	method used to estimate the linear trend: can be `ols` or `gmm`
`e(df)`	degrees of freedom
`e(komit)`	list of lags or leads omitted from regression
`e(kmiss)`	list of lags or leads to omit in the plot
`e(ambiguous)`	list of cross-sectional units omitted because of ambiguous event times
`e(method)`	`ols` or `iv`
`e(pre)`	number of periods with anticipation effects
`e(post)`	number of periods with policy effects
`e(overidpre)`	number of periods to test for pretrends
`e(overidpost)`	number of periods to test for effects leveling off
`e(stub)`	prefix for saved event-time dummy variables

Matrices

`e(b)`	coefficient vector
`e(V)`	variance–covariance matrix
`e(delta)`	coefficient vector of event-time dummies
`e(Vdelta)`	variance–covariance matrix of the event-time dummy coefficients
`e(deltax)`	coefficients for proxy event study to be used in overlay plot
`e(deltaxsc)`	scaled coefficients for proxy event study to be used in overlay plot
`e(deltaov)`	coefficients for event study to be used in overlay plot
`e(Vdeltax)`	variance–covariance matrix of proxy event-study coefficients for overlay plot
`e(Vdeltaov)`	variance–covariance matrix of event-study coefficients for overlay plot
`e(mattrendy)`	matrix with y-axis values of trend for overlay plot; available only when `trend()` is specified
`e(mattrendx)`	matrix with x-axis values of trend for overlay plot; available only when `trend()` is specified
`e(b_ir)`	each column vector contains estimates of each cohort-relative-time interaction and controls included in the interaction regression; the interaction variables are named `_interact_m#_c#` or `_interact_p#_c#`, where `m#` indicates # periods before the policy change, `p#` indicates # periods after the policy change, and `c#` indicates the cohort; available only when `cohort()` and `control_cohort()` or `sunabraham` is specified
`e(V_ir)`	covariance matrix of the cohort-relative-time interactions and controls included in the interaction regression; the interaction variables are named `_interact_m#_c#` or `_interact_p#_c#`, where `m#` indicates # periods before the policy change, `p#` indicates # periods after the policy change, and `c#` indicates the cohort; available only when `cohort()` and `control_cohort()` or `sunabraham` is specified
`e(b_interact)`	each column vector contains estimates of the cohort-specific effect for the given relative time; available only when `cohort()` and `control_cohort()` or `sunabraham` is specified
`e(V_interact)`	each column vector contains variance estimate of the cohort-specific effect estimator for the given relative time; available only when `cohort()` and `control_cohort()` or `sunabraham` is specified
`e(ff_w)`	each column vector contains estimates of cohort shares underlying the given relative time; available only when `cohort()` and `control_cohort()` or `sunabraham` is specified
`e(Sigma_ff)`	variance estimate of the cohort share estimators; available only when `cohort()` and `control_cohort()` or `sunabraham` is specified

Functions

`e(sample)`	marks estimation sample

3.2 The xteventplot command

The `xteventplot` command produces event-study plots after `xtevent`. The syntax is the following:

```
xteventplot [, suptreps(integer) overlay(string) y proxy levels(numlist)
    smpath(type[, subopt]) overidpre(integer) overidpost(integer) noci nosupt
    nozeroline nonormlabel noprepval nopostpval scatterplotopts(string)
    ciplotopts(string) suptciplotopts(string) smplotopts(string)
    trendplotopts(string) staticovplotopts(string) addplots(string)
    textboxoption(string) additional_options]
```

3.2.1 Options

`suptreps(`*integer*`)` specifies the number of repetitions to calculate Montiel Olea and Plagborg-Møller (2019) sup-*t* confidence bands for the dynamic effects. The default is `suptreps(10000)`.

`overlay(`*string*`)` creates overlay plots for trend extrapolation, IV estimation in the presence of pretrends, and constant policy effects over time.

`overlay(trend)` overlays the event-time coefficients for the trajectory of the dependent variable and the extrapolated linear trend. `overlay(trend)` is available only after `xtevent` with option `trend(, saveoverlay)`.

`overlay(iv)` overlays the event-time coefficients trajectory of the dependent variable and the proxy variable used to infer the trend of the confounder. `overlay(iv)` is available only after `xtevent` with options `proxy()` and `proxyiv()`.

`overlay(static)` overlays the event-time coefficients from the fitted model and the coefficients implied by a constant policy effect over time. These coefficients are calculated by 1) fitting a model where the policy affects the outcome contemporaneously and its effect is constant, 2) obtaining predicted values of the outcome variable from this constant effects model, and 3) regressing the predicted values on event-time dummy variables.

`y` creates an event-study plot of the dependent variable in IV estimation. `y` is available only after `xtevent` with options `proxy()` and `proxyiv()`.

`proxy` creates an event-study plot of the proxy variable in IV estimation. `proxy` is available only after `xtevent` with options `proxy()` and `proxyiv()`.

`levels(`*numlist*`)` customizes the confidence level for the confidence intervals in the event-study plot. By default, `xteventplot` draws a standard confidence interval and a sup-*t* confidence band. `levels()` allows different confidence levels for standard confidence intervals. For example, `levels(90 95)` draws both 90% and 95% level confidence intervals along with a sup-*t* confidence band for Stata's default confidence level.

`smpath(`*type*[`,` *subopt*]`)` displays the "least wiggly" path through the Wald confidence region of the event-time coefficients. *type* determines the line type, which may be `scatter` or `line`. `smpath()` is not allowed with `noci`.

The following suboptions for `smpath()` control the optimization process. Because of the nature of the optimization problem, optimization error messages 4 and 5 (missing derivatives) or 8 (flat regions) may be frequent. Nevertheless, the approximate results from the optimization should be close to the results that would be obtained with convergence of the optimization process. Modifying these optimization suboptions may improve optimization behavior.

`postwindow(`*scalar*`)` sets the number of postevent coefficient estimates to use for calculating the smoothest line. The default is to use all the estimates in the postevent window. *scalar* must be greater than 0.

`maxiter(`*integer*`)` sets the maximum number of inner iterations for optimization. The default is `maxiter(100)`.

`maxorder(`*integer*`)` sets the maximum order for the polynomial smoothest line. *integer* must be between 1 and 10. The default is `maxorder(10)`.

`technique(`*string*`)` sets the optimization technique for the inner iterations of the quadratic program. `nr`, `bfgs`, `dfp`, and combinations are allowed. See [R] **Maximize**. The default is `technique(nr 5 bfgs)`.

`overidpre(`*integer*`)` changes the tested coefficients in the pretrends overidentification test. The default is to test all preevent coefficients. `overidpre()` tests whether the coefficients for the earliest *integer* periods before the event are equal to 0, including the endpoints. For example, with a window of 3, `overidpre(2)` tests that the coefficients for event times −4+ (the endpoint) and −3 are jointly equal to 0. *integer* must be greater than 0. See the `xteventtest` command below.

