Package 'DIDmultiplegtDYN'

May 7, 2026

Title Estimation in Difference-in-Difference Designs with Multiple Groups and Periods

Version 2.3.3

Maintainer Anzony Quispe <anzony.quispe@gmail.com>

Description
Estimation of heterogeneity-robust difference-in-differences estimators, with a binary, discrete, or continuous treatment, in designs where past treatments may affect the current outcome.

License MIT + file LICENSE

LazyData true

URL https://github.com/Credible-Answers/did_multiplegt_dyn

Author Anzony Quispe [aut, cre],
Diego Ciccia [aut],
Felix Knau [aut],
Mélitine Malezieux [aut],
Doulo Sow [aut],
Clément de Chaisemartin [aut]

Encoding UTF-8

Imports data.table, MASS, fixest, dplyr, ggplot2, matlib, openxlsx, stats, car, lmtest, sandwich, haven, cowplot, rnames, Rcpp

Suggests polars

LinkingTo Rcpp

RoxygenNote 7.3.3

Additional_repositories https://rpolars.r-universe.dev

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-03-17 12:30:09 UTC

Contents

did_multiplegt_dyn 2
favara_imbs 13
print.did_multiplegt_dyn 14
rnames.did_multiplegt_dyn 14
summary.did_multiplegt_dyn 15

Index **16**

`did_multiplegt_dyn` *Core function for did_multiplegt_dyn*

Description

Estimation of heterogeneity-robust difference-in-differences (DID) estimators, with a binary, discrete, or continuous treatment, in designs where past treatments may affect the current outcome.

Usage

```
did_multiplegt_dyn(
  df,
  outcome,
  group,
  time,
  treatment,
  effects = 1,
  design = NULL,
  normalized = FALSE,
  normalized_weights = FALSE,
  effects_equal = FALSE,
  placebo = 0,
  controls = NULL,
  trends_nonparam = NULL,
  trends_lin = FALSE,
  continuous = NULL,
  weight = NULL,
  cluster = NULL,
  by = NULL,
  by_path = NULL,
  predict_het = NULL,
  predict_het_hc2bm = FALSE,
  date_first_switch = NULL,
  same_switchers = FALSE,
  same_switchers_pl = FALSE,
  switchers = "",
  only_never_switchers = FALSE,
  ci_level = 95,
```

```
  graph_off = FALSE,
  save_results = NULL,
  save_sample = FALSE,
  less_conservative_se = FALSE,
  more_granular_demeaning = FALSE,
  bootstrap = NULL,
  dont_drop_larger_lower = FALSE,
  drop_if_d_miss_before_first_switch = FALSE,
  ggplot_args = NULL
)
```

Arguments

`df`	(dataframe) the estimation dataset.
`outcome`	(char) is the outcome variable.
`group`	(char) is the group variable.
`time`	(char) is the time period variable. The command assumes that the time variable is evenly spaced (e.g.: the panel is at the yearly level, and no year is missing for all groups). When it is not (e.g.: the panel is at the yearly level, but three consecutive years are missing for all groups), the command can still be used, though it requires a bit of tweaking, see FAQ section below.
`treatment`	(char) is the treatment variable.
`effects`	(int) gives the number of event-study effects to be estimated. By default, the command estimates non-normalized event-study effects. Non-normalized event-study effects are averages, across all switchers, of the effect of having received their actual rather than their period-one treatment dose, for ℓ periods. While non-normalized event-study effects can be interpreted as average effects of being exposed to a weakly higher treatment dose for ℓ periods, the magnitude and timing of the incremental treatment doses can vary across groups.
`design`	(2 args: float, char path) this option reports switchers' period-one and subsequent treatments, thus helping the analyst understand the treatment paths whose effect is aggregated in the non-normalized event-study effects. When the number of treatment paths is low, one may consider estimating treatment-path-specific event-study effects to facilitate interpretation, see footnote 10 of de Chaisemartin and D'Haultfoeuille (2024) for detailed instructions. When the number of treatment paths is large, one may specify a number included between 0 and 1 in the float argument. Then the command reports the treatment paths common to at least (*float**100)% of switchers. Results can be printed in the Stata console specifying "console" as the string argument. For example, `design = c(0.5, "console")` reports the treatment paths experienced by at least 50% of the switchers and prints the output in the Stata console. Alternatively, the output can be stored in an Excel file providing a valid file path as the string argument.
`normalized`	(logical) when this option is specified, the command estimates normalized event-study effects, that are equal to a weighted average of the effects of the current treatment and of its $\ell - 1$ first lags on the outcome. See Sections 3.1 and 3.2 of de Chaisemartin and D'Haultfoeuille (2020a) for further details.

