The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–33

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). The package implements various procedures to estimate the underlying policy effects, and allows for non-binary policy variables and estimation adjusting for pre-event trends.

Keywords: st0001, xtevent, xteventplot, xteventtest, get_unit_time_effects, linear panel data models, two-way fixed effects regression, pre-trends, 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 $i \in \{1, ..., N\}$ observed in a sequence of periods $t \in \{1, ..., T\}$. We consider the following model:

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

Here, α_i denotes a unit fixed effect, γ_t a time fixed effect, and q_{it} a vector of controls with conformable coefficients ψ. The scalar C_{it} denotes a (potentially unobserved) confound

 st0001

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):

$$y_{it} = \sum_{k=-G-L_G}^{M+L_M-1} \delta_k \Delta z_{i,t-k} + \delta_{M+L_M} z_{i,t-M-L_M} + \delta_{-G-L_G-1}(1 - z_{i,t+G+L_G}) \\ + \alpha_i + \gamma_t + q_{it}'\psi + C_{it} + \varepsilon_{it}, \tag{2}$$

where 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 $\delta = (\delta_{-G-L_G-1}, \ldots, \delta_{M+L_M})'$ as the event-time path of the outcome, and to its estimated counterpart $\widehat{\delta}$ 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 of 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 pre-event trends ("pre-trends") that are generally inconsistent with the model in (1) (Freyaldenhoven et al. 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 (e.g., non-binary) 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 et al. 2021), `did_imputation` (Borusyak 2023), `didmultiplegt` (de Chaisemartin et al. 2019), `did2s` (Butts 2023), `eventdd` (Clarke and Tapia-Schythe 2021), `eventstudyinteract` (Sun 2023), `jwdid` (Rios-Avila 2022), `lpdid` (Busch and Girardi 2023), `staggered`

(Cáceres Bravo 2023), and `wooldid` (Hegland 2023), as well as the native `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, 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 pre-event trends using approaches based on trend extrapolation (Dobkin et al. 2018) or proxy variables (Freyaldenhoven et al. 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 (2022). 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 δ 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 δ (see Freyaldenhoven et al. Forthcoming for a more detailed discussion). The package `xtevent` includes the following estimators.

Two-way fixed effects estimator. If $C_{it} = 0$, equation (2) may be estimated by 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 2016, 2019) to allow for multiple or high-dimensional fixed effects.

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

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

$$C_{it} = \tilde{\alpha}_i + \tilde{\gamma}_t + q_{it}'\tilde{\psi} + \sum_m \phi' f(m) z_{i,t-m} \tag{3}$$

for a known set of basis functions $f(\cdot)$ and unknown parameters $\tilde{\alpha}_i, \tilde{\gamma}_t$ and $\tilde{\psi}$, then equation (2) may be estimated by including the appropriate terms from (3) directly in a regression model, or by 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 $f(m) = 1$ if $m \in [-3, 3]$ and $f(m) = 0$ otherwise. In this case, we can extrapolate the trend to post-adoption periods and subtract it to account for the confound.

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

$$d_k = \begin{cases} \phi' f_k, & \text{if } k < -G \\ \delta_k + \phi' f_k, & \text{if } -G \leq k \leq M \\ \delta_M + \phi' f_k, & \text{otherwise,} \end{cases} \tag{4}$$

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

Given estimates of d_k, $\widehat{d}_k$, we can recover the trend parameters ϕ from the estimates $\widehat{d}_k$ in the L_G unaffected periods. Let $T_G \leq L_G$ be the number of periods prior to G used to estimate the trend parameters and $T_M \leq M$ be the number of "post-event" periods where the trend is active. We assume $f_k \neq 0$ for $k \in [-G - T_G, T_M]$ and 0 otherwise. We can recover the trend parameters by using the T_G moment conditions $\widehat{d}_k - \phi' f_k = 0$ for $k = -G - T_G, \ldots, -G - 1$. We can then calculate an adjusted estimated event-time path $\widehat{\delta}$ that accounts for the confound by subtracting $\phi' f_k$ 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.