`overidpost(`*integer*`)` changes the coefficients to be tested for the leveling-off overidentification test. The default is to test that the rightmost coefficient and the previous one are equal. `overidpost()` tests whether the coefficients for the latest *integer* periods after the event are equal to each other, including the endpoints. For example, with a window of 3, `overidpost(3)` tests that the coefficients for event-times 4+ (the endpoint), 3, and 2 are equal to each other. *integer* must be greater than 1. See the `xteventtest` command below.

The following options control the appearance of the plot:

`noci` omits the display and calculation of both Wald and sup-*t* confidence band. `noci` overrides `suptreps()` if it is specified. `noci` is not allowed with `smpath()`.

`nosupt` omits the display and calculation of sup-*t* confidence bands. `nosupt` overrides `suptreps()` if it is specified.

`nozeroline` omits the display of the reference line at 0. Note that reference lines with different styles can be obtained by removing the default line with `nozeroline` and adding other lines with `yline()`; see [G-3] ***added_line_options***.

`nonormlabel` suppresses the vertical-axis label for the mean of the dependent variable at event time corresponding to the normalized coefficient.

`noprepval` omits the display of the *p*-value for a test for pretrends. By default, this is a Wald test for all the preevent coefficients being equal to 0, unless `overidpre()` is specified.

`nopostpval` omits the display of the *p*-value for a test for effects leveling off. By default, this is a Wald test for the last postevent coefficients being equal unless `overidpost()` is specified.

`scatterplotopts(`*string*`)` specifies options to be passed to `scatter` for the coefficients' plot.

`ciplotopts(`*string*`)` specifies options to be passed to `rcap` for the confidence interval's plot. These options are disabled if `noci` is specified.

`suptciplotopts(`*string*`)` specifies options to be passed to `rcap` for the sup-*t* confidence band plot. These options are disabled if `nosupt` is specified.

`smplotopts(`*string*`)` specifies options to be passed to `line` for the smoothest path through the confidence region plot. These options are active only if `smpath()` is specified.

`trendplotopts(`*string*`)` specifies options to be passed to `line` for the extrapolated trend overlay plot. These options are active only if `overlay(trend)` is specified.

`staticovplotopts(`*string*`)` specifies options to be passed to `line` for the static effect overlay plot. These options are active only if `overlay(static)` is specified.

`addplots(`*string*`)` specifies additional plots to overlay to the event-study plot.

`textboxoption(`*string*`)` specifies options to pass to the textboxes of the pretrend and leveling-off tests. These options are disabled if `noprepval` and `nopostpval` are specified. See [G-3] ***textbox_options***.

additional_options are additional options to be passed to `twoway`. See [G-2] **graph twoway**.

3.3 The xteventtest command

The `xteventtest` command performs hypothesis testing after the `xtevent` command. The syntax is the following:

`xteventtest` [, `coefs(`*numlist*`)` `cumul` `allpre` `allpost` `linpretrend` `trend(`*#*`)` `constanteff` `overid` `overidpre(`*integer*`)` `overidpost(`*integer*`)` `testopts(`*string*`)`]

3.3.1 Options

`coefs(`*numlist*`)` specifies a numeric list of event times to be tested. These are tested to be equal to 0 jointly unless otherwise requested in `testopts()`.

`cumul` requests a test of equality to 0 for the sum of every coefficient for each event time in `coefs()`.

`allpre` tests that all preevent coefficients are equal to 0. With `cumul`, it tests that the sum of all preevent coefficients is equal to 0.

`allpost` tests that all postevent coefficients are equal to 0. With `cumul`, it tests that the sum of all postevent coefficients is equal to 0.

`linpretrend` requests a specification test to see whether the coefficients follow a linear trend before the event.

`trend(`*#*`)` tests for a linear trend from time period # before the policy change. It uses `xtevent` with `trend(`*#*`, method(ols))` to estimate the trend. # must be less than -1.

`constanteff` tests that all postevent coefficients are equal.

`overid` tests overidentifying restrictions: a test for pretrends and a test for effects leveling off. The periods to be tested are those used in the `xtevent` call.

`overidpre(`*integer*`)` tests the pretrends overidentifying restriction. It tests that the coefficients for the earliest *integer* periods before the event are equal to 0, including the endpoints. For example, with a window of 3, `overidpre(2)` tests that the coefficients for event-times $-4+$ (the endpoint) and -3 are jointly equal to 0. *integer* must be greater than 0.

`overidpost(`*integer*`)` tests the effects leveling-off overidentifying restriction. It tests that the coefficients for the latest *integer* periods after the event are equal, including the endpoints. For example, with a window of 3, `overidpost(3)` tests that the coefficients for event-times 4+ (the endpoint), 3, and 2 are equal to each other. *integer* must be greater than 1.

`testopts(`*string*`)` specifies options to be passed to `test`; see [R] **test**.

3.3.2 Stored results

`xteventtest` stores the following in `r()`:

Scalars	
`r(p)`	two-sided p-value
`r(F)`	F statistic
`r(df)`	test constraints degrees of freedom
`r(df_r)`	residual degrees of freedom
`r(dropped_i)`	index of ith constraint dropped
`r(chi2)`	χ^2
`r(drop)`	`1` if constraints were dropped, `0` otherwise
Macros	
`r(mtmethod)`	method of adjustment for multiple testing; this macro is inherited from `test`
Matrices	
`r(mtest)`	multiple test results; this matrix is inherited from `test`

3.4 The get_unit_time_effects command

The `get_unit_time_effects` command generates group and time effects in a repeated cross-sectional dataset. It produces a dataset with the variables *panelvar*, *timevar*, and `_unittimeeffects`. The variable `_unittimeeffects` contains the group-time effects. Hansen (2007) describes a two-step procedure to obtain the coefficient estimates of covariates that vary at the group level within a repeated cross-sectional framework. The two-step procedure can be used to obtain the coefficient estimates of an event-study when the data are repeated cross-sectionally. `get_unit_time_effects` implements the first part of the two-step procedure. Then `xtevent` can be used for the second part of the procedure to obtain the event-study coefficient estimates. See appendix E in the online supplementary material. The syntax of the `get_unit_time_effects` command is the following:

```
get_unit_time_effects depvar [indepvars] [if] [in] [weight] ,
   panelvar(panelvar) timevar(timevar) [saving(filename[, replace])
   nooutput clear]
```

`aweights`, `fweights`, and `pweights` are allowed; see [U] **11.1.6 weight**.

3.4.1 Options

`panelvar(`*panelvar*`)` specifies the group variable. The policy variable should vary at this group level. `panelvar()` is required.

`timevar(`*timevar*`)` specifies the time variable. `timevar()` is required.

`saving(`*filename*[`, replace`]`)` specifies the name of the dataset to store the unit-time effects estimates. If `saving()` is not specified, the file is saved in the current directory with the name `unit_time_effects.dta`. The suboption `replace` overwrites the unit-time effects file.

`nooutput` omits the regression table.

`clear` replaces the dataset in memory with the unit-time effects file.

4 Examples

This section provides two examples of `xtevent` usage. First, we display the basic functionality of the package using simulated data from Freyaldenhoven et al. (Forthcoming). Then, we show additional options using real data from Martínez (2022a).

