Package ‘etwfe’

January 11, 2023

Type Package

Title Extended Two-Way Fixed Effects

Version 0.2.0

Date 2023-01-11

Description Convenience functions for implementing extended two-way fixed effect regressions a la Wooldridge (2021, 2022) <doi:10.2139/ssrn.3906345>, <doi:10.2139/ssrn.4183726>.

License MIT + file LICENSE

Imports fixest (>= 0.11.0), stats, Formula, marginaleffects (>= 0.8.1)

Suggests did, modelsummary, ggplot2, knitr, rmarkdown, tinytest

Encoding UTF-8

RoxygenNote 7.2.3

URL `https://grantmcdermott.com/etwfe/`

BugReports `https://github.com/grantmcdermott/etwfe/issues`

VignetteBuilder knitr

NeedsCompilation no

Author Grant McDermott [aut, cre] (<`https://orcid.org/0000-0001-7883-8573`>)

Maintainer Grant McDermott <`grantmcd@uoregon.edu`>

Repository CRAN

Date/Publication 2023-01-11 18:00:02 UTC

R topics documented:

emfx *Post-estimation treatment effects for an ETWFE regressions.*

Description

Post-estimation treatment effects for an ETWFE regressions.

Usage

```
emfx(
  object,
  type = c("simple", "group", "calendar", "event"),
  post_only = TRUE,
  ...
)
```

Arguments

`object`	An ‘etwfe‘ model object.
`type`	Character. The desired type of post-estimation aggregation.
`post_only`	Logical. Only keep post-treatment effects. All pre-treatment effects will be zero as a mechanical result of ETWFE's estimation setup, so the default is to drop these nuisance rows from the dataset. But you may want to keep them for presentation reasons (e.g., plotting an event-study); though be warned that this is strictly performative. This argument will only be evaluated if ‘type = "event"‘.
`...`	Additional arguments passed to [‘marginaleffects::marginaleffects‘].

Value

A marginaleffects object.

See Also

[marginaleffects::marginaleffects()]

Examples

```
# We'll use the mpdta dataset from the did package (which you'll need to
# install separately).

# install.packages("did")
data("mpdta", package = "did")

# Run the estimation
mod = etwfe(
    fml  = lemp ~ lpop, # outcome ~ controls
    tvar = year,        # time variable
```

```
        gvar = first.treat, # group variable
        data = mpdta,       # dataset
        vcov = ~countyreal  # vcov adjustment (here: clustered by county)
        )
mod

# We can recover a variety of treatment effects of interest with the
# complementary emfx() function. For example:
emfx(mod, type = "event")
```

`etwfe` *Extended two-way fixed effects*

Description

Extended two-way fixed effects

Usage

```
etwfe(
  fml = NULL,
  tvar = NULL,
  gvar = NULL,
  data = NULL,
  ivar = NULL,
  tref = NULL,
  gref = NULL,
  cgroup = c("notyet", "never"),
  fe = c("vs", "feo", "none"),
  family = NULL,
  ...
)
```

Arguments

`fml`	A formula with the outcome (lhs) and any time-constant controls variables (rhs), e.g. ‘y ~ x1 + x2‘. Please note that time-varying controls are not supported. Similarly, if no additional controls are required, the rhs must take the value of zero, e.g. ‘y ~ 0‘.
`tvar`	Time variable. Can be a string (e.g., "year") or an expression (e.g., year).
`gvar`	Group variable. Can be either a string (e.g., "first_treated") or an expression (e.g., first_treated). In a staggered treatment setting, the group variable typically denotes treatment cohort.
`data`	The data frame that you want to run ETWFE on.

`ivar`	Optional index variable. Can be a string (e.g., "country") or an expression (e.g., country). Leaving as NULL (the default) will result in group-level fixed effects being used, which is more efficient and necessary for nonlinear models (see ‘family‘ argument below). However, you may still want to cluster your standard errors by your index variable through the ‘vcov‘ argument. See examples below.
`tref`	Optional reference value for ‘tvar‘. Defaults to its minimum value (i.e., the first time period observed in the dataset).
`gref`	Optional reference value for ‘gvar‘. You shouldn’t need to provide this if your ‘gvar‘ variable is well specified. But providing an explicit reference value can be useful/necessary if the desired control group takes an unusual value.
`cgroup`	What control group do you wish to use for estimating treatment effects. Either "notyet" treated (the default) or "never" treated.
`fe`	What level of fixed effects should be used? Defaults to "vs" (varying slopes), which is the most efficient in terms of estimation and terseness of the return model object. The other two options, "feo" (fixed effects only) and "none" (no fixed effects whatsoever), trade off efficiency for additional information on other (nuisance) model parameters. Note that the primary treatment parameters of interest should remain unchanged regardless of choice.
`family`	Which [‘family‘] to use for the estimation. Defaults to NULL, in which case [‘fixest::feols‘] is used. Otherwise passed to [‘fixest::feglm‘], so that valid entries include "logit", "poisson", and "negbin". Note that if a non-NULL family entry is detected, ‘ivar‘ will automatically be set to NULL.
`...`	Additional arguments passed to [‘fixest::feols‘] (or [‘fixest::feglm‘]). The most common example would be a ‘vcov‘ argument.

Value

A fixest object with fully saturated interaction effects.

References

Wooldridge, Jeffrey M. (2021). *Two-Way Fixed Effects, the Two-Way Mundlak Regression, and Difference-in-Differences Estimators.* Working paper (version: August 16, 2021). Available: http://dx.doi.org/10.2139/ssrn.3906345

Wooldridge, Jeffrey M. (2022). *Simple Approaches to Nonlinear Difference-in-Differences with Panel Data.* Working paper (version: August 7, 2022). Available: http://dx.doi.org/10.2139/ssrn.4183726

See Also

[fixest::feols()], [fixest::feglm()]

Examples

```
# We’ll use the mpdta dataset from the did package (which you’ll need to
# install separately).

# install.packages("did")
data("mpdta", package = "did")
```

```
# Run the estimation
mod = etwfe(
    fml  = lemp ~ lpop, # outcome ~ controls
    tvar = year,        # time variable
    gvar = first.treat, # group variable
    data = mpdta,       # dataset
    vcov = ~countyreal  # vcov adjustment (here: clustered by county)
    )
mod

# We can recover a variety of treatment effects of interest with the
# complementary emfx() function. For example:
emfx(mod, type = "event")
```

Index