`normalized_weights`
: (logical, requires `normalized = TRUE`) the command reports the weights that normalized effect ℓ puts on the effect of the current treatment, on the effect of the first treatment lag, etc.

`effects_equal`
: (logical or char) when this option is specified and the user requests that at least two effects be estimated, the command performs an F-test that effects are equal. Can be TRUE (or "all") to test equality of all effects, or a string "lb, ub" to test equality of effects from lb to ub (e.g., "2, 5" tests if effects 2 to 5 are equal). When the normalized option is specified, this test can be useful to assess if the current and lagged treatments all have the same effect on the outcome or if their effects differ, see Lemma 3 of de Chaisemartin and D'Haultfoeuille (2020a).

`placebo`
: (int) gives the number of placebo estimators to be computed. Placebos compare the outcome evolution of switchers and of their controls, before switchers' treatment changes for the first time. Under the parallel trends and no-anticipation assumptions underlying the event-study estimators computed by `did_multiplegt_dyn()`, the expectation of the placebos is equal to zero. Thus, placebos can be used to test those assumptions, by testing the null that all placebos are equal to zero. If the user requests that at least two placebos be estimated, the command computes the p-value of a joint test of that null hypothesis. The number of placebos requested can be at most equal to the number of time periods in the data minus 2, though most often only a smaller number of placebos can be computed. Also, the number of placebos requested cannot be larger than the number of effects requested.

`controls`
: (atomic char or vector of char) gives the names of the control variables to be included in the estimation. Estimators with controls are similar to those without controls, except that the first-difference of the outcome is replaced by residuals from regressions of the first-difference of the outcome on the first-differences of the controls and time fixed effects. Those regressions are estimated in the sample of control (g,t)s: (g,t)s such that group g's treatment has not changed yet at t. Those regressions are also estimated separately for each value of the baseline treatment. Estimators with controls are unbiased even if groups experience differential trends, provided such differential trends can be fully explained by a linear model in covariates changes. To control for time-invariant covariates, one needs to interact them with the time variable T, or with time fixed effects. See Section 1.2 of the Web Appendix of de Chaisemartin and D'Haultfoeuille (2020a) for further details.

`trends_nonparam`
: (atomic char or vector of char) when this option is specified, the DID estimators computed by the command only compare switchers to controls whose treatment has not changed yet, with the same baseline treatment, and with the same value of the varlist. Estimators with the `trends_nonparam` option are unbiased even if groups experience differential trends, provided all groups with the same value of the varlist experience parallel trends. The varlist can only include time-invariant variables, and the interaction of those variables has to be coarser than the group variable. For instance, if one works with a county $\times$ year data set and one wants to allow for state-specific trends, one should specify `trends_nonparam = "state"`, where state is the state identifier. Similarly, if one works with firm $\times$ year data set and one wants to allow for industry-specific trends, then one should

write `trends_nonparam = "industry"`. See Section 1.4 of the Web Appendix of de Chaisemartin and D'Haultfoeuille (2024) for further details.