4.1 Simulated data example

We first use the "Jump at the time of the event" data from Freyaldenhoven et al. (2021). The data are a balanced panel of 2,000 observations from 50 units observed over 40 periods, where the policy initially affects the units at different periods. Units are randomly treated in the sample period. The coefficient on the treatment variable is 1.

We start by loading the dataset and specifying the unit and time variables.

```
. use simulation_data_dynamic
. xtset id t
Panel variable: id (strongly balanced)
 Time variable: t, 1 to 40
         Delta: 1 unit
```

Below, we show a glimpse of the dataset. The variable `id` indexes the cross-sectional units, `t` indexes time, `z` is the policy variable, `y` is the outcome variable, and `x` is a control variable.

```
. list id t z y x if id==2 & t<=10, noobs

  ┌─────────────────────────────────────┐
  │ id    t   z          y            x │
  ├─────────────────────────────────────┤
  │  2    1   0   42.02239   -.0102958 │
  │  2    2   0   42.83109    .1713373 │
  │  2    3   0   42.82665    -.197851 │
  │  2    4   1   42.59289   -.5887834 │
  │  2    5   1    43.3557   -.3722385 │
  ├─────────────────────────────────────┤
  │  2    6   1   42.74355     .355894 │
  │  2    7   1   42.82405   -2.047098 │
  │  2    8   1   42.34953   -.3757658 │
  │  2    9   1   42.14841   -1.976451 │
  │  2   10   1   41.79388    -1.16444 │
  └─────────────────────────────────────┘
```

Notice that the time variable `t` is calendar time, not event time. `xtevent` does not require normalization of the time variable, because the specification in (2) allows discrete or continuous policy variables and single or multiple changes.

Basic functionality

We estimate (2) setting $M + L_M = 5$ and $G + L_G = 5$, so we are looking at the effects of the policy on the outcome five periods before and five periods after policy adoption. To estimate a basic panel event study with dynamic effects of policy variable `z` on `y` using `x` as control and plot the results, we write

```
. xtevent y x, panelvar(id) timevar(t) policyvar(z) window(5) impute(nuchange)
> plot
No proxy or instruments provided. Implementing OLS estimator
Linear regression, absorbing indicators           Number of obs     =  2,000
Absorbed variable: id                             No. of categories =     50
                                                  F(52, 1898)       =   2.81
                                                  Prob > F          = 0.0000
                                                  R-squared         = 0.0980
                                                  Adj R-squared     = 0.0500
                                                  Root MSE          = 0.7181
------------------------------------------------------------------------------
           y | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
    _k_eq_m6 |   .2008833   .1138905     1.76   0.078    -.0224803     .424247
    _k_eq_m5 |   .2935611   .1499538     1.96   0.050    -.0005306    .5876528
    _k_eq_m4 |   .1172962   .1490125     0.79   0.431    -.1749493    .4095416
    _k_eq_m3 |   .0874992   .1472188     0.59   0.552    -.2012285    .3762268
    _k_eq_m2 |   .0415972   .1472521     0.28   0.778    -.2471958    .3303902
    _k_eq_p0 |   .8160009   .1472673     5.54   0.000     .5271782    1.104824
    _k_eq_p1 |   .8400974   .1475249     5.69   0.000     .5507695    1.129425
    _k_eq_p2 |   .5699733   .1497108     3.81   0.000     .2763583    .8635883
    _k_eq_p3 |   .2665183   .1507838     1.77   0.077    -.0292011    .5622377
    _k_eq_p4 |   .0915933   .1547402     0.59   0.554    -.2118855    .3950721
    _k_eq_p5 |    .091785   .1564831     0.59   0.558     -.215112     .398682
    _k_eq_p6 |   .1405872   .1192801     1.18   0.239    -.0933466     .374521
           x |   .0076964   .0302145     0.25   0.799    -.0515607    .0669535
             |
           t |
  (output omitted)
             |
       _cons |   41.91144   .1507874   277.95   0.000     41.61572    42.20717
------------------------------------------------------------------------------
F test of absorbed indicators: F(49, 1898) = 1.082            Prob > F = 0.325
```

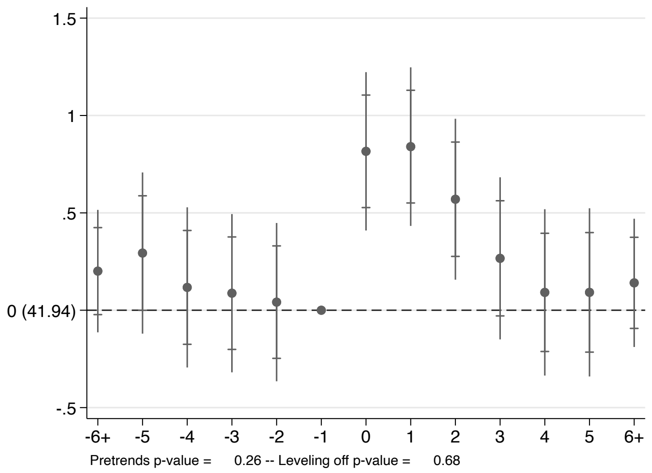


Figure 1. Event-study plot

The panelvar() and timevar() options indicate the cross-sectional and time dimensions of the dataset. The window() option specifies the event-time periods to include. By specifying the impute(nuchange) option, we ask xtevent to assume that the policy does not change outside the estimation window and to impute the leads and lags of the policy variable accordingly. We discuss alternative imputation schemes in appendix D in the online supplementary material.

xtevent automatically creates event-time dummies for event times -5 to 5, denoted by _k_eq_m5,_k_eq_m4,...,_k_eq_p5, plus endpoint dummies for event times ≤ -6 and ≥ 6 (_k_eq_m6,_k_eq_p6), and includes them as independent variables. The normalized coefficient for event time $= -1$ is omitted. By default, xtevent includes unit and time fixed effects in the regression, but these can be excluded using the note and nofe options. In this case, xtevent used areg to estimate the regression, so the unit fixed effects are not reported. The default output includes conventional standard errors.

The plot option requests an event-study plot after estimation, shown in figure 1. The plot can also be obtained by writing xteventplot after the xtevent call. The default plot shows the values of the estimated coefficients along with pointwise confidence intervals (inner whiskers) and sup-t confidence bands for the entire event-time path (outer spikes). By default, xtevent normalizes $\delta_{-1} = 0$, and the y axis includes a parenthetical label indicating the mean of the dependent variable one period before adoption. At the bottom, the graph includes p-values for a pretrends test and a leveling-off test.

