Package: fetwfe (via r-universe)

October 28, 2026

Title Fused Extended Two-Way Fixed Effects

Version 1.56.18.9000

Maintainer Gregory Faletto <gfaletto@gmail.com>

Depends R (>= 4.4.0)

Description Calculates the fused extended two-way fixed effects (FETWFE) estimator for unbiased and efficient estimation of difference-in-differences in panel data with staggered treatment adoption. This estimator eliminates bias inherent in conventional two-way fixed effects estimators, while also employing a novel bridge regression regularization approach to improve efficiency and yield valid standard errors. Also implements extended TWFE (etwfe) and bridge-penalized ETWFE (betwfe). Provides S3 classes for streamlined workflow and supports flexible tuning (ridge and rank-condition guarantees), automatic covariate centering/scaling, and detailed overall and cohort-specific effect estimates with valid standard errors. Includes simulation and formatting utilities, extensive diagnostic tools, vignettes, and examples. See Faletto (2025) (<doi:10.48550/arXiv.2312.05985>).

URL https://gregfaletto.github.io/fetwfePackage/, https://github.com/gregfaletto/fetwfePackage

BugReports https://github.com/gregfaletto/fetwfePackage/issues

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports generics, glmnet, grpreg, Matrix (>= 1.6-0), mvtnorm

Suggests bacondecomp, knitr, rmarkdown, dplyr, did, expm, ggplot2, broom, lme4, testthat (>= 3.2.0), tibble

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/Needs/website pkgdown

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

Date/Publication 2026-10-28 05:00:03 UTC

RemoteUrl https://github.com/gregfaletto/fetwfepackage

RemoteRef HEAD

RemoteSha d0a61a59ff7f528452c8599d3df6e420954d9850

Contents

[.catt_df . . . 3
[[.catt_df . . . 4
[[<-.catt_df . . . 4
[<-.catt_df . . . 5
$.catt_df . . . 6
$<-.catt_df . . . 6
attgtToFetwfeDf . . . 7
augment.betwfe . . . 9
augment.etwfe . . . 10
augment.fetwfe . . . 10
augment.twfeCovs . . . 11
betwfe . . . 12
betwfe-class . . . 20
betwfeWithSimulatedData . . . 20
cohortStudy . . . 27
cohortTimeATTs . . . 28
debiasedATT . . . 30
debiasedATTWithSimulatedData . . . 35
etwfe . . . 36
etwfe-class . . . 42
etwfeToFetwfeDf . . . 42
etwfeWithSimulatedData . . . 44
eventStudy . . . 48
fetwfe . . . 51
fetwfe-class . . . 61
FETWFE_coefs-class . . . 61
FETWFE_simulated-class . . . 62
FETWFE_tes-class . . . 62
fetwfeWithSimulatedData . . . 62
genCoefs . . . 70
genCoefsCore . . . 75
getTes . . . 78
glance.betwfe . . . 80
glance.etwfe . . . 81
glance.fetwfe . . . 81
glance.twfeCovs . . . 82
plot.betwfe . . . 83
plot.etwfe . . . 84

plot.fetwfe . . . 85
plot.simultaneous_cis . . . 87
plot.twfeCovs . . . 87
print.debiased_att . . . 88
print.simultaneous_cis . . . 88
simulateData . . . 89
simulateDataCore . . . 92
simultaneousCIs . . . 95
tidy.betwfe . . . 101
tidy.cohortStudy . . . 102
tidy.cohortTimeATTs . . . 103
tidy.debiased_att . . . 104
tidy.etwfe . . . 104
tidy.eventStudy . . . 105
tidy.fetwfe . . . 106
tidy.FETWFE_tes . . . 107
tidy.simultaneous_cis . . . 108
tidy.twfeCovs . . . 109
twfeCovs . . . 110
twfeCovs-class . . . 116
twfeCovsWithSimulatedData . . . 116

Index **121**

`[.catt_df` *Single-bracket access on a* `catt_df` *object*

Description

Intercepts column selection by the pre-1.11.0 Title-Case names and stops with a migration message. The check fires when an old name appears in the column-selector position: `j` for the `df[i, j]` two-index form, or `i` for the `df[j]` one-index column-selection form. Row-only access (`df[i, ]`) and access by new column names fall through to the `data.frame` method via `NextMethod()`.

Usage

```
## S3 method for class 'catt_df'
x[i, j, ...]
```

Arguments

`x`	A `catt_df` object.
`i, j`	Row / column selectors; see `[.data.frame`.
`...`	Further arguments passed through.

Value

The selected subset, as for `[.data.frame`.

[[.catt_df *Double-bracket access on a* catt_df *object*

Description

Intercepts access by the pre-1.11.0 Title-Case column names (Cohort, Estimated TE, SE, ConfIntLow, ConfIntHigh, P_value) and stops with a migration message pointing to the new snake_case name. All other access falls through to the data.frame method via NextMethod().

Usage

```
## S3 method for class 'catt_df'
x[[i, ...]]
```

Arguments

x	A catt_df object (data frame with class c("catt_df", "data.frame")).
i	Index; passed through to [[.data.frame. Character indices matching an old column name are intercepted; other indices fall through.
...	Further arguments passed through.

Value

The column or value, as for [[.data.frame.

[[<-.catt_df *Double-bracket assignment on a* catt_df *object*

Description

Intercepts assignment by the pre-1.11.0 Title-Case column names (e.g., df[["Estimated TE"]] <- v) and stops with a migration message pointing to the new snake_case name. This closes the gap where a partial migration (RHS updated to new name, LHS still old) would silently append a new column rather than overwriting the renamed one. All other assignment falls through to the data.frame method via NextMethod().

Usage

```
## S3 replacement method for class 'catt_df'
x[[i, ...]] <- value
```

Arguments

x	A catt_df object.
i	Index; passed through to [[<-.data.frame. Character indices matching an old column name are intercepted.
...	Further arguments passed through.
value	The value to assign.

Value

The modified catt_df object, as for [[<-.data.frame.

[<-.catt_df *Single-bracket assignment on a* catt_df *object*

Description

Intercepts column assignment by the pre-1.11.0 Title-Case names and stops with a migration message. The check fires when an old name appears in the column-selector position. Row-only assignment (df[i,] <- v) and assignment by new column names fall through to the data.frame method via NextMethod().

Usage

```
## S3 replacement method for class 'catt_df'
x[i, j, ...] <- value
```

Arguments

x	A catt_df object.
i, j	Row / column selectors; see [<-.data.frame.
...	Further arguments passed through.
value	The value to assign.

Details

The nargs() distinction here mirrors the read-side [.catt_df, but the threshold shifts by one because [<- carries an extra positional value argument: df[i] <- v has nargs() == 3 and i is the column selector; df[i, j] <- v and df[i,] <- v have nargs() == 4 and j (if non-missing) is the column selector.

Value

The modified catt_df object, as for [<-.data.frame.

$.catt_df *Dollar-sign access on a* catt_df *object*

Description

Intercepts access by the pre-1.11.0 Title-Case column names and stops with a migration message. All other access falls through to the data.frame method via NextMethod().

Usage

```
## S3 method for class 'catt_df'
x$name
```

Arguments

x	A catt_df object.
name	Character; the column name being accessed via $.

Value

The column, as for $.data.frame.

$<-.catt_df *Dollar-sign assignment on a* catt_df *object*

Description

Intercepts assignment by the pre-1.11.0 Title-Case column names (e.g., df$Cohort <- v) and stops with a migration message. All other assignment falls through to the data.frame method via NextMethod().

Usage

```
## S3 replacement method for class 'catt_df'
x$name <- value
```

Arguments

x	A catt_df object.
name	Character; the column name being assigned via $.
value	The value to assign.

Value

The modified catt_df object, as for $<-.data.frame.

`attgtToFetwfeDf` *Convert data formatted for* `att_gt()` *to a dataframe suitable for* `fetwfe()` / `etwfe()`

Description

`attgtToFetwfeDf()` reshapes and renames a panel dataset that is already formatted for `did::att_gt()` (Callaway and Sant'Anna 2021) so that it can be passed directly to `fetwfe()` or `etwfe()` from the `fetwfe` package. In particular, it

- creates an *absorbing-state* treatment dummy that equals 1 from the first treated period onward* and 0 otherwise,
- (optionally) drops units that are already treated in the very first period of the sample (because `fetwfe()` removes them internally), and
- returns a tidy dataframe whose column names match the arguments that `fetwfe()`/`etwfe()` expect.

Usage

```
attgtToFetwfeDf(
  data,
  yname,
  tname,
  idname,
  gname,
  covars = character(0),
  drop_first_period_treated = TRUE,
  out_names = list(time = "time_var", unit = "unit_var", treatment = "treatment",
    response = "response"),
  verbose = FALSE
)
```

Arguments

`data`	A `data.frame` in **long** format containing at least the four columns used by `did::att_gt()`: outcome `yname`, time `tname`, unit id `idname`, and the first-treatment period `gname` (which is 0 for the never-treated group).
`yname`	Character scalar. Name of the outcome column.
`tname`	Character scalar. Name of the time variable (numeric or integer). This becomes `time_var` in the returned dataframe.
`idname`	Character scalar. Name of the unit identifier. Converted to character and returned as `unit_var`.
`gname`	Character scalar. Name of the *group* variable holding the first period of treatment. Values must be 0 (or `Inf`, a common never-treated encoding, which is mapped to 0) for never-treated, or a positive integer representing the first treated period.

covars
: Character vector of additional covariate column names to carry through (default `character(0)`). These columns are left untouched and appear *after* the required columns in the returned dataframe.

drop_first_period_treated
: Logical. If `TRUE` (default), units that are already treated in the first sample period are removed *before* creating the treatment dummy. `fetwfe()` would do this internally, but dropping them here keeps the returned dataframe cleaner.

out_names
: A named list giving the column names to use in the resulting dataframe. Defaults are `list(time = "time_var", unit = "unit_var", treatment = "treatment", response = "response")`. Override if you prefer different names (for instance, to keep the original `yname`). The vector *must* contain exactly these four names.

verbose
: Logical. If `TRUE`, a `message()` reports the count of first-period-treated unit-period rows dropped when `drop_first_period_treated = TRUE`. Default `FALSE` (silent).

Value

A `data.frame` with columns `time_var`, `unit_var`, `treatment`, `response`, and any covariates requested in `covars`, ready to be fed to `fetwfe()`/`etwfe()`. All required columns are of the correct type: `time_var` is integer, `unit_var` is character, `treatment` is integer 0/1, and `response` is numeric.

References

Callaway, Brantly and Pedro H.C. Sant'Anna. "Difference-in- Differences with Multiple Time Periods." Journal of Econometrics, Vol. 225, No. 2, pp. 200-230, 2021. doi:10.1016/j.jeconom.2020.12.001, https://arxiv.org/abs/1803.09015.

Examples

```
## toy example ------------------------------------------------------------
## Not run:
library(did)  # provides the mpdta example dataframe
data(mpdta)

head(mpdta)

tidy_df <- attgtToFetwfeDf(
  data  = mpdta,
  yname = "lemp",
  tname = "year",
  idname = "countyreal",
  gname = "first.treat",
  covars = c("lpop"))

head(tidy_df)

## End(Not run)

## Now you can call fetwfe()  ---------------------------------------------
```

```
# res <- fetwfe(
#   pdata      = tidy_df,
#   time_var   = "time_var",
#   unit_var   = "unit_var",
#   treatment  = "treatment",
#   response   = "response",
#   covs       = c("lpop"))
```

`augment.betwfe` *Augment user-supplied data with fitted values and residuals from a betwfe fit*

Description

Same shape as `augment.fetwfe()`, dispatched on class `"betwfe"`. `data` is auto-sorted by `(unit, time)` and any first-period-treated units are auto-trimmed; pass the same raw `pdata` you handed to `betwfe()`.

Usage

```
## S3 method for class 'betwfe'
augment(x, data, ...)
```

Arguments

`x`	An object of class `"betwfe"`.
`data`	A panel `data.frame` with the response column under `x$response_col_name`. Any sort order; first-period-treated units are auto-trimmed.
`...`	Unused.

Value

`data` with `.fitted` and `.resid` columns appended.

Examples

```
## Not run:
  sim <- simulateData(genCoefs(G = 3, T = 6, d = 2, density = 0.5,
                               eff_size = 2),
                      N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  res <- betwfeWithSimulatedData(sim)
  broom::augment(res, data = sim$pdata)

## End(Not run)
```

augment.etwfe — *Augment user-supplied data with fitted values and residuals from an etwfe fit*

Description

Same shape as `augment.fetwfe()`, dispatched on class `"etwfe"`. `data` is auto-sorted by `(unit, time)` and any first-period-treated units are auto-trimmed; pass the same raw `pdata` you handed to `etwfe()`.

Usage

```
## S3 method for class 'etwfe'
augment(x, data, ...)
```

Arguments

`x`	An object of class `"etwfe"`.
`data`	A panel `data.frame` with the response column under `x$response_col_name`. Any sort order; first-period-treated units are auto-trimmed.
`...`	Unused.

Value

`data` with `.fitted` and `.resid` columns appended.

Examples

```
## Not run:
  sim <- simulateData(genCoefs(G = 3, T = 6, d = 2, density = 0.5,
                               eff_size = 2),
                      N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  res <- etwfeWithSimulatedData(sim)
  broom::augment(res, data = sim$pdata)

## End(Not run)
```

augment.fetwfe — *Augment user-supplied data with fitted values and residuals from a fetwfe fit*

Description

Computes `.fitted = X %*% beta_hat + x$y_mean` and `.resid = data[[x$response_col_name]] - .fitted`, then column-binds those two columns onto `data`. The response mean and column name are stored on the fitted object during fitting (the estimator internally centers `y` before solving), so fitted values come back on the original-response scale without the caller having to remember either.

Usage

```
## S3 method for class 'fetwfe'
augment(x, data, ...)
```

Arguments

x	An object of class "fetwfe".
data	A panel data.frame with one row per unit-period (any sort order — augment auto-sorts), containing the response column under the same name used at fit time (see x$response_col_name). First-period-treated units, if present, are auto-trimmed.
...	Unused.

Details

data is auto-handled to match the fitted design: rows are auto-sorted by (unit, time), and any first-period-treated units (whose treatment effect cannot be identified by the estimator) are auto-trimmed via idCohorts(). So you can pass the same raw pdata you handed to fetwfe() — the method takes care of alignment. The only hard requirement is that data contains the response column under its original name.

Value

A copy of data with two extra numeric columns: .fitted and .resid.

Examples

```
## Not run:
  sim <- simulateData(genCoefs(G = 3, T = 6, d = 2, density = 0.5,
                               eff_size = 2),
                      N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  res <- fetwfeWithSimulatedData(sim)
  broom::augment(res, data = sim$pdata)

## End(Not run)
```

---

augment.twfeCovs *Augment is not defined for a twfeCovs fit (documented omission)*

Description

augment() is intentionally not provided for twfeCovs() objects (#58): the fit's coefficient vector lives in a reduced cohort-level basis that the shared fitted-value path (X %*% beta_hat) does not match, so a meaningful .fitted / .resid cannot be reconstructed. Use tidy.twfeCovs(), glance.twfeCovs(), or summary() instead. Calling this method always raises an error.

Usage

```
## S3 method for class 'twfeCovs'
augment(x, data, ...)
```

Arguments

x	An object of class "twfeCovs".
data	Ignored.
...	Ignored.

Value

(none; raises an error).

---

betwfe *Bridge-penalized extended two-way fixed effects*

Description

Implementation of extended two-way fixed effects with a bridge penalty. Estimates overall ATT as well as CATT (cohort average treatment effects on the treated units).

Usage

```
betwfe(
  pdata,
  time_var,
  unit_var,
  treatment,
  response,
  covs = c(),
  indep_counts = NA,
  sig_eps_sq = NA,
  sig_eps_c_sq = NA,
  lambda.max = NA,
  lambda.min = NA,
  nlambda = 100,
  q = 0.5,
  verbose = FALSE,
  alpha = 0.05,
  add_ridge = FALSE,
  allow_no_never_treated = TRUE,
  se_type = "default",
  lambda_selection = "cv",
  cv_folds = 10L,
  cv_seed = NULL,
  ci_type = c("simultaneous", "pointwise")
)
```

Arguments

`pdata`	Dataframe; the panel data set. Each row should represent an observation of a unit at a time. Should contain columns as described below.
`time_var`	Character; the name of a single column containing a variable for the time period. This column is expected to contain integer values (for example, years). Recommended encodings for dates include format YYYY, YYYYMM, or YYYYMMDD, whichever is appropriate for your data.
`unit_var`	Character; the name of a single column containing a variable for each unit. This column is expected to contain character values (i.e. the "name" of each unit).
`treatment`	Character; the name of a single column containing a variable for the treatment dummy indicator. This column is expected to contain integer values, and in particular, should equal 0 if the unit was untreated at that time and 1 otherwise. Treatment should be an absorbing state; that is, if unit `i` is treated at time `t`, then it must also be treated at all times `t + 1`, ..., `T`. Any units treated in the first time period will be removed automatically. Please make sure yourself that at least some units remain untreated at the final time period ("never-treated units").
`response`	Character; the name of a single column containing the response for each unit at each time. The response must be an integer or numeric value.
`covs`	(Optional.) Either a character vector containing the names of the columns for covariates (e.g., `covs = c("x1", "x2")`), or a one-sided formula (e.g., `covs = ~ x1 + x2`) – the formula form mirrors the convention used by `did::att_gt(xformla = ...)`. Only additive bare variable names are supported in the formula form; for derived variables, compute them in the data frame first and pass via the character-vector form. All of these columns are expected to contain integer, numeric, or factor values, and any categorical values will be automatically encoded as binary indicators. If no covariates are provided, the treatment effect estimation will proceed, but it will only be valid under unconditional versions of the parallel trends and no anticipation assumptions. Default is c().
`indep_counts`	(Optional.) Integer; a vector. If you have a sufficiently large number of units, you can optionally randomly split your data set in half (with `N` units in each data set). The data for half of the units should go in the `pdata` argument provided above. For the other `N` units, simply provide the counts for how many units appear in the untreated cohort plus each of the other `G` cohorts in this argument `indep_counts`. The benefit of doing this is that the standard error for the average treatment effect will be (asymptotically) exact instead of conservative. The length of `indep_counts` must equal 1 plus the number of treated cohorts in `pdata`. All entries of `indep_counts` must be strictly positive (if you are concerned that this might not work out, maybe your data set is on the small side and it's best to just leave your full data set in `pdata`). The sum of all the counts in `indep_counts` must match the total number of units in `pdata`. Default is NA (in which case conservative standard errors will be calculated if `q < 1`.)
`sig_eps_sq`	(Optional.) Numeric; the variance of the row-level IID noise assumed to apply to each observation. See Section 2 of Faletto (2025) for details. It is best to provide this variance if it is known (for example, if you are using simulated data). If this variance is unknown, this argument can be omitted, and the variance will be estimated by REML on the linear mixed-effects model `y ~ X + (1 | unit)` via

lme4::lmer (Bates et al. 2015; Patterson & Thompson 1971). When supplied, the value also sets the scale of the bridge penalty's grid (the response is divided by sqrt(sig_eps_sq) before the fit), so at q != 1 a value supplied in the wrong units moves the estimates, not only their standard errors. Default is NA.

sig_eps_c_sq (Optional.) Numeric; the variance of the unit-level IID noise (random effects) assumed to apply to each observation. See Section 2 of Faletto (2025) for details. It is best to provide this variance if it is known (for example, if you are using simulated data). If this variance is unknown, this argument can be omitted, and the variance will be estimated by REML via lme4::lmer on the linear mixed-effects model y ~ X + (1 | unit) (Bates et al. 2015; Patterson & Thompson 1971). Default is NA.

lambda.max (Optional.) Numeric. Used only on the BIC route (lambda_selection = "bic"), which selects lambda by BIC over a grid. The largest lambda in the grid will be lambda.max. If no lambda.max is provided, one will be selected automatically. When q <= 1, the model will be sparse, and ideally all of the following are true at once: the smallest model (the one corresponding to lambda.max) selects close to 0 features, the largest model (the one corresponding to lambda.min) selects close to p features, nlambda is large enough so that models are considered at every feasible model size, and nlambda is small enough so that the computation doesn't become infeasible. You may want to manually tweak lambda.max, lambda.min, and nlambda to try to achieve these goals, particularly if the selected model size is very close to the model corresponding to lambda.max or lambda.min, which could indicate that the range of lambda values was too narrow or coarse. You can use the function outputs lambda.max_model_size, lambda.min_model_size, and lambda_star_model_size to try to assess this. Default is NA.

lambda.min (Optional.) Numeric. Used only on the BIC route: the smallest lambda penalty parameter considered. See the description of lambda.max for details. Default is NA.

nlambda (Optional.) Integer. Used only on the BIC route: the total number of lambda penalty parameters considered. See the description of lambda.max for details. Default is 100.

q (Optional.) Numeric; determines what L_q penalty is used for the regularization. q = 1 is the lasso, and for 0 < q < 1, it is possible to get standard errors and confidence intervals. q = 2 is ridge regression. See Faletto (2025) for details. Default is 0.5.

verbose Logical; if TRUE, more details on the progress of the function will be printed as the function executes. Default is FALSE.

alpha Numeric; function will calculate (1 - alpha) confidence intervals for the cohort average treatment effects that will be returned in catt_df.

add_ridge (Optional.) Logical; if TRUE, adds a small amount of ridge regularization to the (untransformed) coefficients to stabilize estimation. Default is FALSE.

allow_no_never_treated
(Optional.) Logical; if TRUE (default) and the input panel contains no never-treated units, the panel is auto-truncated by dropping time periods at and after the latest cohort's start time — the units in that latest cohort then serve as the

never-treated comparison group in the retained sub-panel — with a warning naming the dropped periods. If FALSE, the estimator stops with an error in this case (the package's behavior prior to version 1.5.6). The argument has no effect when the input already contains never-treated units. Default is TRUE.

se_type Character; one of "default", "conservative", or "cluster". "default" returns the tight Gaussian variance sqrt(att_var_1 + att_var_2) from Theorem (c$'$) under Assumption (Psi-IF); this is asymptotically exact for the package's default cohort sample-proportions estimator and for every standard propensity-score estimator that satisfies (Psi-IF) (multinomial logit, any GLM on W | X, kernel/series regression of 1{W = g} on X). "conservative" returns the Cauchy-Schwarz upper bound from Theorem (c); use only if the propensity-score estimator violates (Psi-IF). "cluster" is an *experimental* unit-clustered Liang-Zeger sandwich SE on the bridge-selected support (see the companion vignette inference_vignette for details). "cluster" is only meaningful when q < 1 (the bridge oracle property is required); for q >= 1 the SE will be NA regardless of se_type. The default "default" was the conservative Cauchy-Schwarz formula in versions <= 1.11.7; v1.12.0 switched the default to the tight Gaussian variance. Default is "default".

lambda_selection Character; method for selecting the bridge penalty parameter lambda. Either "cv" (10-fold cross-validation on cv.grpreg; the v1.13.0+ default) or "bic" (BIC over the grpreg lambda grid; the prior default for v1.12.0 and earlier). The default changed in v1.13.0 to address a finite-sample bias issue documented in simulation studies (see issue #164). See the inference vignette section "Choosing the bridge penalty parameter" for details.

cv_folds Integer; number of folds for the CV path. Ignored when lambda_selection = "bic". Default is 10.

cv_seed Integer or NULL; the seed passed to set.seed() to draw the cross-validation folds. The folds are row-level: the draw is assigned to the rows unit by unit, with the units in an order computed from their data rather than from their IDs. So relabeling the units does not change which lambda cross-validation selects, except through REML's small dependence on unit order when the noise variances are estimated (#535). If supplied, must be within +/- .Machine$integer.max. If NULL (the default), the seed defaults internally to as.integer(N * T). Ignored when lambda_selection = "bic".

ci_type Character; one of "simultaneous" (default) or "pointwise". Controls the confidence-interval bounds reported for the cohort-specific ATTs (in catt_df) and the event-study effects (from eventStudy(), shown by print / summary / plot, and surfaced by broom::tidy() on the fitted object and on the eventStudy() / cohortStudy() outputs). "simultaneous" reports parametric simultaneous (family-wise, uniform) bands computed via simultaneousCIs(): each family's band covers all of its effects jointly with probability 1 - alpha, matching the default presentation of did::aggte(cband = TRUE). "pointwise" reports per-effect Wald intervals (each covers its own effect with probability 1 - alpha, no joint guarantee — the behavior of versions <= 1.15.1). Both the interval bounds and the per-cohort p-values (p_value) follow ci_type: under "simultaneous" the p_value is the single-step max-T multiplicity- adjusted p-value matching the

band, under "pointwise" the per-cohort Wald p-value (#200). The standard errors (se) and selection flags (selected) are identical under both settings, and the overall-ATT confidence interval (a single scalar) is unaffected. When standard errors are unavailable (q >= 1, or a rank-deficient design) the bounds are NA under both settings. Default is "simultaneous".