`trends_lin` (logical) when this option is specified, the estimation of the treatment effects allows for group-specific linear trends. Estimators with linear trends start by computing event-study effects on the outcome's first-difference, rather than on the outcome itself, thus allowing for group-specific linear trends. Then, to recover event-study effect ℓ on the outcome, event-study effects on the outcome's first-difference are summed from 1 to ℓ. See Section 1.3 of the Web Appendix of de Chaisemartin and D'Haultfoeuille (2024) for further details. When this option is specified, the estimated average total effect per unit of treatment is not computed.

`continuous` (int) allows to use the command even when groups' period-one treatment is continuous, meaning that all groups have a different period-one treatment value. With a discrete period-one treatment, the command compares the outcome evolution of switchers and non-switchers with the same period-one treatment. But with a truly continuous period-one treatment, there will be no two groups with the same period-one treatment. The command assumes that group's status-quo outcome evolution is a polynomial in their period-one treatment. The user's chosen polynomial order is the option's int argument. See Section 1.10 of the Web Appendix of de Chaisemartin and D'Haultfoeuille (2024) for further details. Unlike the other variance estimators computed by the command, those computed when the continuous option is specified are not backed by a proven asymptotic normality result. Preliminary simulation evidence indicates that when the option is used with a correctly-specified polynomial order, those variance estimators are conservative. On the other hand, when the specified polynomial order is strictly larger than needed, those variance estimators can become liberal. Thus, when this option is specified, we recommend using the `bootstrap` option for inference. At least, one should perform a robustness check where one compares the analytic variance computed by the command to a bootstrapped variance.

`weight` (char) gives the name of a variable to be used to weight the data. For instance, if one works with a district $\times$ year data set and one wants to weight the estimation by each district $\times$ year's population, one should write `weight = "population"`, where population is the population of each district $\times$ year. If the data set is at a more disaggregated level than group $\times$ time, the command aggregates it at the group $\times$ time level internally, and weights the estimation by the number of observations in each group $\times$ time cell if the weight option is not specified, or by the sum of the weights of the observations in each group $\times$ time cell if the weight option is specified.

`cluster` (char) can be used to cluster the estimators' standard errors. Only one clustering variable is allowed. A common practice in DID analysis is to cluster standard errors at the group level. Such clustering is implemented by default by the command. Standard errors can be clustered at a more aggregated level than the group level, but they cannot be clustered at a more disaggregated level.

`by` (char) when this option is specified, the command estimates all the effects by the different levels of the specified variable. If the variable is a binary variable for example, then the estimation is carried out once for the sample of groups with var=0 and once for the sample of groups with var=1. Then, the command reports

on a graph event-study plots for all values of the by argument, thus allowing to assess effect heterogeneity by the specified variable.

by_path (integer) when this option is specified, the command estimates all the effects separately for the # most common treatment paths from F_{g-1} to $F_{g-1+\ell}$, where ℓ is the argument inputted to the effects option. If you want to estimate effects separately for all treatment paths, you can input -1 as the option's argument. This option can not be combined with the by option.

predict_het (list with 2 args: char or vector of char, -1 or vector of positive integers) when this option is specified, the command outputs tables showing whether the group-level and time-invariant variables in the char varlist predict groups' estimated event-study effects. With the second argument set to -1, with this option the command produces one table per event-study effect estimated, each displaying the coefficients from regressions of the group-level estimate of the event-study effect on the variables in the char varlist. This method to analyze heterogeneous treatment effects assumes that switchers' counterfactual outcome evolutions is uncorrelated with the variables in varlist. To placebo test this condition, the command also shows placebo regression tables, where switchers' outcome evolutions before their treatment changed is regressed on the covariates. The p-value of a test on the null hypothesis that all heterogeneity estimates are equal to zero is shown below each table. If you are only interested in predicting a subset of the event-study effects estimated, you can specify those inside an integer vector as the second argument. This option cannot be specified with normalized = TRUE and when the controls option is specified. See Section 1.5 of the Web Appendix of de Chaisemartin and D'Haultfoeuille (2024) for further details.

predict_het_hc2bm (logical) when this option is specified together with predict_het, the command uses HC2 Bell-McCaffrey degrees of freedom adjusted standard errors for the predict_het regressions, which are more robust for small samples.