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

$$x_{it} = \alpha_i^x + \gamma_t^x + \phi^x q_{it} + \Xi^x C_{it} + u_{it} \tag{5}$$

with unknown parameters $\alpha_i^x, \gamma_t^x, \Xi^x$ and an unobserved vector u_{it} (which is uncorrelated across proxies), then `xtevent` permits estimating equation (2) by two-stage least squares, including one of the proxies in (2), and using the other proxy variable as an excluded instrument (Freyaldenhoven et al. 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 equation (5) conditionally mean-independent of the policy, then `xtevent` permits estimating equation (2) using a two-stage least squares estimator, instrumenting for the proxy with leads of the policy variable (Freyaldenhoven et al. 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 pre-trends. Evidence of such pre-trends is, in practice, often seen as evidence for the presence of a confound that invalidates the research design (Freyaldenhoven et al. 2019). The estimating equation in (2) also includes L_M additional periods after the policy effects end to assess if 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 "pre-trends" ($\delta_k = 0$ 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 of 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, this suggests that 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, this may indicate that a confound cannot plausibly explain the entire event-time path of the outcome and, therefore, that 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 if 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 equation (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 $t \in \{\underline{t}, \dots, \overline{t}\}$, 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:

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

b. Automatically impute missing values of the policy variable *inside* the observed data range. For example, if $z_{jt} = 1$ and $z_{j,t+2} = 1$, under staggered adoption, this 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 $t \in \{\underline{t} - G - L_G, ..., \underline{t} - 1\}$.

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:

$$y_{it} = \sum_{k=-G-l_G}^{M+L_M-1} \delta_k \Delta z_{s(i),t-k} + \delta_{M+L_M} z_{s(i),t-M-L_M} + \delta_{-G-L_G-1}(1 - z_{s(i),t+G+L_G}) \\ +\alpha_{s(i)} + \gamma_t + q'_{it}\psi + C_{s(i)t} + \varepsilon_{it}, \tag{6}$$

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 q_{it} that vary at the individual level, and then estimate (2) with the estimated state-time effects as dependent variables, and 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 equation (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 equation (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 to estimate event studies for each treated cohort separately, comparing each one to an untreated cohort, and then to average 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 each of these commands.

3.1 The xtevent command

The `xtevent` command has the following syntax:

`xtevent` *depvar* [*indepvars*] [*if*] [*in*] [*weight*] , `policyvar(`*varname*`)` `panelvar(`*varname*`)` `timevar(`*varname*`)` [*options*]

Options

Main

`policyvar(`*varname*`)` 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 `xtset`.

`timevar(`*varname*`)` specifies the time variable. `timevar()` is required if the data have not been previously `xtset`. See `xtset`.

`window(`*numlist*`)` 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 $k = 2$, there will be five coefficients in the window $(-2, -1, 0, 1, 2)$ and two endpoints: -3 and $+3$.

`window(k1 k2)` 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 $k_1 = -1$ and $k_2 = 2$, 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 only allowed if the policy follows staggered adoption and 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 only allowed 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` offer an alternative way to specify the estimation window:

`pre` is the number of pre-event periods where anticipation effects are allowed. With `window`, `pre` is 0.

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

`overidpre` is the number of pre-event periods for an overidentification test of pre-trends. With `window`, `overidpre` is the number of periods before the event.

`overidpost` is the number of post-event 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` estimates 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 the actual *policyvar*. `type` determines the imputation rule. The suboption `saveimp` adds the new variable to the database 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: i) in periods before the first observed value, the policy value is the same as the first observed value and; ii) 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: i) *policyvar* must be binary, and ii) 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 section 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 to normalize the coefficient on -1.

`diffavg` calculates the difference in averages between the post-event estimated coefficients and the pre-event 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` are 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 estimating the model. This option is helpful if the users want to customize their regressions and plots.

`saveinteract` saves interaction variables if `cohort` and `control_cohort`, or `sunabraham` are 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 `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` to be installed. For IV estimation, it also requires `ivreghdfe` and `ivreg2` to be installed. Note that standard errors may differ and singleton clusters may be dropped using `reghdfe`. See Correia (2016).

`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: 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.

Instrumental variable estimation with proxy variables (Freyaldenhoven et al. Forthcoming)

`proxyiv(`*proxyiv_spec*`)` specifies instruments for the proxy variable for the policy. `proxyiv()` admits three syntaxes to use either leads of the policy variable or additional variables as instruments. `proxy` 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. `proxyiv(select)` is the default for the one proxy, one instrument case, and it is only available in this case.

`proxyiv(# ...)` 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.

Controlling for event-time trends

`trend(`*#1* `[,`*subopt*`])` extrapolates a linear trend using the periods from period #1 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. #1 must be less than -1. `trend` is only available 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 Ordinary Least Squares (*ols*) or Generalized Method of Moments (*gmm*). (*ols*) omits the event-time dummies from trend(*#1*) to *-1* and adds a linear trend (_ttrend) to the regression. (*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, overlay(`*trend*`)`.