With the reghdfe option, the user can ask xtevent to fit the model using the community-contributed command reghdfe from Correia (2016b). The user can also specify additional variables to be absorbed through the option addabsorb(). To ask xtevent to estimate with reghdfe and additionally absorb the variable eta_r2, we write

```
. generate eta_r2=round((eta_r+1)*2)
. quietly xtevent y x, panelvar(id) timevar(t) policyvar(z) window(5)
> reghdfe addabsorb(eta_r2) impute(nuchange)
```

Choosing different windows

We can also specify an asymmetric window. For instance, for an estimation of four preevent periods, seven postevent periods, and two endpoints, we write

```
. quietly xtevent y x, panelvar(id) timevar(t) policyvar(z) window(-4 7)
> impute(nuchange)
```

With the notation from (2), this estimation corresponds to setting G, the number of preevent periods where anticipated effects can occur, to 0 and setting L_G, the number of preevent periods to use for visualizing preevent effects, to 4. Analogously, it sets M, the number of postevent periods for lagged effects, to 7 and sets L_M, the number of periods to test whether effects are leveling off, to 1. This is equivalent to explicitly specifying the values of G, L_G, M, and L_M:

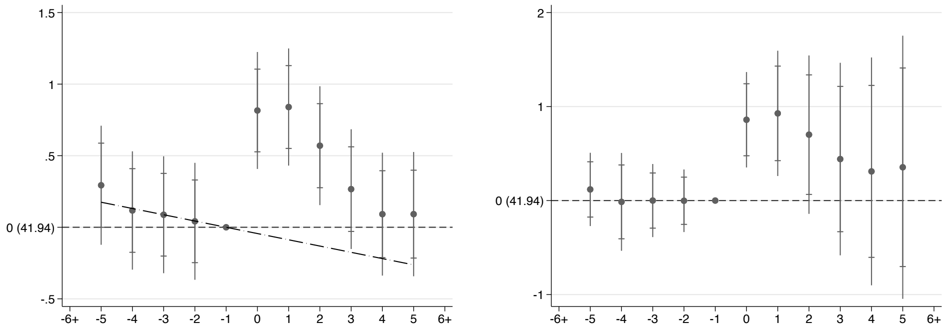
```
. quietly xtevent y x, panelvar(id) timevar(t) policyvar(z) pre(0) overidpre(4)
> post(7) overidpost(1) impute(nuchange)
```

Linear trend adjustment

The option `trend(`*#* $\left[\right.$`,` *subopt*$\left.\right]$`)` allows extrapolation of a linear trend in event-time from period *#* before the policy change, as in Dobkin et al. (2018). The estimated effect of the policy is the deviation from the extrapolated linear trend. We estimate the linear event-time trend using three preevent periods $(-3, -2, -1)$ and using the `method(gmm)` suboption to use a GMM estimator. We also save the overlay data for plotting:

```
. xtevent y x, panelvar(id) timevar(t) policyvar(z) window(5)
> impute(nuchange) trend(-3, method(gmm) saveoverlay)
No proxy or instruments provided. Implementing OLS estimator
Linear regression, absorbing indicators            Number of obs     =  2,000
Absorbed variable: id                              No. of categories =     50
                                                   F(52, 1898)       =   2.81
                                                   Prob > F          = 0.0000
                                                   R-squared         = 0.0980
                                                   Adj R-squared     = 0.0500
                                                   Root MSE          = 0.7181

------------------------------------------------------------------------------
           y | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
    _k_eq_m6 |  -.0178682   .3123965    -0.06   0.954    -.6305449    .5948085
    _k_eq_m5 |   .1185599   .1499569     0.79   0.429    -.1755379    .4126576
    _k_eq_m4 |  -.0139547   .2002518    -0.07   0.944    -.4066915     .378782
    _k_eq_m3 |  -1.45e-06   .1496187    -0.00   1.000    -.2934359     .293433
    _k_eq_m2 |  -.0021531   .1281336    -0.02   0.987    -.2534507    .2491444
    _k_eq_p0 |   .8597512   .1953144     4.40   0.000     .4766977    1.242805
    _k_eq_p1 |    .927598    .256436     3.62   0.000     .4246719    1.430524
    _k_eq_p2 |   .7012242   .3242324     2.16   0.031      .065335    1.337113
    _k_eq_p3 |   .4415195   .3940669     1.12   0.263    -.3313303    1.214369
    _k_eq_p4 |   .3103449   .4663886     0.67   0.506    -.6043433    1.225033
    _k_eq_p5 |   .3542868   .5386234     0.66   0.511    -.7020694    1.410643
    _k_eq_p6 |   .4468393   .6017334     0.74   0.458     -.733289    1.626968
           x |   .0076964   .0302145     0.25   0.799    -.0515607    .0669535
             |
           t |
          2  |    .090792   .1439829     0.63   0.528    -.1915894    .3731734
  (output omitted)
------------------------------------------------------------------------------
```

To visualize the original estimates, the extrapolated trend, and the linear-trend-adjusted estimates, `xtevent` can produce an overlay plot: we use `xteventplot` to produce an overlay plot with the extrapolated trend and an adjusted plot. In both figures, `xtevent` excludes the endpoints. We show the overlay and adjusted event-study plots in figure 2.



(a) Overlay extrapolation line (b) Subtract extrapolated trend

Figure 2. Linear trend adjustment

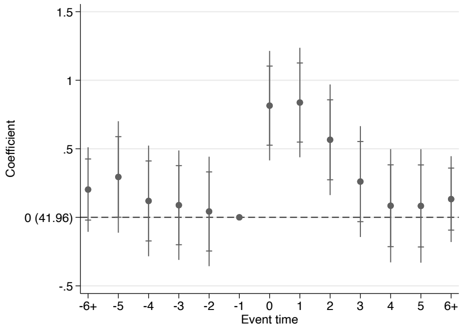
Pretrends adjustment with proxy variables

`xtevent` allows estimation when there is a pretrend present using the IV estimator of Freyaldenhoven, Hansen, and Shapiro (2019). For this, we need to specify the option `proxy()` to indicate the proxy variable or variables for the confound. As a default, `xtevent` regresses the proxy variable on leads of the differenced policy variable and chooses the lead with the highest absolute t statistic to use as an instrument for the proxy variable. Alternatively, the user can specify a specific lead or different variables to be used as instruments in the `proxyiv()` option. `xtevent` uses `xtivreg` to fit this model.