date_first_switch (2 args: char in ("", "by_baseline_treat"), char path) the option reports the dates at which switchers experience their first treatment change, and how many groups experienced a first change at each date. The reference population are switchers for which the last event-study effect can be estimated. If "by_baseline_treat" is specified as the first argument, separate tables are displayed for each level of the period-one treatment. Results can be printed in the Stata console specifying "console" in the second argument. Alternatively, the output can be stored in an Excel file providing a valid file path in the second argument.

same_switchers (logical) if this option is specified and the user requests that at least two effects be estimated, the command will restrict the estimation of the event-study effects to switchers for which all effects can be estimated, to avoid compositional changes.

same_switchers_pl (logical, requires same_switchers = TRUE) the command restricts the estimation of event-study and placebo effects to switchers for which all event-study and placebos effects can be estimated.

switchers (char in ("", "in", "out")) one may be interested in estimating separately the treatment effect of switchers-in, whose average treatment after they switch is

larger than their baseline treatment, and of switchers-out, whose average treatment after they switch is lower than their baseline treatment. In that case, one should run the command first with the `switchers = "in"` option, and then with the `switchers = "out"` option.

`only_never_switchers`
(logical) if this option is specified, the command estimates the event-study effects using only never-switchers as control units.

`ci_level` (int) with this option you can change the level of the confidence intervals displayed in the output tables and the graphs. The default value is fixed at 95, yielding a 95% coverage.

`graph_off` (logical) when this option is specified, the command does not print a graph. Regardless, a ggplot object will be still generated and stored in the did_multiplegt_dyn class object.

`save_results` (char) if this option is specified, the command saves the estimators requested, their standard error, their 95% confidence interval, and the number of observations used in the estimation in a separate data set, at the location specified in the char argument.

`save_sample` (logical) if this option is specified, the command generates a (numeric) variable *did_sample*, tagging all (g, t) cells used in the estimation. This variable may take on three non-missing values: "Control" for (g, t) cells used as controls, "Switcher-in" for (g,t) cells used as switchers-in, and "Switcher-out" for cells used as switchers-out. This variable is missing for all (g, t) cells not used in the estimation. For (g,t) cells used as switchers-in or switchers-out, the variable *did_effect* also indicates the number of the event-study effect for which the cell is used in the estimation.

`less_conservative_se`
(logical) when groups' treatment can change multiple times, the standard errors reported by default by the command may be conservative. Then, less conservative standard errors can be obtained by specifying this option. See de Chaisemartin et al. (2024) for further details.

`more_granular_demeaning`
(logical) when this option is specified, the command uses more granular demeaning for variance estimation. This option automatically enables `less_conservative_se`.

`bootstrap` (integer or list) when this option is specified, bootstrapped instead of analytical standard errors are reported. Can be specified as an integer (number of replications) or as a list with two elements: `list(reps, seed)` where reps is the number of replications and seed is the random seed for reproducibility. If the `cluster` option is also requested, the bootstrap is clustered at the level requested in the `cluster` option.

`dont_drop_larger_lower`
(logical) by default, the command drops all the (g, t) cells such that at t, group g has experienced both a strictly larger and a strictly lower treatment than its baseline treatment. de Chaisemartin and D'Haultfoeuille (2020a) recommend this procedure, if you are interested in more details you can see their Section 3.1. The option `dont_drop_larger_lower` allows to overwrite this procedure and keeps (g, t) cells such that at t, group g has experienced both a strictly

larger and a strictly lower treatment than its baseline treatment in the estimation sample.

`drop_if_d_miss_before_first_switch`
: (logical) This option is relevant when the treatment of some groups is missing at some time periods. Then, the command imputes some of those missing treatments. Those imputations are detailed in Appendix A of de Chaisemartin et al (2024). In designs where groups' treatments can change at most once, all those imputations are justified by the design. In other designs, some of those imputations may be liberal. `drop_if_d_miss_before_first_switch` can be used to overrule the potentially liberal imputations that are not innocuous for the non-normalized event-study estimators. See Appendix A of de Chaisemartin et al (2024) for further details.