Value

An object of class betwfe containing the following elements:

- **att_hat** The estimated overall average treatment effect for a randomly selected treated unit.
- **att_se** If q < 1, a standard error for the ATT. If indep_counts was provided, this standard error is asymptotically exact; if not, it is asymptotically conservative. If q >= 1, this will be NA.
- **att_p_value** A two-sided p-value for the overall ATT against the null H_0: tau = 0, computed as 2 * pnorm(-|att_hat / att_se|). NA if att_se is zero or NA (e.g., under the bridge solver's selected-out fallback).
- **att_selected** Logical scalar; TRUE if att_hat is not exactly zero, FALSE otherwise. BETWFE uses bridge regression directly on the coefficients (rather than on the fused restrictions used by FETWFE); under the bridge oracle property of Kock (2013), att_selected = FALSE is an analogous asymptotic statement that the truth is zero under a sparsity assumption different from the one Theorem 6.2 establishes for FETWFE. For ridge (q = 2) the bridge solver does not zero coefficients, so this will typically be TRUE.
- **catt_hats** A named vector containing the estimated average treatment effects for each cohort.
- **catt_ses** If q < 1, a named vector containing the (asymptotically exact, non-conservative) standard errors for the estimated average treatment effects within each cohort.
- **cohort_probs** A vector of the estimated probabilities of being in each cohort conditional on being treated, which was used in calculating att_hat. If indep_counts was provided, cohort_probs was calculated from that; otherwise, it was calculated from the counts of units in each treated cohort in pdata.
- **catt_df** A data frame (with S3 class c("catt_df", "data.frame")) displaying the cohort names (cohort), average treatment effects (estimate), standard errors (se), 1 - alpha confidence interval bounds (ci_low, ci_high), per-cohort p-values (p_value), and a selected logical flag (TRUE when the bridge penalty left the cohort's CATT nonzero). For selected-out cohorts (selected = FALSE), p_value is NA. The catt_df S3 class makes [[/ $ / [access on the pre-1.11.0 Title-Case column names (Cohort, Estimated TE, SE, ConfIntLow, ConfIntHigh, P_value) stop() with a migration message pointing to the new name. See NEWS.md for the rename table.
- **beta_hat** The full vector of estimated coefficients.
- **treat_inds** The indices of beta_hat corresponding to the treatment effects for each cohort at each time.
- **treat_int_inds** The indices of beta_hat corresponding to the interactions between the treatment effects for each cohort at each time and the covariates.

sig_eps_sq — Either the provided sig_eps_sq or the estimated one, if a value wasn't provided.

sig_eps_c_sq — Either the provided sig_eps_c_sq or the estimated one, if a value wasn't provided.

lambda.max — The largest lambda of the grid used, which is a supplied lambda.max only on the BIC route. (This is returned to help with getting a reasonable range of lambda values for grid search.)

lambda.max_model_size — The number of selected features (excluding the always-present intercept) at lambda.max (for q <= 1, the smallest model). As mentioned above, for q <= 1 ideally this value is close to 0.

lambda.min — Either the provided lambda.min or the one that was used, if a value wasn't provided.

lambda.min_model_size — The number of selected features (excluding the always-present intercept) at lambda.min (for q <= 1, the largest model). As mentioned above, for q <= 1 ideally this value is close to p.

lambda_star — The value of lambda chosen by the method recorded in lambda_selection. If this value is close to lambda.min or lambda.max, that could suggest that the range of lambda values should be expanded.

lambda_star_model_size — The number of selected features (excluding the always-present intercept) in the chosen model. If this value is close to lambda.max_model_size or lambda.min_model_size, that could suggest that the range of lambda values should be expanded.

lambda_selection — Character scalar; either "cv" or "bic". Mirrors the lambda_selection argument the user passed.

cv_folds — Integer scalar; the cv_folds value used when lambda_selection = "cv", NA_integer_ when lambda_selection = "bic".

cv_seed — Integer scalar; the seed actually fed to set.seed() to draw the cross-validation folds. Defaults to as.integer(N * T) when the user did not pass a seed. NA_integer_ when lambda_selection = "bic".

ci_type — Character scalar; the ci_type argument the user passed ("simultaneous" or "pointwise"), controlling whether the reported catt_df / eventStudy() confidence-interval bounds are simultaneous (family-wise) or pointwise.

catt_band_applied — Logical scalar; TRUE when the fit-time cohort-family simultaneous band was computed and written into catt_df, and FALSE otherwise — including when ci_type = "pointwise", when standard errors were unavailable, and when the band construction degraded. It records only whether the band was applied: it does not imply the band is wider than the pointwise interval (they coincide when fewer than two effects have positive variance) and does not imply the band is informative (a degenerate fit can carry an all-zero band with TRUE).

X_ints — The design matrix created containing all interactions, time and cohort dummies, etc.

y — The vector of responses, containing nrow(X_ints) entries.

`X_final` The design matrix after applying the change in coordinates to fit the model and also multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.

`y_final` The final response after multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.

`N` The final number of units that were in the data set used for estimation (after any units may have been removed because they were treated in the first time period).

`T` The number of time periods in the final data set.

`G` The final number of treated cohorts that appear in the final data set.

`R` Deprecated alias for `G`, retained for backward compatibility; populated with the same value. Use `G`. Will be removed in a future release.

`d` The final number of covariates that appear in the final data set (after any covariates may have been removed because they contained missing values or all contained the same value for every unit).

`p` The final number of columns in the full set of covariates used to estimate the model.

`y_mean` Numeric scalar; mean of the original (pre-centering) response. Stored so downstream methods (`augment()`, `predict()`) can return fitted values on the original-response scale.

`response_col_name`
Character scalar; the response column name in the original `pdata`. Consumed by `augment.betwfe()`.

`time_var`, `unit_var`, `treatment`
Character scalars; the corresponding arguments the user passed. Consumed by `augment.betwfe()` when auto-aligning a user-supplied panel to the fitted design.

`covs` Character vector; the original `covs` argument (pre-factor- expansion). Consumed by `augment.betwfe()`.

`alpha` The alpha level used for confidence intervals.

`calc_ses` Logical indicating whether standard errors were calculated.

`cohort_probs_overall`
A vector of the estimated cohort probabilities on the overall sample (treated and untreated), used in computing the variance of the overall ATT.

`indep_counts_used`
Logical scalar; `TRUE` if a valid `indep_counts` argument was provided and used for asymptotically-exact ATT inference, `FALSE` otherwise.

`se_type` Character scalar; the `se_type` argument the user passed (`"default"`, `"conservative"`, or `"cluster"`).

`internal` A list containing internal outputs that are typically not needed for interpretation, packaged here for parity with `fetwfe()` so downstream consumers can use a single canonical access path across all four estimator classes (#144). The first five sub-slots (`X_ints`, `y`, `X_final`, `y_final`, `calc_ses`) are also duplicated at top level for backward compat; `variance_components` and `first_year` live only under `$internal`:

X_ints The design matrix containing all interactions, time and cohort dummies, etc. Same value as top-level `X_ints`.

y The vector of responses. Same as top-level `y`.

X_final The design matrix after the change-of-coordinates step. Same as top-level `X_final`.

y_final The transformed response vector. Same as top-level `y_final`.

calc_ses Logical indicating whether standard errors were calculated. Same as top-level `calc_ses`.

variance_components A list exposing the two variance pieces (`att_var_1`, `att_var_2`) plus paper-notation counterparts (`V_1`, `V_2`) and unit-scaled variance estimators (`tilde_v_N`, `hat_v_N`, `tilde_v_N_C`, `tilde_v_N_C_pi_hat`, `tilde_v_N_C_pi_hat_cons`, `tilde_v_N_cons`). The Wald CI is `[hat_T_N +- qnorm(1-alpha/2)` (paper Eq. `conf.int.form`). New in v1.12.0 (issue #141 + #146).

first_year Integer or numeric scalar; the first (earliest) `time_var` value in the panel after `idCohorts()` processing. Consumed by `eventStudy()` to map `cohort_probs`' cohort labels (treatment-start years) to 1-based panel-time-index offsets when the labels are integer-coercible. New in v1.13.3 (issue #174).

Author(s)

Gregory Faletto

References

Faletto, G (2025). Fused Extended Two-Way Fixed Effects for Difference-in-Differences with Staggered Adoptions. *arXiv preprint arXiv:2312.05985*. https://arxiv.org/abs/2312.05985.

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2015). Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software*, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Patterson, H. D., & Thompson, R. (1971). Recovery of inter-block information when block sizes are unequal. *Biometrika*, 58(3), 545-554.

Pinheiro, J. C., & Bates, D. M. (2000). *Mixed-Effects Models in S and S-PLUS*. Springer.

Examples

```
# `bacondecomp` (which supplies the `castle` data) is a Suggests-only
# dependency, so guard the example on its availability.
if (requireNamespace("bacondecomp", quietly = TRUE)) {
  library(bacondecomp)

  data(castle)

  # Response: the log homicide rate. Treatment: `cdl` records the share of
  # the year the castle-doctrine law was in effect, so `cdl > 0` gives the
  # absorbing 0/1 treatment indicator. No `covs`: castle's smallest
  # adoption cohorts contain a single state, so the design is
  # rank-deficient once any covariate is added.
  castle$l_homicide <- log(castle$homicide)
```

```
castle$treated <- as.integer(castle$cdl > 0)

# On this panel betwfe's bridge penalty selects every cohort out, so the
# estimated ATT and cohort effects below are all zero.
res <- betwfe(
    pdata = castle,
    time_var = "year",
    unit_var = "state",
    treatment = "treated",
    response = "l_homicide",
    verbose = TRUE)

# Average treatment effect on the treated units (in percentage point
# units)
100 * res$att_hat

# Conservative 95% confidence interval for ATT (in percentage point units)

low_att <- 100 * (res$att_hat - qnorm(1 - 0.05 / 2) * res$att_se)
high_att <- 100 * (res$att_hat + qnorm(1 - 0.05 / 2) * res$att_se)

c(low_att, high_att)

# Cohort average treatment effects and confidence intervals (in percentage
# point units)

catt_df_pct <- res$catt_df
catt_df_pct[["estimate"]] <- 100 * catt_df_pct[["estimate"]]
catt_df_pct[["se"]] <- 100 * catt_df_pct[["se"]]
catt_df_pct[["ci_low"]] <- 100 * catt_df_pct[["ci_low"]]
catt_df_pct[["ci_high"]] <- 100 * catt_df_pct[["ci_high"]]

catt_df_pct
}
```

---

betwfe-class *Bridge-Penalized Extended Two-Way Fixed Effects Output Class*

Description

S3 class for the output of `betwfe()`.

---

betwfeWithSimulatedData *Run BETWFE on Simulated Data*

Description

This function runs the bridge-penalized extended two-way fixed effects estimator (betwfe()) on simulated data. It is simply a wrapper for betwfe(): it accepts an object of class "FETWFE_simulated" (produced by simulateData()) and unpacks the necessary components to pass to betwfe(). So the outputs match betwfe(), and the needed inputs match their counterparts in betwfe().

Usage

```
betwfeWithSimulatedData(
  simulated_obj,
  lambda.max = NA,
  lambda.min = NA,
  nlambda = 100,
  q = 0.5,
  verbose = FALSE,
  alpha = 0.05,
  add_ridge = FALSE,
  allow_no_never_treated = TRUE,
  se_type = "default",
  lambda_selection = "cv",
  cv_folds = 10L,
  cv_seed = NULL,
  ci_type = c("simultaneous", "pointwise")
)
```

Arguments

simulated_obj	An object of class "FETWFE_simulated" containing the simulated panel data and design matrix.
lambda.max	(Optional.) Numeric. Used only on the BIC route (lambda_selection = "bic"), which selects lambda by BIC over a grid. The largest lambda in the grid will be lambda.max. If no lambda.max is provided, one will be selected automatically. When q <= 1, the model will be sparse, and ideally all of the following are true at once: the smallest model (the one corresponding to lambda.max) selects close to 0 features, the largest model (the one corresponding to lambda.min) selects close to p features, nlambda is large enough so that models are considered at every feasible model size, and nlambda is small enough so that the computation doesn't become infeasible. You may want to manually tweak lambda.max, lambda.min, and nlambda to try to achieve these goals, particularly if the selected model size is very close to the model corresponding to lambda.max or lambda.min, which could indicate that the range of lambda values was too narrow. You can use the function outputs lambda.max_model_size, lambda.min_model_size, and lambda_star_model_size to try to assess this. Default is NA.
lambda.min	(Optional.) Numeric. Used only on the BIC route: the smallest lambda penalty parameter considered. See the description of lambda.max for details. Default is NA.

nlambda (Optional.) Integer. Used only on the BIC route: the total number of lambda penalty parameters considered. See the description of lambda.max for details. Default is 100.

q (Optional.) Numeric; determines what L_q penalty is used for the fusion regularization. q = 1 is the lasso, and for 0 < q < 1, it is possible to get standard errors and confidence intervals. q = 2 is ridge regression. See Faletto (2025) for details. Default is 0.5.

verbose Logical; if TRUE, more details on the progress of the function will be printed as the function executes. Default is FALSE.

alpha Numeric; function will calculate (1 - alpha) confidence intervals for the cohort average treatment effects that will be returned in catt_df.

add_ridge (Optional.) Logical; if TRUE, adds a small amount of ridge regularization to the (untransformed) coefficients to stabilize estimation. Default is FALSE.

allow_no_never_treated
(Optional.) Logical; if TRUE (default) and the input panel contains no never-treated units, the panel is auto-truncated by dropping time periods at and after the latest cohort's start time — the units in that latest cohort then serve as the never-treated comparison group in the retained sub-panel — with a warning naming the dropped periods. If FALSE, the estimator stops with an error in this case (the package's behavior prior to version 1.5.6). The argument has no effect when the input already contains never-treated units. Default is TRUE.

se_type Character; one of "default", "conservative", or "cluster". "default" returns the tight Gaussian variance sqrt(att_var_1 + att_var_2) from Theorem (c$'$) under Assumption (Psi-IF); this is asymptotically exact for the package's default cohort sample-proportions estimator and for every standard propensity-score estimator that satisfies (Psi-IF) (multinomial logit, any GLM on W | X, kernel/series regression of 1{W = g} on X). "conservative" returns the Cauchy-Schwarz upper bound from Theorem (c); use only if the propensity-score estimator violates (Psi-IF). "cluster" is an *experimental* unit-clustered Liang-Zeger sandwich SE on the bridge-selected support (see the companion vignette inference_vignette for details). "cluster" is only meaningful when q < 1 (the bridge oracle property is required); for q >= 1 the SE will be NA regardless of se_type. The default "default" was the conservative Cauchy-Schwarz formula in versions <= 1.11.7; v1.12.0 switched the default to the tight Gaussian variance. Default is "default".

lambda_selection
Character; method for selecting the bridge penalty parameter lambda. Either "cv" (10-fold cross-validation on cv.grpreg; the v1.13.0+ default) or "bic" (BIC over the grpreg lambda grid; the prior default for v1.12.0 and earlier). The default changed in v1.13.0 to address a finite-sample bias issue documented in simulation studies (see issue #164). See the inference vignette section "Choosing the bridge penalty parameter" for details.

cv_folds Integer; number of folds for the CV path. Ignored when lambda_selection = "bic". Default is 10.

cv_seed Integer or NULL; the seed passed to set.seed() to draw the cross-validation folds. The folds are row-level: the draw is assigned to the rows unit by unit, with

the units in an order computed from their data rather than from their IDs. So relabeling the units does not change which lambda cross-validation selects, except through REML's small dependence on unit order when the noise variances are estimated (#535). If supplied, must be within +/- .Machine$integer.max. If NULL (the default), the seed defaults internally to as.integer(N * T). Ignored when lambda_selection = "bic".

ci_type — Character; one of "simultaneous" (default) or "pointwise". Controls the confidence-interval bounds reported for the cohort-specific ATTs (in catt_df) and the event-study effects (from eventStudy(), shown by print / summary / plot, and surfaced by broom::tidy() on the fitted object and on the eventStudy() / cohortStudy() outputs). "simultaneous" reports parametric simultaneous (family-wise, uniform) bands computed via simultaneousCIs(): each family's band covers all of its effects jointly with probability 1 - alpha, matching the default presentation of did::aggte(cband = TRUE). "pointwise" reports per-effect Wald intervals (each covers its own effect with probability 1 - alpha, no joint guarantee — the behavior of versions <= 1.15.1). Both the interval bounds and the per-cohort p-values (p_value) follow ci_type: under "simultaneous" the p_value is the single-step max-T multiplicity- adjusted p-value matching the band, under "pointwise" the per-cohort Wald p-value (#200). The standard errors (se) and selection flags (selected) are identical under both settings, and the overall-ATT confidence interval (a single scalar) is unaffected. When standard errors are unavailable (q >= 1, or a rank-deficient design) the bounds are NA under both settings. Default is "simultaneous".

Value

An object of class betwfe containing the following elements:

att_hat — The estimated overall average treatment effect for a randomly selected treated unit.

att_se — If q < 1, a standard error for the ATT. If indep_counts was provided, this standard error is asymptotically exact; if not, it is asymptotically conservative. If q >= 1, this will be NA.

att_p_value — A two-sided p-value for the overall ATT against the null H_0: tau = 0, computed as 2 * pnorm(-|att_hat / att_se|). NA if att_se is zero or NA (e.g., under the bridge solver's selected-out fallback).

att_selected — Logical scalar; TRUE if att_hat is not exactly zero, FALSE otherwise. BETWFE uses bridge regression directly on the coefficients (rather than on the fused restrictions used by FETWFE); under the bridge oracle property of Kock (2013), att_selected = FALSE is an analogous asymptotic statement that the truth is zero under a sparsity assumption different from the one Theorem 6.2 establishes for FETWFE. For ridge (q = 2) the bridge solver does not zero coefficients, so this will typically be TRUE.

catt_hats — A named vector containing the estimated average treatment effects for each cohort.

catt_ses — If q < 1, a named vector containing the (asymptotically exact, non-conservative) standard errors for the estimated average treatment effects within each cohort.

cohort_probs A vector of the estimated probabilities of being in each cohort conditional on being treated, which was used in calculating att_hat. If indep_counts was provided, cohort_probs was calculated from that; otherwise, it was calculated from the counts of units in each treated cohort in pdata.

catt_df A data frame (with S3 class c("catt_df", "data.frame")) displaying the cohort names (cohort), average treatment effects (estimate), standard errors (se), 1 - alpha confidence interval bounds (ci_low, ci_high), per-cohort p-values (p_value), and a selected logical flag (TRUE when the bridge penalty left the cohort's CATT nonzero). For selected-out cohorts (selected = FALSE), p_value is NA. The catt_df S3 class makes [[/ $ / [access on the pre-1.11.0 Title-Case column names (Cohort, Estimated TE, SE, ConfIntLow, ConfIntHigh, P_value) stop() with a migration message pointing to the new name. See NEWS.md for the rename table.

beta_hat The full vector of estimated coefficients.

treat_inds The indices of beta_hat corresponding to the treatment effects for each cohort at each time.

treat_int_inds The indices of beta_hat corresponding to the interactions between the treatment effects for each cohort at each time and the covariates.

sig_eps_sq Either the provided sig_eps_sq or the estimated one, if a value wasn't provided.

sig_eps_c_sq Either the provided sig_eps_c_sq or the estimated one, if a value wasn't provided.

lambda.max The largest lambda of the grid used, which is a supplied lambda.max only on the BIC route. (This is returned to help with getting a reasonable range of lambda values for grid search.)

lambda.max_model_size
The number of selected features (excluding the always-present intercept) at lambda.max (for q <= 1, the smallest model). As mentioned above, for q <= 1 ideally this value is close to 0.

lambda.min Either the provided lambda.min or the one that was used, if a value wasn't provided.

lambda.min_model_size
The number of selected features (excluding the always-present intercept) at lambda.min (for q <= 1, the largest model). As mentioned above, for q <= 1 ideally this value is close to p.

lambda_star The value of lambda chosen by the method recorded in lambda_selection. If this value is close to lambda.min or lambda.max, that could suggest that the range of lambda values should be expanded.

lambda_star_model_size
The number of selected features (excluding the always-present intercept) in the chosen model. If this value is close to lambda.max_model_size or lambda.min_model_size, that could suggest that the range of lambda values should be expanded.

lambda_selection
Character scalar; either "cv" or "bic". Mirrors the lambda_selection argument the user passed.

`cv_folds` Integer scalar; the `cv_folds` value used when `lambda_selection = "cv"`, `NA_integer_` when `lambda_selection = "bic"`.

`cv_seed` Integer scalar; the seed actually fed to `set.seed()` to draw the cross-validation folds. Defaults to `as.integer(N * T)` when the user did not pass a seed. `NA_integer_` when `lambda_selection = "bic"`.

`ci_type` Character scalar; the `ci_type` argument the user passed (`"simultaneous"` or `"pointwise"`), controlling whether the reported `catt_df` / `eventStudy()` confidence-interval bounds are simultaneous (family-wise) or pointwise.

`catt_band_applied`
Logical scalar; `TRUE` when the fit-time cohort-family simultaneous band was computed and written into `catt_df`, and `FALSE` otherwise — including when `ci_type = "pointwise"`, when standard errors were unavailable, and when the band construction degraded. It records only whether the band was applied: it does not imply the band is wider than the pointwise interval (they coincide when fewer than two effects have positive variance) and does not imply the band is informative (a degenerate fit can carry an all-zero band with `TRUE`).

`X_ints` The design matrix created containing all interactions, time and cohort dummies, etc.

`y` The vector of responses, containing `nrow(X_ints)` entries.

`X_final` The design matrix after applying the change in coordinates to fit the model and also multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.

`y_final` The final response after multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.

`N` The final number of units that were in the data set used for estimation (after any units may have been removed because they were treated in the first time period).

`T` The number of time periods in the final data set.

`G` The final number of treated cohorts that appear in the final data set.

`R` Deprecated alias for `G`, retained for backward compatibility; populated with the same value. Use `G`. Will be removed in a future release.

`d` The final number of covariates that appear in the final data set (after any covariates may have been removed because they contained missing values or all contained the same value for every unit).

`p` The final number of columns in the full set of covariates used to estimate the model.

`alpha` The alpha level used for confidence intervals.

`calc_ses` Logical indicating whether standard errors were calculated.

`cohort_probs_overall`
A vector of the estimated cohort probabilities on the overall sample (treated and untreated), used in computing the variance of the overall ATT.

`indep_counts_used`
Logical scalar; `TRUE` if a valid `indep_counts` argument was provided and used for asymptotically-exact ATT inference, `FALSE` otherwise.

`se_type` Character scalar; the `se_type` argument the user passed (`"default"`, `"conservative"`, or `"cluster"`).

`y_mean` Numeric scalar; mean of the original (pre-centering) response. Stored so downstream methods (`augment()`, `predict()`) can return fitted values on the original-response scale.

`response_col_name` Character scalar; the response column name in the original `pdata`. Consumed by `augment.betwfe()`.

`time_var`, `unit_var`, `treatment` Character scalars; the corresponding arguments the user passed.

`covs` Character vector; the original `covs` argument (pre-factor- expansion).

`internal` A list containing internal outputs that are typically not needed for interpretation, packaged here for parity with `fetwfe()` so downstream consumers can use a single canonical access path across all four estimator classes (#144). The first five sub-slots (`X_ints`, `y`, `X_final`, `y_final`, `calc_ses`) are also duplicated at top level for backward compat; `variance_components` and `first_year` live only under `$internal`:

- **X_ints** The design matrix containing all interactions, time and cohort dummies, etc. Same value as top-level `X_ints`.
- **y** The vector of responses. Same as top-level `y`.
- **X_final** The design matrix after the change-of-coordinates step. Same as top-level `X_final`.
- **y_final** The transformed response vector. Same as top-level `y_final`.
- **calc_ses** Logical indicating whether standard errors were calculated. Same as top-level `calc_ses`.
- **variance_components** A list exposing the two variance pieces (`att_var_1`, `att_var_2`) plus paper-notation counterparts (`V_1`, `V_2`) and unit-scaled variance estimators (`tilde_v_N`, `hat_v_N`, `tilde_v_N_C`, `tilde_v_N_C_pi_hat`, `tilde_v_N_C_pi_hat_cons`, `tilde_v_N_cons`). The Wald CI is `[hat_T_N +- qnorm(1-alpha/2)` (paper Eq. `conf.int.form`). New in v1.12.0 (issue #141 + #146).
- **first_year** Integer or numeric scalar; the first (earliest) `time_var` value in the panel after `idCohorts()` processing. Consumed by `eventStudy()` to map `cohort_probs`' cohort labels (treatment-start years) to 1-based panel-time-index offsets when the labels are integer-coercible. New in v1.13.3 (issue #174).

Examples

```
## Not run:
  # Generate coefficients
  coefs <- genCoefs(G = 5, T = 30, d = 12, density = 0.1, eff_size = 2, seed = 123)

  # Simulate data using the coefficients
  sim_data <- simulateData(coefs, N = 120, sig_eps_sq = 5, sig_eps_c_sq = 5, seed = 123)

  result <- betwfeWithSimulatedData(sim_data)
```

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

`cohortStudy` *Per-cohort average treatment effects*

Description

Extracts per-cohort ATT estimates from a fitted FETWFE / ETWFE / BETWFE / twfeCovs object as a tidy data frame. Parallel to `eventStudy()` for event-time aggregation: the per-cohort information is already available via `result$catt_df`, and `cohortStudy()` surfaces it through a discoverable function with its own help page (so users can reach it via `?cohortStudy` without having to know the slot name).

The function is a pass-through on `result$catt_df` modulo class: the columns, their values, and their order are unchanged. The returned object carries class `c("cohortStudy", "catt_df", "data.frame")`. The `cohortStudy` class dispatches the broom tidier `tidy.cohortStudy()`; the `catt_df` class preserves the helpful-error layer (introduced in fetwfe 1.11.0) that intercepts pre-1.11.0 Title-Case column names (`Cohort`, `Estimated TE`, etc.) with a migration message; the `data.frame` base preserves the standard data-frame methods (`print`, `head`, `nrow`, `dplyr::filter`, etc.).

Usage

```
cohortStudy(result)
```

Arguments

`result`	A fitted object from `fetwfe()`, `etwfe()`, `betwfe()`, or `twfeCovs()` (or their `*WithSimulatedData()` wrapper analogs, which return the same classes).

Value

A data frame with class `c("cohortStudy", "catt_df", "data.frame")` containing one row per treated cohort and columns:

cohort Character; the cohort label (the calendar time at which the cohort first received treatment).

estimate Numeric; the per-cohort ATT estimate.

se Numeric; standard error for the per-cohort ATT (`NA` when the Gram matrix is singular or, for `fetwfe()` / `betwfe()`, the bridge penalty zeroed out the cohort).

ci_low, ci_high Numeric; the stored lower and upper confidence-interval bounds, reflecting the fit's `ci_type` – simultaneous (family-wise) by default, or pointwise `1 - alpha` Wald bounds when the fit used `ci_type = "pointwise"` (`alpha` is the value passed at fit time).

p_value Numeric; follows the fit's `ci_type`. Under `"pointwise"`, the two-sided Wald p-value (`2 * pnorm(-|estimate / se|)`); under `"simultaneous"` (the default), the single-step max-T multiplicity-adjusted (family-wise) p-value matching the simultaneous band (#200). `NA` when `se` is `0` or `NA`.

selected (`fetwfe()` / `betwfe()` only.) Logical; `TRUE` when the bridge penalty left the cohort's ATT nonzero. Absent for `etwfe()` and `twfeCovs()`, which do not perform selection.