Heterogeneous treatment effects (Sun and Abraham 2021)

`cohort(`*cohort_spec*`)` specifies how to identify the treatment cohorts used for estimation of heterogenous effects by cohort using the estimator from Sun and Abraham (2021). `cohort` requires the Stata module `avar`.

`cohort(variable` ***varname***`,[,force])` specifies that the categorical variable `varname` identifies 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 heterogenous 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 only available in the staggered adoption case.

`control_cohort(control_cohort_spec)` specifies how to identify the control cohort used for estimation of heterogenous 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* identifies 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 heterogenous 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 control cohort 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 heterogenous treatment effects by cohort using the estimator from Sun and Abraham (2021). `sunabraham` is equivalent to `cohort(create)` and `control_cohort(create)`.

Estimation with repeated cross sectional data

`repeatedcs` indicates that the dataset in memory is repeated cross-sectional. 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`.

Saved Results

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

Scalars

`e(lwindow)`	left endpoint for estimation window
`e(rwindow)`	right endpoint for estimation window

Macros

`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(cmd)`	estimation command: can be "regress", "areg", "ivregress", "xtivreg", or "reghdfe"
`e(df)`	degrees of freedom
`e(komit)`	list of lags/leads omitted from regression
`e(kmiss)`	list of lags/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(cmd2)`	"xtevent"
`e(depvar)`	dependent variable
`e(pre)`	number of periods with anticipation effects
`e(post)`	number of periods with policy effects
`e(overidpre)`	number of periods to test for pre-trends
`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 dummies 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, only when `trend(#1)` is specified
`e(mattrendx)`	matrix with x-axis values of trend for overlay plot, only when `trend(#1)` 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` are 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` are specified
`e(b_interact)`	each column vector contains estimates of cohort-specific effect for the given relative time, only when `cohort` and `control_cohort`, or `opt sunabraham` are specified
`e(V_interact)`	each column vector contains variance estimate of the cohort-specific effect estimator for the given relative time, only when `cohort` and `control_cohort`, or `sunabraham` are specified
`e(ff_w)`	each column vector contains estimates of cohort shares underlying the given relative time, only when `cohort` and `control_cohort`, or `sunabraham` are specified
`e(Sigma_ff)`	variance estimate of the cohort share estimators, only when `cohort` and `control_cohort`, or `sunabraham` are 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,` [*options*]

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 10000.

`overlay(`*string*`)` creates overlay plots for trend extrapolation, instrumental variables estimation in the presence of pre-trends, 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 only available after `xtevent, 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 only available after `xtevent, proxy() proxyiv()`.

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

`y` creates an event-study plot of the dependent variable in instrumental variables estimation. `y` is only available after `xtevent, proxy() proxyiv()`.

`proxy` creates an event-study plot of the proxy variable in instrumental variables estimation. `proxy` is only available after `xtevent, proxy() 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 > 0*`)` sets the number of post-event coefficient estimates to use for calculating the smoothest line. The default is to use all the estimates in the post-event window.

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

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

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

`overidpre` changes the tested coefficients in the pre-trends overidentification test. The default is to test all pre-event coefficients. `overidpre(`*#1*`)` tests if the coefficients for the earliest *#1* 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*. *#1* must be greater than *0*. See the `xteventtest` command below.

`overidpost` 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(`*#1*`)` tests if the coefficients for the latest *#1* 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. *#1* 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 `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 pre-trends. By default, this is a Wald test for all the pre-event 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 post-event coefficients being equal unless `overidpost` is specified.

`scatterplotopts` specifies options to be passed to `scatter` for the coefficients' plot.

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

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

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

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

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

`addplots` specifies additional plots to overlay to the event-study plot.

`textboxoption` specifies options to pass to the textbox of the pre-trend and leveling-off tests. These options are disabled if `noprepval` and `nopostval` are specified. See `textbox_options`.

additional_options: Additional options to be passed to `twoway`. See `twoway`.

3.3 The xteventtest command

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

`xteventtest,` [*options*]

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 pre-event coefficients are equal to *0*. With `cumul`, it tests that the sum of all pre-event coefficients is equal to *0*.

`allpost` tests that all post-event coefficients are equal to *0*. With `cumul`, it tests that the sum of all post-event coefficients is equal to *0*.

