

# Package ‘etwfe’

September 3, 2025

**Type** Package

**Title** Extended Two-Way Fixed Effects

**Version** 0.6.0

**Date** 2025-09-02

**Description** Convenience functions for implementing extended two-way fixed effect regressions a la Wooldridge (2021, 2023)  
<[doi:10.2139/ssrn.3906345](https://doi.org/10.2139/ssrn.3906345)>, <[doi:10.1093/ectj/utad016](https://doi.org/10.1093/ectj/utad016)>.

**License** MIT + file LICENSE

**Imports** fixest (>= 0.11.2), stats, data.table, Formula,  
marginaleffects (>= 0.29.0), tinyplot (>= 0.4.2)

**Suggests** did, broom, modelsummary (>= 2.2.0), ggplot2, knitr,  
rmarkdown, tinytest

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**URL** <https://grantmcdermott.com/etwfe/>

**BugReports** <https://github.com/grantmcdermott/etwfe/issues>

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Grant McDermott [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-7883-8573>>),  
Frederic Kluser [ctb]

**Maintainer** Grant McDermott <gmcd@amazon.com>

**Repository** CRAN

**Date/Publication** 2025-09-03 06:50:13 UTC

## Contents

emfx . . . . .	2
etwfe . . . . .	7
plot.emfx . . . . .	12

emfx

*Post-estimation aggregation of ETWFE results***Description**

Companion function to `etwfe`, enabling the recovery of aggregate treatment effects along different dimensions of interest (e.g. an event study of dynamic average treatment effects). `emfx` is a light wrapper around the [slopes](#) function from the **marginaleffects** package.

**Usage**

```
emfx(
  object,
  type = c("simple", "group", "calendar", "event"),
  by_xvar = "auto",
  compress = "auto",
  collapse = compress,
  predict = c("response", "link"),
  post_only = TRUE,
  window = NULL,
  lean = FALSE,
  ...
)
```

**Arguments**

<code>object</code>	An <code>etwfe</code> model object.
<code>type</code>	Character. The desired type of post-estimation aggregation.
<code>by_xvar</code>	Logical. Should the results account for heterogeneous treatment effects? Only relevant if the preceding <code>etwfe</code> call included a specified <code>xvar</code> argument, i.e. interacted categorical covariate. The default behaviour ("auto") is to automatically estimate heterogeneous treatment effects for each level of <code>xvar</code> if these are detected as part of the underlying <code>etwfe</code> model object. Users can override by setting to either <code>FALSE</code> or <code>TRUE</code> . See the "Heterogeneous treatment effects" section below.
<code>compress</code>	Logical. Compress the data by (period by cohort) groups before calculating marginal effects? This trades off a slight loss in precision (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 compress if the original dataset has more than 500,000 rows. Users can override by setting either <code>FALSE</code> or <code>TRUE</code> . Note that collapsing by group is only valid if the preceding <code>etwfe</code> call was run with <code>ivar = NULL</code> (the default). See the "Performance tips" section below.

collapse	Logical. An alias for compress (only used for backwards compatibility and ignored if both arguments are provided). The behaviour is identical, but it will trigger a message nudging users to rather use the compress argument.
predict	Character. The type (scale) of prediction used to compute the marginal effects. If "response" (the default), then the output is at the level of the response variable, i.e. it is the expected predictor $E(Y X)$ . If "link", the value returned is the linear predictor of the fitted model, i.e. $X \cdot \beta$ . The difference should only matter for nonlinear models. (Note: This argument is typically called type when use in <code>predict</code> or <code>slopes</code> , but we rename it here to avoid a clash with the top-level type argument above.)
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 (TRUE) 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 FALSE is <i>strictly</i> performative: the "zero" treatment effects for any pre-treatment periods is purely an artefact of the estimation setup.
window	Numeric of length 1 or 2. Limits the temporal window of consideration around treatment. <ul style="list-style-type: none"> <li>• NULL (default): Include all available periods.</li> <li>• Length 1: Truncate to a symmetric window around the treatment event. E.g., window = 2 will truncate to two pre-treatment periods and two post-treatment periods.</li> <li>• Length 2: Asymmetric window, where the first number gives the maximum number of pre-treatment periods and the second number gives the maximum number of post-treatment periods. E.g., window = c(5, 2) will truncate to five pre-treatment periods and two post-treatment periods.</li> </ul> <p>Note that the pre-treatment truncation is only ever binding in cases where the "never" treated group is used as a control, i.e., <code>etwfe(..., cgroup = "never")</code> in the original call.</p>
lean	Logical. Default is FALSE. Switching to TRUE enforces a lean return object; namely a simple data.frame of the main results, stripped of ancillary attributes. Note that this will disable some advanced <code>margineffects</code> post-processing features, but those are unlikely to be used in the emfx context. The upside is a potentially dramatic reduction in the size of the return object. Consequently, we may change the default to TRUE in a future version of <b>etwfe</b> .
...	Additional arguments passed to <code>margineffects::slopes</code> . For example, you can pass <code>vcov = FALSE</code> 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. <code>hypothesis = "b1 = b2"</code> .