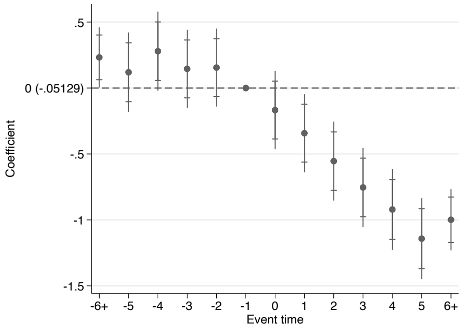
```
. xtevent y, panelvar(id) timevar(t) policyvar(z) window(5) impute(nuchange)
> proxy(x)
Proxy for the confound specified. Implementing FHS estimator
proxyiv=select. Selecting lead order of differenced policy variable to use as
> instrument.
Lead 4 selected.
The coefficient at -1 is normalized to zero.
For estimation with proxy variables, an additional coefficient needs to be
> normalized to zero.
The coefficient at -4 was selected to be normalized to zero.
Fixed-effects (within) IV regression           Number of obs     =      1,800
Group variable: id                             Number of groups  =         50
R-squared:                                     Obs per group:
     Within  =      .                                        min =         36
     Between = 0.0557                                        avg =       36.0
     Overall = 0.0335                                        max =         36
                                               Wald chi2(47)     =   5.83e+06
corr(u_i, Xb) = -0.0155                        Prob > chi2       =     0.0000

------------------------------------------------------------------------------
           y | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
           x |   .3772094   .5069244     0.74   0.457    -.6163441    1.370763
    _k_eq_m6 |   .1009198   .1046026     0.96   0.335    -.1040974    .3059371
    _k_eq_m5 |   .2344818   .1356115     1.73   0.084    -.0313119    .5002755
    _k_eq_m3 |   .0187142   .1328999     0.14   0.888    -.2417649    .2791932
    _k_eq_m2 |   .0189579     .13511     0.14   0.888    -.2458528    .2837686
    _k_eq_p0 |   .8122546   .1805533     4.50   0.000     .4583766    1.166133
    _k_eq_p1 |   .9546866   .2887189     3.31   0.001     .3888079    1.520565
    _k_eq_p2 |   .7687619   .3744836     2.05   0.040     .0347875    1.502736
    _k_eq_p3 |   .4715529   .4518123     1.04   0.297     -.413983    1.357089
  (output omitted)
          32 |  -.1106802   .1948427    -0.57   0.570     -.492565    .2712045
          33 |  -.0372759   .2005944    -0.19   0.853    -.4304338    .3558819
          34 |  -.0579759   .1700988    -0.34   0.733    -.3913634    .2754116
          35 |  -.1346377   .1744911    -0.77   0.440     -.476634    .2073585
          36 |   .2059306   .1766346     1.17   0.244    -.1402669    .5521281
             |
       _cons |   41.96931   .1359876   308.63   0.000     41.70278    42.23584
-------------+----------------------------------------------------------------
     sigma_u |  .12123774
     sigma_e |  .74115015
         rho |  .02606123   (fraction of variance due to u_i)
------------------------------------------------------------------------------
 F test that all u_i=0: F(49,1703) =     0.88             Prob > F    = 0.7137
------------------------------------------------------------------------------
Endogenous: x
Exogenous:  _k_eq_m6 _k_eq_m5 _k_eq_m3 _k_eq_m2 _k_eq_p0 _k_eq_p1 _k_eq_p2
            _k_eq_p3 _k_eq_p4 _k_eq_p5 _k_eq_p6 2.t 3.t 4.t 5.t 6.t 7.t 8.t
            9.t 10.t 11.t 12.t 13.t 14.t 15.t 16.t 17.t 18.t 19.t 20.t 21.t
            22.t 23.t 24.t 25.t 26.t 27.t 28.t 29.t 30.t 31.t 32.t 33.t 34.t
            35.t 36.t _fd4__00000M
```

`xteventplot` can create several plots to illustrate this estimator. First, we use `xteventplot, y` to create an unadjusted event-study plot for the outcome. This plot is shown in figure 3 panel a. Second, using the option `proxy()`, we illustrate the dynamics of the proxy by creating an event-study plot for the proxy (shown in figure 3 panel b). Third, using the option `overlay(iv)`, we create a plot that aligns the dynamics of the proxy and the outcome between the coefficient used as an instrument and the normalized coefficient (shown in figure 3 panel c). Finally, using `xteventplot` and no additional options, we create a plot that shows the coefficients of the outcome after subtracting the rescaled event-study coefficients for the proxy (shown in figure 3 panel d).

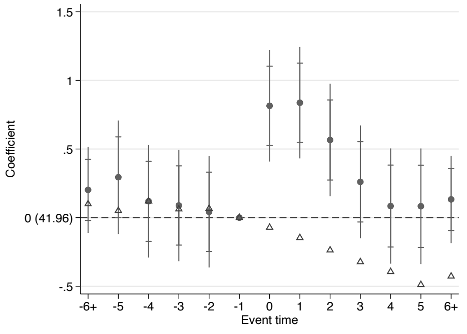
```
. xteventplot, y ytitle("Coefficient") xtitle("Event time")
. xteventplot, proxy ytitle("Coefficient") xtitle("Event time")
. xteventplot, overlay(iv) ytitle("Coefficient") xtitle("Event time")
. xteventplot, ytitle("Coefficient") xtitle("Event time")
```



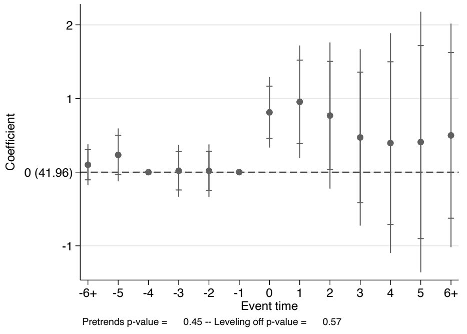
(a) Event-study plot for outcome



(b) Event-study plot for proxy



(c) Align proxy to outcome



(d) Subtract rescaled confound from outcome

Figure 3. Pretrends adjustment using proxy variables for the confound following Freyaldenhoven, Hansen, and Shapiro (2019)

Estimation with heterogeneous effects by cohort

`xtevent` allows estimation in settings with heterogeneous effects that vary by cohort using Sun and Abraham's (2021) estimator through the `cohort()` and `control_cohort()` options. In the `cohort()` option, the user must specify the variable that identifies the cohorts, and in the `control_cohort()` option, the user must specify the variable that identifies the cohort to use as the control group. In the following example, we first create the `time_of_treat` variable to identify the cohorts and then create the variable `last_treat` to indicate the control group, which in this case are the last-treated units.

```
. generate timet = t if z == 1
(1,059 missing values generated)
. by id: egen time_of_treat = min(timet)
. generate last_treat = time_of_treat == 39
. replace time_of_treat = . if last_treat
(80 real changes made, 80 to missing)
. xtevent y, panelvar(id) timevar(t) policyvar(z) window(5) impute(nuchange)
> cohort(variable time_of_treat) control_cohort(variable last_treat)
No proxy or instruments provided. Implementing OLS estimator
You have specified the cohort or the sunabraham option
Event-time coefficients will be estimated with the Interaction Weighted
> Estimator of Sun and Abraham (2021)
Linear regression, absorbing indicators            Number of obs     =  2,000
Absorbed variable: id                              No. of categories =     50
                                                   F(329, 1621)      =   1.33
                                                   Prob > F          = 0.0003
                                                   R-squared         = 0.2347
                                                   Adj R-squared     = 0.0562
                                                   Root MSE          = 0.7158

------------------------------------------------------------------------------
           y | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
     _k_eq_m6 |   .1688391   .1394638     1.21   0.226    -.1047091    .4423872
     _k_eq_m5 |   .3559271   .2014006     1.77   0.077    -.0391058    .7509601
     _k_eq_m4 |   .1412944   .1802456     0.78   0.433    -.2122445    .4948332
     _k_eq_m3 |   .1332897   .1908158     0.70   0.485    -.2409819    .5075613
     _k_eq_m2 |   .1268691   .1961503     0.65   0.518    -.2578657    .5116038
     _k_eq_p0 |   .8289953   .1805352     4.59   0.000     .4748883    1.183102
     _k_eq_p1 |   .8549653   .1870247     4.57   0.000     .4881298    1.221801
     _k_eq_p2 |   .6015642   .1858662     3.24   0.001      .237001    .9661274
     _k_eq_p3 |    .287782   .1883399     1.53   0.127    -.0816333    .6571973
     _k_eq_p4 |    .090265    .185755     0.49   0.627    -.2740801      .45461
     _k_eq_p5 |    .090156   .2027774     0.44   0.657    -.3075774    .4878894
     _k_eq_p6 |   .2033672   .1394957     1.46   0.145    -.0702436    .4769781
------------------------------------------------------------------------------
```

The output is analogous to standard `xtevent` output. The estimated event-time path corresponds to the weighted average of event-study estimates comparing each treatment cohort with the control cohort. `xtevent` stores the estimates by cohort in the matrices `e(b_interact)` and `e(V_interact)`.