`ggplot_args`
: (list) This option allows you to enter additional ggplot features to the event-study graph produced by the command. Enter all your arguments in the list, as you would list them with a + in general. For instance, you can modify legends by using `ggplot_args = list(labs(. . .))`. More pervasive changes can be done by directly interacting with the ggplot object stored in the `$plot` sub-list of the assigned `did_multiplegt_dyn` object.

Value

A list of class did_multiplegt_dyn containing the arguments used, the results for the estimation requested and a ggplot object with the event-study graph. If the by option is specified, the did_multiplegt_dyn object will contain the arguments, a list with the levels of the by option, a sublist for each of these levels with the results and ggplot objects from these by-estimations and a ggplot object for the combined event-study graph. The class did_multiplegt_dyn is assigned to enable customized print and summary methods.

Overview

`did_multiplegt_dyn()` estimates the effect of a treatment on an outcome, using group-(e.g. county- or state-) level panel data. The command computes the DID event-study estimators introduced in de Chaisemartin and D'Haultfoeuille (2024). Like other recently proposed DID estimation commands (`did`, `didimputation`, ...), `did_multiplegt_dyn()` can be used with a binary and staggered (absorbing) treatment. But unlike those other commands, `did_multiplegt_dyn()` can also be used with a non-binary treatment (discrete or continuous) that can increase or decrease multiple times. Lagged treatments may affect the outcome, and the current and lagged treatments may have heterogeneous effects, across space and/or over time. The event-study estimators computed by the command rely on a no-anticipation and parallel trends assumptions. The panel may be unbalanced: not all groups have to be observed at every period. The data may also be at a more disaggregated level than the group level (e.g. individual-level wage data to measure the effect of a regional-level minimum-wage on individuals' wages).

Further detail

Non-normalized event-study estimators (the default):

For all "switchers", namely groups that experience a change of their treatment over the study period, let F_g denote the first time period when g's treatment changes. The command computes

the non-normalized event-study estimators DID_ℓ. DID_1 is the average, across all switchers, of DID estimators comparing the $F_g - 1$ to F_g outcome evolution of g to that of groups with the same period-one treatment as g but whose treatment has not changed yet at F_g. More generally, DID_ℓ is the average, across all switchers, of DID estimators comparing the $F_g - 1$ to $F_g - 1 + \ell$ outcome evolution of g to that of groups with the same period-one treatment as g but whose treatment has not changed yet at $F_g - 1 + \ell$. Those estimators are unbiased for non-normalized event-study effects, which are average effects of having been exposed to a weakly higher treatment dose for ℓ periods, where the magnitude and timing of the incremental treatment doses can vary across groups.

Normalized event-study estimators:

The command also computes the normalized event-study estimators DID^n_ℓ, that normalize DID_ℓ by the average cumulative (total) incremental treatment dose received by switchers from $F_g - 1$ to $F_g - 1 + \ell$ with respect to their period-one treatment. This normalization ensures that DID^n_ℓ estimates a weighted average of the effects of the current treatment and of its $\ell - 1$ first lags on the outcome. While the effects of the current and lagged treatments cannot be separately estimated, the weight that DID^n_ℓ puts on the effect of each lag can be estimated.

Average cumulative (total) effect per dose:

The command also computes an estimated average cumulative (total) effect per unit of treatment, where "cumulative effect" refers to the sum of the effects of a treatment dose, at the time when it takes place and at later periods, see Section 3.3 of de Chaisemartin and D'Haultfoeuille (2024) for further details. The command also shows the number of time periods over which the effect of a dose is accumulated, on average across all incremental doses received by switchers over the study period. By dividing the average cumulative effect by the average number of periods across which effects are accumulated, one can get an estimator of the effect of being exposed to one more dose for one more period.