Use `tidy(cohortStudy(result))` (with the `broom` package loaded) to reshape to broom convention (`term`, `estimate`, `std.error`, `statistic`, `p.value`, `conf.low`, `conf.high`, optionally `selected`); see `tidy.cohortStudy()`.

See Also

`eventStudy()` for the parallel event-time accessor; `cohortTimeATTs()` for the fully disaggregated per-(cohort, time) accessor; `tidy.cohortStudy()` for broom-shape translation.

Examples

```
## Not run:
  coefs <- genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2)
  dat <- simulateData(coefs, N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  res <- fetwfeWithSimulatedData(dat)
  cs <- cohortStudy(res)
  cs
  # Broom-shape translation:
  if (requireNamespace("broom", quietly = TRUE)) {
    broom::tidy(cs)
  }

## End(Not run)
```

---

cohortTimeATTs *Per-(cohort, time) average treatment effects*

Description

Extracts the fully disaggregated treatment-effect estimates from a fitted FETWFE / ETWFE / BETWFE object: one row for every (`cohort`, `time`) cell, with no averaging over cohorts or over event time. This is the finest-grained view of the estimated effects — the `num_treats` underlying parameters themselves — complementing `cohortStudy()` (which averages each cohort's cells over time) and `eventStudy()` (which averages over cohorts at each event time).

Like `eventStudy()`, this accessor is not available for `twfeCovs()` objects: that estimator has a single treatment-effect parameter per cohort (no per-time disaggregation), so its finest granularity is already `cohortStudy()`.

Standard errors are the per-cell regression standard errors $\sqrt{\sigma_\varepsilon^2\, \psi' G^{-1} \psi / (NT)}$, recomputed at call time from the fit's stored design (the same Gram-matrix machinery `eventStudy()` uses; nothing is added to the fitted object). Because each cell is a single cohort-time parameter, the cohort-probability sampling variance that contributes to the aggregated `cohortStudy()` / `eventStudy()` standard errors is identically zero here, so a cell's SE is a single coefficient's regression SE.

Confidence intervals and p-values are *pointwise* $1-\alpha$ Wald quantities (`estimate +/- z * se`). For simultaneous (family-wise) bands over the cell family, use `simultaneousCIs(result, family = "all_post_treatment")`.

Usage

```
cohortTimeATTs(result, alpha = NULL)
```

Arguments

result	A fitted object from fetwfe(), etwfe(), or betwfe() (or their *WithSimulatedData() wrapper analogs, which return the same classes). twfeCovs() objects are not supported (see Description).
alpha	Numeric in (0, 1); the pointwise confidence level is 1 - alpha. Defaults to the alpha stored on the fit.

Details

The cell standard error is computed from psi, the cell's row of the (selected) treatment-effect design — for fetwfe() the relevant row of the inverse fusion transform D^{-1} in the transformed (theta) coordinate space, for betwfe() / etwfe() a unit selector in the original (beta) coordinate space restricted to the selected support. It is never gated on the point estimate: a cell whose estimate is exactly zero because the penalty fused it away has an all-zero psi and therefore se = 0 (the correct degenerate value), while a cell whose estimate happens to be near zero for other reasons still receives its proper nonzero SE.

Value

A data frame with class c("cohortTimeATTs", "data.frame") containing one row per (cohort, time) treatment-effect cell, sorted by cohort then time, with columns:

cohort Character; the cohort label (the calendar time at which the cohort first received treatment), matching cohortStudy().

time Numeric; the calendar time of the cell, equal to the cohort's adoption time plus the event time (0, 1, ...). Real panels carry their actual calendar times. For synthetic genCoefs() / simulateData() fixtures (whose panel runs 1, ..., T, so the stored first year is 1) this coincides with the 1-based panel-time index. (Only a hand-built or legacy fit with no stored first year falls back to that panel-time index directly.)

estimate Numeric; the cell's ATT estimate.

se Numeric; the pointwise standard error. 0 for a cell zeroed out by the fusion/bridge penalty while other cells survive (fetwfe() / betwfe()); NA when standard errors are unavailable — the fit was computed with q >= 1, the Gram matrix on the selected support is singular, or the penalty zeroed the *entire* treatment block (no cells selected, so there is no support to recompute the Gram from; this matches eventStudy()).

ci_low, ci_high Numeric; the pointwise 1 - alpha Wald bounds estimate -/+ qnorm(1 - alpha/2) * se. (0, 0) for a fused-away cell; NA when se is NA.

p_value Numeric; the two-sided pointwise Wald p-value 2 * pnorm(-|estimate / se|). NA when se is 0 or NA.

selected (fetwfe() / betwfe() only.) Logical; TRUE when the bridge penalty left the cell's estimate nonzero. Absent for etwfe(), which does not perform selection.

Use tidy(cohortTimeATTs(result)) (with the broom package loaded) to reshape to broom convention; see tidy.cohortTimeATTs().

See Also

cohortStudy() for the per-cohort (time-averaged) accessor; eventStudy() for the per-event-time (cohort-averaged) accessor; simultaneousCIs() for simultaneous (family-wise) bands over the cell family (family = "all_post_treatment"); tidy.cohortTimeATTs() for broom-shape translation.

Examples

```
## Not run:
  coefs <- genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2)
  dat <- simulateData(coefs, N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  res <- fetwfeWithSimulatedData(dat)
  cta <- cohortTimeATTs(res)
  cta
  # Broom-shape translation:
  if (requireNamespace("broom", quietly = TRUE)) {
    broom::tidy(cta)
  }

## End(Not run)
```

---

debiasedATT *Debiased overall-ATT with a uniformly-valid standard error*

Description

Computes a *debiased* estimate of the overall average treatment effect on the treated (ATT) from a fitted fetwfe() object, together with a uniformly-valid standard error. This is a complement to the fused plug-in interval reported on the fit (fitatt_hat / fitatt_se): the fused SE is the within-selection variance and tends to *under-cover* the aggregated overall ATT (it omits the between-selection term), whereas debiasedATT() returns an interval with asymptotically nominal coverage at roughly the efficiency of unrestricted ETWFE.

The debiased point estimate **differs** from the fused fit$att_hat: when p < NT and the centered design has full column rank, by the OLS identity (paper eq. debiased.ols.identity) it equals the unrestricted ETWFE/OLS estimate in the ATT direction, up to a numerical ridge (see Assumptions). You therefore get a dual offering — fitatt_hat / fitatt_se (fused, efficient, pointwise) and debiasedATT(fit) (debiased, ETWFE-efficient, uniformly valid). It is exposed as a separate accessor (rather than a se_type option) precisely because the debiased SE accompanies a different point estimate than the fused one.

When p >= NT the unrestricted ETWFE is infeasible, so the OLS identity no longer applies; the accessor instead builds the debiasing direction by a nodewise (desparsified-lasso) relaxed inverse (**Theorem** debiased.highdim.thm of Faletto (2025)). The point estimate, SE formula, and return value are otherwise identical — "one estimator, two regimes." The high-dimensional overall-ATT interval is **uniformly valid** under that theorem, and its coverage is validated near-nominally in simulation (0.94 at the p >= NT anchor of Faletto (2025), robust across the nodewise penalty scales studied); see Assumptions.

Usage

```
debiasedATT(
  fit,
  alpha = NULL,
  method = c("analytic", "bootstrap"),
  B = 1000L,
  seed = NULL,
  multiplier = c("webb", "rademacher", "mammen"),
  lambda_c = 1,
  riesz_max_iter = 5000L,
  riesz_tol = 1e-09,
  cv_time_budget = Inf
)
```

Arguments

fit	A fitted object from `fetwfe()`. In the fixed-p (`p < NT`) regime the SE needs either bridge selection (`q < 1`, `gls = TRUE`) or an un-whitened fit (`gls = FALSE`, any `q`, the Omega-free cluster-robust path; #312); a GLS-whitened `q >= 1` fit is rejected. The high-dimensional (`p >= NT`) path accepts any fit. `etwfe()` / `betwfe()` / `twfeCovs()` are not supported (the construction is specific to the FETWFE transformed-coefficient space).
alpha	Numeric in `(0, 1)`; the confidence level is `1 - alpha`. Defaults to the `alpha` stored on the fit.
method	How to form the confidence interval. `"analytic"` (default) uses the two-channel unit-clustered sandwich SE with a Gaussian critical value. `"bootstrap"` replaces the Gaussian critical value with a studentized score / influence-function **wild cluster bootstrap** (#360), **floored at the Gaussian quantile** so the interval is never narrower than the analytic Wald interval. The point estimate and the reported `se` are unchanged; only the critical value changes. **This is not a few-clusters remedy.** As an *unrestricted, no-refit* score bootstrap it does not reproduce the tail inflation of the *restricted* wild-cluster bootstrap-t, and under heterogeneous cluster influence its critical value (before the floor) falls *below* the Gaussian — so under the floor it reduces to the analytic interval exactly in the small-`N` regime; it only ever *widens* the interval when cluster influence is near-homogeneous. For a genuine few-clusters correction the restricted bootstrap-t or a CR2-type analytic adjustment is required (tracked as future work, #361). Supported for both single-sample and `indep_counts` (two-sample) fits: for a two-sample fit the cohort-weight channel is perturbed on its own independent multiplier stream over the `N` count-sample units (`sum(indep_counts) == N` is enforced), yielding the two-sample propensity variance.
B	Integer; the number of wild-bootstrap replicates (default `1000`). Must be a positive integer no greater than `.Machine$integer.max`. Ignored unless `method = "bootstrap"`.
seed	`NULL` (draw from the ambient RNG, the default) or a single integer within `+/- .Machine$integer.max` for a reproducible bootstrap (the RNG state is saved and restored). Ignored unless `method = "bootstrap"`.

`multiplier` The wild-bootstrap weight distribution: `"webb"` (default), `"rademacher"`, or `"mammen"`. `"rademacher"` (±1) gives a constant studentization denominator (a *percentile* bootstrap); `"webb"` and `"mammen"` vary it (a *studentized* statistic). Because the critical value is floored at the Gaussian quantile, the multiplier only affects the near-homogeneous case where the bootstrap widens above the floor; in the small-`N` / heterogeneous regime the interval is the analytic one regardless of the weight. Ignored unless `method = "bootstrap"`.

`lambda_c` The leading constant of the high-dimensional (`p >= NT`) nodewise penalty `lambda_node = lambda_c * ma` (`N` = number of units). Either a single positive number (a fixed constant; default `1.0` = the theory scale) or the string `"cv"`, which selects the constant **per fit** by cross-validating the unit-level Riesz loss `0.5 v'Sigma v - a'v` over the KKT-feasible region (the desparsified-lasso / auto-DML standard; van de Geer; Chernozhukov-Newey-Singh), falling back to the theory scale `1.0` when no grid penalty is feasible (#295). Larger values shrink the debiasing direction more. **Ignored when** `p < NT`**.** The default stays the fixed `1.0` (the theory scale); the high-dimensional coverage validated in Faletto (2025) is robust across the penalty values studied — a cross-validated selector below the theory scale and a fixed constant above it, which bracket the default `1.0` — so it needs no tuning for valid inference.

`riesz_max_iter, riesz_tol`
Integer / numeric; coordinate-descent controls for the high-dimensional nodewise solver. **Ignored when** `p < NT`**.**

`cv_time_budget` Numeric; a wall-clock backstop (in seconds) for the `lambda_c = "cv"` cross-validation. `Inf` (the default) leaves selection fully deterministic / reproducible; a finite value stops the CV early and falls back to the theory scale (`lambda_c = 1.0`) with a warning, so an adversarial high-dimensional draw cannot spin indefinitely (#384). Whether a finite budget fires depends on machine speed, so the result can too. Ignored unless `lambda_c = "cv"`.

Details

The standard error has two channels that **add** under marginal cohort assignment / Assumption (Psi-IF): a regression (outcome) channel, per-unit-clustered, and a cohort-weight channel for the sampling noise in the estimated cohort weights. See *Assumptions* below.

The cohort-weight channel is the fit's own plug-in propensity variance (`att_var_2`), the same quantity `fit$att_se` uses — so it automatically carries the correct single-sample formula and, for `indep_counts` fits, the two-sample formula (the weight channel concerns the estimated cohort weights, not the selection, so it is shared by the fused and debiased estimators). The fit's `se_type` does **not** affect the regression channel — `debiasedATT()` always computes its own per-unit-clustered version. Fits made with `add_ridge = TRUE` are **not** supported (the ridge-augmented design and post-fit un-shrink are outside the validated construction); the accessor errors on them.

Value

An object of S3 class `"debiased_att"` (a named list, with a `print` and a `tidy` method) with elements:

att Numeric scalar; the debiased overall-ATT point estimate.

se Numeric scalar; the uniformly-valid standard error, `sqrt(var_reg + var_weight)`.

ci_low, ci_high Numeric; the `1 - alpha` interval. With `method = "analytic"` (default) this is the Wald interval `att +/- qnorm(1 - alpha/2) * se`; with `method = "bootstrap"` the critical value is the bootstrap `crit_value` (floored at `qnorm(1 - alpha/2)`, so never narrower than the Wald interval) in place of `qnorm(1 - alpha/2)`.

var_reg Numeric; the regression (outcome) channel's contribution to `se^2`, per-unit-clustered.

var_weight Numeric; the cohort-weight channel's contribution to `se^2` — the fit's plug-in propensity variance (`att_var_2`), which carries the correct single- or two-sample (`indep_counts`) formula.

`var_reg` and `var_weight` are the two `se^2`**-scale** (unit-scale) components, so `se = sqrt(var_reg + var_weight)`. They are *not* the paper's CLT-scale variances `V_1^full` / `V_2` from Theorem `debiased.att.thm`, which are larger by a factor on the order of `NT`.

For high-dimensional (`p >= NT`) fits the list additionally contains `feasibility` (`= ||Sigma v - a||_inf`, the nodewise KKT certificate, which should be `<= lambda_node`), `converged` (the coordinate-descent flag), `lambda_node` (the penalty used), `lambda_c` (the leading constant used), and `lambda_c_selection` (`"fixed"` or `"cv"`). When `lambda_c = "cv"` it also contains `lambda_cv`, the cross-validation diagnostics (the grid, the per-grid feasibility flags and CV losses, and whether the theory-scale fallback fired). These are absent for `p < NT` fits.

With `method = "bootstrap"` the list additionally contains `method`, `crit_value` (the studentized wild-bootstrap critical value, floored at `qnorm(1 - alpha/2)`), `B`, `multiplier`, and `alpha`. The analytic default adds none of these, so its `names()` are unchanged.

Assumptions

The uniformly-valid SE relies on the hypotheses of paper Theorem `debiased.att.thm`:

- **Marginal cohort assignment / (Psi-IF).** Cohort membership is independent of the outcome noise, so the regression and cohort-weight variance channels are uncorrelated and add. The package's default cohort weight estimator (sample proportions) satisfies this. Under *confounded* assignment (cohort membership correlated with treatment-effect heterogeneity) the two channels correlate and the additive SE can mis-state uncertainty; a combined per-unit score is then required (not yet implemented).
- **Correctly specified conditional mean** of the outcome. Neyman-orthogonality removes first-order sensitivity to the selected nuisance coefficients, but not to mean-misspecification. The regression channel is the unit-clustered sandwich, valid as the number of units `N -> infinity` with **no** model for within-unit dependence (so a `gls = FALSE` fit, which skips GLS whitening, yields a valid cluster-robust SE; #307, paper Decision D1).
- **(Low-dimensional `p < NT` only) Asymptotically negligible ridge,** `lambda = o((NT)^(-1/2))` (the theoretical condition; the leading case is the exact inverse `lambda = 0`). The implementation does **not** use a vanishing schedule — it adds a fixed numerical stabilizer `lambda = 1e-6 * mean(diag(Sigma))` to the Gram before solving. The same `lambda` is added for every column, so a covariate in very large or very small units can make it pull the estimate back toward the fused one (#512).
- **Growing number of clusters,** `N -> infinity`. The CLT is over the `N` independent units, not the `NT` rows. With few treated units the cluster approximation is poor and the interval can

under-cover; a genuine few-clusters correction (a restricted wild-cluster bootstrap-t or a CR2-type analytic adjustment, #361) is future work. The `method = "bootstrap"` option does *not* remedy this (see its documentation).

- **Regularity / two regimes.** When `p < NT`, Theorem `debiased.att.thm` assumes the limiting full design Gram is nonsingular; in a sample, the exact inverse needs the full-rank condition stated in the Description, and the accessor then reduces to debiased ETWFE. When `p >= NT` (the high-dimensional FETWFE theory) the Gram is singular and the direction is the **nodewise (desparsified-lasso) relaxed inverse** of equation `debiased.highdim.v` (the same estimate and SE, only `v` differs — "one estimator, two regimes"); it requires sparsity (`s_N log(p_N) / sqrt(N) -> 0`), a restricted-eigenvalue condition over sparse cones, `||v*||_1 = O(1)`, and a bounded limiting variance `a' Sigma_theta^(-1) a`. The high-dimensional branch realizes **Theorem** `debiased.highdim.thm` (Faletto 2025) with the theory-scaled penalty; its overall-ATT interval is **uniformly valid**, with coverage validated near-nominally in simulation (0.94 at the `p >= NT` anchor, robust across the nodewise penalty scales studied). Inspect the returned `feasibility` / `converged` diagnostics to confirm the per-fit nodewise directions are well-behaved.

The two regimes use different nuisances. The **fixed-p** branch uses the fit's bridge (`q < 1`) `theta_hat`: Theorem `debiased.att.thm` needs only *consistency* of the nuisance, and with the exact inverse this is the OLS identity. The **high-dimensional** branch instead fits an internal `q = 1` fused lasso (`grpreg::cv.grpreg(penalty = "gBridge", gamma = 1)`, CV-selected `lambda`) as the nuisance, because the high-dimensional FETWFE theory controls the orthogonalization remainder through the nuisance's `l1` *rate* (`||theta_hat - theta*||_1 = O_p(s_N lambda_theta)`) — the rate the `q = 1` fused lasso enjoys under a restricted-eigenvalue condition (the standard desparsified-lasso rate; cf. Negahban et al. 2012), the `p >= NT` extension the paper points to; the `q < 1` bridge is super-efficient / non-uniform — the wrong nuisance for a uniformly-valid CI (#303). The *input* fit must still be `q < 1` (it supplies the cohort-weight variance channel `att_var_2`); only the high-dimensional debiasing nuisance is `q = 1`. That nuisance's cross-validation uses a fixed, data-derived seed, so it is deterministic across calls (`debiasedATT()` and the high-dimensional `simultaneousCIs()` band center agree exactly) — but it is not tunable via the fit's `cv_seed`.

References

Faletto, G (2025). Fused Extended Two-Way Fixed Effects for Difference-in-Differences with Staggered Adoptions. *arXiv preprint arXiv:2312.05985*. https://arxiv.org/abs/2312.05985.

See Also

`fetwfe()` for the fused fit; `cohortTimeATTs()` / `eventStudy()` / `cohortStudy()` for the disaggregated fused effects.

Examples

```
## Not run:
  coefs <- genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2, seed = 1)
  dat <- simulateData(coefs, N = 200, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 1)
  fit <- fetwfeWithSimulatedData(dat, q = 0.5)
  fit$att_hat # fused (pointwise)
  debiasedATT(fit) # debiased (uniformly valid)
```

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

`debiasedATTWithSimulatedData` *Run debiasedATT() on simulated data*

Description

Convenience wrapper mirroring the `*WithSimulatedData()` pattern: fits `fetwfe()` on a `"FETWFE_simulated"` object (via `fetwfeWithSimulatedData()`) and returns `debiasedATT()` on the fit.

Usage

```
debiasedATTWithSimulatedData(simulated_obj, q = 0.5, alpha = NULL, ...)
```

Arguments

`simulated_obj`	An object of class `"FETWFE_simulated"` (from `simulateData()`).
`q`	Numeric; the `L_q` bridge penalty exponent. Must be `< 1` (the debiased SE requires bridge selection). Defaults to `0.5`.
`alpha`	Numeric in `(0, 1)` or `NULL`; confidence level `1 - alpha`. Default `NULL`, inheriting the `alpha` stored on the fit.
`...`	Further arguments forwarded to `fetwfeWithSimulatedData()` (e.g. `fusion_structure`, `lambda_selection`, `cv_seed`).

Value

An object of class `"debiased_att"` (see `debiasedATT()`).

See Also

`debiasedATT()`, `fetwfeWithSimulatedData()`.

Examples

```
## Not run:
  coefs <- genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2, seed = 1)
  dat <- simulateData(coefs, N = 200, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 1)
  debiasedATTWithSimulatedData(dat)

## End(Not run)
```

`etwfe` *Extended two-way fixed effects*

Description

Implementation of extended two-way fixed effects. Estimates overall ATT as well as CATT (cohort average treatment effects on the treated units).

Usage

```
etwfe(
  pdata,
  time_var,
  unit_var,
  treatment,
  response,
  covs = c(),
  indep_counts = NA,
  sig_eps_sq = NA,
  sig_eps_c_sq = NA,
  verbose = FALSE,
  alpha = 0.05,
  add_ridge = FALSE,
  allow_no_never_treated = TRUE,
  se_type = "default",
  ci_type = c("simultaneous", "pointwise")
)
```

Arguments

`pdata`	Dataframe; the panel data set. Each row should represent an observation of a unit at a time. Should contain columns as described below.
`time_var`	Character; the name of a single column containing a variable for the time period. This column is expected to contain integer values (for example, years). Recommended encodings for dates include format YYYY, YYYYMM, or YYYYMMDD, whichever is appropriate for your data.
`unit_var`	Character; the name of a single column containing a variable for each unit. This column is expected to contain character values (i.e. the "name" of each unit).
`treatment`	Character; the name of a single column containing a variable for the treatment dummy indicator. This column is expected to contain integer values, and in particular, should equal 0 if the unit was untreated at that time and 1 otherwise. Treatment should be an absorbing state; that is, if unit `i` is treated at time `t`, then it must also be treated at all times `t` + 1, ..., `T`. Any units treated in the first time period will be removed automatically. Please make sure yourself that at least some units remain untreated at the final time period ("never-treated units").

response — Character; the name of a single column containing the response for each unit at each time. The response must be an integer or numeric value.

covs — (Optional.) Either a character vector containing the names of the columns for covariates (e.g., covs = c("x1", "x2")), or a one-sided formula (e.g., covs = ~ x1 + x2) – the formula form mirrors the convention used by did::att_gt(xformla = ...). Only additive bare variable names are supported in the formula form; for derived variables, compute them in the data frame first and pass via the character-vector form. All of these columns are expected to contain integer, numeric, or factor values, and any categorical values will be automatically encoded as binary indicators. If no covariates are provided, the treatment effect estimation will proceed, but it will only be valid under unconditional versions of the parallel trends and no anticipation assumptions. Default is c().

indep_counts — (Optional.) Integer; a vector. If you have a sufficiently large number of units, you can optionally randomly split your data set in half (with N units in each data set). The data for half of the units should go in the pdata argument provided above. For the other N units, simply provide the counts for how many units appear in the untreated cohort plus each of the other G cohorts in this argument indep_counts. The benefit of doing this is that the standard error for the average treatment effect will be (asymptotically) exact instead of conservative. The length of indep_counts must equal 1 plus the number of treated cohorts in pdata. All entries of indep_counts must be strictly positive (if you are concerned that this might not work out, maybe your data set is on the small side and it's best to just leave your full data set in pdata). The sum of all the counts in indep_counts must match the total number of units in pdata. Default is NA (in which case conservative standard errors will be calculated if q < 1.)

sig_eps_sq — (Optional.) Numeric; the variance of the row-level IID noise assumed to apply to each observation. See Section 2 of Faletto (2025) for details. It is best to provide this variance if it is known (for example, if you are using simulated data). If this variance is unknown, this argument can be omitted, and the variance will be estimated by REML on the linear mixed-effects model y ~ X + (1 | unit) via lme4::lmer (Bates et al. 2015; Patterson & Thompson 1971). Default is NA.

sig_eps_c_sq — (Optional.) Numeric; the variance of the unit-level IID noise (random effects) assumed to apply to each observation. See Section 2 of Faletto (2025) for details. It is best to provide this variance if it is known (for example, if you are using simulated data). If this variance is unknown, this argument can be omitted, and the variance will be estimated by REML via lme4::lmer on the linear mixed-effects model y ~ X + (1 | unit) (Bates et al. 2015; Patterson & Thompson 1971). Default is NA.

verbose — Logical; if TRUE, more details on the progress of the function will be printed as the function executes. Default is FALSE.

alpha — Numeric; function will calculate (1 - alpha) confidence intervals for the cohort average treatment effects that will be returned in catt_df.

add_ridge — (Optional.) Logical; if TRUE, adds a small amount of ridge regularization to the (untransformed) coefficients to stabilize estimation. Default is FALSE.

allow_no_never_treated — (Optional.) Logical; if TRUE (default) and the input panel contains no never-treated units, the panel is auto-truncated by dropping time periods at and after

the latest cohort's start time — the units in that latest cohort then serve as the never-treated comparison group in the retained sub-panel — with a warning naming the dropped periods. If FALSE, the estimator stops with an error in this case (the package's behavior prior to version 1.5.6). The argument has no effect when the input already contains never-treated units. Default is TRUE.

se_type — Character; one of "default", "conservative", or "cluster". "default" returns the tight Gaussian variance sqrt(att_var_1 + att_var_2) from Theorem (c$'$) under Assumption (Psi-IF); this is asymptotically exact for the package's default cohort sample-proportions estimator and for every standard propensity-score estimator that satisfies (Psi-IF) (multinomial logit, any GLM on W | X, kernel/series regression of 1{W = g} on X). "conservative" returns the Cauchy-Schwarz upper bound sqrt(att_var_1 + att_var_2 + 2 * sqrt(att_var_1 * att_var_2)) from Theorem (c); use only if the propensity-score estimator violates (Psi-IF) (e.g., a Robins-Rotnitzky-augmented doubly-robust estimator, which the package does not currently implement). "cluster" is an *experimental* unit-clustered Liang-Zeger sandwich SE on the OLS-selected support (see the companion vignette inference_vignette for the formula, the assumptions, and the theory-pending caveat). The default value of "default" corresponds to the new tight Gaussian default introduced in version 1.12.0; previous versions used the conservative Cauchy-Schwarz formula as the default. To recover the prior conservative default behavior, pass se_type = "conservative".

ci_type — Character; one of "simultaneous" (default) or "pointwise". Controls the confidence-interval bounds reported for the cohort-specific ATTs (in catt_df) and the event-study effects (from eventStudy(), shown by print / summary / plot, and surfaced by broom::tidy() on the fitted object and on the eventStudy() / cohortStudy() outputs). "simultaneous" reports parametric simultaneous (family-wise, uniform) bands computed via simultaneousCIs(): each family's band covers all of its effects jointly with probability 1 - alpha, matching the default presentation of did::aggte(cband = TRUE). "pointwise" reports per-effect Wald intervals (each covers its own effect with probability 1 - alpha, no joint guarantee — the behavior of versions <= 1.15.1). Both the interval bounds and the per-cohort p-values (p_value) follow ci_type (max-T multiplicity-adjusted under "simultaneous", per-cohort Wald under "pointwise"; #200); the standard errors (se) and the overall-ATT confidence interval (a single scalar) are identical under both settings. When standard errors are unavailable (e.g., a rank-deficient design) the bounds are NA under both settings. Default is "simultaneous".