Least wiggly path of confounds consistent with the estimates

`xteventplot` allows for estimation and display of the least wiggly path of confound consistent with the estimates discussed in section 2 through the `smpath()` option. The default plottype is `line`. With the additional suboptions `maxorder()`, `maxiter()`, and `technique()`, we can control the maximum polynomial order for the confound path and the optimization process.

```
. quietly xtevent y x , panelvar(id) timevar(t) policyvar(z) window(5)
> impute(nuchange)
. xteventplot, ytitle("Coefficient") xtitle("Event time")
> smpath(line, maxorder(9) maxiter(200) technique(nr 10 dfp 10))
Note: Smoothest line drawn for system confidence level = 95%
Wald Critical Value 21.0260698
Order 0
Wald value 99.1427915
Order 1
Wald value 99.1021008
Order 2
Wald value 77.664952
Order 3
Wald value 64.4768055
Order 4
Wald value 50.446398
Order 5
Wald value 26.7712056
Order 6
Wald value 19.9659905
(setting technique to nr)
Iteration 0:  f(p) =  42803.512  (not concave)
  (output omitted)
```

In this case, the minimum polynomial order required to pass through the confidence region is of order 6. Figure 4 shows the resulting smoothest path. If all the dynamics of the estimated event-time path of the outcome variable were due to this unobserved confound, the jump at the time of the event implies that the confound would also have to jump at the time of the event. Such a confound path suggests that the estimated effects may be due to an effect of the policy and not to a confound.

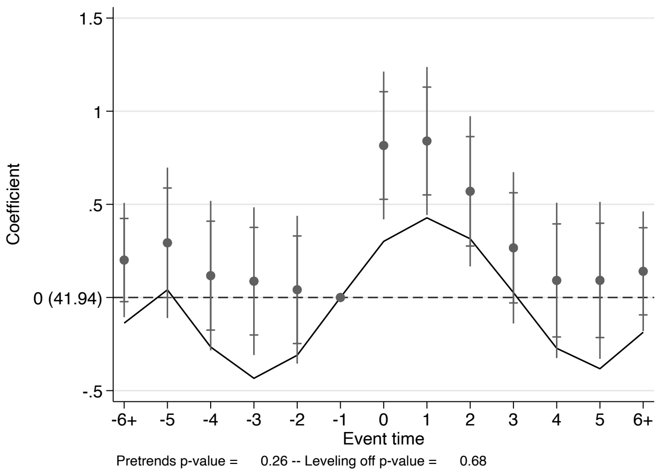


Figure 4. Least wiggly path through the confidence region

4.2 Empirical application

Martínez (2022a) analyzes the effect of a tax reform in the Swiss canton of Obwalden in 2006. The reform modified the income tax schedule, reducing the tax rate for high-income taxpayers. We focus on the reform's effect on Obwalden's tax revenue from wealth taxes, following figure 9 in Martínez (2022a). We use the data from Martínez (2022b).

The data are a balanced panel of 702 observations from 26 cantons from 1990 to 2016. Only one canton, Obwalden, is treated. We estimate a version of (2):

$$y_{it} = \sum_{k=-5, k\neq -1}^{9} \delta_k \Delta z_{i,t-k} + \delta_{10} z_{i,t-10} + \delta_{-6}(1 - z_{i,t+6}) + \alpha_i + \gamma_t + \varepsilon_{it} \qquad (5)$$

Here the outcome y_{it} is per-capita revenue from wealth taxes in canton i and year t, normalized to 100 in 2005. The policy variable z_{it} is 1 for Obwalden in 2006 or after and 0 otherwise. The δ parameters are estimates of the cumulative effect of the reform at various horizons. We normalize $\delta_{-1} = 0$. The parameters α_i and γ_t represent canton and year fixed effects, respectively, and ε_{it} is an error term.

We start by loading the dataset, specifying the unit and time variables, and displaying some values of the dependent and policy variables around the treatment year:

```
. use martinez, clear

. xtset cant year

Panel variable: cant (strongly balanced)
 Time variable: year, 1990 to 2016
         Delta: 1 unit

. list cant year pcrev_weatax policyvar if cant==6 & inrange(year,2003,2009),
> abbreviate(19) noobs separator(7)

  ┌─────────────────────────────────────────┐
  │ cant   year   pcrev_weatax   policyvar  │
  ├─────────────────────────────────────────┤
  │   OW   2003       98.06313           0  │
  │   OW   2004       99.06337           0  │
  │   OW   2005            100           0  │
  │   OW   2006       92.90456           1  │
  │   OW   2007       87.39193           1  │
  │   OW   2008        64.9511           1  │
  │   OW   2009       73.20993           1  │
  └─────────────────────────────────────────┘
```

We now estimate (5) to capture the dynamic effects of the reform on tax revenues. To adjust for pretrends, we use a linear trend adjustment based on the five immediate preevent periods. The unit and time variables do not need to be specified because the data were `xtset` previously. With the option `reghdfe`, we can estimate this equation using the `reghdfe` command of Correia (2016b). Using `reghdfe` enables two-way clustered standard errors through the `vce()` option. We also specify the imputation rule `impute(stag)` and add weights.

```
. xtevent pcrev_weatax [aweight = weight_pcrev_weatax], panelvar(cant)
> timevar(year) policyvar(policyvar) window(-5 9) impute(stag) reghdfe
> vce(cluster cant) trend(-5, method(ols))
No proxy or instruments provided. Implementing OLS estimator
(MWFE estimator converged in 3 iterations)
HDFE Linear regression                            Number of obs   =        702
Absorbing 2 HDFE groups                           F(  13,     25) =      98.29
Statistics robust to heteroskedasticity           Prob > F        =     0.0000
                                                  R-squared       =     0.8426
                                                  Adj R-squared   =     0.8267
                                                  Within R-sq.    =     0.0043
Number of clusters (cant)    =         26         Root MSE        =    29.9937
                                  (Std. err. adjusted for 26 clusters in cant)
------------------------------------------------------------------------------
             |               Robust
pcrev_weatax | Coefficient  std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
    _k_eq_m6 |   19.42696   9.159563     2.12   0.044     .5624837    38.29143
    _k_eq_p0 |    -21.213    5.02208    -4.22   0.000    -31.55617   -10.86983
    _k_eq_p1 |  -40.46391   7.878604    -5.14   0.000     -56.6902   -24.23762
    _k_eq_p2 |  -69.30378   11.09302    -6.25   0.000    -92.15027   -46.45728
    _k_eq_p3 |  -65.53987   15.00997    -4.37   0.000    -96.45348   -34.62627
    _k_eq_p4 |  -55.14003   17.70663    -3.11   0.005    -91.60752   -18.67253
    _k_eq_p5 |  -57.77175   21.48921    -2.69   0.013    -102.0296   -13.51388
    _k_eq_p6 |  -51.15706   21.70713    -2.36   0.027    -95.86374   -6.450375
    _k_eq_p7 |  -43.37057   24.54034    -1.77   0.089    -93.91234     7.17121
    _k_eq_p8 |  -54.18095   29.12688    -1.86   0.075    -114.1689    5.806996
    _k_eq_p9 |  -53.77577   29.91591    -1.80   0.084    -115.3887      7.8372
   _k_eq_p10 |  -44.05725   14.69578    -3.00   0.006    -74.32377   -13.79073
     _ttrend |   2.528577    2.67721     0.94   0.354    -2.985241    8.042394
       _cons |   123.7051    9.13919    13.54   0.000     104.8826    142.5276
------------------------------------------------------------------------------
Absorbed degrees of freedom:
-----------------------------------------------------+
 Absorbed FE | Categories  - Redundant  = Num. Coefs |
-------------+---------------------------------------|
        cant |        26          26           0    *|
        year |        27           1          26     |
-----------------------------------------------------+
* = FE nested within cluster; treated as redundant for DoF computation
```