## Value

A `data.frame` of aggregated treatment effects along the dimension(s) of interested. Note that this `data.frame` will have been overloaded with the `slopes` class, and so will come with a special print method. But the underlying columns will usually include:

- `term`
- `contrast`
- `<type>` (i.e., the name of your type string)
- `estimate`
- `std.error`
- `statistic`
- `p.value`
- `s.value`
- `conf.low`
- `conf.high`

## 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).

## References

Wong, Jeffrey *et al.* (2021). *You Only Compress Once: Optimal Data Compression for Estimating Linear Models*. Working paper (version: March 16, 2021). Available: <https://doi.org/10.48550/arXiv.2102.11297>

## See Also

`marginaleffects::slopes` which does the heavily lifting behind the scenes. `etwfe` is the companion estimating function that should be run before `emfx`.

## 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 consecutive function calls:
# 1) `etwfe` and 2) `emfx`

# 1) `etwfe`: Estimate a regression model with saturated interaction terms.
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) `emfx`: Recover the treatment effects of interest.

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

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

# To visualize results, use the native plot method (see `?plot.emfx`)
plot(mod_es)

# Notice that we don't get any pre-treatment effects with the default
# "notyet" treated control group. Switch to the "never" treated control
# group if you want this.
etwfe(
  lemp ~ lpop, tvar = year, gvar = first.treat, data = mpdta,
  vcov = ~countyreal,
  cgroup = "never"    ## <= use never treated group as control
) |>
  emfx("event") |>
  plot()

#
# 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")

plot(emfx(hmod))

#
# 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

Estimates an "extended" two-way fixed effects regression, with fully saturated interaction effects *a la* Wooldridge (2021, 2023). At its heart, `etwfe` is a convenience function that automates a number of tedious and error prone preparation steps involving both the data and model formulae. Computation is passed on to the `feols` (linear) / `feglm` (nonlinear) functions from the `fixest` package. `etwfe` should be paired with its companion `emfx` function.

## Usage

```

etwfe(
  fml = NULL,
  tvar = NULL,
  gvar = NULL,
  data = NULL,
  ivar = NULL,
  xvar = NULL,
  tref = NULL,
  gref = NULL,
  cgroup = c("notyet", "never"),
  fe = NULL,
  family = NULL,
  ...
)

```

## Arguments

`fml` A two-side formula representing the outcome (lhs) and any control variables (rhs), e.g.  $y \sim x_1 + x_2$ . If no controls are required, the rhs must take the value of 0 or 1, e.g.  $y \sim 0$ .

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 <i>interacted</i> 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 category of fixed effects should be used? One of either "vs" (varying slopes), "feo" (fixed effects only), or "none" (no fixed effects whatsoever). If left as NULL (i.e., no explicit choice) then will default to "vs" for linear/Gaussian models, since this is the most efficient estimation option and further limits the number of "nuisance" parameters in the return model object. However, for non-Gaussian families will default to "none", since the downstream <code>emfx</code> function cannot compute standard errors for these models in the presence of fixed-effects. (See: <a href="https://github.com/vincentarelbundock/marginaleffects/issues/1487">https://github.com/vincentarelbundock/marginaleffects/issues/1487</a> ) Please note that the primary treatment parameters of interest should remain unchanged regardless of fe argument choice.
family	Which <code>family</code> to use for the estimation. Defaults to NULL, in which case <code>fixest::feols</code> is used. Otherwise passed to <code>fixest::feglm</code> , so that valid entries include "logit", "poisson", and "negbin". Note that if a non-NULL family entry is detected, <code>ivar</code> will automatically be set to NULL.
...	Additional arguments passed to <code>fixest::feols</code> (or <code>fixest::feglm</code> ). The most common example would be a <code>vcov</code> argument.