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

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

`constanteff` tests that all post-event coefficients are equal.

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

`overidpre(`*#1*`)` tests the pre-trends overidentifying restriction. It tests that the coefficients for the earliest *#1* 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*. *#1* must be greater than *0*.

`overidpost(`*#1*`)` tests the effects leveling off overidentifying restriction. It tests that the coefficients for the latest *#1* 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. *#1* must be greater than *1*.

`testopts(`*string*`)` specifies options to be passed to `test`. See `test`.

Saved 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)`	chi-squared
`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 Stata data file 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 is repeated cross-sectional. `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(*varname*) **timevar**(*varname*) [*options*]

Options

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

`timevar(`*varname*`)` specifies the time variable.

`saving(`*filename* `[, replace])` specifies the name of the Stata data file 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 (2022).

4.1 Simulated data example

We first use the "Jump at the time of the event" data from Freyaldenhoven et al. (2022). The data is 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 one.

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

```
. use simulation_data_dynamic.dta, clear
. 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, `y` is the outcome variable, `z` is the policy 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, as the specification in equation (2) allows discrete or continuous policy variables and single or multiple changes.

Basic functionality

We estimate equation (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
```

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 pre-trends test and a leveling-off test.

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

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

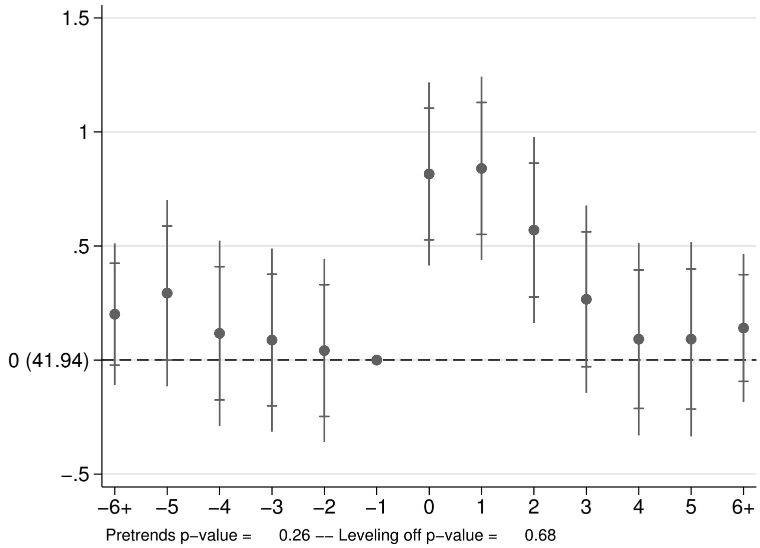


Figure 1: Event-study plot

Choosing different windows

We can also specify an asymmetric window. For instance, for an estimation of 4 pre-event periods, 7 post-event periods, and two endpoints:

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

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

```
. qui 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(#1 [, subopt])` allows extrapolation of a linear trend in event-time from period #1 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 pre-event periods $(-3, -2, -1)$ and using the `method(gmm)` suboption to use a Generalized Method of Moments estimator. We also save the overlay data for plotting:

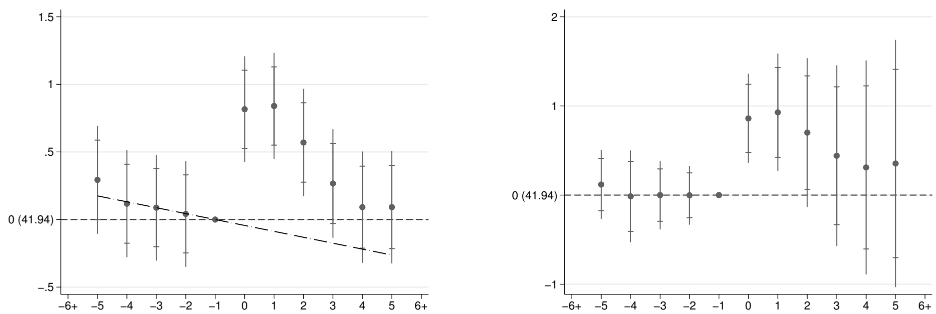
```
. xtevent y x, panelvar(id) timevar(t) policyvar(z) window(5) ///
>  impute(nuchange) trend(-3, method(gmm) saveoverlay)
No proxy or instruments provided. Implementing OLS estimator
Linear regression, absorbing indicators           Number of obs     =  2,000
Absorbed variable: id                             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.