We use `xteventplot` to display an event-study plot. `xtevent` has several options to modify the plot's appearance. We modify the plot to suppress the sup-t confidence bands and the p-values from overidentification tests. We also change the colors and add axis titles.

```
. xteventplot, ytitle("Coefficient") xtitle("Event time")
> nosupt noprepval nopostpval
> scatterplotopts(lcolor(gray) recast(connected) mcolor(gray) msymbol(circle))
> ciplotopts(recast(rarea) fcolor(gray*0.2))
> graphregion(fcolor(white))
option nosupt has been specified. Sup-t confidence bands won't be
> displayed or calculated
option noprepval has been specified. The p-value for a pretrends test
> won't be displayed
option nopostpval has been specified. The p-value for a test of
> effects leveling-off won't be displayed
```

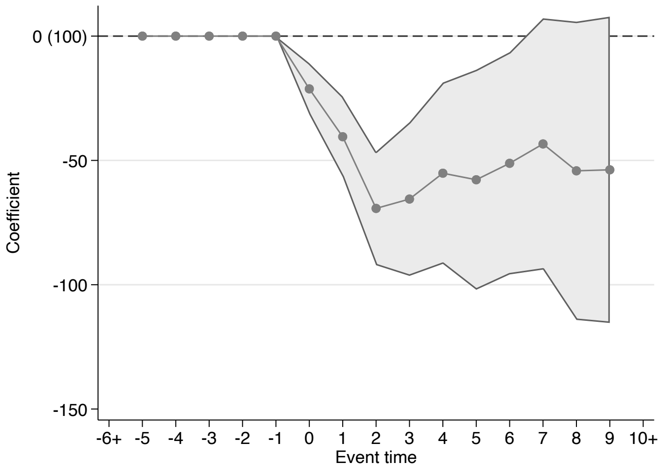


Figure 5. Dynamic effect of a Swiss tax reform following Martínez (2022a)

Figure 5 indicates that the introduction of the regressive tax reform in Obwalden decreased its government's tax revenues (measured per capita and relative to the level in 2005). The most substantial decrease was 69% and came two years after the reform.

Hypothesis tests

We can use `xteventtest` to test for different hypotheses about the event-study coefficients after `xtevent`. For instance, we repeat the estimation with standard errors clustered by canton and use the `coefs()` option to test whether the coefficients for event times 0, 1, 2, and 3 are equal to 0 jointly.

```
. xteventtest, coefs(0 1 2 3)
 ( 1)  _k_eq_p0 = 0
 ( 2)  _k_eq_p1 = 0
 ( 3)  _k_eq_p2 = 0
 ( 4)  _k_eq_p3 = 0
       F(  4,    25) =   22.82
            Prob > F =    0.0000
```

The test indicates that the effects are different from zero. We can also test whether the estimated policy effects are constant across time:

```
. xteventtest, constanteff
Test for constant post-event coefficients
 ( 1)  _k_eq_p0 - _k_eq_p1 = 0
 ( 2)  _k_eq_p0 - _k_eq_p2 = 0
 ( 3)  _k_eq_p0 - _k_eq_p3 = 0
 ( 4)  _k_eq_p0 - _k_eq_p4 = 0
 ( 5)  _k_eq_p0 - _k_eq_p5 = 0
 ( 6)  _k_eq_p0 - _k_eq_p6 = 0
 ( 7)  _k_eq_p0 - _k_eq_p7 = 0
 ( 8)  _k_eq_p0 - _k_eq_p8 = 0
 ( 9)  _k_eq_p0 - _k_eq_p9 = 0
       F(  9,    25) =   73.46
            Prob > F =    0.0000
```

This test suggests that the effects are not constant over time. Last, we conduct an overidentification test to see whether the effects level off. To do this, we ask `xteventtest` to use the last two postevent coefficients.

```
. xteventtest, overidpost(2)
Overidentification test for effects leveling off: 2 last post-event
> coefficients are equal
 ( 1)  - _k_eq_p8 + _k_eq_p9 = 0
       F(  1,    25) =    0.01
            Prob > F =    0.9086
```

This test suggests that the effects level off and that the number of postevent periods in the model may be sufficient to capture the dynamic effects of the policy.

5 Conclusions

This article introduced the `xtevent` command, which fits linear panel models and visualizes the results in event-study plots. The package allows for estimation and plotting with general policy variables, accommodating settings such as continuous policy variables and reversible treatments.

The versatility of the package and the estimation approaches we suggest rely on the assumptions behind the structure of (1), such as the fact that the policy variable has homogeneous linear effects that are separable from those of the confound. There are other approaches in the literature, such as Callaway, Goodman-Bacon, and Sant'Anna (2024), to estimate treatment effects in settings with continuous treatments and where (1) may not hold. The development of tools to implement these approaches and to estimate and visualize policy effects in general settings while relaxing parametric assumptions like those in (1) is a potential area for future research.

6 Acknowledgments

We thank our dedicated research assistants for their contributions to this project. We also thank the editor and one anonymous referee for their comments. We are grateful to the many `xtevent` users that have provided feedback. We acknowledge financial support from the SUPER RA Program at Harvard (Shapiro). The views expressed herein are those of the authors and do not necessarily reflect the views of the Federal Reserve Bank of Philadelphia, the Federal Reserve System, Banco de México, or the funding sources.

7 Programs and supplemental material

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

```
. net sj 25-1
. net install st0767      (to install program files, if available)
. net get st0767          (to install ancillary files, if available)
```

To get the latest stable versions of `xtevent`, check the installation instructions at https://github.com/JMSLab/xtevent#installation. We update the stable website version more frequently than the Statistical Software Components version.