## Value

A `fixest` object with fully saturated interaction effects, and a few additional attributes used for post-estimation in `emfx`.



## 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 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. (2023). *Simple Approaches to Nonlinear Difference-in-Differences with Panel Data*. *The Econometrics Journal*, 26(3), C31-C66. Available: <https://doi.org/10.1093/ectj/utad016>

## See Also

`fixest::feols()`, `fixest::feglm()` which power the underlying estimation routines. `emfx` is a companion function that handles post-estimation aggregation to extract quantities of interest.

## 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 consecutive function calls:
# 1) `etwfe` and 2) `emfx`

# 1) `etwfe`: Estimate a regression model with saturated interaction terms.
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) `emfx`: Recover the treatment effects of interest.

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

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

# To visualize results, use the native plot method (see `?plot.emfx`)
plot(mod_es)

# Notice that we don't get any pre-treatment effects with the default
# "notyet" treated control group. Switch to the "never" treated control
```

```

# group if you want this.
etwfe(
  lemp ~ lpop, tvar = year, gvar = first.treat, data = mpdta,
  vcov = ~countyreal,
  cgroup = "never"    ## <= use never treated group as control
) |>
emfx("event") |>
plot()

#
# 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")

plot(emfx(hmod))

#
# 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)
```

---

```
plot.emfx
```

```
Plot method for emfx objects
```

---

## Description

Visualize the results of an [emfx](#) call.

## Usage

```
## S3 method for class 'emfx'
plot(
  x,
  type = c("pointrange", "errorbar", "ribbon"),
  pch = 16,
  zero = TRUE,
  grid = TRUE,
  ref = -1,
  ...
)
```

## Arguments

x	An emfx object.
type	Character. The type of plot display. One of "pointrange" (default), "errorbar", or "ribbon".
pch	Integer or character. Which plotting character or symbol to use (see <a href="#">points</a> ). Defaults to 16 (i.e., small solid circle). Ignored if type = "ribbon".
zero	Logical. Should 0-zero line be emphasized? Default is TRUE.
grid	Logical. Should a background grid be displayed? Default is TRUE.
ref	Integer. Reference line marker for event-study plot. Default is -1 (i.e., the period immediately preceding treatment). To remove completely, set to NA, NULL, or FALSE. Only used if the underlying object was computed using <code>emfx(..., type = "event")</code> .
...	Additional arguments passed to <code>tinyplot::tinyplot</code> .

## Value

No return value, called for side effect of producing a plot.

**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 consecutive function calls:
# 1) `etwfe` and 2) `emfx`

# 1) `etwfe`: Estimate a regression model with saturated interaction terms.
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) `emfx`: Recover the treatment effects of interest.

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

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

# To visualize results, use the native plot method (see `?plot.emfx`)
plot(mod_es)

# Notice that we don't get any pre-treatment effects with the default
# "notyet" treated control group. Switch to the "never" treated control
# group if you want this.
etwfe(
  lemp ~ lpop, tvar = year, gvar = first.treat, data = mpdta,
  vcov = ~countyreal,
  cgroup = "never" ## <= use never treated group as control
) |>
emfx("event") |>
plot()

#
# 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")

plot(emfx(hmod))

#
# 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

`emfx`, [2](#), [7](#), [8](#), [10](#), [12](#)  
`etwfe`, [5](#), [7](#)

`family`, [8](#)  
`feglm`, [7](#)  
`feols`, [7](#)  
`fixest`, [8](#)  
`fixest::feglm`, [8](#)  
`fixest::feglm()`, [10](#)  
`fixest::feols`, [8](#)  
`fixest::feols()`, [10](#)

`marginaleffects::slopes`, [3](#), [5](#)

`plot.emfx`, [12](#)  
`points`, [12](#)  
`predict`, [3](#)

`slopes`, [2–4](#)

`tinypplot::tinypplot`, [12](#)