Pretrends adjustment with proxy variables

`xtevent` allows estimation when a pre-trend is present using the instrumental variables estimator of Freyaldenhoven et al. (2019). For this, we need to specify the option `proxy(varname)` to indicate the proxy variable(s) 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(string)` option. `xtevent` uses `xtivregress` to estimate this model.

```
. xtevent y, panelvar(id) timevar(t) policyvar(z) window(5) impute(nuchange) ///
```



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

Figure 2: Linear trend adjustment

```
> proxy(x)
Proxy for the confound specified. Implementing FHS estimator
proxyiv=select. Selecting lead order of differenced policy variable to use as inst
> rument.
Lead 4 selected.
The coefficient at -1 is normalized to zero.
For estimation with proxy variables, an additional coefficient needs to be normali
> zed 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. We first 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)).

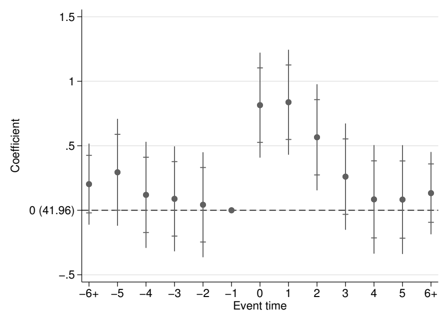
```
. xteventplot, y ytitle("Coefficient") xtitle("Event time")
. xteventplot, proxy ytitle("Coefficient") xtitle("Event time")
. xteventplot, overlay(iv) ytitle("Coefficient") xtitle("Event time")
. xteventplot, ytitle("Coefficient") xtitle("Event time")
```

Estimation with heterogeneous effects by cohort

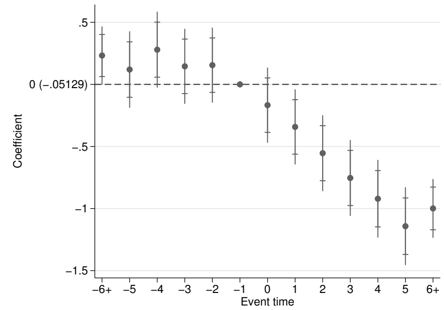
`xtevent` allows estimation in settings with heterogeneous effects that vary by cohort using Sun and Abraham's (2021) estimator through the `cohort` and `control_cohort` options.

In the `cohort` option, the user must specify the variable that identifies the cohorts, and in the `control_cohort` option, the 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.

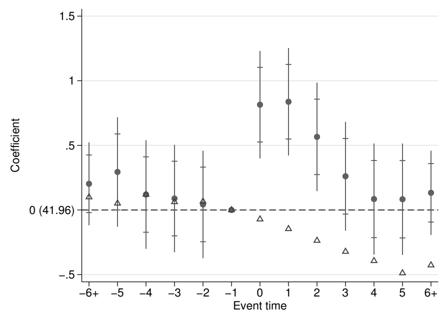
```
. gen timet=t if z==1
(1,059 missing values generated)
. by id: egen time_of_treat=min(timet)
. gen last_treat=time_of_treat==39
. replace time_of_treat = . if last_treat
(80 real changes made, 80 to missing)
. xtevent y, panelvar(id) timevar(t) policyvar(z) window(5) impute(nuchange) ///
> cohort(variable 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
```



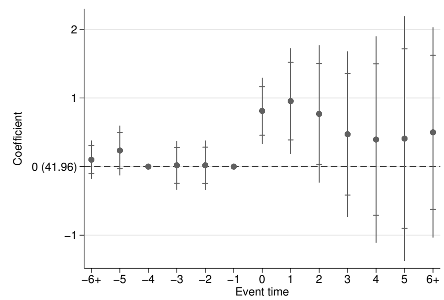
(a) Event-study plot for outcome



(b) Event-study plot for proxy



(c) Align proxy to outcome



(d) Subtract rescaled confound from outcome

Figure 3: Pre-trends adjustment using proxy variables for the confound following Freyaldenhoven et al. (2019)