8 References

Amemiya, T. 1978. A note on a random coefficients model. *International Economic Review* 19: 793–796. https://doi.org/10.2307/2526342.

Athey, S., and G. W. Imbens. 2022. Design-based analysis in difference-in-differences settings with staggered adoption. *Journal of Econometrics* 226: 62–79. https://doi.org/10.1016/j.jeconom.2020.10.012.

Baum, C. F., and M. E. Schaffer. 2013. avar: Stata module to perform asymptotic covariance estimation for iid and non-iid data robust to heteroskedasticity, autocorrelation, 1- and 2-way clustering, and common cross-panel autocorrelated disturbances.

Statistical Software Components S457689, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s457689.html.

Baum, C. F., M. E. Schaffer, and S. Stillman. 2002. ivreg2: Stata module for extended instrumental variables/2SLS and GMM estimation. Statistical Software Components S425401, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s425401.html.

Borusyak, K. 2023. did_imputation. GitHub. https://github.com/borusyak/did_imputation/.

Busch, A., and D. Girardi. 2023. lpdid: Stata module implementing local projections difference-in-differences (LP-DiD) estimator. Statistical Software Components S459273, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s459273.html.

Butts, K. 2021. did2s. GitHub. https://github.com/kylebutts/did2s/.

Caceres Bravo, M. 2023. staggered. GitHub. https://github.com/mcaceresb/stata-staggered/.

Callaway, B., A. Goodman-Bacon, and P. H. C. Sant'Anna. 2024. Difference-in-differences with a continuous treatment. NBER Working Paper 32117, National Bureau of Economic Research. https://doi.org/10.3386/w32117.

Callaway, B., and P. H. C. Sant'Anna. 2021. Difference-in-differences with multiple time periods. *Journal of Econometrics* 225: 200–230. https://doi.org/10.1016/j.jeconom.2020.12.001.

Clarke, D., and K. Tapia-Schythe. 2021. Implementing the panel event study. *Stata Journal* 21: 853–884. https://doi.org/10.1177/1536867X211063144.

Correia, S. 2014. reghdfe: Stata module to perform linear or instrumental-variable regression absorbing any number of high-dimensional fixed effects. Statistical Software Components S457874, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s457874.html.

———. 2016a. ftools: Stata module to provide alternatives to common Stata commands optimized for large datasets. Statistical Software Components S458213, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s458213.html.

———. 2016b. Linear models with high-dimensional fixed effects: An efficient and feasible estimator. https://scorreia.com/research/hdfe.pdf.

———. 2018. ivreghdfe: Stata module for extended instrumental variable regressions with multiple levels of fixed effects. Statistical Software Components S458530, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s458530.html.

de Chaisemartin, C., X. D'Haultfœuille, and Y. Guyonvarch. 2019. did_multiplegt: Stata module to estimate sharp difference-in-difference designs with multiple groups and periods. Statistical Software Components S458643, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s458643.html.

Dobkin, C., A. Finkelstein, R. Kluender, and M. J. Notowidigdo. 2018. The economic consequences of hospital admissions. *American Economic Review* 108: 308–352. https://doi.org/10.1257/aer.20161038.

Freyaldenhoven, S., C. Hansen, J. Pérez, and J. M. Shapiro. Forthcoming. "Visualization, identification, and estimation in the linear panel event-study design". In *Twelfth World Congress*. Advances in Economics and Econometrics, edited by V. Chernozhukov, E. Hörner, Johannes La Ferrara, and I. Werning, vol. 2. Cambridge: Cambridge University Press.

Freyaldenhoven, S., C. Hansen, J. P. Pérez, and J. M. Shapiro. 2021. Data appendix for "Visualization, identification, and estimation in the linear panel event-study design". National Bureau of Economic Research. https://data.nber.org/data-appendix/w29170/.

Freyaldenhoven, S., C. Hansen, and J. M. Shapiro. 2019. Pre-event trends in the panel event-study design. *American Economic Review* 109: 3307–3338. https://doi.org/10.1257/aer.20180609.

Freyberger, J., and Y. Rai. 2018. Uniform confidence bands: Characterization and optimality. *Journal of Econometrics* 204: 119–130. https://doi.org/10.1016/j.jeconom.2018.01.006.

Goodman-Bacon, A. 2021. Difference-in-differences with variation in treatment timing. *Journal of Econometrics* 225: 254–277. https://doi.org/10.1016/j.jeconom.2021.03.014.

Guimarães, P., and P. Portugal. 2010. A simple feasible procedure to fit models with high-dimensional fixed effects. *Stata Journal* 10: 628–649. https://doi.org/10.1177/1536867X1101000406.

Hansen, C. B. 2007. Generalized least squares inference in panel and multilevel models with serial correlation and fixed effects. *Journal of Econometrics* 140: 670–694. https://doi.org/10.1016/j.jeconom.2006.07.011.

Hegland, T. A. 2023. wooldid: Stata module to estimate difference-in-differences treatment effects with staggered treatment onset using heterogeneity-robust two-way fixed effects regressions. Statistical Software Components S459238, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s459238.html.

Martínez, I. Z. 2022a. Mobility responses to the establishment of a residential tax haven: Evidence from Switzerland. *Journal of Urban Economics* 129: art. 103441. https://doi.org/10.1016/j.jue.2022.103441.

———. 2022b. Replication materials for "Mobility responses to the establishment of a residential tax haven: Evidence from Switzerland". Dropbox. https://www.dropbox.com/s/3v0fgzy07jmm3xp/.

Montiel Olea, J. L., and M. Plagborg-Møller. 2019. Simultaneous confidence bands: Theory, implementation, and an application to SVARs. *Journal of Applied Econometrics* 34: 1–17. https://doi.org/10.1002/jae.2656.

Rios-Avila, F. 2022. jwdid: Stata module to estimate difference-in-difference models using Mundlak approach. Statistical Software Components S459114, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s459114.html.

Rios-Avila, F., P. H. C. Sant'Anna, and B. Callaway. 2021. csdid: Stata module for the estimation of difference-in-difference models with multiple time periods. Statistical Software Components S458976, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s458976.html.

Schmidheiny, K., and S. Siegloch. 2023. On event studies and distributed-lags in two-way fixed effects models: Identification, equivalence, and generalization. *Journal of Applied Econometrics* 38: 695–713. https://doi.org/10.1002/jae.2971.

Sun, L. 2021. eventstudyinteract. GitHub. https://github.com/lsun20/EventStudyInteract/.

Sun, L., and S. Abraham. 2021. Estimating dynamic treatment effects in event studies with heterogeneous treatment effects. *Journal of Econometrics* 225: 175–199. https://doi.org/10.1016/j.jeconom.2020.09.006.

About the authors

Simon Freyaldenhoven is a senior machine learning economist at the Federal Reserve Bank of Philadelphia.

Christian B. Hansen is the Wallace W. Booth Professor of Econometrics and Statistics at the University of Chicago Booth School of Business.

Jorge Pérez Pérez is a research economist at Banco de México.

Jesse M. Shapiro is the George Gund Professor of Economics and Business Administration at Harvard University and a research associate at the National Bureau of Economic Research.

Constantino Carreto is an economist at Banco de México.