Package: etwfe (via r-universe)

August 25, 2024

Type Package

Title Extended Two-Way Fixed Effects

Version 0.4.0

Date 2024-02-26

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 (>= 0.11.2), stats, data.table, Formula, marginaleffects (>= 0.10.0)

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

Encoding UTF-8

RoxygenNote 7.3.1

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

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

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Repository https://grantmcdermott.r-universe.dev

RemoteUrl https://github.com/grantmcdermott/etwfe

RemoteRef HEAD

RemoteSha ac7fe39832d57315e5a038c4f5dbd4164538e31a

Contents

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"),
  by_xvar = "auto",
  collapse = "auto",
  post_only = TRUE,
  ...
)
```

Arguments

`object` An `etwfe` model object.

`type` Character. The desired type of post-estimation aggregation.

`by_xvar` Logical. Should the results account for heterogeneous treatment effects? Only relevant if the preceding `etwfe` call included a specified `xvar` argument, i.e. interacted categorical covariate. The default behaviour ("auto") is to automatically estimate heterogeneous treatment effects for each level of `xvar` if these are detected as part of the underlying `etwfe` model object. Users can override by setting to either FALSE or TRUE. See the section on Heterogeneous treatment effects below.

`collapse` Logical. Collapse the data by (period by cohort) groups before calculating marginal effects? This trades off a loss in estimate accuracy (typically around the 1st or 2nd significant decimal point) for a substantial improvement in estimation time for large datasets. The default behaviour ("auto") is to automatically collapse if the original dataset has more than 500,000 rows. Users can override by setting either FALSE or TRUE. Note that collapsing by group is only valid if the preceding `etwfe` call was run with "ivar = NULL" (the default). See the section on Performance tips below.

`post_only` Logical. Drop pre-treatment ATTs? Only evaluated if (a) `type = "event"` and (b) the original `etwfe` model object was estimated using the default "notyet" treated control group. If conditions (a) and (b) are met then the pre-treatment effects will be zero as a mechanical result of ETWFE's estimation setup. The default behaviour (`FALSE`) is thus to drop these nuisance rows from the dataset. The `post_only` argument recognises that you may still want to keep them for presentation purposes (e.g., plotting an event-study). Nevertheless, be forewarned that enabling that behaviour via `TRUE` is *strictly* performative: the "zero" treatment effects for any pre-treatment periods is purely an artefact of the estimation setup.

`...` Additional arguments passed to `marginaleffects::slopes`. For example, you can pass `vcov = FALSE` to dramatically speed up estimation times of the main marginal effects (but at the cost of not getting any information about standard errors; see Performance tips below). Another potentially useful application is testing whether heterogeneous treatment effects (i.e. the levels of any `xvar` covariate) are equal by invoking the `hypothesis` argument, e.g. `hypothesis = "b1 = b2"`.

Value

A `slopes` object from the `marginaleffects` package.

Performance tips

Under most situations, `etwfe` should complete very quickly. For its part, `emfx` is quite performant too and should take a few seconds or less for datasets under 100k rows. However, `emfx`'s computation time does tend to scale non-linearly with the size of the original data, as well as the number of interactions from the underlying `etwfe` model. Without getting too deep into the weeds, the numerical delta method used to recover the ATEs of interest has to estimate two prediction models for *each* coefficient in the model and then compute their standard errors. So, it's a potentially expensive operation that can push the computation time for large datasets (> 1m rows) up to several minutes or longer.

Fortunately, there are two complementary strategies that you can use to speed things up. The first is to turn off the most expensive part of the whole procedure—standard error calculation—by calling `emfx(..., vcov = FALSE)`. Doing so should bring the estimation time back down to a few seconds or less, even for datasets in excess of a million rows. While the loss of standard errors might not be an acceptable trade-off for projects where statistical inference is critical, the good news is this first strategy can still be combined our second strategy. It turns out that collapsing the data by groups prior to estimating the marginal effects can yield substantial speed gains of its own. Users can do this by invoking the `emfx(..., collapse = TRUE)` argument. While the effect here is not as dramatic as the first strategy, our second strategy does have the virtue of retaining information about the standard errors. The trade-off this time, however, is that collapsing our data does lead to a loss in accuracy for our estimated parameters. On the other hand, testing suggests that this loss in accuracy tends to be relatively minor, with results equivalent up to the 1st or 2nd significant decimal place (or even better).

Summarizing, here's a quick plan of attack for you to try if you are worried about the estimation time for large datasets and models:

1. Estimate `mod = etwfe(...)` as per usual.
2. Run `emfx(mod, vcov = FALSE, ...)`.
3. Run `emfx(mod, vcov = FALSE, collapse = TRUE, ...)`.
4. Compare the point estimates from steps 1 and 2. If they are are similar enough to your satisfaction, get the approximate standard errors by running `emfx(mod, collapse = TRUE, ...)`.

Heterogeneous treatment effects

Specifying `etwfe(..., xvar = <xvar>)` will generate interaction effects for all levels of <xvar> as part of the main regression model. The reason that this is useful (as opposed to a regular, non-interacted covariate in the formula RHS) is that it allows us to estimate heterogeneous treatment effects as part of the larger ETWFE framework. Specifically, we can recover heterogeneous treatment effects for each level of <xvar> by passing the resulting `etwfe` model object on to `emfx()`.

For example, imagine that we have a categorical variable called "age" in our dataset, with two distinct levels "adult" and "child". Running `emfx(etwfe(..., xvar = age))` will tell us how the efficacy of treatment varies across adults and children. We can then also leverage the in-built hypothesis testing infrastructure of `marginaleffects` to test whether the treatment effect is statistically different across these two age groups; see Examples below. Note the same principles carry over to categorical variables with multiple levels, or even continuous variables (although continuous variables are not as well supported yet).

See Also

`marginaleffects::slopes()`

Examples