```
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 to 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([type, subopt])`) option. The default plot type 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.

```
. qui 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 of 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.

4.2 Empirical application

Martínez (2022) analyzes the effect of a tax reform in the Swiss canton of Obwalden

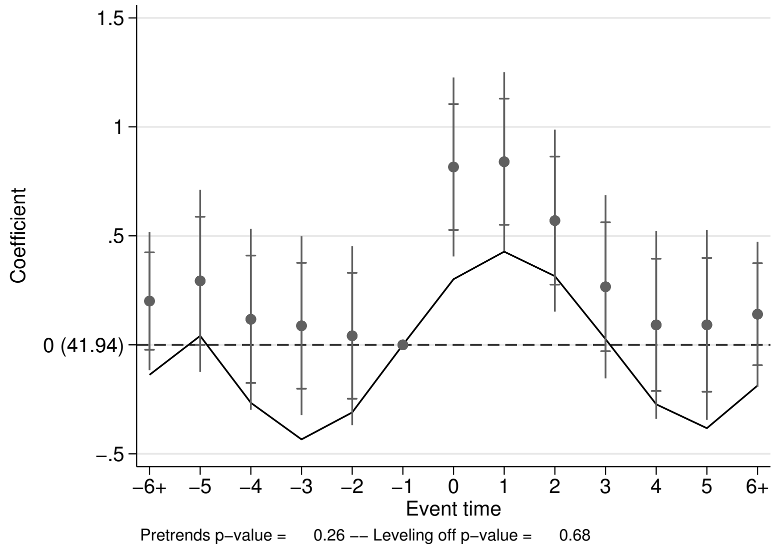


Figure 4: Least wiggly path through the confidence region

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 (2022). We use the data from Martínez (2023).

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

$$y_{it} = \sum_{k=-5, k\neq -1}^{9} \delta_k \Delta z_{i,t-k} + \delta_{10} z_{i,t-10} + \delta_{-6}(1 - z_{i,t+6}) + \alpha_i + \gamma_t + \varepsilon_{it}. \tag{7}$$

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 one for Obwalden in 2006 or after and zero 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.dta, 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),
> ab(19) noo sep(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 equation (7) 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 pre-event periods. The unit and time variables do not need to be specified because the data was `xtset` previously. With the option `reghdfe`, we can estimate this equation using the `reghdfe` command of Correia (2016). Using `reghdfe` enables two-way clustered standard errors through the `vce` option. We also specify the imputation rule `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` allows for 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(maroon) recast(connected) mcolor(maroon) msymbol(circle))
>  ///
> ciplotopts(recast(rarea) fcolor(maroon*0.2)) ///
> graphregion(fcolor(white))
option nosupt has been specified. Sup-t confidence intervals won´t be displayed or
>  calculated
option noprepval has been specified. The p-value for a pretrends test won´t be dis
> played
option nopostpval has been specified. The p-value for a test of effects leveling-o
> ff won´t be displayed
```

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 if the coefficients for event times 0, 1, 2, and 3 are equal to zero 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 if the estimated policy effects are constant across time:

```
. xteventtest, constanteff
```

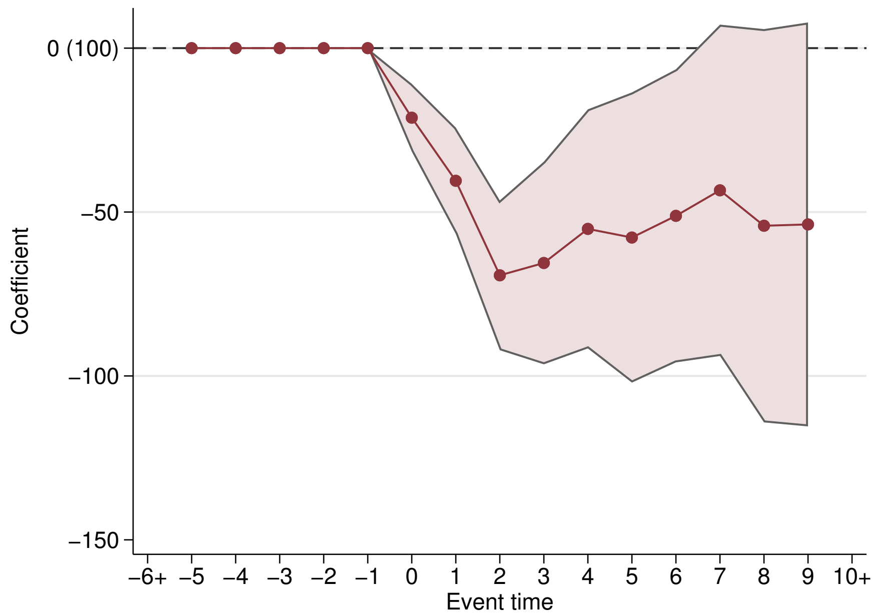


Figure 5: Dynamic effect of a Swiss tax reform following Martínez (2022)

```
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 if the effects level off. To do this, we ask `xteventtest` to use the last two post-event coefficients.