Placebos:

The command also computes placebo estimators, that average DIDs comparing the outcome evolution of switcher g and of its control groups, from $F_g - 1$ to $F_g - 1 - \ell$, namely before g's treatment changes for the first time. Those placebos can be used to test the parallel trends and no-anticipation assumptions under which the estimators computed by `did_multiplegt_dyn` are unbiased.

Designs compatible with the command:

The command accommodates many DID designs. It covers the canonical binary and absorbing treatment case. It also covers more complicated treatment paths: groups may have heterogeneous treatments at period one, their treatment may change at different dates, some groups may experience increases in their treatment while other groups experience decreases, some groups may experience more than one change of their treatment, and finally some groups may experience larger treatment changes than others. The command can also be used to separately estimate the effects of several treatment variables, see references in the FAQ section. The only requirement is that not all groups experience their first treatment change at the same date. If groups' period-one treatment is continuously distributed, meaning essentially that all groups have a different period-one treatment, the option `continuous` needs to be used.

Relaxing parallel trends assumptions:

This command allows for many relaxations of the parallel-trends assumption: see the controls option for estimators allowing for time-varying covariates, see the `trends_lin` option for estimators allowing for group-specific linear trends, and see the trends_nonparam option for estimators allowing to interact time fixed effects with time-invariant variables (e.g. industry×year effect with firm-level panel data).

Contacts

Github repository: chaisemartinPackages/did_multiplegt_dyn

Mail: chaisemartin.packages@gmail.com

FAQ

`did_multiplegt_dyn()` does not output exactly the same results as `did_multiplegt()`, is this normal?:

Yes, the two commands can sometimes output different results. This is mostly due to different conventions in the way the two commands deal with missing values. See Appendix B of de Chaisemartin et al (2024) for further details.

Do I have to include group and time fixed effects as controls when using `did_multiplegt_dyn()`?:

No, you do not have to. Group and time fixed effects are automatically controlled for.

My group-level panel is unbalanced: some groups (e.g. counties) are not observed in every year. Can I still use the command?:

You can. A frequent case of unbalancedness is when some groups are not observed over the full duration of the panel. For instance, your data may be a yearly county-level panel from 1990 to 2000, where some counties appear after 1990 while some exit before 2000. Then, the command just redefines group's period-one treatment as their treatment at the first period when they are observed.

It may also be that some groups enter and exit the data multiple times. For instance, you observe a county in 1990, 1991, 1994, 1996, and 2000. Then, the command may impute some of that county's missing treatments. Those imputations are detailed in Appendix A of de Chaisemartin et al (2024). In designs where groups' treatments can change at most once, all those imputations are justified by the design. In other designs, some of those imputations may be liberal.drop_if_d_miss_before_first_switch can be used to overrule the potentially liberal imputations that are not innocuous for the non-normalized event-study estimators. See Appendix A of de Chaisemartin et al (2024) for further details.

Finally, it may also be the case that the data is fully missing at one or several time periods. For instance, you have data for 1990, 1991, and 1993, but 1992 is missing for every group. Then, it is important to fill the gap in the data, as otherwise the estimation will assume that 1991 and 1993 are as far apart as 1990 and 1991. There are two ways of doing so. First, you can append to your data a data set identical to your 1991 data, but with the year equal to 1992, and the outcome missing for every observation. This is a conservative solution, where no first treatment change occurring between 1991 and 1993 will be used in the estimation, which may be reasonable because the year in which the change occurred is effectively unknown. Second, you can append to your data a data set identical to your 1993 data, with the year equal to 1992, and the outcome missing for every observation. Then, treatment changes occurring between 1991 and 1993 will be used in the estimation, assuming they all took place between 1991 and 1992.