```
## Not run:
# We'll use the mpdta dataset from the did package (which you'll need to
# install separately).

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

#
# Basic example
#

# The basic ETWFE workflow involves two steps:

# 1) Estimate the main regression model with etwfe().

mod = etwfe(
    fml  = lemp ~ lpop, # outcome ~ controls (use 0 or 1 if none)
    tvar = year,        # time variable
    gvar = first.treat, # group variable
    data = mpdta,       # dataset
    vcov = ~countyreal  # vcov adjustment (here: clustered by county)
    )

# mod ## A fixest model object with fully saturated interaction effects.

# 2) Recover the treatment effects of interest with emfx().

emfx(mod, type = "event") # dynamic ATE a la an event study

# Etc. Other aggregation type options are "simple" (the default), "group"
# and "calendar"


#
# Heterogeneous treatment effects
#
```

```
# Example where we estimate heterogeneous treatment effects for counties
# within the 8 US Great Lake states (versus all other counties).

gls = c("IL" = 17, "IN" = 18, "MI" = 26, "MN" = 27,
        "NY" = 36, "OH" = 39, "PA" = 42, "WI" = 55)

mpdta$gls = substr(mpdta$countyreal, 1, 2) %in% gls

hmod = etwfe(
   lemp ~ lpop, tvar = year, gvar = first.treat, data = mpdta,
   vcov = ~countyreal,
   xvar = gls           ## <= het. TEs by gls
   )

# Heterogeneous ATEs (could also specify "event", etc.)

emfx(hmod)

# To test whether the ATEs across these two groups (non-GLS vs GLS) are
# statistically different, simply pass an appropriate "hypothesis" argument.

emfx(hmod, hypothesis = "b1 = b2")


#
# Nonlinear model (distribution / link) families
#

# Poisson example

mpdta$emp = exp(mpdta$lemp)

etwfe(
   emp ~ lpop, tvar = year, gvar = first.treat, data = mpdta,
   vcov = ~countyreal,
   family = "poisson"   ## <= family arg for nonlinear options
   ) |>
   emfx("event")