```
. xteventtest, overidpost(2)
Overidentification test for effects leveling off: 2 last post-event coefficients a
> re 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 post-event periods in the model may be sufficient to capture the dynamic effects of the policy.

5 Conclusions

This article introduces the `xtevent` command to estimate linear panel models and visualize 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 equation (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 et al. (2024), to estimate treatment effects in settings with continuous treatments and where equation (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 equation (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 users of `xtevent` 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 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* 793–796.

Athey, S., and G. W. Imbens. 2022. Design-based analysis in difference-in-differences settings with staggered adoption. *Journal of Econometrics* 226(1): 62–79.

Borusyak, K. 2023. did_imputation and event_plot https://github.com/borusyak/did_imputation, accessed September 2023.

Busch, A., and D. Girardi. 2023. LPDID: Stata module implementing Local Projections Difference-in-Differences (LP-DiD) estimator. Statistical Software Components, Boston College Department of Economics. https://ideas.repec.org/c/boc/bocode/s459273.html.

Butts, K. 2023. did2s https://github.com/kylebutts/did2s_stata, accessed march 2024.

Callaway, B., A. Goodman-Bacon, and P. H. C. Sant'Anna. 2024. Difference-in-differences with a Continuous Treatment. Working Paper 32117, National Bureau of Economic Research.

Callaway, B., and P. H. Sant'Anna. 2021. Difference-in-differences with multiple time periods. *Journal of Econometrics* 225(2): 200–230.

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

Clarke, D., and K. Tapia-Schythe. 2021. Implementing the panel event study. *The Stata Journal* 21(4): 853–884. https://doi.org/10.1177/1536867X211063144.

Correia, S. 2016. Linear Models with High-Dimensional Fixed Effects: An Efficient and Feasible Estimator. Technical report. Working Paper.

———. 2019. REGHDFE: Stata module to perform linear or instrumental-variable regression absorbing any number of high-dimensional fixed effects .

Cáceres Bravo, M. 2023. staggered https://github.com/mcaceresb/stata-staggered, accessed march 2024.

Dobkin, C., A. Finkelstein, R. Kluender, and M. J. Notowidigdo. 2018. The economic consequences of hospital admissions. *American Economic Review* 108(2): 308–352.

Freyaldenhoven, S., C. Hansen, J. P. Pérez, and J. M. Shapiro. 2022. Replication data for "Visualization, identification, and estimation in the linear panel event-study design". https://data.nber.org/data-appendix/w29170/. Accessed in September 2022.

———. Forthcoming. Visualization, identification, and estimation in the linear panel event-study design. In *Advances in Economics and Econometrics: Twelfth World Congress*, ed. V. Chernozhukov, E. Hörner, Johannes La Ferrara, and I. Werning. Cambridge, UK: Cambridge University Press.

Freyaldenhoven, S., C. Hansen, and J. M. Shapiro. 2019. Pre-event trends in the panel event-study design. *American Economic Review* 109(9): 3307–38.

Freyberger, J., and Y. Rai. 2018. Uniform confidence bands: Characterization and optimality. *Journal of Econometrics* 204(1): 119–130.

Goodman-Bacon, A. 2021. Difference-in-differences with variation in treatment timing. *Journal of Econometrics* 225(2): 254–277.

Guimarães, P., and P. Portugal. 2010. A simple feasible procedure to fit models with high-dimensional fixed effects. *The Stata Journal* 10(4): 628–649.

Hansen, C. B. 2007. Generalized least squares inference in panel and multilevel models with serial correlation and fixed effects. *Journal of econometrics* 140(2): 670–694.

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, Boston College Department of Economics. https://ideas.repec.org/c/boc/bocode/s459238.html.

Martínez, I. Z. 2022. Mobility responses to the establishment of a residential tax haven: Evidence from Switzerland. *Journal of Urban Economics* 129: 103441.

———. 2023. Replication data for "Mobility responses to the establishment of a residential tax haven: Evidence from Switzerland". https://www.dropbox.com/s/3v0fgzy07jmm3xp/. Accessed in April 2023.

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): 1–17.