Value

An object of class etwfe containing the following elements:

att_hat — The estimated overall average treatment effect for a randomly selected treated unit.

att_se — A standard error for the ATT. If the Gram matrix is not invertible, this will be NA.

att_p_value — A two-sided p-value for the overall ATT against the null H_0: tau = 0, computed as 2 * pnorm(-|att_hat / att_se|). NA if att_se is zero or NA. Standard post-OLS interpretation; ETWFE does not perform selection.

`catt_hats` A named vector containing the estimated average treatment effects for each cohort.

`catt_ses` A named vector containing the (asymptotically exact) standard errors for the estimated average treatment effects within each cohort.

`cohort_probs` A vector of the estimated probabilities of being in each cohort conditional on being treated, which was used in calculating `att_hat`. If `indep_counts` was provided, `cohort_probs` was calculated from that; otherwise, it was calculated from the counts of units in each treated cohort in `pdata`.

`catt_df` A data frame (with S3 class `c("catt_df", "data.frame")`) displaying the cohort names (`cohort`), average treatment effects (`estimate`), standard errors (`se`), `1 - alpha` confidence interval bounds (`ci_low`, `ci_high`), and per-cohort p-values (`p_value`). No `selected` column; ETWFE does not perform selection. The `catt_df` S3 class makes `[[` / `$` / `[` access on the pre-1.11.0 Title-Case column names (`Cohort`, `Estimated TE`, `SE`, `ConfIntLow`, `ConfIntHigh`, `P_value`) `stop()` with a migration message pointing to the new name. See `NEWS.md` for the rename table.

`beta_hat` The full vector of estimated coefficients.

`treat_inds` The indices of `beta_hat` corresponding to the treatment effects for each cohort at each time.

`treat_int_inds` The indices of `beta_hat` corresponding to the interactions between the treatment effects for each cohort at each time and the covariates.

`sig_eps_sq` Either the provided `sig_eps_sq` or the estimated one, if a value wasn't provided.

`sig_eps_c_sq` Either the provided `sig_eps_c_sq` or the estimated one, if a value wasn't provided.

`X_ints` The design matrix created containing all interactions, time and cohort dummies, etc.

`y` The vector of responses, containing `nrow(X_ints)` entries.

`X_final` The design matrix after applying the change in coordinates to fit the model and also multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.

`y_final` The final response after multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.

`N` The final number of units that were in the data set used for estimation (after any units may have been removed because they were treated in the first time period).

`T` The number of time periods in the final data set.

`G` The final number of treated cohorts that appear in the final data set.

`R` Deprecated alias for `G`, retained for backward compatibility; populated with the same value. Use `G`. Will be removed in a future release.

`d` The final number of covariates that appear in the final data set (after any covariates may have been removed because they contained missing values or all contained the same value for every unit).

`p` The final number of columns in the full set of covariates used to estimate the model.

`alpha` The alpha level used for confidence intervals.

`calc_ses` Logical indicating whether standard errors were calculated.

`cohort_probs_overall`
A vector of the estimated cohort probabilities on the overall sample (treated and untreated), used in computing the variance of the overall ATT.

`indep_counts_used`
Logical scalar; `TRUE` if a valid `indep_counts` argument was provided and used for asymptotically-exact ATT inference, `FALSE` otherwise.

`se_type` Character scalar; the `se_type` argument the user passed (`"default"`, `"conservative"`, or `"cluster"`).

`ci_type` Character scalar; the `ci_type` argument the user passed (`"simultaneous"` or `"pointwise"`), controlling whether the reported `catt_df` / `eventStudy()` confidence-interval bounds are simultaneous (family-wise) or pointwise.

`catt_band_applied`
Logical scalar; `TRUE` when the fit-time cohort-family simultaneous band was computed and written into `catt_df`, and `FALSE` otherwise — including when `ci_type = "pointwise"`, when standard errors were unavailable, and when the band construction degraded. It records only whether the band was applied: it does not imply the band is wider than the pointwise interval (they coincide when fewer than two effects have positive variance) and does not imply the band is informative (a degenerate fit can carry an all-zero band with `TRUE`).

`y_mean` Numeric scalar; the mean of the original (pre-centering) response. Stored so downstream methods (`augment()`, `predict()`) can return fitted values on the original-response scale.

`response_col_name`
Character scalar; the name of the response column in the original `pdata`. Consumed by `augment.<class>()`.

`time_var`, `unit_var`, `treatment`
Character scalars; the `time_var` / `unit_var` / `treatment` arguments the user passed. Consumed by `augment.<class>()` when auto-aligning a user-supplied panel to the fitted design.

`covs` Character vector; the original `covs` argument the user passed (before any factor expansion the estimator performed internally). Consumed by `augment.<class>()`.

`internal` A list containing internal outputs that are typically not needed for interpretation, packaged here for parity with `fetwfe()` so downstream consumers can use a single canonical access path across all four estimator classes (#144). The first five sub-slots (`X_ints`, `y`, `X_final`, `y_final`, `calc_ses`) are also duplicated at top level for backward compat; `variance_components` and `first_year` live only under `$internal`:

- **X_ints** The design matrix containing all interactions, time and cohort dummies, etc. Same value as top-level `X_ints`.
- **y** The vector of responses. Same as top-level `y`.
- **X_final** The design matrix after the change-of-coordinates step. Same as top-level `X_final`.
- **y_final** The transformed response vector. Same as top-level `y_final`.

calc_ses Logical indicating whether standard errors were calculated. Same as top-level `calc_ses`.

variance_components A list exposing the two variance pieces (`att_var_1`, `att_var_2`) plus paper-notation counterparts (`V_1`, `V_2`) and unit-scaled variance estimators (`tilde_v_N`, `hat_v_N`, `tilde_v_N_C`, `tilde_v_N_C_pi_hat`, `tilde_v_N_C_pi_hat_cons`, `tilde_v_N_cons`). The Wald CI is `[hat_T_N +- qnorm(1-alpha/2)` (paper Eq. `conf.int.form`). New in v1.12.0 (issue #141 + #146).

first_year Integer or numeric scalar; the first (earliest) `time_var` value in the panel after `idCohorts()` processing. Consumed by `eventStudy()` to map `cohort_probs`' cohort labels (treatment-start years) to 1-based panel-time-index offsets when the labels are integer-coercible. New in v1.13.3 (issue #174).

Author(s)

Gregory Faletto

References

Wooldridge, J. M. (2021). Two-way fixed effects, the two-way mundlak regression, and difference-in-differences estimators. *Available at SSRN 3906345*. doi:10.2139/ssrn.3906345.

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2015). Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software*, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Patterson, H. D., & Thompson, R. (1971). Recovery of inter-block information when block sizes are unequal. *Biometrika*, 58(3), 545-554.

Pinheiro, J. C., & Bates, D. M. (2000). *Mixed-Effects Models in S and S-PLUS*. Springer.

Examples

```
## Not run:
library(bacondecomp)

data(castle)

# Response: the log homicide rate. Treatment: `cdl` records the share of
# the year the castle-doctrine law was in effect, so `cdl > 0` gives the
# absorbing 0/1 treatment indicator.
castle$l_homicide <- log(castle$homicide)
castle$treated <- as.integer(castle$cdl > 0)

# No `covs` here: etwfe is pure OLS (no bridge penalty), and castle's
# smallest adoption cohorts contain a single state, so the design is
# rank-deficient once any covariate is added.
res <- etwfe(
    pdata = castle,
    time_var = "year",
    unit_var = "state",
    treatment = "treated",
    response = "l_homicide",
    verbose = TRUE)
```

```
# Print results
print(res, max_cohorts = Inf)

## End(Not run)
```

`etwfe-class` *Extended Two-Way Fixed Effects Output Class*

Description

S3 class for the output of `etwfe()`.

`etwfeToFetwfeDf` *Convert data prepared for* `etwfe::etwfe()` *to the format required by* `fetwfe()` *and* `fetwfe::etwfe()`

Description

`etwfeToFetwfeDf()` reshapes and renames a panel dataset that is already formatted for `etwfe::etwfe()` (McDermott 2024) so that it can be passed directly to `fetwfe()` or `etwfe()` from the `fetwfe` package. In particular, it

- creates an *absorbing-state* treatment dummy that equals 1 from the first treated period onward* and 0 otherwise,
- (optionally) drops units that are already treated in the very first period of the sample (because `fetwfe()` removes them internally), and
- returns a tidy dataframe whose column names match the arguments that `fetwfe()`/`etwfe()` expect.

Usage

```
etwfeToFetwfeDf(
  data,
  yvar,
  tvar,
  idvar,
  gvar,
  covars = character(0),
  drop_first_period_treated = TRUE,
  out_names = list(time = "time_var", unit = "unit_var", treatment = "treatment",
    response = "response"),
  verbose = FALSE
)
```

Arguments

data	A long-format data.frame that you could already feed to etwfe().
yvar	Character. Column name of the outcome (left-hand side in your fml).
tvar	Character. Column name of the time variable that you pass to etwfe() as tvar.
idvar	Character. Column name of the unit identifier (the variable you would cluster on, or pass to etwfe(..., ivar = idvar) if you were using unit FEs).
gvar	Character. Column name of the "first treated" cohort variable passed to etwfe() as gvar. Must be 0 (or Inf, which is mapped to 0) for never-treated units, or the (strictly positive) first treated period.
covars	Character vector of *additional* covariate columns to keep (default character(0)).
drop_first_period_treated	Logical. Should units already treated in the very first sample period be removed? (fetwfe() will drop them internally anyway, but doing it here keeps the returned dataframe clean.) Default TRUE.
out_names	Named list giving the column names that the returned dataframe should have. The default (time_var, unit_var, treatment, response) matches the arguments usually supplied to fetwfe(). **Do not change the *names* of this list** – only the *values* – and keep all four.
verbose	Logical. If TRUE, a message() reports the count of first-period-treated unit-period rows dropped when drop_first_period_treated = TRUE. Default FALSE (silent).

Value

A tidy data.frame with (in this order)

- time_var integer,
- unit_var character,
- treatment integer 0/1 absorbing-state dummy,
- response numeric outcome,
- any covariates requested in covars. Ready to pass straight to fetwfe() or fetwfe::etwfe().

References

McDermott G (2024). *etwfe: Extended Two-Way Fixed Effects*. doi:10.32614/CRAN.package.etwfe doi:10.32614/CRAN.package.etwfe, R package version 0.5.0, https://CRAN.R-project.org/package=etwfe.

Examples

```
## toy example ------------------------------------------------------------
## Not run:
library(did)  # provides the mpdta example dataframe
data(mpdta)

head(mpdta)
```

```
tidy_df <- etwfeToFetwfeDf(
  data  = mpdta,
  yvar = "lemp",
  tvar = "year",
  idvar = "countyreal",
  gvar = "first.treat",
  covars = c("lpop"))

head(tidy_df)


## End(Not run)
## Now you can call fetwfe()  ----------------------------------------------
# res <- fetwfe(
#   pdata      = tidy_df,
#   time_var   = "time_var",
#   unit_var   = "unit_var",
#   treatment  = "treatment",
#   response   = "response",
#   covs       = c("lpop"))
```

`etwfeWithSimulatedData` *Run ETWFE on Simulated Data*

Description

This function runs the extended two-way fixed effects estimator (`etwfe()`) on simulated data. It is simply a wrapper for `etwfe()`: it accepts an object of class `"FETWFE_simulated"` (produced by `simulateData()`) and unpacks the necessary components to pass to `etwfe()`. So the outputs match `etwfe()`, and the needed inputs match their counterparts in `etwfe()`.

Usage

```
etwfeWithSimulatedData(
  simulated_obj,
  verbose = FALSE,
  alpha = 0.05,
  add_ridge = FALSE,
  allow_no_never_treated = TRUE,
  se_type = "default",
  ci_type = c("simultaneous", "pointwise")
)
```

Arguments

simulated_obj	An object of class "FETWFE_simulated" containing the simulated panel data and design matrix.
verbose	Logical; if TRUE, more details on the progress of the function will be printed as the function executes. Default is FALSE.
alpha	Numeric; function will calculate (1 - alpha) confidence intervals for the cohort average treatment effects that will be returned in catt_df.
add_ridge	(Optional.) Logical; if TRUE, adds a small amount of ridge regularization to the (untransformed) coefficients to stabilize estimation. Default is FALSE.
allow_no_never_treated	(Optional.) Logical; if TRUE (default) and the input panel contains no never-treated units, the panel is auto-truncated by dropping time periods at and after the latest cohort's start time — the units in that latest cohort then serve as the never-treated comparison group in the retained sub-panel — with a warning naming the dropped periods. If FALSE, the estimator stops with an error in this case (the package's behavior prior to version 1.5.6). The argument has no effect when the input already contains never-treated units. Default is TRUE.
se_type	Character; one of "default", "conservative", or "cluster". "default" returns the tight Gaussian variance sqrt(att_var_1 + att_var_2) from Theorem (c$'$) under Assumption (Psi-IF); this is asymptotically exact for the package's default cohort sample-proportions estimator and for every standard propensity-score estimator that satisfies (Psi-IF) (multinomial logit, any GLM on W | X, kernel/series regression of 1{W = g} on X). "conservative" returns the Cauchy-Schwarz upper bound sqrt(att_var_1 + att_var_2 + 2 * sqrt(att_var_1 * att_var_2)) from Theorem (c); use only if the propensity-score estimator violates (Psi-IF) (e.g., a Robins-Rotnitzky-augmented doubly-robust estimator, which the package does not currently implement). "cluster" is an *experimental* unit-clustered Liang-Zeger sandwich SE on the OLS-selected support (see the companion vignette inference_vignette for the formula, the assumptions, and the theory-pending caveat). The default value of "default" corresponds to the new tight Gaussian default introduced in version 1.12.0; previous versions used the conservative Cauchy-Schwarz formula as the default. To recover the prior conservative default behavior, pass se_type = "conservative".
ci_type	Character; one of "simultaneous" (default) or "pointwise". Controls the confidence-interval bounds reported for the cohort-specific ATTs (in catt_df) and the event-study effects (from eventStudy(), shown by print / summary / plot, and surfaced by broom::tidy() on the fitted object and on the eventStudy() / cohortStudy() outputs). "simultaneous" reports parametric simultaneous (family-wise, uniform) bands computed via simultaneousCIs(): each family's band covers all of its effects jointly with probability 1 - alpha, matching the default presentation of did::aggte(cband = TRUE). "pointwise" reports per-effect Wald intervals (each covers its own effect with probability 1 - alpha, no joint guarantee — the behavior of versions <= 1.15.1). Both the interval bounds and the per-cohort p-values (p_value) follow ci_type (max-T multiplicity-adjusted under "simultaneous", per-cohort Wald under "pointwise"; #200); the standard errors (se) and the overall-ATT confidence interval (a single scalar)

are identical under both settings. When standard errors are unavailable (e.g., a rank-deficient design) the bounds are NA under both settings. Default is "simultaneous".

Value

An object of class etwfe containing the following elements:

att_hat	The estimated overall average treatment effect for a randomly selected treated unit.
att_se	A standard error for the ATT. If the Gram matrix is not invertible, this will be NA.
att_p_value	A two-sided p-value for the overall ATT against the null H_0: tau = 0, computed as 2 * pnorm(-|att_hat / att_se|). NA if att_se is zero or NA. Standard post-OLS interpretation; ETWFE does not perform selection.
catt_hats	A named vector containing the estimated average treatment effects for each cohort.
catt_ses	A named vector containing the (asymptotically exact) standard errors for the estimated average treatment effects within each cohort.
cohort_probs	A vector of the estimated probabilities of being in each cohort conditional on being treated, which was used in calculating att_hat. If indep_counts was provided, cohort_probs was calculated from that; otherwise, it was calculated from the counts of units in each treated cohort in pdata.
catt_df	A data frame (with S3 class c("catt_df", "data.frame")) displaying the cohort names (cohort), average treatment effects (estimate), standard errors (se), 1 - alpha confidence interval bounds (ci_low, ci_high), and per-cohort p-values (p_value). No selected column; ETWFE does not perform selection. The catt_df S3 class makes [[/ $ / [access on the pre-1.11.0 Title-Case column names (Cohort, Estimated TE, SE, ConfIntLow, ConfIntHigh, P_value) stop() with a migration message pointing to the new name. See NEWS.md for the rename table.
beta_hat	The full vector of estimated coefficients.
treat_inds	The indices of beta_hat corresponding to the treatment effects for each cohort at each time.
treat_int_inds	The indices of beta_hat corresponding to the interactions between the treatment effects for each cohort at each time and the covariates.
sig_eps_sq	Either the provided sig_eps_sq or the estimated one, if a value wasn't provided.
sig_eps_c_sq	Either the provided sig_eps_c_sq or the estimated one, if a value wasn't provided.
X_ints	The design matrix created containing all interactions, time and cohort dummies, etc.
y	The vector of responses, containing nrow(X_ints) entries.
X_final	The design matrix after applying the change in coordinates to fit the model and also multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.

`y_final` The final response after multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.

`N` The final number of units that were in the data set used for estimation (after any units may have been removed because they were treated in the first time period).

`T` The number of time periods in the final data set.

`G` The final number of treated cohorts that appear in the final data set.

`R` Deprecated alias for `G`, retained for backward compatibility; populated with the same value. Use `G`. Will be removed in a future release.

`d` The final number of covariates that appear in the final data set (after any covariates may have been removed because they contained missing values or all contained the same value for every unit).

`p` The final number of columns in the full set of covariates used to estimate the model.

`alpha` The alpha level used for confidence intervals.

`calc_ses` Logical indicating whether standard errors were calculated.

`cohort_probs_overall`
A vector of the estimated cohort probabilities on the overall sample (treated and untreated), used in computing the variance of the overall ATT.

`indep_counts_used`
Logical scalar; `TRUE` if a valid `indep_counts` argument was provided and used for asymptotically-exact ATT inference, `FALSE` otherwise.

`se_type` Character scalar; the `se_type` argument the user passed (`"default"`, `"conservative"`, or `"cluster"`).

`ci_type` Character scalar; the `ci_type` argument the user passed (`"simultaneous"` or `"pointwise"`), controlling whether the reported `catt_df` / `eventStudy()` confidence-interval bounds are simultaneous (family-wise) or pointwise.

`catt_band_applied`
Logical scalar; `TRUE` when the fit-time cohort-family simultaneous band was computed and written into `catt_df`, and `FALSE` otherwise — including when `ci_type = "pointwise"`, when standard errors were unavailable, and when the band construction degraded. It records only whether the band was applied: it does not imply the band is wider than the pointwise interval (they coincide when fewer than two effects have positive variance) and does not imply the band is informative (a degenerate fit can carry an all-zero band with `TRUE`).

`y_mean` Numeric scalar; the mean of the original (pre-centering) response. Stored so downstream methods (`augment()`, `predict()`) can return fitted values on the original-response scale.

`response_col_name`
Character scalar; the name of the response column in the original `pdata`. Consumed by `augment.<class>()`.

`time_var`, `unit_var`, `treatment`
Character scalars; the `time_var` / `unit_var` / `treatment` arguments the user passed.

covs Character vector; the original covs argument the user passed (before any factor expansion the estimator performed internally).

internal A list containing internal outputs that are typically not needed for interpretation, packaged here for parity with fetwfe() so downstream consumers can use a single canonical access path across all four estimator classes (#144). The first five sub-slots (X_ints, y, X_final, y_final, calc_ses) are also duplicated at top level for backward compat; variance_components and first_year live only under $internal:

X_ints The design matrix containing all interactions, time and cohort dummies, etc. Same value as top-level X_ints.

y The vector of responses. Same as top-level y.

X_final The design matrix after the change-of-coordinates step. Same as top-level X_final.

y_final The transformed response vector. Same as top-level y_final.

calc_ses Logical indicating whether standard errors were calculated. Same as top-level calc_ses.

variance_components A list exposing the two variance pieces (att_var_1, att_var_2) plus paper-notation counterparts (V_1, V_2) and unit-scaled variance estimators (tilde_v_N, hat_v_N, tilde_v_N_C, tilde_v_N_C_pi_hat, tilde_v_N_C_pi_hat_cons, tilde_v_N_cons). The Wald CI is [hat_T_N +- qnorm(1-alpha/2) (paper Eq. conf.int.form). New in v1.12.0 (issue #141 + #146).

first_year Integer or numeric scalar; the first (earliest) time_var value in the panel after idCohorts() processing. Consumed by eventStudy() to map cohort_probs' cohort labels (treatment-start years) to 1-based panel-time-index offsets when the labels are integer-coercible. New in v1.13.3 (issue #174).

Examples

```
## Not run:
  # Generate coefficients
  coefs <- genCoefs(G = 5, T = 30, d = 12, density = 0.1, eff_size = 2, seed = 123)

  # Simulate data using the coefficients
  sim_data <- simulateData(coefs, N = 120, sig_eps_sq = 5, sig_eps_c_sq = 5, seed = 123)

  result <- etwfeWithSimulatedData(sim_data)

## End(Not run)
```

---

eventStudy *Compute pooled event-time treatment-effect estimates*

Description

For a fitted object from `fetwfe()`, `etwfe()`, or `betwfe()`, computes the pooled event-time treatment-effect estimates `tau_E(e)`, defined as cohort-weighted averages of the cell-level treatment-effect estimates at each post-treatment event time `e = t - g` (where `t` is calendar time and `g` is the cohort's first-treated calendar time). Weights are sample-cohort-size weights (matching `did::aggte(type = "dynamic")` convention).

Standard errors combine two terms, mirroring the package's existing overall-ATT SE machinery: `var_1(e)` from regression-coefficient noise (computed via the same `gram_inv` machinery the package uses for cohort SEs, or the cluster-robust sandwich under `se_type = "cluster"`), and `var_2(e)` from cohort-probability noise (analog of the existing `getSecondVarTermOLS` / `getSecondVarTermDataApp` machinery, with the multinomial Jacobian restricted to cohorts valid at event time `e`). Combined as `sqrt(var_1 + var_2)` by default (asymptotically exact under paper Theorem `te.asym.norm.thm`(c') / Assumption (Ψ-IF), which the package's default cohort-sample-proportions estimator satisfies); the conservative Cauchy-Schwarz bound `sqrt(var_1 + var_2 + 2 sqrt(var_1 * var_2))` is available via `se_type = "conservative"` (for users with non-(Ψ-IF) propensity-score estimators). When `indep_counts` was supplied at fit time, the tight formula applies regardless of `se_type` (two-sample regime, Theorem (b)).

Usage

```
eventStudy(x, alpha = NULL, ci_type = NULL)
```

Arguments

`x`	A fitted object of class `"fetwfe"`, `"etwfe"`, or `"betwfe"`.
`alpha`	(Optional) Significance level for confidence intervals. Defaults to `x$alpha` (the alpha used at fit time).
`ci_type`	(Optional) Character; one of `"simultaneous"` or `"pointwise"`, or `NULL` (default). Controls whether the returned `ci_low` / `ci_high` are the event-study-family simultaneous (family-wise, uniform) band or the per-event-time pointwise Wald intervals. `NULL` inherits the fit's stored `ci_type` (so a fit made with the default `ci_type = "simultaneous"` yields simultaneous event-study bands, which `print` / `summary` / `plot` then display); for objects fitted before version 1.16.0 (no `ci_type` slot) `NULL` falls back to `"pointwise"`. The `se` column is identical under both settings; the interval bounds, their critical-value multiplier, and the `p_value` differ (under `"simultaneous"` the `p_value` is the single-step max-T multiplicity- adjusted dual of the band, under `"pointwise"` the per-effect Wald p-value; #200). Degenerate event times (no valid contributing cohorts) carry `NA` bounds and `NA` `p_value` under both settings. **High-dimensional (`p >= NT`):** the `"simultaneous"` band here is the analytic post-selection band on the selected support (or `NA` bounds on a `gls = FALSE` fit), *not* the uniformly-valid desparsified band. For the valid high-dimensional event-study band call `simultaneousCIs()` with `family = "event_study"`, `method = "bootstrap"` (Theorem `debiased.highdim.joint.thm`, validated near-nominally in Faletto 2025). Such a call **emits a warning**; see Details.

Details

High-dimensional (p >= NT) bands warn. When the returned "simultaneous" bounds are the p >= NT post-selection fallback rather than a uniformly-valid band – which is every high-dimensional fit that reaches the band at all, since this accessor always takes the analytic route – the call emits a warning naming the remedy (#433). A high-dimensional fit under ci_type = "pointwise", or one whose standard errors are unavailable, never builds a simultaneous band and so does not warn. A #308 coverage study measures the fallback band at roughly 0.09 against 0.95 nominal, even when the selected support is low-dimensional. The remedy is simultaneousCIs() with family = "event_study" and method = "bootstrap" on a fetwfe() fit; no valid high-dimensional band exists for the other estimators.

The warning is the same classed condition simultaneousCIs() signals, "fetwfe_highdim_postselection_band", so it can be caught or muffled precisely:

```
withCallingHandlers(
  eventStudy(fit),
  fetwfe_highdim_postselection_band = function(w) invokeRestart("muffleWarning")
)
```

print(), summary(), and plot() call this function internally and muffle that condition, so they do not repeat it on every render; print() and summary() render a two-line caveat under their CATT preview instead, and plot() renders none.