Related to imbalanced panels, my outcomes (and potentially the control variables) are measured less frequently than the treatment. For instance, the outcome is measured every two years, but I know the treatment of every group in every year. How should I proceed?:

To fix ideas, let us first assume that the outcome is measured every two years, but you know the treatment of every group in every year. Then, you should split the sample into two subsamples, and run the command twice, one time on each of the subsamples. In the first estimation, you should include all group $\times$ time cells (g,t) such that at t, g's treatment has never changed since the start of the panel, and all (g,t)s such that i) g's treatment has changed at least once at t and ii) the change occurred at a period where the outcome is observed. Since the outcome is measured every two years, in that subsample the first event-study effect (denoted effect_1) is the effect of being exposed to a higher treatment for one period, the second effect (effect_2) is the effect of being exposed to a higher treatment for three periods, etc. In the second estimation, you should include all group $\times$ time cells (g,t) such that at t, g's treatment has never changed since the start of the panel, and all (g,t)s such that i) g's treatment has changed at least once at t and ii) the change occurred at a period where the outcome is not observed. In that subsample, the first event-study effect (denoted effect_1) is the effect of being exposed to a higher treatment for two periods, the second effect (effect_2) is the effect of being exposed to a higher treatment for four periods, etc. You may then combine the two sets of estimated effects into one event-study graph, with the only caveat that the "odd" and "even" effects are estimated on different subsamples. Importantly, the two estimations have to be run on a dataset at the same bi-yearly level as the outcome variable: the yearly level treatment information should only be used to select the relevant subsamples.

If the treatment is observed three times more often than the treatment, you can follow the same logic, splitting the sample into three subsamples and running the command three times, etc.

A short do file with a simple example where the treatment status is observed in each period while the outcome is only observed every second period can be found here.

What is the maximum number of event-study effects I can estimate?:

With a balanced panel of groups, the maximum number of event-study effects one can estimate can be determined as follows. For each value of the period-one treatment d, start by computing the difference between the last period at which at least one group has had treatment d since period 1, and the first period at which a group with treatment d at period 1 changed its treatment. Add one to this difference. Then, the maximum number of event-study effects is equal to the maximum of the obtained values, across all values of the period-one treatment. With an unbalanced panel, this method can still be used to derive an upper bound of the maximum number of event-study effects one can estimate.

How many control variables can I include in the estimation?:

Estimators with control variables are similar to those without controls, except that the first-difference of the outcome is replaced by residuals from regressions of the first-difference of the outcome on the first-differences of the controls and time fixed effects. Those regressions are estimated in the sample of control (g,t)s: (g,t)s such that group g's treatment has not changed yet at period t. Those regressions are also estimated separately for each value of the period-one treatment. If at period one, treatment takes values 0, 1, 2, 3, and 4, one regression is estimated for control (g,t)s with a period-one treatment equal to 0, one regression is estimated for control (g,t)s with a period-one treatment equal to 1, etc. The number of control variables needs to be significantly smaller than the number of control (g,t)s in each of those regressions. Otherwise, those regressions will overfit and produce noisy estimates. If the number of observations is lower than the number of variables in one of those regressions, the command will run but will not take

into account all the controls for all values of the period-one treatment. An error message will let the user know that they are encountering this situation, and may thus want to reduce their number of control variables.

My design is such that treatment is binary, and groups can enter the treatment, and then leave it once. Can I use the command to separately estimate the effect of joining and leaving the treatment?:

Yes you can. See Section 1.6 of the Web Appendix of de Chaisemartin and D'Haultfoeuille (2024) for further details.

My design has several treatments. Can I use the command to estimate the event-study effects of a treatment controlling for other treatments?:

Yes, if those treatments follow binary and staggered designs. See Section 3.2 of the Web Appendix of de Chaisemartin and D'Haultfoeuille (2023) for further details, keeping in mind that the did_multiplegt command referenced at the time is now superseded by this command.

Can I perform triple difference-in-differences with the command?:

Yes. Suppose for instance your third difference is across men and women in the same (g, t) cell. Then, for each (g, t) cell, you just need to compute the difference between the average outcome of men and women in cell (g, t). Then, you simply run the command with this new outcome.