## End(Not run)
```

`etwfe` *Extended two-way fixed effects*

Description

Extended two-way fixed effects

Usage

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

Arguments

`fml` A two-side formula representing the outcome (lhs) and any control variables (rhs), e.g. `y ~ x1 + x2`. If no controls are required, the rhs must take the value of 0 or 1, 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.

`xvar` Optional interacted categorical covariate for estimating heterogeneous treatment effects. Enables recovery of the marginal treatment effect for distinct levels of `xvar`, e.g. "child", "teenager", or "adult". Note that the "x" prefix in "xvar" represents a covariate that is *interacted* with treatment, as opposed to a regular control variable.

`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.

Heterogeneous treatment effects

Specifying `etwfe(..., xvar = <xvar>)` will generate interaction effects for all levels of `<xvar>` as part of the main regression model. The reason that this is useful (as opposed to a regular, non-interacted covariate in the formula RHS) is that it allows us to estimate heterogeneous treatment effects as part of the larger ETWFE framework. Specifically, we can recover heterogeneous treatment effects for each level of `<xvar>` by passing the resulting `etwfe` model object on to `emfx()`.

For example, imagine that we have a categorical variable called "age" in our dataset, with two distinct levels "adult" and "child". Running `emfx(etwfe(..., xvar = age))` will tell us how the efficacy of treatment varies across adults and children. We can then also leverage the in-built hypothesis testing infrastructure of `marginaleffects` to test whether the treatment effect is statistically different across these two age groups; see Examples below. Note the same principles carry over to categorical variables with multiple levels, or even continuous variables (although continuous variables are not as well supported yet).

Performance tips

Under most situations, `etwfe` should complete very quickly. For its part, `emfx` is quite performant too and should take a few seconds or less for datasets under 100k rows. However, `emfx`'s computation time does tend to scale non-linearly with the size of the original data, as well as the number of interactions from the underlying `etwfe` model. Without getting too deep into the weeds, the numerical delta method used to recover the ATEs of interest has to estimate two prediction models for *each* coefficient in the model and then compute their standard errors. So, it's a potentially expensive operation that can push the computation time for large datasets (> 1m rows) up to several minutes or longer.

Fortunately, there are two complementary strategies that you can use to speed things up. The first is to turn off the most expensive part of the whole procedure—standard error calculation—by calling `emfx(..., vcov = FALSE)`. Doing so should bring the estimation time back down to a few seconds or less, even for datasets in excess of a million rows. While the loss of standard errors might not be an acceptable trade-off for projects where statistical inference is critical, the good news is this first strategy can still be combined our second strategy. It turns out that collapsing the data by groups prior to estimating the marginal effects can yield substantial speed gains of its own. Users can do this by invoking the `emfx(..., collapse = TRUE)` argument. While the effect here is not as dramatic as the first strategy, our second strategy does have the virtue of retaining information about the standard errors. The trade-off this time, however, is that collapsing our data does lead to a loss in accuracy for our estimated parameters. On the other hand, testing suggests that this loss in accuracy tends to be relatively minor, with results equivalent up to the 1st or 2nd significant decimal place (or even better).

Summarizing, here's a quick plan of attack for you to try if you are worried about the estimation time for large datasets and models:

1. Estimate `mod = etwfe(...)` as per usual.
2. Run `emfx(mod, vcov = FALSE, ...)`.
3. Run `emfx(mod, vcov = FALSE, collapse = TRUE, ...)`.
4. Compare the point estimates from steps 1 and 2. If they are are similar enough to your satisfaction, get the approximate standard errors by running `emfx(mod, collapse = TRUE, ...)`.

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.* The Econometrics Journal (forthcoming). Available: http://dx.doi.org/10.2139/ssrn.4183726

See Also

`fixest::feols()`, `fixest::feglm()`

Examples

```
## Not run:
# We'll use the mpdta dataset from the did package (which you'll need to
# install separately).

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

#
# Basic example
#

# The basic ETWFE workflow involves two steps:

# 1) Estimate the main regression model with etwfe().

mod = etwfe(
    fml  = lemp ~ lpop, # outcome ~ controls (use 0 or 1 if none)
    tvar = year,        # time variable
    gvar = first.treat, # group variable
    data = mpdta,       # dataset
    vcov = ~countyreal  # vcov adjustment (here: clustered by county)
    )
```

```
# mod ## A fixest model object with fully saturated interaction effects.

# 2) Recover the treatment effects of interest with emfx().

emfx(mod, type = "event") # dynamic ATE a la an event study

# Etc. Other aggregation type options are "simple" (the default), "group"
# and "calendar"


#
# Heterogeneous treatment effects
#

# Example where we estimate heterogeneous treatment effects for counties
# within the 8 US Great Lake states (versus all other counties).

gls = c("IL" = 17, "IN" = 18, "MI" = 26, "MN" = 27,
        "NY" = 36, "OH" = 39, "PA" = 42, "WI" = 55)

mpdta$gls = substr(mpdta$countyreal, 1, 2) %in% gls

hmod = etwfe(
   lemp ~ lpop, tvar = year, gvar = first.treat, data = mpdta,
   vcov = ~countyreal,
   xvar = gls           ## <= het. TEs by gls
   )

# Heterogeneous ATEs (could also specify "event", etc.)

emfx(hmod)

# To test whether the ATEs across these two groups (non-GLS vs GLS) are
# statistically different, simply pass an appropriate "hypothesis" argument.

emfx(hmod, hypothesis = "b1 = b2")


#
# Nonlinear model (distribution / link) families
#

# Poisson example

mpdta$emp = exp(mpdta$lemp)

etwfe(
   emp ~ lpop, tvar = year, gvar = first.treat, data = mpdta,
   vcov = ~countyreal,
   family = "poisson"   ## <= family arg for nonlinear options
   ) |>
   emfx("event")
```

```
## End(Not run)
```

Index