Value

A data frame with class c("eventStudy", "data.frame") and columns:

event_time Integer; event time e = t - g, ranging from 0 to T - 2.

n_cohorts Integer; number of cohorts contributing to the pooled estimate at event time e.

estimate Numeric; the pooled event-time ATT estimate.

se Numeric; combined standard error.

ci_low Numeric; lower bound of the (1 - alpha) Wald CI.

ci_high Numeric; upper bound of the (1 - alpha) Wald CI.

p_value Numeric; follows the fit's ci_type. Under "pointwise", the two-sided Wald p-value (2 * pnorm(-|estimate / se|)); under "simultaneous" (the default), the single-step max-T multiplicity-adjusted (family-wise) p-value matching the simultaneous band (#200). NA when se is 0 or NA.

The returned frame also carries the logical attribute attr(., "band_applied") (#460), TRUE when its ci_low / ci_high ARE the event-study-family simultaneous band and FALSE when they are the pointwise Wald bounds — because the fit asked for pointwise intervals, because standard errors were unavailable on the selected support, or because the band construction degraded. It is not the same question as the fit's ci_type, which records what was requested, nor as the fit's catt_band_applied slot, which answers it for the cohort family: the two families are built from different contrast matrices and fail independently, so one can be applied while the other is not. **It is an attribute rather than a slot because no slot could hold it.** This band is recomputed per call with that call's arguments, so one fit answers differently for different calls: on a fit carrying

`catt_band_applied = TRUE`, `attr(eventStudy(fit), "band_applied")` is `TRUE` while `attr(eventStudy(fit, ci_type = "pointwise"), "band_applied")` is `FALSE`. Like the slot, it records only that the band was applied: `TRUE` does not imply the bounds are wider than the pointwise ones, and does not imply every row is informative — degenerate event times are masked to `NA` while the band is still reported as applied. Only post-treatment event times (`e >= 0`) are included; pre-treatment placebo periods would require an extended regression specification and are out of scope for this initial release.

See Also

`cohortStudy()` for the parallel per-cohort accessor; `cohortTimeATTs()` for the fully disaggregated per-(cohort, time) accessor.

Examples

```
## Not run:
  coefs <- genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2)
  dat <- simulateData(coefs, N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  res <- fetwfeWithSimulatedData(dat)
  eventStudy(res)

## End(Not run)
```

`fetwfe` *Fused extended two-way fixed effects*

Description

Implementation of fused extended two-way fixed effects. Estimates overall ATT as well as CATT (cohort average treatment effects on the treated units).

The treatment-effect fusion penalty defaults to a within-/between-cohort geometry (`fusion_structure = "cohort"`) and also supports an event-study geometry (`fusion_structure = "event_study"`, fusing effects at the same time since treatment across cohorts) or a fully custom `fusion_matrix`. See the `fusion_structure` / `fusion_matrix` arguments below and `vignette("fusion_structure_vignette", package = "fetwfe")` for guidance on choosing.

Usage

```
fetwfe(
  pdata,
  time_var,
  unit_var,
  treatment,
  response,
  covs = c(),
  indep_counts = NA,
  sig_eps_sq = NA,
  sig_eps_c_sq = NA,
  lambda.max = NA,
```

```
  lambda.min = NA,
  nlambda = 100,
  q = 0.5,
  verbose = FALSE,
  alpha = 0.05,
  add_ridge = FALSE,
  allow_no_never_treated = TRUE,
  se_type = "default",
  lambda_selection = "cv",
  cv_folds = 10L,
  cv_seed = NULL,
  ci_type = c("simultaneous", "pointwise"),
  fusion_structure = c("cohort", "event_study"),
  fusion_matrix = NULL,
  gls = TRUE
)
```

Arguments

pdata	Dataframe; the panel data set. Each row should represent an observation of a unit at a time. Should contain columns as described below.
time_var	Character; the name of a single column containing a variable for the time period. This column is expected to contain integer values (for example, years). Recommended encodings for dates include format YYYY, YYYYMM, or YYYYMMDD, whichever is appropriate for your data.
unit_var	Character; the name of a single column containing a variable for each unit. This column is expected to contain character values (i.e. the "name" of each unit).
treatment	Character; the name of a single column containing a variable for the treatment dummy indicator. This column is expected to contain integer values, and in particular, should equal 0 if the unit was untreated at that time and 1 otherwise. Treatment should be an absorbing state; that is, if unit `i` is treated at time `t`, then it must also be treated at all times `t` + 1, ..., `T`. Any units treated in the first time period will be removed automatically. Please make sure yourself that at least some units remain untreated at the final time period ("never-treated units").
response	Character; the name of a single column containing the response for each unit at each time. The response must be an integer or numeric value.
covs	(Optional.) Either a character vector containing the names of the columns for covariates (e.g., `covs = c("x1", "x2")`), or a one-sided formula (e.g., `covs = ~ x1 + x2`) – the formula form mirrors the convention used by `did::att_gt(xformla = ...)`. Only additive bare variable names are supported in the formula form; for derived variables, compute them in the data frame first and pass via the character-vector form. All of these columns are expected to contain integer, numeric, or factor values, and any categorical values will be automatically encoded as binary indicators. If no covariates are provided, the treatment effect estimation will proceed, but it will only be valid under unconditional versions of the parallel trends and no anticipation assumptions. Default is c().

indep_counts (Optional.) Integer; a vector. If you have a sufficiently large number of units, you can optionally randomly split your data set in half (with N units in each data set). The data for half of the units should go in the pdata argument provided above. For the other N units, simply provide the counts for how many units appear in the untreated cohort plus each of the other G cohorts in this argument indep_counts. The benefit of doing this is that the standard error for the average treatment effect will be (asymptotically) exact instead of conservative. The length of indep_counts must equal 1 plus the number of treated cohorts in pdata. All entries of indep_counts must be strictly positive (if you are concerned that this might not work out, maybe your data set is on the small side and it's best to just leave your full data set in pdata). The sum of all the counts in indep_counts must match the total number of units in pdata. Default is NA (in which case conservative standard errors will be calculated if q < 1.)

sig_eps_sq (Optional.) Numeric; the variance of the row-level IID noise assumed to apply to each observation. See Section 2 of Faletto (2025) for details. It is best to provide this variance if it is known (for example, if you are using simulated data). If this variance is unknown, this argument can be omitted, and the variance will be estimated by REML on the linear mixed-effects model y ~ X + (1 | unit) via lme4::lmer (Bates et al. 2015; Patterson & Thompson 1971). When supplied, the value also sets the scale of the bridge penalty's grid (the response is divided by sqrt(sig_eps_sq) before the fit), so at q != 1 a value supplied in the wrong units moves the estimates, not only their standard errors. Default is NA.

sig_eps_c_sq (Optional.) Numeric; the variance of the unit-level IID noise (random effects) assumed to apply to each observation. See Section 2 of Faletto (2025) for details. It is best to provide this variance if it is known (for example, if you are using simulated data). If this variance is unknown, this argument can be omitted, and the variance will be estimated by REML via lme4::lmer on the linear mixed-effects model y ~ X + (1 | unit) (Bates et al. 2015; Patterson & Thompson 1971). Default is NA.

lambda.max (Optional.) Numeric. Used only on the BIC route (lambda_selection = "bic"), which selects lambda by BIC over a grid. The largest lambda in the grid will be lambda.max. If no lambda.max is provided, one will be selected automatically. When q <= 1, the model will be sparse, and ideally all of the following are true at once: the smallest model (the one corresponding to lambda.max) selects close to 0 features, the largest model (the one corresponding to lambda.min) selects close to p features, nlambda is large enough so that models are considered at every feasible model size, and nlambda is small enough so that the computation doesn't become infeasible. You may want to manually tweak lambda.max, lambda.min, and nlambda to try to achieve these goals, particularly if the selected model size is very close to the model corresponding to lambda.max or lambda.min, which could indicate that the range of lambda values was too narrow or coarse. You can use the function outputs lambda.max_model_size, lambda.min_model_size, and lambda_star_model_size to try to assess this. Default is NA.

lambda.min (Optional.) Numeric. Used only on the BIC route: the smallest lambda penalty parameter considered. See the description of lambda.max for details. Default is NA.

nlambda (Optional.) Integer. Used only on the BIC route: the total number of lambda penalty parameters considered. See the description of lambda.max for details. Default is 100.

q (Optional.) Numeric; determines what L_q penalty is used for the fusion regularization. q = 1 is the lasso, and for 0 < q < 1, it is possible to get standard errors and confidence intervals. q = 2 is ridge regression. See Faletto (2025) for details. Default is 0.5.

verbose Logical; if TRUE, more details on the progress of the function will be printed as the function executes. Default is FALSE.

alpha Numeric; function will calculate (1 - alpha) confidence intervals for the cohort average treatment effects that will be returned in catt_df.

add_ridge (Optional.) Logical; if TRUE, adds a small amount of ridge regularization to the (untransformed) coefficients to stabilize estimation. Default is FALSE.

allow_no_never_treated
(Optional.) Logical; if TRUE (default) and the input panel contains no never-treated units, the panel is auto-truncated by dropping time periods at and after the latest cohort's start time — the units in that latest cohort then serve as the never-treated comparison group in the retained sub-panel — with a warning naming the dropped periods. If FALSE, the estimator stops with an error in this case (the package's behavior prior to version 1.5.6). The argument has no effect when the input already contains never-treated units. Default is TRUE.

se_type Character; one of "default", "conservative", or "cluster". "default" returns the tight Gaussian variance sqrt(att_var_1 + att_var_2) from Theorem (c$'$) under Assumption (Psi-IF); this is asymptotically exact for the package's default cohort sample-proportions estimator and for every standard propensity-score estimator that satisfies (Psi-IF) (multinomial logit, any GLM on W | X, kernel/series regression of 1{W = g} on X). "conservative" returns the Cauchy-Schwarz upper bound sqrt(att_var_1 + att_var_2 + 2 * sqrt(att_var_1 * att_var_2)) from Theorem (c); use only if the propensity-score estimator violates (Psi-IF) (e.g., a Robins-Rotnitzky-augmented doubly-robust estimator, which the package does not currently implement). "cluster" is an *experimental* unit-clustered Liang-Zeger sandwich SE on the bridge-selected support (see the companion vignette inference_vignette for the formula, the assumptions, and the theory-pending caveat); only meaningful when q < 1 (the bridge oracle property is required), and for q >= 1 the SE will be NA regardless of se_type. The default value of "default" corresponds to the new tight Gaussian default introduced in version 1.12.0; previous versions used the conservative Cauchy-Schwarz formula as the default. To recover the prior conservative default behavior, pass se_type = "conservative".

lambda_selection
Character; method for selecting the bridge penalty parameter lambda. Either "cv" (10-fold cross-validation on cv.grpreg; the v1.13.0+ default) or "bic" (BIC over the grpreg lambda grid; the prior default for v1.12.0 and earlier). The default changed in v1.13.0 to address a finite-sample bias issue documented in simulation studies (see issue #164): under the prior BIC default, the overall-ATT estimator was biased toward zero at moderate sample sizes, producing 95% confidence intervals whose empirical coverage was as low as 0.00 in some regimes.

Cross-validation restores near-nominal coverage in every regime tested. See the inference vignette section "Choosing the bridge penalty parameter" for details.

cv_folds
: Integer; number of folds for the CV path. Ignored when `lambda_selection = "bic"`. Default is 10.

cv_seed
: Integer or `NULL`; the seed passed to `set.seed()` to draw the cross-validation folds. The folds are row-level: the draw is assigned to the rows unit by unit, with the units in an order computed from their data rather than from their IDs. So relabeling the units does not change which `lambda` cross-validation selects, except through REML's small dependence on unit order when the noise variances are estimated (#535). If supplied, must be within `+/- .Machine$integer.max`. If `NULL` (the default), the seed defaults internally to `as.integer(N * T)` so consecutive calls on the same dataset are reproducible without the user having to specify a seed. The seed actually used is stored on the returned object as `cv_seed`. Ignored when `lambda_selection = "bic"`.

ci_type
: Character; one of `"simultaneous"` (default) or `"pointwise"`. Controls the confidence-interval bounds reported for the cohort-specific ATTs (in `catt_df`) and the event-study effects (from `eventStudy()`, shown by `print` / `summary` / `plot`, and surfaced by `broom::tidy()` on the fitted object and on the `eventStudy()` / `cohortStudy()` outputs). `"simultaneous"` reports parametric simultaneous (family-wise, uniform) bands computed via `simultaneousCIs()`: each family's band covers all of its effects jointly with probability `1 - alpha`, matching the default presentation of `did::aggte(cband = TRUE)`. `"pointwise"` reports per-effect Wald intervals (each covers its own effect with probability `1 - alpha`, no joint guarantee — the behavior of versions <= 1.15.1). Both the interval bounds and the per-cohort p-values (`p_value`) follow `ci_type`: under `"simultaneous"` the `p_value` is the single-step max-T multiplicity- adjusted p-value matching the band, under `"pointwise"` the per-cohort Wald p-value (#200). The standard errors (`se`) and selection flags (`selected`) are identical under both settings, and the overall-ATT confidence interval (a single scalar) is unaffected. When standard errors are unavailable (`q >= 1`, or a rank-deficient design) the bounds are `NA` under both settings. Default is `"simultaneous"`.

fusion_structure
: Character; one of `"cohort"` or `"event_study"`. `"cohort"` (the default) uses the within-cohort / between-cohort two-way fusion penalty. `"event_study"` instead fuses treatment effects at the same time since treatment (event time `e = t - g`) across cohorts. The event-study penalty carries the same theoretical guarantees as the default (Faletto 2025); only the treatment-effect fusion structure changes.

fusion_matrix
: (Optional.) Numeric matrix or `NULL` (the default). An advanced-use override: a user-supplied `num_treats x num_treats` forward differences matrix `D_N` for the treatment-effect block, encoding an arbitrary fusion structure beyond the two built-ins. When non-`NULL` it overrides `fusion_structure` for the treatment-effect block only (the fixed-effect blocks are unchanged); the estimator uses `solve(fusion_matrix)` internally. The rows/columns are interpreted in the cohort-major `(g, t)` order used internally for the treatment effects (the order `getFirstInds()` / `getTreatInds()` encode): row/column `i` corresponds to base treatment effect `i`, with cohort `g` occupying rows `first_inds[g]:(first_inds[g`

+ 1] - 1) ordered by event time. num_treats equals T * G - G * (G + 1) / 2. fusion_matrix must be a finite, invertible numeric matrix of that exact dimension (otherwise fetwfe() stops). Under the paper's fixed-dimension scoping, *any* finite invertible D_N of that dimension inherits the paper's inferential guarantees: the theory depends on D_N only through its invertibility and singular-value bounds (Assumption (D) of Faletto 2025), which a fixed invertible matrix automatically satisfies, and swapping in a different D_N from this class changes only constant factors. A numerically near-singular (ill-conditioned) D_N still yields a valid point estimator but emits a warning() that its inverse may be unreliable. Default is NULL (use the built-in fusion_structure).

gls (Optional.) Logical; default TRUE. When TRUE, the design is GLS-whitened using REML-estimated (or supplied) variance components — the standard, efficient path. When FALSE, fetwfe() **skips GLS whitening and variance-component estimation entirely**, fitting on the un-whitened (fusion-transformed) design. This is the high-dimensional (p >= NT) path: REML cannot estimate the variance components there (the p < N(T - 1) REML guard would otherwise stop the fit), and the debiasedATT() cluster-robust sandwich standard error needs no Omega (paper Decision D1). A gls = FALSE fit has calc_ses = FALSE (no within-selection oracle standard errors) and un-whitened internal$X_final / internal$y_final; pass it to debiasedATT() for a valid cluster-robust SE. add_ridge = TRUE is not supported, and supplied sig_eps_sq / sig_eps_c_sq are ignored, under gls = FALSE.

Value

An object of class fetwfe containing the following elements:

att_hat The estimated overall average treatment effect for a randomly selected treated unit.

att_se If q < 1, a standard error for the ATT. Under the default se_type = "default", the SE is the tight Gaussian variance sqrt(att_var_1 + att_var_2) (Theorem (c$'$) under Assumption (Psi-IF); paper line 1233 onwards). Assumption (Psi-IF) is satisfied by the package's default cohort sample-proportions estimator hat_pi_g = N_g / N (and by multinomial logit, any GLM on W | X, and kernel/series regression of 1{W = g} on X), so the default SE is asymptotically exact for the package's default estimator. Under se_type = "conservative" (or in version <= 1.11.7 by default), the SE is the Cauchy-Schwarz upper bound sqrt(att_var_1 + att_var_2 + 2 * sqrt(att_var_1 * att_var_2)) from Theorem (c). When indep_counts is provided, the two-sample exact formula sqrt(att_var_1 + att_var_2) is used regardless of se_type. If q >= 1, this will be NA.

att_p_value A two-sided p-value for the overall ATT against the null H_0: tau = 0, computed as 2 * pnorm(-|att_hat / att_se|). NA if att_se is zero or NA (e.g., under the bridge solver's selected-out fallback). See the package vignette section "Testing the zero-effect null" for interpretation guidance under selection consistency.

att_selected Logical scalar; TRUE if att_hat is not exactly zero (i.e., at least one cohort's bridge-penalized coefficient survived selection), FALSE otherwise. Under FETWFE

Theorem 6.2 (restriction selection consistency), att_selected = FALSE is the asymptotic statement that the truth is zero. For ridge (q = 2) the bridge solver does not zero coefficients, so this will typically be TRUE.

catt_hats — A named vector containing the estimated average treatment effects for each cohort.

catt_ses — If q < 1, a named vector containing the (asymptotically exact, non-conservative) standard errors for the estimated average treatment effects within each cohort.

cohort_probs — A vector of the estimated probabilities of being in each cohort conditional on being treated, which was used in calculating att_hat. If indep_counts was provided, cohort_probs was calculated from that; otherwise, it was calculated from the counts of units in each treated cohort in pdata.

catt_df — A data frame (with S3 class c("catt_df", "data.frame")) displaying the cohort names (cohort), average treatment effects (estimate), standard errors (se), 1 - alpha confidence interval bounds (ci_low, ci_high), per-cohort p-values (p_value), and a selected logical flag (TRUE when the bridge penalty left the cohort's CATT nonzero). For selected-out cohorts (selected = FALSE), p_value is NA — the inferential content lives in selected. The catt_df S3 class makes [[/ $ / [access on the pre-1.11.0 Title-Case column names (Cohort, Estimated TE, SE, ConfIntLow, ConfIntHigh, P_value) stop() with a migration message pointing to the new name. See NEWS.md for the rename table.

beta_hat — The full vector of estimated coefficients.

treat_inds — The indices of beta_hat corresponding to the treatment effects for each cohort at each time.

treat_int_inds — The indices of beta_hat corresponding to the interactions between the treatment effects for each cohort at each time and the covariates.

sig_eps_sq — Either the provided sig_eps_sq or the estimated one, if a value wasn't provided.

sig_eps_c_sq — Either the provided sig_eps_c_sq or the estimated one, if a value wasn't provided.

lambda.max — The largest lambda of the grid used, which is a supplied lambda.max only on the BIC route. (This is returned to help with getting a reasonable range of lambda values for grid search.)

lambda.max_model_size — The number of selected features (excluding the always-present intercept) at lambda.max (for q <= 1, this will be the smallest model size). As mentioned above, for q <= 1 ideally this value is close to 0.

lambda.min — Either the provided lambda.min or the one that was used, if a value wasn't provided.

lambda.min_model_size — The number of selected features (excluding the always-present intercept) at lambda.min (for q <= 1, this will be the largest model size). As mentioned above, for q <= 1 ideally this value is close to p.

lambda_star — The value of lambda chosen by the selection method recorded in lambda_selection. If this value is close to lambda.min or lambda.max, that could suggest that the range of lambda values should be expanded.

lambda_star_model_size
: The number of selected features (excluding the always-present intercept) in the chosen model. If this value is close to lambda.max_model_size or lambda.min_model_size, that could suggest that the range of lambda values should be expanded.

lambda_selection
: Character scalar; either "cv" (10-fold cross-validation on cv.grpreg; v1.13.0+ default) or "bic" (BIC over the grpreg lambda grid; the prior default). Mirrors the lambda_selection argument the user passed.

cv_folds
: Integer scalar; the cv_folds value used when lambda_selection = "cv", NA_integer_ when lambda_selection = "bic".

cv_seed
: Integer scalar; the seed actually fed to set.seed() to draw the cross-validation folds. Defaults to as.integer(N * T) when the user did not pass a seed. NA_integer_ when lambda_selection = "bic".

fusion_structure
: Character scalar; the fusion_structure argument the user passed ("cohort" or "event_study"), recording which fusion-penalty differences matrix was used for the treatment effects.

fusion_matrix
: The user-supplied custom forward differences matrix D_N (a num_treats x num_treats numeric matrix), or NULL if none was supplied. When non-NULL it overrode fusion_structure for the treatment-effect block; the estimator used solve(fusion_matrix) internally. See the fusion_matrix argument.

N
: The final number of units that were in the data set used for estimation (after any units may have been removed because they were treated in the first time period).

T
: The number of time periods in the final data set.

G
: The final number of treated cohorts that appear in the final data set.

R
: Deprecated alias for G, retained for backward compatibility; populated with the same value. Use G. Will be removed in a future release.

d
: The final number of covariates that appear in the final data set (after any covariates may have been removed because they contained missing values or all contained the same value for every unit).

p
: The final number of columns in the full set of covariates used to estimate the model.

alpha
: The alpha level used for confidence intervals.

calc_ses
: Logical indicating whether standard errors were calculated. Same as $internal$calc_ses; duplicated at top level for parity with etwfe(), betwfe(), and twfeCovs() (#180).

cohort_probs_overall
: A vector of the estimated cohort probabilities on the overall sample (treated and untreated), used in computing the variance of the overall ATT.

indep_counts_used
: Logical scalar; TRUE if a valid indep_counts argument was provided and used for asymptotically-exact ATT inference, FALSE otherwise.

se_type
: Character scalar; the se_type argument the user passed ("default", "conservative", or "cluster").

ci_type Character scalar; the ci_type argument the user passed ("simultaneous" or "pointwise"), controlling whether the reported catt_df / eventStudy() confidence-interval bounds are simultaneous (family-wise) or pointwise.

catt_band_applied
Logical scalar; TRUE when the fit-time cohort-family simultaneous band was computed and written into catt_df, and FALSE otherwise — including when ci_type = "pointwise", when standard errors were unavailable, and when the band construction degraded. It records only whether the band was applied: it does not imply the band is wider than the pointwise interval (they coincide when fewer than two effects have positive variance) and does not imply the band is informative (a degenerate fit can carry an all-zero band with TRUE).

y_mean Numeric scalar; the mean of the original (pre-centering) response. Stored so downstream methods (augment(), predict()) can return fitted values on the original-response scale.

response_col_name
Character scalar; the name of the response column in the original pdata. Consumed by augment.<class>().

time_var, unit_var, treatment
Character scalars; the time_var / unit_var / treatment arguments the user passed. Consumed by augment.<class>() when auto-aligning a user-supplied panel to the fitted design (e.g., dropping first-period-treated units the estimator removed internally, and sorting rows to match the design matrix's internal (unit, time) order).

covs Character vector; the original covs argument the user passed (before any factor expansion the estimator performed internally). Consumed by augment.<class>().

internal A list containing internal outputs that are typically not needed for interpretation:

X_ints The design matrix created containing all interactions, time and cohort dummies, etc.

y The vector of responses, containing nrow(X_ints) entries.

X_final The design matrix after applying the change in coordinates to fit the model and also multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.

y_final The final response after multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.

theta_hat The vector of estimated coefficients in the transformed (fused) space, including the intercept as the first element.

calc_ses Logical indicating whether standard errors were calculated.

variance_components A list exposing the two variance pieces (att_var_1, att_var_2) plus their paper-notation counterparts (V_1, V_2) and the unit-scaled variance estimators (tilde_v_N, hat_v_N, tilde_v_N_C, tilde_v_N_C_pi_hat, tilde_v_N_C_pi_hat_cons, tilde_v_N_cons) catalogued at paper line 2006. The Wald CI is [hat_T_N +- qnorm(1-alpha/2) * sqrt(tilde_v_N / N)] (paper Eq. conf.int.form). New in v1.12.0 (issue #141 + #146).

first_year Integer or numeric scalar; the first (earliest) time_var value in the panel after idCohorts() processing. Consumed by eventStudy() to map

cohort_probs' cohort labels (treatment-start years) to 1-based panel-time-index offsets when the labels are integer-coercible. New in v1.13.3 (issue #174).

d_inv_treat The inverted custom treatment-effect fusion block `solve(fusion_matrix)` (a `num_treats` x `num_treats` numeric matrix), or `NULL` if no `fusion_matrix` was supplied. Consumed by `eventStudy()` and `simultaneousCIs()` so the access-time bands reuse the same fusion block the fit used (#236).

The object has methods for `print()`, `summary()`, and `coef()`. By default, `print()` and `summary()` only show the essential outputs. To see internal details, use `print(x, show_internal = TRUE)` or `summary(x, show_internal = TRUE)`. The `coef()` method returns the vector of estimated coefficients (`beta_hat`).

Author(s)

Gregory Faletto

References

Faletto, G (2025). Fused Extended Two-Way Fixed Effects for Difference-in-Differences with Staggered Adoptions. *arXiv preprint arXiv:2312.05985*. https://arxiv.org/abs/2312.05985.

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2015). Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software*, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Patterson, H. D., & Thompson, R. (1971). Recovery of inter-block information when block sizes are unequal. *Biometrika*, 58(3), 545-554.

Pinheiro, J. C., & Bates, D. M. (2000). *Mixed-Effects Models in S and S-PLUS*. Springer.

See Also

`vignette("fusion_structure_vignette", package = "fetwfe")` for guidance on choosing between the cohort (default) and event-study fusion penalties and on supplying a custom `fusion_matrix`.