Is it possible to compute switchers' average counterfactual outcome at periods F_g, F_{g+1}, ..., $F_{g-1+\ell}$, so as to then express the event-study effects in percentage points of the counterfactual outcome level?:

Yes. You just need to define a new outcome variable $Y' = -1t < F_g Y$, where F_g is the first date at which g's treatment has changed. Essentially, you replace the outcome by 0 after the treatment change, and by $-Y$ before the treatment change. Then, you compute non-normalized event-study estimators with Y' as the outcome.

Can the command be used in fuzzy designs, where the treatment varies within group $\times$ time cells?:

Yes it can, see Section 1.7 of the Web Appendix of de Chaisemartin and D'Haultfoeuille (2024) for further details.

Authors

- Clément de Chaisemartin, Economics Department, Sciences Po, France.
- Diego Ciccia, Northwestern University, USA.
- Xavier D'Haultfoeuille, CREST-ENSAE, France.
- Felix Knau, LMU Munich, Germany.
- Mélitine Malézieux, Stockholm School of Economics, Sweden.
- Doulo Sow, Princeton University, USA.

References

de Chaisemartin, C, D'Haultfoeuille, X (2024). Difference-in-Differences Estimators of Intertemporal Treatment Effects doi:10.2139/ssrn.3731856. Forthcoming, Review of Economics and Statistics.

de Chaisemartin, C, D'Haultfoeuille, X (2023). Two-way fixed effects regressions with several treatments doi:10.2139/ssrn.3751060. Journal of Econometrics.

de Chaisemartin, C, Ciccia, D, D'Haultfoeuille, X, Knau, F, Malézieux, M, Sow, D (2024). Estimators and Variance Estimators Computed by the did_multiplegt_dyn Command.

Examples

```
# See the did_multiplegt_dyn GitHub page for examples and details.
```

`favara_imbs` *Favara and Imbs (2015)*

Description

Favara and Imbs (2015) use 1994-to-2005 county-level data to estimate the effect of the number of regulations lifted on the growth of mortgages originated by banks, and on the growth of houses prices. Their findings are revisited in de Chaisemartin and D'Haultfoeuille (2024a). This dataset includes only the first 10 states. The full dataset is available on GitHub.

Usage

```
data(favara_imbs)
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 1157 rows and 7 columns.

References

Favara and Imbs (2015) (AER)

print.did_multiplegt_dyn *A print method for did_multiplegt_dyn*

Description

A customized printed display for did_multiplegt_dyn output

Usage

```
## S3 method for class 'did_multiplegt_dyn'
print(x, ...)
```

Arguments

x	A did_multiplegt_dyn object
...	Undocumented

Value

No return, custom print method for did_multiplegt_dyn objects. Estimation tables are fetched from the object and displayed in the same style as the Stata did_multiplegt_dyn command.

rnames.did_multiplegt_dyn *rnames method for did_multiplegt_dyn*

Description

A customized rnames method for did_multiplegt_dyn output

Usage

```
## S3 method for class 'did_multiplegt_dyn'
rnames(obj, ignore = c("plot", "args"), ...)
```

Arguments

obj	A did_multiplegt_dyn object
ignore	Sublists to be ignored
...	Undocumented

Value

The same output as rnames.

summary.did_multiplegt_dyn *A summary method for did_multiplegt_dyn*

Description

A customized summary display for did_multiplegt_dyn output

Usage

```
## S3 method for class 'did_multiplegt_dyn'
summary(object, ...)
```

Arguments

object	A did_multiplegt_dyn object
...	Undocumented

Value

No return, custom summary method for did_multiplegt_dyn objects. Estimation tables are fetched from the object and displayed in the same style as the Stata did_multiplegt_dyn command.

Index

∗ **datasets**
favara_imbs, 13

did_multiplegt_dyn, 2

favara_imbs, 13

print.did_multiplegt_dyn, 14

rnames.did_multiplegt_dyn, 14

summary.did_multiplegt_dyn, 15