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

Rios-Avila, F., P. H. Sant'Anna, and B. Callaway. 2021. CSDID: Stata module for the estimation of Difference-in-Difference models with multiple time periods. Statistical Software Components, Boston College Department of Economics. 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(5): 695–713.

Sun, L. 2023. EventStudyInteract https://github.com/lsun20/EventStudyInteract, accessed September 2023.

Sun, L., and S. Abraham. 2021. Estimating dynamic treatment effects in event studies with heterogeneous treatment effects. *Journal of Econometrics* 225(2): 175–199.

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.

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

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

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

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

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

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

A Details on trend extrapolation

The default method to extrapolate a linear trend uses GMM and is implemented as follows. Let $T_G \leq L_G$ be the number of periods prior to G used to estimate the trend parameters and $T_M \leq M$ be the number of "post-event" periods where the trend is active. We assume $f_k \neq 0$ for $k \in [-G - T_G, T_M]$ and 0 otherwise. We then have moments given by

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

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

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

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

Solving the FOC gives

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

Under suitable regularity conditions, we have that

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

where

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

B Sup-t confidence bands

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

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

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

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

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

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

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

C Details for least wiggly path

C.1 The least wiggly path proposal

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

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

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

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

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

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

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

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

In practice we normalize the event path, such that $\delta_k = 0$ for at least one k (e.g. usually at $k = -1$). We will use $\mathcal{N}$ to denote the set of size $|\mathcal{N}|$ that collects all normalized coefficients, such that $\delta^*_k(v) = 0$ for $k \in \mathcal{N}$. Throughout, we only consider the case $|\mathcal{N}| \in \{1, 2\}$, i.e., we allow for at most two normalizations.

C.2 Implementation

C.2.1 Finding p^*

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

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

Algorithm 1 Finding p^*

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

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

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

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

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

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

From the Lagrangian, the first-order conditions are:

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

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

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

Thus, the solution for v is given by

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

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

C.2.2 Finding the optimal path given p^*

Once we have found a solution to (1) using Algorithm 1, the next step is to find the polynomial with the lowest coefficient on the p^* term that is still contained in the Wald region (see equation 2). First note that by construction $v^2_{p^*} \neq 0$ (If not, Algorithm 1 would have found a solution at $p^* - 1$). v^* can then be found through a simple constrained minimization on the vector v (of dimension p^*):

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

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

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

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

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

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

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

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

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

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

It follows that

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

and the constraint in (5) becomes

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

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

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

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

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

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

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

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

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

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

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

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

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

D Policy variable imputation

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

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

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

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

```
. use simulation_data_dynamic.dta, clear

. qui xtset id t

. qui replace z=. if id==19 & (t==35 | t>=39)

. qui gen z_d=d.z

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

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

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

id	t	z	z_d	v_eq_m6	v_eq_m5	v_eq_m4	v_eq_m3	v_eq_m2	v_eq_m1

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

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

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

```
. cap drop v*

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

. qui gen z_imputed_d=d.z_imputed
```

Below, we compare the original policy variable, the imputed policy variable, the differentiated imputed policy variable, some event-time dummies, and the left endpoint generated using the imputed policy variable. First, the policy variable has been imputed in the observed time range. Nonetheless, the imputation also assumes that the policy variable in periods after $t = 40$ would have the same value as the one in that last period. This imputation can be seen in the event-time dummies, which now have zeros corresponding to leads of the differentiated policy variable in periods after 40.

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

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

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

```
. cap drop v* z_imputed z_imputed_d

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

. qui gen z_imputed_d=d.z_imputed
```

Below, we compare the original policy variable, the imputed policy variable, the differentiated imputed policy variable, some event-time dummies, and the left endpoint generated with the imputed policy variable. First, the imputed policy variable does not have missing values inside or outside the event-time range. As in the example using `impute(stag)`, the event-time dummies have zeros corresponding to leads of the differentiated policy variable in periods greater than 40. Additionally, now the event-time dummies do not have missing values inside the event-time range.

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

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

E Estimation in repeated cross-sectional datasets

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

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

```
. gen state=eventtime

. xtset, clear
```

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

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

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

. keep state t z

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

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

F Estimation with heterogenous treatment effects in staggered adoption settings

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

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

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

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

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

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

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

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

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

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

To permit greater flexibility, `xtevent` also allows estimation of (10) outside staggered adoption settings or in settings where the user wants to aggregate treatment cohorts for user-provided cohort and control cohort variables through the `cohort(variable varname, [,force])` and `control_cohort(variable varname, [,force])` options.