Examples

```
# `bacondecomp` (which supplies the `divorce` data) is a Suggests-only
# dependency, so guard the example on its availability. The fit is wrapped in
# \donttest{} because it is slower than a toy example.

if (requireNamespace("bacondecomp", quietly = TRUE)) {
  library(bacondecomp)

  data(divorce)

  # Stevenson & Wolfers (2006): the effect of unilateral ("no-fault") divorce
  # reforms on female suicide rates. Restrict to the female subset
  # (`sex == 2`); `changed` is already the absorbing 0/1 reform indicator, and
  # the elasticity-scaled female suicide rate is the response.
  divorce_f <- divorce[divorce$sex == 2, ]

  # Follows the data and specification of Faletto (2025, Sec. 8.2). The 9
```

```
  # states already treated by 1964 are auto-dropped as first-period-treated,
  # and `murderrate` is auto-dropped (missing in 1964 for one state); both are
  # reported as (expected) warnings. The noise variances are supplied
  # (precomputed by REML) to keep the example fast and reproducible; the
  # default lambda_selection is "cv" (10-fold cross-validation).
  res <- fetwfe(
      pdata = divorce_f,
      time_var = "year",
      unit_var = "st",
      treatment = "changed",
      covs = c("murderrate", "lnpersinc", "afdcrolls"),
      response = "suiciderate_elast_jag",
      sig_eps_sq = 0.0344,
      sig_eps_c_sq = 0.1507,
      q = 0.5)

  # FETWFE estimates a negative overall ATT on the elasticity-scaled female
  # suicide rate, with a 95% confidence interval that excludes zero.
  # The selection step retains heterogeneous cohort effects (several cohorts
  # are pruned to exactly zero), rather than fusing to a single common effect.
  print(res, max_cohorts = Inf)
}
```

`fetwfe-class` *Fused Extended Two-Way Fixed Effects Output Class*

Description

S3 class for the output of `fetwfe()`.

`FETWFE_coefs-class` *FETWFE Coefficient-Vector Class*

Description

S3 class for objects returned by `genCoefs()`. Compact `print` method summarizes the coefficient vector and its sparsity pattern instead of dumping the full `beta` and `theta` vectors.

FETWFE_simulated-class *Simulated Panel-Data Class*

Description

S3 class for objects returned by `simulateData()`. Compact `print` method summarizes the panel's dimensions and cohort structure instead of dumping the full `N*T x p` design matrix (which the default `print.list` would do).

FETWFE_tes-class *Compute True Treatment Effects Output Class*

Description

S3 class for the output of `getTes()`.

fetwfeWithSimulatedData *Run FETWFE on Simulated Data*

Description

This function runs the fused extended two-way fixed effects estimator (`fetwfe()`) on simulated data. It is simply a wrapper for `fetwfe()`: it accepts an object of class `"FETWFE_simulated"` (produced by `simulateData()`) and unpacks the necessary components to pass to `fetwfe()`. So the outputs match `fetwfe()`, and the needed inputs match their counterparts in `fetwfe()`.

Usage

```
fetwfeWithSimulatedData(
  simulated_obj,
  lambda.max = NA,
  lambda.min = NA,
  nlambda = 100,
  q = 0.5,
  verbose = FALSE,
  alpha = 0.05,
  add_ridge = FALSE,
  allow_no_never_treated = TRUE,
  se_type = "default",
  lambda_selection = "cv",
  cv_folds = 10L,
```

```
  cv_seed = NULL,
  ci_type = c("simultaneous", "pointwise"),
  fusion_structure = c("cohort", "event_study"),
  fusion_matrix = NULL,
  gls = TRUE
)
```

Arguments

simulated_obj	An object of class "FETWFE_simulated" containing the simulated panel data and design matrix.
lambda.max	(Optional.) Numeric. Used only on the BIC route (lambda_selection = "bic"), which selects lambda by BIC over a grid. The largest lambda in the grid will be lambda.max. If no lambda.max is provided, one will be selected automatically. When q <= 1, the model will be sparse, and ideally all of the following are true at once: the smallest model (the one corresponding to lambda.max) selects close to 0 features, the largest model (the one corresponding to lambda.min) selects close to p features, nlambda is large enough so that models are considered at every feasible model size, and nlambda is small enough so that the computation doesn't become infeasible. You may want to manually tweak lambda.max, lambda.min, and nlambda to try to achieve these goals, particularly if the selected model size is very close to the model corresponding to lambda.max or lambda.min, which could indicate that the range of lambda values was too narrow. You can use the function outputs lambda.max_model_size, lambda.min_model_size, and lambda_star_model_size to try to assess this. Default is NA.
lambda.min	(Optional.) Numeric. Used only on the BIC route: the smallest lambda penalty parameter considered. See the description of lambda.max for details. Default is NA.
nlambda	(Optional.) Integer. Used only on the BIC route: the total number of lambda penalty parameters considered. See the description of lambda.max for details. Default is 100.
q	(Optional.) Numeric; determines what L_q penalty is used for the fusion regularization. q = 1 is the lasso, and for 0 < q < 1, it is possible to get standard errors and confidence intervals. q = 2 is ridge regression. See Faletto (2025) for details. Default is 0.5.
verbose	Logical; if TRUE, more details on the progress of the function will be printed as the function executes. Default is FALSE.
alpha	Numeric; function will calculate (1 - alpha) confidence intervals for the cohort average treatment effects that will be returned in catt_df.
add_ridge	(Optional.) Logical; if TRUE, adds a small amount of ridge regularization to the (untransformed) coefficients to stabilize estimation. Default is FALSE.
allow_no_never_treated	(Optional.) Logical; if TRUE (default) and the input panel contains no never-treated units, the panel is auto-truncated by dropping time periods at and after the latest cohort's start time — the units in that latest cohort then serve as the never-treated comparison group in the retained sub-panel — with a warning

naming the dropped periods. If FALSE, the estimator stops with an error in this case (the package's behavior prior to version 1.5.6). The argument has no effect when the input already contains never-treated units. Default is TRUE.

se_type
: Character; one of "default", "conservative", or "cluster". "default" returns the tight Gaussian variance sqrt(att_var_1 + att_var_2) from Theorem (c$'$) under Assumption (Psi-IF); this is asymptotically exact for the package's default cohort sample-proportions estimator and for every standard propensity-score estimator that satisfies (Psi-IF) (multinomial logit, any GLM on W | X, kernel/series regression of 1{W = g} on X). "conservative" returns the Cauchy-Schwarz upper bound sqrt(att_var_1 + att_var_2 + 2 * sqrt(att_var_1 * att_var_2)) from Theorem (c); use only if the propensity-score estimator violates (Psi-IF) (e.g., a Robins-Rotnitzky-augmented doubly-robust estimator, which the package does not currently implement). "cluster" is an *experimental* unit-clustered Liang-Zeger sandwich SE on the bridge-selected support (see the companion vignette inference_vignette for the formula, the assumptions, and the theory-pending caveat); only meaningful when q < 1 (the bridge oracle property is required), and for q >= 1 the SE will be NA regardless of se_type. The default value of "default" corresponds to the new tight Gaussian default introduced in version 1.12.0; previous versions used the conservative Cauchy-Schwarz formula as the default. To recover the prior conservative default behavior, pass se_type = "conservative".

lambda_selection
: Character; method for selecting the bridge penalty parameter lambda. Either "cv" (10-fold cross-validation on cv.grpreg; the v1.13.0+ default) or "bic" (BIC over the grpreg lambda grid; the prior default for v1.12.0 and earlier). The default changed in v1.13.0 to address a finite-sample bias issue documented in simulation studies (see issue #164): under the prior BIC default, the overall-ATT estimator was biased toward zero at moderate sample sizes, producing 95% confidence intervals whose empirical coverage was as low as 0.00 in some regimes. Cross-validation restores near-nominal coverage in every regime tested. See the inference vignette section "Choosing the bridge penalty parameter" for details.

cv_folds
: Integer; number of folds for the CV path. Ignored when lambda_selection = "bic". Default is 10.

cv_seed
: Integer or NULL; the seed passed to set.seed() to draw the cross-validation folds. The folds are row-level: the draw is assigned to the rows unit by unit, with the units in an order computed from their data rather than from their IDs. So relabeling the units does not change which lambda cross-validation selects, except through REML's small dependence on unit order when the noise variances are estimated (#535). If supplied, must be within +/- .Machine$integer.max. If NULL (the default), the seed defaults internally to as.integer(N * T) so consecutive calls on the same dataset are reproducible without the user having to specify a seed. The seed actually used is stored on the returned object as cv_seed. Ignored when lambda_selection = "bic".

ci_type
: Character; one of "simultaneous" (default) or "pointwise". Controls the confidence-interval bounds reported for the cohort-specific ATTs (in catt_df) and the event-study effects (from eventStudy(), shown by print / summary / plot, and surfaced by broom::tidy() on the fitted object and on the eventStudy()

/ cohortStudy() outputs). "simultaneous" reports parametric simultaneous (family-wise, uniform) bands computed via simultaneousCIs(): each family's band covers all of its effects jointly with probability 1 - alpha, matching the default presentation of did::aggte(cband = TRUE). "pointwise" reports per-effect Wald intervals (each covers its own effect with probability 1 - alpha, no joint guarantee — the behavior of versions <= 1.15.1). Both the interval bounds and the per-cohort p-values (p_value) follow ci_type: under "simultaneous" the p_value is the single-step max-T multiplicity- adjusted p-value matching the band, under "pointwise" the per-cohort Wald p-value (#200). The standard errors (se) and selection flags (selected) are identical under both settings, and the overall-ATT confidence interval (a single scalar) is unaffected. When standard errors are unavailable (q >= 1, or a rank-deficient design) the bounds are NA under both settings. Default is "simultaneous".

fusion_structure

Character; one of "cohort" or "event_study". "cohort" (the default) uses the within-cohort / between-cohort two-way fusion penalty. "event_study" instead fuses treatment effects at the same time since treatment (event time e = t - g) across cohorts. The event-study penalty carries the same theoretical guarantees as the default (Faletto 2025); only the treatment-effect fusion structure changes.

fusion_matrix (Optional.) Numeric matrix or NULL (the default). An advanced-use override: a user-supplied num_treats x num_treats forward differences matrix D_N for the treatment-effect block, encoding an arbitrary fusion structure beyond the two built-ins. When non-NULL it overrides fusion_structure for the treatment-effect block only (the fixed-effect blocks are unchanged); the estimator uses solve(fusion_matrix) internally. The rows/columns are interpreted in the cohort-major (g, t) order used internally for the treatment effects (the order getFirstInds() / getTreatInds() encode): row/column i corresponds to base treatment effect i, with cohort g occupying rows first_inds[g]:(first_inds[g + 1] - 1) ordered by event time. num_treats equals T * G - G * (G + 1) / 2. fusion_matrix must be a finite, invertible numeric matrix of that exact dimension (otherwise fetwfe() stops). Under the paper's fixed-dimension scoping, *any* finite invertible D_N of that dimension inherits the paper's inferential guarantees: the theory depends on D_N only through its invertibility and singular-value bounds (Assumption (D) of Faletto 2025), which a fixed invertible matrix automatically satisfies, and swapping in a different D_N from this class changes only constant factors. A numerically near-singular (ill-conditioned) D_N still yields a valid point estimator but emits a warning() that its inverse may be unreliable. Default is NULL (use the built-in fusion_structure).

gls (Optional.) Logical; default TRUE. When TRUE, the design is GLS-whitened using REML-estimated (or supplied) variance components — the standard, efficient path. When FALSE, fetwfe() **skips GLS whitening and variance-component estimation entirely**, fitting on the un-whitened (fusion-transformed) design. This is the high-dimensional (p >= NT) path: REML cannot estimate the variance components there (the p < N(T - 1) REML guard would otherwise stop the fit), and the debiasedATT() cluster-robust sandwich standard error needs no Omega (paper Decision D1). A gls = FALSE fit has calc_ses = FALSE (no within-selection oracle standard errors) and un-whitened internal$X_final /

internal$y_final; pass it to debiasedATT() for a valid cluster-robust SE. add_ridge = TRUE is not supported, and supplied sig_eps_sq / sig_eps_c_sq are ignored, under gls = FALSE.

Value

An object of class fetwfe containing the following elements:

att_hat — The estimated overall average treatment effect for a randomly selected treated unit.

att_se — If q < 1, a standard error for the ATT. Under the default se_type = "default", the SE is the tight Gaussian variance sqrt(att_var_1 + att_var_2) (Theorem (c$'$) under Assumption (Psi-IF); paper line 1233 onwards). Assumption (Psi-IF) is satisfied by the package's default cohort sample-proportions estimator hat_pi_g = N_g / N (and by multinomial logit, any GLM on W | X, and kernel/series regression of 1{W = g} on X), so the default SE is asymptotically exact for the package's default estimator. Under se_type = "conservative" (or in version <= 1.11.7 by default), the SE is the Cauchy-Schwarz upper bound sqrt(att_var_1 + att_var_2 + 2 * sqrt(att_var_1 * att_var_2)) from Theorem (c). When indep_counts is provided, the two-sample exact formula sqrt(att_var_1 + att_var_2) is used regardless of se_type. If q >= 1, this will be NA.

att_p_value — A two-sided p-value for the overall ATT against the null H_0: tau = 0, computed as 2 * pnorm(-|att_hat / att_se|). NA if att_se is zero or NA (e.g., under the bridge solver's selected-out fallback). See the package vignette section "Testing the zero-effect null" for interpretation guidance under selection consistency.

att_selected — Logical scalar; TRUE if att_hat is not exactly zero (i.e., at least one cohort's bridge-penalized coefficient survived selection), FALSE otherwise. Under FETWFE Theorem 6.2 (restriction selection consistency), att_selected = FALSE is the asymptotic statement that the truth is zero. For ridge (q = 2) the bridge solver does not zero coefficients, so this will typically be TRUE.

catt_hats — A named vector containing the estimated average treatment effects for each cohort.

catt_ses — If q < 1, a named vector containing the (asymptotically exact, non-conservative) standard errors for the estimated average treatment effects within each cohort.

cohort_probs — A vector of the estimated probabilities of being in each cohort conditional on being treated, which was used in calculating att_hat. If indep_counts was provided, cohort_probs was calculated from that; otherwise, it was calculated from the counts of units in each treated cohort in pdata.

catt_df — A data frame (with S3 class c("catt_df", "data.frame")) displaying the cohort names (cohort), average treatment effects (estimate), standard errors (se), 1 - alpha confidence interval bounds (ci_low, ci_high), per-cohort p-values (p_value), and a selected logical flag (TRUE when the bridge penalty left the cohort's CATT nonzero). For selected-out cohorts (selected = FALSE), p_value is NA — the inferential content lives in selected. The catt_df S3 class makes [[/ $ / [access on the pre-1.11.0 Title-Case column names (Cohort,

Estimated TE, SE, ConfIntLow, ConfIntHigh, P_value) stop() with a migration message pointing to the new name. See NEWS.md for the rename table.

beta_hat — The full vector of estimated coefficients.

treat_inds — The indices of beta_hat corresponding to the treatment effects for each cohort at each time.

treat_int_inds — The indices of beta_hat corresponding to the interactions between the treatment effects for each cohort at each time and the covariates.

sig_eps_sq — Either the provided sig_eps_sq or the estimated one, if a value wasn't provided.

sig_eps_c_sq — Either the provided sig_eps_c_sq or the estimated one, if a value wasn't provided.

lambda.max — The largest lambda of the grid used, which is a supplied lambda.max only on the BIC route. (This is returned to help with getting a reasonable range of lambda values for grid search.)

lambda.max_model_size — The number of selected features (excluding the always-present intercept) at lambda.max (for q <= 1, this will be the smallest model size). As mentioned above, for q <= 1 ideally this value is close to 0.

lambda.min — Either the provided lambda.min or the one that was used, if a value wasn't provided.

lambda.min_model_size — The number of selected features (excluding the always-present intercept) at lambda.min (for q <= 1, this will be the largest model size). As mentioned above, for q <= 1 ideally this value is close to p.

lambda_star — The value of lambda chosen by the selection method recorded in lambda_selection. If this value is close to lambda.min or lambda.max, that could suggest that the range of lambda values should be expanded.

lambda_star_model_size — The number of selected features (excluding the always-present intercept) in the chosen model. If this value is close to lambda.max_model_size or lambda.min_model_size, that could suggest that the range of lambda values should be expanded.

lambda_selection — Character scalar; either "cv" (10-fold cross-validation on cv.grpreg; v1.13.0+ default) or "bic" (BIC over the grpreg lambda grid; the prior default). Mirrors the lambda_selection argument the user passed.

cv_folds — Integer scalar; the cv_folds value used when lambda_selection = "cv", NA_integer_ when lambda_selection = "bic".

cv_seed — Integer scalar; the seed actually fed to set.seed() to draw the cross-validation folds. Defaults to as.integer(N * T) when the user did not pass a seed. NA_integer_ when lambda_selection = "bic".

fusion_structure — Character scalar; the fusion_structure argument the user passed ("cohort" or "event_study"), recording which fusion-penalty differences matrix was used for the treatment effects.

`fusion_matrix` The user-supplied custom forward differences matrix `D_N` (a `num_treats x num_treats` numeric matrix), or `NULL` if none was supplied. When non-`NULL` it overrode `fusion_structure` for the treatment-effect block; the estimator used `solve(fusion_matrix)` internally. See the `fusion_matrix` argument.

`N` The final number of units that were in the data set used for estimation (after any units may have been removed because they were treated in the first time period).

`T` The number of time periods in the final data set.

`G` The final number of treated cohorts that appear in the final data set.

`R` Deprecated alias for `G`, retained for backward compatibility; populated with the same value. Use `G`. Will be removed in a future release.

`d` The final number of covariates that appear in the final data set (after any covariates may have been removed because they contained missing values or all contained the same value for every unit).

`p` The final number of columns in the full set of covariates used to estimate the model.

`alpha` The alpha level used for confidence intervals.

`calc_ses` Logical indicating whether standard errors were calculated. Same as `$internal$calc_ses`; duplicated at top level for parity with `etwfe()`, `betwfe()`, and `twfeCovs()` (#180).

`cohort_probs_overall`
A vector of the estimated cohort probabilities on the overall sample (treated and untreated), used in computing the variance of the overall ATT.

`indep_counts_used`
Logical scalar; `TRUE` if a valid `indep_counts` argument was provided and used for asymptotically-exact ATT inference, `FALSE` otherwise.

`se_type` Character scalar; the `se_type` argument the user passed (`"default"`, `"conservative"`, or `"cluster"`).

`ci_type` Character scalar; the `ci_type` argument the user passed (`"simultaneous"` or `"pointwise"`), controlling whether the reported `catt_df` / `eventStudy()` confidence-interval bounds are simultaneous (family-wise) or pointwise.

`catt_band_applied`
Logical scalar; `TRUE` when the fit-time cohort-family simultaneous band was computed and written into `catt_df`, and `FALSE` otherwise — including when `ci_type = "pointwise"`, when standard errors were unavailable, and when the band construction degraded. It records only whether the band was applied: it does not imply the band is wider than the pointwise interval (they coincide when fewer than two effects have positive variance) and does not imply the band is informative (a degenerate fit can carry an all-zero band with `TRUE`).

`y_mean` Numeric scalar; the mean of the original (pre-centering) response. Stored so downstream methods (`augment()`, `predict()`) can return fitted values on the original-response scale.

`response_col_name`
Character scalar; the name of the response column in the original `pdata`. Consumed by `augment.<class>()`.

time_var, unit_var, treatment
: Character scalars; the time_var / unit_var / treatment arguments the user passed. Consumed by augment.<class>() when auto-aligning a user-supplied panel to the fitted design (e.g., dropping first-period-treated units the estimator removed internally, and sorting rows to match the design matrix's internal (unit, time) order).

covs
: Character vector; the original covs argument the user passed (before any factor expansion the estimator performed internally). Consumed by augment.<class>().

internal
: A list containing internal outputs that are typically not needed for interpretation:

  X_ints The design matrix created containing all interactions, time and cohort dummies, etc.

  y The vector of responses, containing nrow(X_ints) entries.

  X_final The design matrix after applying the change in coordinates to fit the model and also multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.

  y_final The final response after multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.

  theta_hat The vector of estimated coefficients in the transformed (fused) space, including the intercept as the first element.

  calc_ses Logical indicating whether standard errors were calculated.

  variance_components A list exposing the two variance pieces (att_var_1, att_var_2) plus their paper-notation counterparts (V_1, V_2) and the unit-scaled variance estimators (tilde_v_N, hat_v_N, tilde_v_N_C, tilde_v_N_C_pi_hat, tilde_v_N_C_pi_hat_cons, tilde_v_N_cons) catalogued at paper line 2006. The Wald CI is [hat_T_N +- qnorm(1-alpha/2) * sqrt(tilde_v_N / N)] (paper Eq. conf.int.form). New in v1.12.0 (issue #141 + #146).

  first_year Integer or numeric scalar; the first (earliest) time_var value in the panel after idCohorts() processing. Consumed by eventStudy() to map cohort_probs' cohort labels (treatment-start years) to 1-based panel-time-index offsets when the labels are integer-coercible. New in v1.13.3 (issue #174).

  d_inv_treat The inverted custom treatment-effect fusion block solve(fusion_matrix) (a num_treats x num_treats numeric matrix), or NULL if no fusion_matrix was supplied. Consumed by eventStudy() and simultaneousCIs() so the access-time bands reuse the same fusion block the fit used (#236).

The object has methods for print(), summary(), and coef(). By default, print() and summary() only show the essential outputs. To see internal details, use print(x, show_internal = TRUE) or summary(x, show_internal = TRUE). The coef() method returns the vector of estimated coefficients (beta_hat).

Examples

```
## Not run:
  # Generate coefficients
  coefs <- genCoefs(G = 5, T = 30, d = 12, density = 0.1, eff_size = 2, seed = 123)

  # Simulate data using the coefficients
```

```
  sim_data <- simulateData(coefs, N = 120, sig_eps_sq = 5, sig_eps_c_sq = 5, seed = 123)

  result <- fetwfeWithSimulatedData(sim_data)

## End(Not run)
```

---

genCoefs *Generate Coefficient Vector for Data Generation*

Description

This function generates a coefficient vector `beta` for simulation studies of the fused extended two-way fixed effects estimator. It returns an S3 object of class `"FETWFE_coefs"` containing `beta` along with simulation parameters `G`, `T`, and `d`. See the simulation studies section of Faletto (2025) for details.

Usage

```
genCoefs(
  G = NULL,
  T,
  d,
  density,
  eff_size,
  fusion_structure = c("cohort", "event_study"),
  assignment_type = c("marginal", "multinomial", "ordered"),
  assignment_strength = 1,
  assignment_interactions = NULL,
  assignment_interaction_strength = NULL,
  n_signal_cohorts = NULL,
  treat_base_levels = NULL,
  seed = NULL,
  verbose = FALSE,
  R = NULL
)
```

Arguments

`G`	Optional integer. The number of treated cohorts (treatment is assumed to start in periods 2 to `G + 1`). Defaults to `NULL`; supply either `G` or the deprecated alias `R` (described below).
`T`	Integer. The total number of time periods.
`d`	Integer. The number of time-invariant covariates. If `d > 0`, additional terms corresponding to covariate main effects and interactions are included in `beta`.

`density` Numeric in (0,1]. The probability that any given entry in the initial coefficient vector `theta` is nonzero. `density = 1` gives a fully dense (non-sparse) coefficient vector. A `density` so small that an all-zero draw is near-certain is rejected rather than retried forever: the coefficient draw is retried at most 10,000 times and then errors.

`eff_size` Numeric. The magnitude used to scale nonzero entries in `theta`. Each nonzero entry is set to `eff_size` or `-eff_size` (with a 60 percent chance for a positive value).

`fusion_structure`

Character. One of `"cohort"` (default) or `"event_study"`. Controls the basis in which the true treatment-effect coefficients are sparse. Under `"cohort"` the sparse transformed coefficients `theta` are mapped back through the default two-way (cohort) inverse fusion transform (byte-identical to previous behavior); under `"event_study"` they are mapped through the event-study inverse fusion transform, so the true treatment effects are sparse in the event-study basis (effects sharing the same time since treatment, $e = t - g$, tend to be equal across cohorts). This is the simulation-side companion to the `fusion_structure` argument of `fetwfe()`.

`assignment_type`

Character. One of `"marginal"` (default), `"multinomial"`, or `"ordered"`. Selects the data-generating process for cohort assignment in `simulateData()`.

- `"marginal"`: each unit's cohort is drawn uniformly, independent of its covariates. Original pre-1.14.0 behavior, preserved byte-identically.
- `"multinomial"`: cohort assignment follows a multinomial-logit propensity-score model $\pi_g(x) = \exp(\gamma_g^\top x) / \sum_{g'} \exp(\gamma_{g'}^\top x)$ with $\gamma_0 \equiv 0$ (never-treated reference). Requires $d \geq 1$.
- `"ordered"`: cohort assignment follows a proportional-odds (ordered-logit) model with cumulative probabilities $P(W \leq g|x) = \text{plogis}(\alpha_g - \gamma^\top x)$. Cutpoints α_g are chosen so the marginal cohort probabilities are approximately uniform 1/(G + 1). Requires $d \geq 1$.

`assignment_strength`

Non-negative numeric scalar. Scales the logit coefficients in the propensity-score model. `0` reduces both non-marginal types to the uniform marginal distribution by construction. Larger values produce stronger covariate-cohort coupling. Defaults to `1.0`. Ignored when `assignment_type = "marginal"`.

`assignment_interactions`

Optional. A list of length-2 integer vectors, each naming a pair of covariate indices (j, k) in $[1, d]$ whose elementwise product $x_{i,j} \cdot x_{i,k}$ enters the propensity model as an additional column. Self-interactions `c(j, j)` are allowed and yield a quadratic term x_j^2. Unordered pairs are canonicalized to `c(min(j, k), max(j, k))` and duplicates are silently deduplicated (inspect `coefs$assignment_coefs$interactions` on the returned object to verify the retained list). The interaction columns enter the propensity model only; the outcome model continues to use the original covariates. Defaults to `NULL` (no interactions — v1.14.0 behavior is preserved byte-identically). Passing `list()` (empty list) is treated as equivalent to `NULL` — no interactions specified, behavior is identical to the v1.14.0 marginal-cohort path within `multinomial` / `ordered` DGPs. Errors when `assignment_type =`

"marginal" (the marginal DGP has no propensity model to augment). New in 1.14.1.

assignment_interaction_strength
: Optional non-negative numeric scalar. Scales the Gaussian draws of the interaction coefficients independently of assignment_strength. Defaults to NULL, which means "fall through to assignment_strength". Useful when stress-testing whether the nonlinear-propensity angle alone drives downstream differences (without simultaneously cranking the linear angle). Ignored when assignment_interactions = NULL. New in 1.14.1.

n_signal_cohorts
: (Optional) Targeted-sparsity mode (#332). A single positive integer k in 1..(G - 1). When supplied, the coefficient vector is built *deterministically* with the treatment signal placed on exactly k cohorts' fused base levels (and the covariate / interaction blocks left at zero), giving ||theta||_0 = k non-zeros with a guaranteed non-zero, *heterogeneous* cohort effect (overall ATT $\neq 0$ and positive cohort-weight variance $V_2 > 0$). This produces the simultaneously sparse and non-degenerate high-dimensional ($p \geq NT$) data-generating process the desparsified / nodewise debiased-ATT coverage study needs, which the default uniform density placement cannot (it almost never lands signal on the tiny treatment-effect coordinate block). Cohort 1 is held at the baseline and cohorts 2..(k + 1) receive the distinct magnitudes eff_size * (1, 2, ..., k) (so eff_size must be non-zero). Defaults to NULL; NULL (with treat_base_levels = NULL) is the unchanged, byte-identical uniform-density behavior. Mutually exclusive with treat_base_levels.

treat_base_levels
: (Optional) Targeted-sparsity mode (#332), explicit form. A finite numeric vector of length G giving each cohort's fused base level directly (placed on getTreatInds()[getFirstInds()]) every other coordinate is left at zero, so ||theta||_0 is the number of non-zero entries. Note these are *fused* base levels, not the per-cohort ATTs: under fusion_structure = "cohort" a vector (b_1, ..., b_G) maps to the cumulative cohort effects cumsum(b) (e.g. c(0, m, 0) gives cohort ATTs (0, m, m)), and "event_study" uses a different but still heterogeneity-preserving map. The result must be non-degenerate and heterogeneous ($V_2 > 0$); an all-zero or constant-effect specification is rejected. Defaults to NULL. Mutually exclusive with n_signal_cohorts.

seed
: (Optional) Integer. Seed for reproducibility. If supplied, must be within +/- .Machine$integer.max — and because the two offsets below are derived from it, the usable maximum here is .Machine$integer.max - 2. Three deterministic offsets share this seed: the main coefficient draw uses seed; the assignment coefficients use seed + 1L; the Monte Carlo integration in getTes() uses seed + 2L. NA (or NULL) means "draw from the ambient random-number generator" — no set.seed() is called, so any preceding set.seed() is respected and the result varies across calls.

verbose
: Logical. If TRUE, emit a message() when assignment_interactions canonicalization removes duplicate or unordered pairs (e.g., when the user passes both c(1, 2) and c(2, 1)). Default FALSE (silent — users can verify the final canonical list via coefs$assignment_coefs$interactions).

R — Deprecated. The former name for G; still accepted with a deprecation warning, and will be removed in a future release. Use G.

Details

Optional arguments assignment_type and assignment_strength control whether cohort membership in the simulated panel is drawn marginally (independent of the covariates, the original behavior) or from a covariate-dependent propensity-score model — either a multinomial-logit or an ordered-logit (proportional-odds) model. The default assignment_type = "marginal" preserves the pre-1.14.0 behavior byte-identically. See vignette("simulation_vignette", package = "fetwfe") for worked examples.

The length of beta is given by

$$p = G + (T-1) + d + dG + d(T-1) + num_treats + (num_treats \times d)$$

, where the number of treatment parameters is defined as

$$num_treats = T \times G - \frac{G(G+1)}{2}$$

.

The function operates in two steps:

1. It first creates a sparse vector theta of length p, with nonzero entries occurring with probability density. Nonzero entries are set to eff_size or -eff_size (with a 60\
2. The full coefficient vector beta is then computed by applying an inverse fusion transform to theta using internal routines: genBackwardsInvFusionTransformMat() for the fixed-effect blocks and, for the treatment-effect block, genInvTwoWayFusionTransformMat() when fusion_structure = "cohort" or genInvEventStudyFusionTransformMat() when fusion_structure = "event_study".

The multinomial-logit and proportional-odds reference DGPs are the canonical parametric propensity-score models named in Faletto (2025) line 1016; the propensity-weighted population-truth aggregation matches Eq. att.estimator.weighted (line 837).

Note on the random-number generator: passing an explicit numeric seed calls set.seed(seed) internally and **leaves the global RNG advanced** after the call returns. This is deliberate — it makes a simulation both reproducible (the same seed always yields the same coefficients) and varying (drawing data afterwards consumes the advanced stream). To draw from / preserve the ambient random stream instead — without calling set.seed() — pass seed = NA (or seed = NULL).

Value

An object of class "FETWFE_coefs", which is a list containing:

beta A numeric vector representing the full coefficient vector after the inverse fusion transform.

theta A numeric vector representing the coefficient vector in the transformed feature space. theta is a sparse vector, which aligns with an assumption that deviations from the restrictions encoded in the FETWFE model are sparse. beta is derived from theta.

fusion_structure The fusion structure ("cohort" or "event_study") used to build the treatment-effect coefficients.

G The provided number of treated cohorts.

R Deprecated alias for `G`, retained for backward compatibility; populated with the same value. Use `G`. Will be removed in a future release.

T The provided number of time periods.

d The provided number of covariates.

seed The provided seed.

assignment_type The selected cohort-assignment DGP (`"marginal"` / `"multinomial"` / `"ordered"`). New in 1.14.0.

assignment_strength The scaling factor applied to the assignment coefficients. New in 1.14.0.

assignment_interaction_strength The scaling factor applied to the interaction coefficients. `NULL` when no interactions were specified or when the user passed `NULL` (the fall-through default). New in 1.14.1.

assignment_coefs `NULL` when `assignment_type = "marginal"`; otherwise a list with elements `type`, `strength`, `coefs` (the gamma matrix or vector), and (for ordered) `cutpoints`. Starting in 1.14.1, `assignment_coefs` also carries the sub-slots `interactions` (the canonicalized + deduplicated list of pairs, or `NULL`), `delta` (the interaction coefficient matrix for multinomial or vector for ordered, or `NULL`), and `interaction_strength` (the effective scaling factor used for the `delta` draws, or `NULL` when no interactions). New in 1.14.0; `interactions`, `delta`, and `interaction_strength` sub-slots new in 1.14.1.

References

Faletto, G (2025). Fused Extended Two-Way Fixed Effects for Difference-in-Differences with Staggered Adoptions. *arXiv preprint arXiv:2312.05985*. https://arxiv.org/abs/2312.05985.

See Also

`fetwfe`, whose `fusion_structure` argument this mirrors on the estimation side; `vignette("fusion_structure_vignette", package = "fetwfe")` for the cohort-vs-event-study distinction, and `vignette("simulation_vignette", package = "fetwfe")` for the full simulation pipeline.

Examples

```
## Not run:
  # Generate coefficients
  coefs <- genCoefs(G = 5, T = 30, d = 12, density = 0.1, eff_size = 2, seed = 123)

  # Simulate data using the coefficients
  sim_data <- simulateData(coefs, N = 120, sig_eps_sq = 5, sig_eps_c_sq = 5, seed = 123)

  # Event-study-sparse truth: treatment effects that share the same time
  # since treatment are fused across cohorts (the simulation-side companion
  # to fetwfe()'s fusion_structure = "event_study").
  coefs_es <- genCoefs(
    G = 5, T = 30, d = 12, density = 0.1, eff_size = 2,
    fusion_structure = "event_study", seed = 123
  )
```

```
  # Covariate-dependent cohort assignment: multinomial-logit DGP
  coefs_mn <- genCoefs(
    G = 5, T = 30, d = 12, density = 0.1, eff_size = 2,
    assignment_type = "multinomial", assignment_strength = 1.0,
    seed = 123
  )
  sim_mn <- simulateData(coefs_mn, N = 120, sig_eps_sq = 5, sig_eps_c_sq = 5, seed = 123)

  # Covariate-dependent cohort assignment with nonlinear propensity
  # (multinomial-logit + a single x1*x2 interaction term in the propensity
  # model only; outcome model continues to use plain X):
  coefs_int <- genCoefs(
    G = 5, T = 30, d = 12, density = 0.1, eff_size = 2,
    assignment_type = "multinomial",
    assignment_interactions = list(c(1, 2)),
    assignment_interaction_strength = 1.5,
    seed = 123
  )

## End(Not run)
```

---

genCoefsCore *Generate Coefficient Vector for Data Generation (core)*

Description

This function generates a coefficient vector `beta` along with a sparse auxiliary vector `theta` for simulation studies of the fused extended two-way fixed effects estimator. The returned `beta` is formatted to align with the design matrix created by `simulateDataCore()`, and is a valid input for the `beta` argument of that function. The vector `theta` is sparse, with nonzero entries occurring with probability `density` and scaled by `eff_size`. See the simulation studies section of Faletto (2025) for details.

Usage

```
genCoefsCore(
  G = NULL,
  T,
  d,
  density,
  eff_size,
  fusion_structure = c("cohort", "event_study"),
  n_signal_cohorts = NULL,
  treat_base_levels = NULL,
  seed = NULL,
  R = NULL
)
```

Arguments

G	Integer. The number of treated cohorts (treatment is assumed to start in periods 2 to G + 1). Defaults to NULL; supply either G or the deprecated alias R (described below).
T	Integer. The total number of time periods.
d	Integer. The number of time-invariant covariates. If d > 0, additional terms corresponding to covariate main effects and interactions are included in beta.
density	Numeric in (0,1]. The probability that any given entry in the initial coefficient vector theta is nonzero. density = 1 gives a fully dense (non-sparse) coefficient vector. A density so small that an all-zero draw is near-certain is rejected rather than retried forever: the coefficient draw is retried at most 10,000 times and then errors.
eff_size	Numeric. The magnitude used to scale nonzero entries in theta. Each nonzero entry is set to eff_size or -eff_size (with a 60 percent chance for a positive value).
fusion_structure	Character. One of "cohort" (default) or "event_study". Selects the inverse fusion transform applied to the treatment-effect block, controlling the basis in which the true treatment effects are sparse. "cohort" is byte-identical to previous behavior; "event_study" fuses effects at the same time since treatment across cohorts. See genCoefs() for details.
n_signal_cohorts, treat_base_levels	(Optional) Targeted-sparsity mode (#332). Both NULL (the default) is the unchanged, byte-identical uniform-density path. Otherwise the treatment signal is placed deterministically on the per-cohort fused base levels for a sparse, non-degenerate, heterogeneous high-dimensional DGP. See genCoefs() for the full description; the two are mutually exclusive.
seed	(Optional) Integer. Seed for reproducibility. If supplied, must be within +/- .Machine$integer.max. NA (or NULL) means "draw from the ambient random-number generator" — no set.seed() is called.
R	Deprecated. The former name for G; still accepted with a deprecation warning, and will be removed in a future release. Use G.

Details

The length of beta is given by

$$p = G + (T-1) + d + dG + d(T-1) + num_treats + (num_treats \times d)$$

, where the number of treatment parameters is defined as

$$num_treats = T \times G - \frac{G(G+1)}{2}$$

.

The function operates in two steps:

1. It first creates a sparse vector `theta` of length p, with nonzero entries occurring with probability `density`. Nonzero entries are set to `eff_size` or `-eff_size` (with a 60\
2. The full coefficient vector `beta` is then computed by applying an inverse fusion transform to `theta` using internal routines: `genBackwardsInvFusionTransformMat()` for the fixed-effect blocks and, for the treatment-effect block, `genInvTwoWayFusionTransformMat()` when `fusion_structure = "cohort"` or `genInvEventStudyFusionTransformMat()` when `fusion_structure = "event_study"`.

Value

A list with two elements:

`beta` A numeric vector representing the full coefficient vector after the inverse fusion transform.

theta A numeric vector representing the coefficient vector in the transformed feature space. `theta` is a sparse vector, which aligns with an assumption that deviations from the restrictions encoded in the FETWFE model are sparse. `beta` is derived from `theta`.

References

Faletto, G (2025). Fused Extended Two-Way Fixed Effects for Difference-in-Differences with Staggered Adoptions. *arXiv preprint arXiv:2312.05985*. https://arxiv.org/abs/2312.05985.

Examples

```
## Not run:
  # Set parameters for the coefficient generation
  G <- 3         # Number of treated cohorts
  T <- 6         # Total number of time periods
  d <- 2         # Number of covariates
  density <- 0.1 # Probability that an entry in the initial vector is nonzero
  eff_size <- 1.5  # Scaling factor for nonzero coefficients
  seed <- 789    # Seed for reproducibility

  # Generate coefficients using genCoefsCore()
  coefs_core <- genCoefsCore(G = G, T = T, d = d, density = density,
  eff_size = eff_size, seed = seed)
  beta <- coefs_core$beta
  theta <- coefs_core$theta

  # For diagnostic purposes, compute the expected length of beta.
  # The length p is defined internally as:
  #   p = G + (T - 1) + d + d*G + d*(T - 1) + num_treats + num_treats*d,
  # where num_treats = T * G - (G*(G+1))/2.
  num_treats <- T * G - (G * (G + 1)) / 2
  p_expected <- G + (T - 1) + d + d * G + d * (T - 1) + num_treats + num_treats * d

  cat("Length of beta:", length(beta), "\nExpected length:", p_expected, "\n")

## End(Not run)
```

getTes *Compute True Treatment Effects*

Description

This function extracts the true treatment effects from a full coefficient vector as generated by `genCoefs()`. It returns the per-cohort CATTs and an overall ATT. Under the default marginal cohort-assignment DGP, the overall ATT is the equal-weighted mean of the cohort-specific effects. Under the covariate-dependent DGPs introduced in 1.14.0, the overall ATT is a propensity-weighted mean using cohort weights $E[\pi_g(X)] / \sum_{g' \text{ treated}} E[\pi_{g'}(X)]$, matching Faletto (2025) Eq. `att.estimator.weighted` (line 837) at the population level. The expected propensities are computed by Monte Carlo integration over the covariate distribution.

Usage

```
getTes(coefs_obj, distribution = "gaussian")
```

Arguments

`coefs_obj`	An object of class `"FETWFE_coefs"` containing the coefficient vector and simulation parameters.
`distribution`	Character; the covariate distribution to integrate the propensity-weighted truth over. Must match the `distribution` passed to `simulateData` for the panel whose truth you want. One of `"gaussian"` (default) or `"uniform"`. Only affects covariate-dependent assignment (`assignment_type != "marginal"`); it is ignored under marginal assignment, where the cohort weights are uniform.

Details

The function internally uses auxiliary routines `getNumTreats()`, `getP()`, `getFirstInds()`, `getTreatInds()`, and `getActualCohortTes()` to determine the correct indices of treatment effect coefficients in `beta`. The overall treatment effect is computed as a weighted average of the cohort-specific effects (uniform weights under the marginal DGP, propensity weights otherwise).

Under non-marginal DGPs, $E[\pi_g(X)]$ is estimated by Monte Carlo integration over the X distribution set by `distribution` (`"gaussian"` by default) with `M = 10000` draws. The Monte Carlo seed is offset from the main `coefs_obj$seed` by `+ 2L` per the documented seed-offset convention.

Value

An object of class `"FETWFE_tes"`, which is a list with the following elements:

att_true A numeric value representing the overall average treatment effect on the treated. Under the marginal DGP this is the equal-weighted mean of the cohort-specific effects; under covariate-dependent DGPs it is the propensity-weighted mean using `cohort_weights`.

actual_cohort_tes A numeric vector of length `G` containing the true cohort-specific treatment effects, calculated by averaging the coefficients corresponding to the treatment dummies for each cohort. Intrinsic to β; does not depend on the assignment DGP.

actual_event_time_tes A named numeric vector of length `T - 1` giving the true treatment effect at each event time (periods since adoption) `e = 0, 1, ..., T - 2`, with names `"0", ..., "T-2"`. Each event time's effect is the mean of the true per-`(g, t)` cell effects over the cohorts still observed at that event time, weighted in proportion to `cohort_weights` — the same aggregation `eventStudy()` estimates on a fitted panel. `NA` at an event time no cohort reaches (where `eventStudy()` instead reports `0`). New in 1.56.7.

cohort_times An integer vector of length `G` giving the calendar time period at which each treated cohort first adopts treatment. In the simulator's convention cohort `g` adopts at calendar time `g + 1` (cohort 0 is never-treated).

cohort_weights Numeric vector of length `G` summing to 1. Under the marginal DGP this is uniform `1/G`. Under `assignment_type = "multinomial"` or `"ordered"` it is $E[\pi_g(X)] / \sum_{g' \text{ treated}} E[\pi_{g'}(X)]$. New in 1.14.0.

assignment_type Character; the cohort-assignment DGP carried over from `coefs_obj` (one of `"marginal"`, `"multinomial"`, or `"ordered"`). Determines whether `cohort_weights` is uniform (marginal) or propensity-weighted. New in 1.18.1.

assignment_strength Numeric; the assignment-strength scaling carried over from `coefs_obj` (meaningful only when `assignment_type != "marginal"`). `NULL` for `FETWFE_coefs` objects saved before 1.14.0. New in 1.18.1.

G, T, d, seed The generating parameters carried over from `coefs_obj` so that `print()` and `summary()` on the returned object are self-describing.

R Deprecated alias for `G`, retained for backward compatibility; populated with the same value. Use `G`. Will be removed in a future release.

Use `print()` or `summary()` on the returned object for a formatted display.

References

Faletto, G (2025). Fused Extended Two-Way Fixed Effects for Difference-in-Differences with Staggered Adoptions. *arXiv preprint arXiv:2312.05985*. https://arxiv.org/abs/2312.05985.

Examples

```
## Not run:
# Generate coefficients
coefs <- genCoefs(G = 5, T = 30, d = 12, density = 0.1, eff_size = 2, seed = 123)

# Compute the true treatment effects:
te_results <- getTes(coefs)

# Overall average treatment effect on the treated:
print(te_results$att_true)

# Cohort-specific treatment effects:
print(te_results$actual_cohort_tes)

# True effect at each event time (periods since adoption):
print(te_results$actual_event_time_tes)

# Or use the new print method for a self-describing display:
```

```
print(te_results)

# Propensity-weighted truth under covariate-dependent DGP:
coefs_mn <- genCoefs(G = 3, T = 5, d = 2, density = 0.5, eff_size = 2,
                     assignment_type = "multinomial", assignment_strength = 1.0,
                     seed = 42)
te_mn <- getTes(coefs_mn)
te_mn$att_true       # propensity-weighted overall ATT
te_mn$cohort_weights # length G; sums to 1

## End(Not run)
```

`glance.betwfe` *Glance a* `betwfe` *fitted object*

Description

Same schema as `glance.fetwfe()` (BETWFE also has regularization).

Usage

```
## S3 method for class 'betwfe'
glance(x, ...)
```

Arguments

`x`	An object of class `"betwfe"`.
`...`	Unused.

Value

A one-row data frame with 16 columns.

Examples

```
## Not run:
  res <- betwfeWithSimulatedData(
    simulateData(genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  )
  broom::glance(res)

## End(Not run)
```

glance.etwfe *Glance an* etwfe *fitted object*

Description

Like glance.fetwfe() but omits the lambda_star / lambda_star_model_size columns — ETWFE has no regularization.

Usage

```
## S3 method for class 'etwfe'
glance(x, ...)
```

Arguments

x	An object of class "etwfe".
...	Unused.

Value

A one-row data frame with 11 columns.

Examples

```
## Not run:
  res <- etwfeWithSimulatedData(
    simulateData(genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  )
  broom::glance(res)

## End(Not run)
```

glance.fetwfe *Glance an* fetwfe *fitted object*

Description

Returns a one-row broom-style summary data frame with model-level scalars: panel-shape counts (nobs, n_units, n_periods, n_cohorts, n_covs, n_features), bridge-regression tuning (lambda_star, lambda_star_model_size), variance components (sig_eps_sq, sig_eps_c_sq), and inference settings (alpha, se_type, indep_counts_used).

Usage

```
## S3 method for class 'fetwfe'
glance(x, ...)
```

Arguments

x	An object of class "fetwfe".
...	Unused.

Value

A one-row data frame with 16 columns.

Examples

```
## Not run:
  res <- fetwfeWithSimulatedData(
    simulateData(genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  )
  broom::glance(res)

## End(Not run)
```

---

glance.twfeCovs *Glance a* twfeCovs *fitted object*

Description

Like glance.etwfe() (and with the same schema): omits the lambda_star / lambda_star_model_size columns, since twfeCovs performs no regularization.

Usage

```
## S3 method for class 'twfeCovs'
glance(x, ...)
```

Arguments

x	An object of class "twfeCovs".
...	Unused.

Value

A one-row data frame with 11 columns.

Examples

```
## Not run:
  res <- twfeCovsWithSimulatedData(
    simulateData(genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  )
  broom::glance(res)

## End(Not run)
```

`plot.betwfe` *Plot CATT or event-study estimates from a fitted BETWFE*

Description

Parallel to `plot.fetwfe()`. BETWFE uses the same bridge-penalty selection mechanism, so `selected` is encoded the same way (TRUE = nonzero, FALSE = zeroed by the bridge penalty).

Usage

```
## S3 method for class 'betwfe'
plot(x, type = c("event_study", "catt"), conf_int = TRUE, alpha = NULL, ...)
```

Arguments

`x`	A fitted object from `fetwfe()`, `etwfe()`, or `betwfe()`. (`twfeCovs` is not currently supported – see GitHub issue #58 for the broader treatment of `twfeCovs` class methods.)
`type`	Character; either `"event_study"` (default; event-time pooled coefficients from `eventStudy()`) or `"catt"` (per-cohort ATTs from `result$catt_df`).
`conf_int`	Logical; if `TRUE` (default), include confidence interval error bars.
`alpha`	Numeric; overrides the fit's alpha for CI computation. `NULL` (default) uses the fit's alpha (so 95% CIs when the fit's alpha is 0.05). With the default `NULL`, both views show the fit's `ci_type` band (simultaneous by default): the `"catt"` view reads the bounds stored in `catt_df`, and the `"event_study"` view re-derives them via `eventStudy()` (which inherits the fit's `ci_type`). **Asymmetry under an explicit `alpha`:** for the `"event_study"` view, the explicit alpha is forwarded to `eventStudy()`, which re-runs the joint machinery and so still produces a *simultaneous* band at that alpha (when the fit's `ci_type` is `"simultaneous"`); for the `"catt"` view, an explicit alpha recomputes `ci_low` / `ci_high` from `estimate +/- qnorm(1 - alpha/2) * se`, i.e. a *pointwise* band at that alpha (the stored simultaneous bounds are at the fit's alpha and are not re-derived at a new alpha for the catt view). To plot simultaneous catt bands at a different alpha, refit at that alpha or call `simultaneousCIs()` directly.
`...`	Currently unused; reserved for future arguments.

Value

A ggplot object.

See Also

plot.fetwfe() for the full documentation; eventStudy(); cohortStudy().

Examples

```
## Not run:
  coefs <- genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2)
  dat <- simulateData(coefs, N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  res <- betwfeWithSimulatedData(dat)
  if (requireNamespace("ggplot2", quietly = TRUE)) {
    plot(res)
  }

## End(Not run)
```

---

plot.etwfe *Plot CATT or event-study estimates from a fitted ETWFE*

Description

Parallel to plot.fetwfe(). ETWFE does not perform selection, so all points are uniformly styled (no selected = TRUE / FALSE encoding).

Usage

```
## S3 method for class 'etwfe'
plot(x, type = c("event_study", "catt"), conf_int = TRUE, alpha = NULL, ...)
```

Arguments

x	A fitted object from fetwfe(), etwfe(), or betwfe(). (twfeCovs is not currently supported – see GitHub issue #58 for the broader treatment of twfeCovs class methods.)
type	Character; either "event_study" (default; event-time pooled coefficients from eventStudy()) or "catt" (per-cohort ATTs from result$catt_df).
conf_int	Logical; if TRUE (default), include confidence interval error bars.
alpha	Numeric; overrides the fit's alpha for CI computation. NULL (default) uses the fit's alpha (so 95% CIs when the fit's alpha is 0.05). With the default NULL, both views show the fit's ci_type band (simultaneous by default): the "catt" view reads the bounds stored in catt_df, and the "event_study" view re-derives them via eventStudy() (which inherits the fit's ci_type). **Asymmetry under an explicit alpha:** for the "event_study" view, the explicit alpha is forwarded

to eventStudy(), which re-runs the joint machinery and so still produces a *simultaneous* band at that alpha (when the fit's ci_type is "simultaneous"); for the "catt" view, an explicit alpha recomputes ci_low / ci_high from estimate +/- qnorm(1 - alpha/2) * se, i.e. a *pointwise* band at that alpha (the stored simultaneous bounds are at the fit's alpha and are not re-derived at a new alpha for the catt view). To plot simultaneous catt bands at a different alpha, refit at that alpha or call simultaneousCIs() directly.

... Currently unused; reserved for future arguments.

Value

A ggplot object.

See Also

plot.fetwfe() for the full documentation; eventStudy(); cohortStudy().

Examples

```
## Not run:
  coefs <- genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2)
  dat <- simulateData(coefs, N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  res <- etwfeWithSimulatedData(dat)
  if (requireNamespace("ggplot2", quietly = TRUE)) {
    plot(res)
  }

## End(Not run)
```

---

plot.fetwfe *Plot CATT or event-study estimates from a fitted FETWFE / ETWFE / BETWFE*

Description

Returns a ggplot object showing either event-study coefficients (default; type = "event_study") or per-cohort average treatment effects (type = "catt") from a fitted estimator. Mirrors the visualization style of did::ggdid() from the Callaway-Sant'Anna did package, providing a single-call route from a fitted object to a publication-ready visualization.

For fetwfe / betwfe (the bridge-penalty estimators) in the CATT view, points are shape- and color-coded by whether the bridge penalty left that cohort's ATT nonzero (selected = TRUE) or zeroed it out (selected = FALSE). For etwfe (no selection), all points are uniformly styled.

Usage

```
## S3 method for class 'fetwfe'
plot(x, type = c("event_study", "catt"), conf_int = TRUE, alpha = NULL, ...)
```

Arguments

x	A fitted object from fetwfe(), etwfe(), or betwfe(). (twfeCovs is not currently supported – see GitHub issue #58 for the broader treatment of twfeCovs class methods.)
type	Character; either "event_study" (default; event-time pooled coefficients from eventStudy()) or "catt" (per-cohort ATTs from result$catt_df).
conf_int	Logical; if TRUE (default), include confidence interval error bars.
alpha	Numeric; overrides the fit's alpha for CI computation. NULL (default) uses the fit's alpha (so 95% CIs when the fit's alpha is 0.05). With the default NULL, both views show the fit's ci_type band (simultaneous by default): the "catt" view reads the bounds stored in catt_df, and the "event_study" view re-derives them via eventStudy() (which inherits the fit's ci_type). **Asymmetry under an explicit** alpha**:** for the "event_study" view, the explicit alpha is forwarded to eventStudy(), which re-runs the joint machinery and so still produces a *simultaneous* band at that alpha (when the fit's ci_type is "simultaneous"); for the "catt" view, an explicit alpha recomputes ci_low / ci_high from estimate +/- qnorm(1 - alpha/2) * se, i.e. a *pointwise* band at that alpha (the stored simultaneous bounds are at the fit's alpha and are not re-derived at a new alpha for the catt view). To plot simultaneous catt bands at a different alpha, refit at that alpha or call simultaneousCIs() directly.
...	Currently unused; reserved for future arguments.

Value

A ggplot object. Users can customize further via standard ggplot layer-addition syntax (e.g., plot(res) + ggplot2::theme_classic()).

See Also

cohortStudy() for the per-cohort accessor; eventStudy() for the event-time accessor; plot.etwfe(), plot.betwfe() for the parallel methods.

Examples

```
## Not run:
  coefs <- genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2)
  dat <- simulateData(coefs, N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  res <- fetwfeWithSimulatedData(dat)
  if (requireNamespace("ggplot2", quietly = TRUE)) {
    plot(res)                          # default: event-study coefficients
    plot(res, type = "catt")           # per-cohort ATTs
    plot(res, conf_int = FALSE)        # point estimates only
    plot(res, alpha = 0.1)             # 90% CIs
  }

## End(Not run)
```

plot.simultaneous_cis *Plot a simultaneous-confidence-interval object*

Description

Draws the per-effect estimates with both bands overlaid – the wider simultaneous band (error bars) on top of the narrower pointwise band (thick line range) – for a `simultaneousCIs()` result, so the family-wise vs per-effect coverage trade-off is visible.

Usage

```
## S3 method for class 'simultaneous_cis'
plot(x, ...)
```

Arguments

`x`	A `"simultaneous_cis"` object from `simultaneousCIs()`.
`...`	Ignored.

Value

A `ggplot` object.

Note

Requires the `ggplot2` package (in Suggests). Install via `install.packages("ggplot2")` if it is not already installed. Matches `R/plot.R`'s `.plot_estimator` precedent for ggplot2-dependent plot methods.

plot.twfeCovs *Plot is not defined for a twfeCovs fit (documented omission)*

Description

`plot()` is intentionally not provided for `twfeCovs()` objects (#58): `twfeCovs()` estimates one pooled effect per cohort, so there is no per-(cohort, time) / event-study structure to plot. Use `summary()` or `tidy.twfeCovs()` for the cohort effects. Calling this method always raises an error.

Usage

```
## S3 method for class 'twfeCovs'
plot(x, ...)
```

Arguments

x	An object of class "twfeCovs".
...	Ignored.

Value

(none; raises an error).

---

print.debiased_att *Print a debiased overall-ATT estimate*

Description

Compact display of debiasedATT()'s estimate, standard error, and Wald confidence interval. In the high-dimensional (p >= NT) regime it also surfaces the nodewise-direction diagnostics (lambda_c / lambda_node, KKT feasibility, convergence) the documentation tells users to inspect; the fixed-p print shows only the estimate block.

Usage

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

Arguments

x	A "debiased_att" object from debiasedATT().
...	Ignored.

Value

x, invisibly.

---

print.simultaneous_cis *Print a simultaneous-confidence-interval object*

Description

Compact display of a simultaneousCIs() result: the family, K, and alpha; the simultaneous / pointwise / Bonferroni critical values; the high-dimensional desparsified-band diagnostics (only in the p >= NT regime); and the per-effect interval table.

Usage

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

Arguments

x	A "simultaneous_cis" object from simultaneousCIs().
...	Ignored.

Value

x, invisibly.

---

simulateData *Generate Random Panel Data for FETWFE Simulations*

Description

Generates a random panel data set for simulation studies of the fused extended two-way fixed effects (FETWFE) estimator by taking an object of class "FETWFE_coefs" (produced by genCoefs()) and using it to simulate data. The function creates a balanced panel with N units over T time periods, assigns treatment status across G treated cohorts (with equal marginal probabilities for treatment and non-treatment), and constructs a design matrix along with the corresponding outcome. The covariates are generated according to the specified distribution: by default, covariates are drawn from a normal distribution; if distribution = "uniform", they are drawn uniformly from $[-\sqrt{3}, \sqrt{3}]$. When $d = 0$ (i.e. no covariates), no covariate-related columns or interactions are generated. See the simulation studies section of Faletto (2025) for details.

Usage

```
simulateData(
  coefs_obj,
  N,
  sig_eps_sq,
  sig_eps_c_sq,
  distribution = "gaussian",
  guarantee_rank_condition = FALSE,
  seed = NULL
)
```

Arguments

coefs_obj	An object of class "FETWFE_coefs" containing the coefficient vector and simulation parameters.
N	Integer. Number of units in the panel. Units are allocated among the G + 1 groups by a retry that is bounded at 1,000,000 attempts; an N at or very near the smallest admissible size (G + 1, or (G + 1) * (d + 1) when guarantee_rank_condition = TRUE) can exhaust that budget and error rather than returning. Raising N is the fix, and the error says so.
sig_eps_sq	Numeric. Variance of the idiosyncratic (observation-level) noise.

`sig_eps_c_sq` Numeric. Variance of the unit-level random effects. Must be non-negative; `0` is allowed (yields a panel with no unit-level random effects).

`distribution` Character. Distribution to generate covariates. Defaults to `"gaussian"`. If set to `"uniform"`, covariates are drawn uniformly from $[-\sqrt{3}, \sqrt{3}]$. To obtain a matching ground truth from `getTes`, pass the same `distribution` value there.

`guarantee_rank_condition` (Optional). Logical. If TRUE, the returned data set is guaranteed to have at least `d + 1` units per cohort, which is necessary for the final design matrix to have full column rank. Default is FALSE, in which case only `>= 1` unit per cohort is required – this permits small cohorts and therefore high-dimensional (`p > NT`) panels, which `fetwfe()` fits in its regularized regime (and on which the high-dimensional `debiasedATT()` path can be exercised, using the known data-generating coefficient vector; recovering the truth requires an adequate number of units).

`seed` (Optional) Controls the random-number generator for the simulated panel. As of fetwfe 1.24.0 the default is `NULL`, which draws from the ambient random-number generator (respecting any preceding `set.seed()`) and emits a warning; pass an integer within `+/- .Machine$integer.max` for a reproducible panel, or `NA` to draw from the ambient generator silently. `simulateData()` no longer reuses `coefs_obj$seed` (the seed `genCoefs()` used to build the coefficients). Default `NULL`.

Details

This function extracts simulation parameters from the `FETWFE_coefs` object and passes them, along with additional simulation parameters, to the internal function `simulateDataCore()`. It validates that all necessary components are returned and assigns the S3 class `"FETWFE_simulated"` to the output.

The random draw is controlled by the `seed` argument, not by `coefs_obj$seed`. By default (`seed = NULL`) `simulateData()` draws from the ambient random-number generator (so a preceding `set.seed()` is respected and repeated calls return different panels) and emits a warning noting that this default changed in fetwfe 1.24.0. Pass an integer `seed` for a reproducible panel (the same integer always yields the same panel), or `seed = NA` to use the ambient generator without the warning. To vary the panel across Monte Carlo replications, pass a different `seed` each replication.

Passing an explicit numeric `seed` calls `set.seed(seed)` internally and **leaves the global RNG advanced** after the call returns. This is deliberate — it makes a simulation both reproducible (the same `seed` always yields the same panel) and varying (subsequent draws consume the advanced stream). Use `seed = NA` (or the default `seed = NULL`) to draw from / preserve the ambient stream without calling `set.seed()`.

The argument `distribution` controls the generation of covariates. For `"gaussian"`, covariates are drawn from `rnorm`; for `"uniform"`, they are drawn from `runif` on the interval $[-\sqrt{3}, \sqrt{3}]$ (which ensures that the covariates have unit variance regardless of which distribution is chosen).

When $d = 0$ (i.e. no covariates), the function omits any covariate-related columns and their interactions.

Value

An object of class `"FETWFE_simulated"`, which is a list containing:

pdata A dataframe containing generated data that can be passed to `fetwfe()`.

X The design matrix X, with p columns with interactions.

y A numeric vector of length $N \times T$ containing the generated responses.

covs A character vector containing the names of the generated features (if $d > 0$), or simply an empty vector (if $d = 0$)

time_var The name of the time variable in pdata

unit_var The name of the unit variable in pdata

treatment The name of the treatment variable in pdata

response The name of the response variable in pdata

coefs The coefficient vector β used for data generation.

first_inds A vector of indices indicating the first treatment effect for each treated cohort.

N_UNTREATED The number of never-treated units.

assignments A vector of counts (of length $G + 1$) indicating how many units fall into the never-treated group and each of the G treated cohorts.

indep_counts Independent cohort assignments (for auxiliary purposes).

p The number of columns in the design matrix X.

N Number of units.

T Number of time periods.

G Number of treated cohorts.

R Deprecated alias for `G`, retained for backward compatibility; populated with the same value. Use `G`. Will be removed in a future release.

d Number of covariates.

sig_eps_sq The idiosyncratic noise variance.

sig_eps_c_sq The unit-level noise variance.

References

Faletto, G (2025). Fused Extended Two-Way Fixed Effects for Difference-in-Differences with Staggered Adoptions. *arXiv preprint arXiv:2312.05985*. https://arxiv.org/abs/2312.05985.

Examples

```
## Not run:
  # Generate coefficients
  coefs <- genCoefs(G = 5, T = 30, d = 12, density = 0.1, eff_size = 2, seed = 123)

  # Simulate data using the coefficients
  sim_data <- simulateData(coefs, N = 120, sig_eps_sq = 5, sig_eps_c_sq = 5)

## End(Not run)
```

`simulateDataCore` *Generate Random Panel Data for FETWFE Simulations (core)*

Description

Generates a random panel data set for simulation studies of the fused extended two-way fixed effects (FETWFE) estimator. The function creates a balanced panel with N units over T time periods, assigns treatment status across G treated cohorts (with equal marginal probabilities for treatment and non-treatment), and constructs a design matrix along with the corresponding outcome. When `gen_ints = TRUE` the full design matrix is returned (including interactions between covariates and fixed effects and treatment indicators). When `gen_ints = FALSE` the design matrix is generated in a simpler format (with no interactions) as expected by `fetwfe()`. Moreover, the covariates are generated according to the specified `distribution`: by default, covariates are drawn from a normal distribution; if `distribution = "uniform"`, they are drawn uniformly from $[-\sqrt{3}, \sqrt{3}]$.

When $d = 0$ (i.e. no covariates), no covariate-related columns or interactions are generated.

See the simulation studies section of Faletto (2025) for details.

Usage

```
simulateDataCore(
  N,
  T,
  G = NULL,
  d,
  sig_eps_sq,
  sig_eps_c_sq,
  beta,
  seed = NULL,
  gen_ints = FALSE,
  distribution = "gaussian",
  guarantee_rank_condition = FALSE,
  assignment_type = "marginal",
  assignment_coefs = NULL,
  R = NULL
)
```

Arguments

`N`	Integer. Number of units in the panel. Units are allocated among the `G + 1` groups by a retry that is bounded at 1,000,000 attempts; an `N` at or very near the smallest admissible size (`G + 1`, or `(G + 1) * (d + 1)` when `guarantee_rank_condition = TRUE`) can exhaust that budget and error rather than returning. Raising `N` is the fix, and the error says so.
`T`	Integer. Number of time periods.
`G`	Integer. Number of treated cohorts (with treatment starting in periods 2 to T).

d — Integer. Number of time-invariant covariates.

sig_eps_sq — Numeric. Variance of the idiosyncratic (observation-level) noise.

sig_eps_c_sq — Numeric. Variance of the unit-level random effects. Must be non-negative; 0 is allowed (yields a panel with no unit-level random effects).

beta — Numeric vector. Coefficient vector for data generation. Its required length depends on the value of gen_ints:

- If gen_ints = TRUE and d > 0, the expected length is $p = G + (T-1) + d + dG + d(T-1) + num_treats + num_treats \times d$, where $num_treats = T \times G - \frac{G(G+1)}{2}$.
- If gen_ints = TRUE and d = 0, the expected length is $p = G + (T-1) + num_treats$.
- If gen_ints = FALSE, the expected length is $p = G + (T-1) + d + num_treats$.

seed — (Optional) Integer. Seed for reproducibility. If supplied, must be within +/- .Machine$integer.max.

gen_ints — Logical. If TRUE, generate the full design matrix with interactions; if FALSE (the default), generate a design matrix without any interaction terms.

distribution — Character. Distribution to generate covariates. Defaults to "gaussian". If set to "uniform", covariates are drawn uniformly from $[-\sqrt{3}, \sqrt{3}]$. To obtain a matching ground truth from getTes, pass the same distribution value there.

guarantee_rank_condition — (Optional). Logical. If TRUE, the returned data set is guaranteed to have at least d + 1 units per cohort, which is necessary for the final design matrix to have full column rank. Default is FALSE, in which case only >= 1 unit per cohort is required – this permits small cohorts and therefore high-dimensional (p > NT) panels, which fetwfe() fits in its regularized regime (and on which the high-dimensional debiasedATT() path can be exercised, using the known data-generating coefficient vector; recovering the truth requires an adequate number of units).

assignment_type — Character. One of "marginal" (default), "multinomial", or "ordered". Selects the cohort-assignment DGP. "marginal" preserves the pre-1.14.0 behavior. The non-marginal types require a non-NULL assignment_coefs argument (typically pulled from a FETWFE_coefs object built with the matching assignment_type).

assignment_coefs — Optional list returned by .gen_assignment_coefs() (an internal helper). Required when assignment_type != "marginal".

R — Deprecated. The former name for G; still accepted with a deprecation warning, and will be removed in a future release. Use G.

Details

When gen_ints = TRUE, the function constructs the design matrix by first generating base fixed effects and a long-format covariate matrix (via generateBaseEffects()), then appending interactions between the covariates and cohort/time fixed effects (via generateFEInts()) and finally treatment indicator columns and treatment-covariate interactions (via genTreatVarsSim()

and genTreatInts()). When gen_ints = FALSE, the design matrix consists only of the base fixed effects, covariates, and treatment indicators.

The argument distribution controls the generation of covariates. For "gaussian", covariates are drawn from rnorm; for "uniform", they are drawn from runif on the interval $[-\sqrt{3}, \sqrt{3}]$.

When $d = 0$ (i.e. no covariates), the function omits any covariate-related columns and their interactions.

Value

An object of class "FETWFE_simulated", which is a list containing:

pdata A dataframe containing generated data that can be passed to fetwfe().

X The design matrix. When gen_ints = TRUE, X has p columns with interactions; when gen_ints = FALSE, X has no interactions.

y A numeric vector of length $N \times T$ containing the generated responses.

covs A character vector containing the names of the generated features (if $d > 0$), or simply an empty vector (if $d = 0$)

time_var The name of the time variable in pdata

unit_var The name of the unit variable in pdata

treatment The name of the treatment variable in pdata

response The name of the response variable in pdata

coefs The coefficient vector β used for data generation.

first_inds A vector of indices indicating the first treatment effect for each treated cohort.

N_UNTREATED The number of never-treated units.

assignments A vector of counts (of length $G + 1$) indicating how many units fall into the never-treated group and each of the G treated cohorts.

indep_counts Independent cohort assignments (for auxiliary purposes).

p The number of columns in the design matrix X.

N Number of units.

T Number of time periods.

G Number of treated cohorts.

R Deprecated alias for G, retained for backward compatibility; populated with the same value. Use G. Will be removed in a future release.

d Number of covariates.

sig_eps_sq The idiosyncratic noise variance.

sig_eps_c_sq The unit-level noise variance.

References

Faletto, G (2025). Fused Extended Two-Way Fixed Effects for Difference-in-Differences with Staggered Adoptions. *arXiv preprint arXiv:2312.05985*. https://arxiv.org/abs/2312.05985.

Examples

```
## Not run:
  # Set simulation parameters
  N <- 100           # Number of units in the panel
  T <- 5             # Number of time periods
  G <- 3             # Number of treated cohorts
  d <- 2             # Number of time-invariant covariates
  sig_eps_sq <- 1    # Variance of observation-level noise
  sig_eps_c_sq <- 0.5  # Variance of unit-level random effects

  # Generate coefficient vector using genCoefsCore()
  # (Here, density controls sparsity and eff_size scales nonzero entries)
  coefs_core <- genCoefsCore(G = G, T = T, d = d, density = 0.2, eff_size = 2, seed = 123)

  # Now simulate the data. Setting gen_ints = TRUE generates the full design
  matrix with interactions.
  sim_data <- simulateDataCore(
    N = N,
    T = T,
    G = G,
    d = d,
    sig_eps_sq = sig_eps_sq,
    sig_eps_c_sq = sig_eps_c_sq,
    beta = coefs_core$beta,
    seed = 456,
    gen_ints = TRUE,
    distribution = "gaussian"
  )

  # Examine the returned list:
  str(sim_data)

## End(Not run)
```

`simultaneousCIs` *Parametric simultaneous (1 - alpha) confidence intervals over a family of treatment effects*

Description

Computes simultaneous (family-wise) confidence intervals for a user- specified family of treatment effects from a fitted FETWFE / ETWFE / BETWFE / twfeCovs object. The simultaneous critical value `c_{1 - alpha}` is the `(1 - alpha)` quantile of `max_k |Z_k|` where `Z` follows a multivariate normal with correlation matrix `cov2cor(Sigma)`; it is computed deterministically via `mvtnorm::qmvnorm()`. Under Faletto (2025) Theorem (c') tight Gaussianity and Assumption (Psi-IF), the family of psi-linear effects is asymptotically multivariate normal with a covariance that is estimable from the package's existing variance machinery; under the paper's fixed-dim framing no high-dimensional correction is needed.

The pointwise critical value `qnorm(1 - alpha/2)` (per-effect coverage) and the Bonferroni-conservative critical value `qnorm(1 - alpha/(2K))` (family- wise coverage with no correlation assumption) are returned for side-by-side comparison; the simultaneous critical value is always between them when the effects are positively correlated (as is typical in difference-in- differences, where effects share the regression-coefficient variance piece).

Usage

```
simultaneousCIs(
  result,
  family = c("event_study", "cohort", "all_post_treatment", "custom"),
  alpha = NULL,
  contrasts = NULL,
  method = c("analytic", "bootstrap"),
  B = 1000L,
  seed = NULL,
  multiplier = c("rademacher", "mammen", "webb"),
  lambda_c = 1,
  riesz_max_iter = 5000L,
  riesz_tol = 1e-09,
  cv_time_budget = Inf
)
```

Arguments

`result`	A fitted object of class `"fetwfe"`, `"etwfe"`, `"betwfe"`, or `"twfeCovs"`.
`family`	Character; one of `"event_study"`, `"cohort"`, `"all_post_treatment"`, or `"custom"`. See Details for each family's resolution.
`alpha`	Numeric in `(0, 1)`; significance level. Default `NULL`, which inherits the fit's own `alpha` (the level its `catt_df` / displayed simultaneous bands were built at), so `simultaneousCIs(fit)` reproduces the fit's shown simultaneous bounds. Pass an explicit value to override.
`contrasts`	For `family = "custom"`, a `K x num_treats` matrix whose rows give the `K` linear combinations of the underlying per-`(g, t)` treatment-effect vector (the `multcomp::glht()` convention; `num_treats` is the number of estimated effects, equal to `G` for `twfeCovs`). Ignored for the other families. Note that the `"custom"` family omits the cohort-probability variance term (`Sigma_2 = 0`), so a custom contrast that pools across cohorts in a probability-weighted way is anti-conservative (its band can under-cover); use `family = "cohort"` for cohort-pooled effects.
`method`	Character; how the simultaneous critical value is computed. `"analytic"` (default) uses the exact multivariate-normal sup-t quantile via `mvtnorm::qmvnorm()`. `"bootstrap"` uses the multiplier bootstrap of Chernozhukov, Chetverikov & Kato (2013): it perturbs the per-unit influence functions and reads the sup-t critical value off the bootstrap distribution. The two are asymptotically equivalent; the bootstrap scales better to large effect families and is heteroskedasticity/cluster-robust by construction (it always reports cluster-robust per-unit standard errors,

regardless of the fit's se_type). method = "bootstrap" supports all four families. For family = "event_study" the bootstrap additionally perturbs the per-unit cohort-probability (propensity) influence function with its own independent multiplier stream – a two-channel bootstrap matching the analytic variance Sigma = Sigma_1 + Sigma_2 (the event-study family is the one whose Sigma_2 is non-zero, because each event-time effect pools across cohorts weighted by the estimated cohort probabilities). When the (full) design is **high-dimensional** (p >= NT) – where the analytic Gram inverse need not exist – the bootstrap uses the full-design **desparsified** construction of debiasedATT() (per-effect nodewise directions) generalized to the family. This desparsified p >= NT band path (fetwfe() fits only) generalizes the debiasedATT() construction — whose overall-ATT coverage is validated near-nominally in simulation (Theorem debiased.highdim.thm, Faletto 2025) — to a family of functionals. Its family-wise coverage is likewise validated near-nominally (Theorem debiased.highdim.joint.thm): ~0.92 over the **event-study** family (K = 7) at the p >= NT anchor of Faletto (2025), the band sibling of the scalar result (~0.94); the cohort and all-post-treatment families are covered by the same theorem but were not separately simulated. As with the scalar, validity rests on the sparsity / restricted-eigenvalue primitives (assumed, not proved), so inspect the returned feasibility / converged diagnostics. The desparsified path covers all four families (a high-dimensional family = "event_study" fit additionally carries the propensity channel F_pi). In the high-dimensional regime the band is centered on the **debiased** estimate (the high-dimensional FETWFE theory correction, equal to debiasedATT()'s point estimate for the matching contrast), not the post-selection bridge estimate; fixed-p bands center on the (unbiased) bridge estimate as before. Because this desparsified bootstrap band is built from the empirical per-unit influence function – needing neither GLS whitening nor sig_eps_sq – it also accepts a high-dimensional gls = FALSE fetwfe fit (calc_ses = FALSE), the band analog of debiasedATT()'s Omega-free SE (#307, #313). method = "analytic" still requires valid analytic SEs (a q < 1, gls = TRUE, rank-satisfied fit). **Which method is valid at p >= NT is a property of the function rather than of this argument alone; see the "High-dimensional (p >= NT) bands" paragraph of Details, which also documents the warning such a call emits.**

B — Integer; number of multiplier-bootstrap replicates (method = "bootstrap" only). Must be a positive integer no greater than .Machine$integer.max. Default 1000.

seed — Optional integer within +/- .Machine$integer.max. If supplied, the bootstrap draws are reproducible (the ambient random-number stream is saved and restored around them). If NULL (default), the draws come from the ambient generator, so results vary run to run. Ignored when method = "analytic".

multiplier — Character; the multiplier-weight distribution for the bootstrap: "rademacher" (default, +/-1), "mammen" (the Mammen 1993 two-point distribution), or "webb" (the Webb 2013 six-point distribution, preferred with very few clusters). Ignored when method = "analytic".

lambda_c, riesz_max_iter, riesz_tol — Controls for the high-dimensional (p >= NT) bootstrap, where each effect's debiasing direction is a nodewise (desparsified-lasso) riesz_lasso() solve. lambda_c is the leading constant of the penalty lambda_node = lambda_c * max(|a|) * sqrt(log p / N):

either a single positive number (the default `1.0` = theory scale) or the string `"cv"`, which selects the constant **per fit** by cross-validation (the same CV `debiasedATT()` uses, on the overall-ATT direction, so one constant serves the point estimate and every band effect; #295). **Smaller fixed values can leave directions infeasible (a warning fires and those bands are unreliable).** Ignored when `p < NT` or `method = "analytic"`. The default stays the fixed `1.0`; the family-wise coverage validated in Faletto (2025) used the `"cv"` selector (`lambda_c` ~ 0.54, feasibility 1.0), the same constant that serves the scalar `debiasedATT()` and every band effect.

`cv_time_budget` Numeric; a wall-clock backstop (in seconds) for the `lambda_c = "cv"` cross-validation (the same #384 backstop `debiasedATT()` offers). `Inf` (the default) leaves selection fully deterministic / reproducible; a finite value stops the CV early and falls back to the theory scale (`lambda_c = 1.0`) with a warning, so an adversarial high-dimensional draw cannot spin indefinitely (#384). Whether a finite budget fires depends on machine speed, so the result can too. Ignored unless `lambda_c = "cv"` and `p >= NT`.

Details

Family resolution and `K`. `"event_study"` resolves to one effect per post-treatment event time `e = 0, ..., T - 2` (`K = T - 1`); `"cohort"` to one effect per treated cohort (`K = G`); `"all_post_treatment"` to one effect per `(g, t)` cell (`K = num_treats`); `"custom"` to the `K = nrow(contrasts)` user-supplied contrasts.

Joint covariance. The `K x K` covariance `Sigma = Sigma_1 + Sigma_2` is reconstructed at call time from the fit's stored slots (design matrix, selected support, `theta_hat` / `beta_hat`, `cohort_probs_overall`, `sig_eps_sq`). `Sigma_1` is the regression-coefficient piece and `Sigma_2` the cohort-probability piece, generalizing the package's per-point variance machinery (the same machinery `eventStudy()` uses). By construction `sqrt(diag(Sigma))` equals the package's existing per-point standard errors for the corresponding effects. The `Sigma` blocks are not persisted on the fit; re-derivation is sub-second.

Degenerate effects. An effect whose entire contribution to the selected support is zeroed by the bridge penalty – or, in scattered-cohort panels, an event time with an empty valid-cohort set – has a standard error of exactly 0 by construction, so its simultaneous and pointwise CIs collapse to a point at the estimate and it is excluded from the joint correlation matrix (it adds no family-wise risk; the critical value is computed over the non-degenerate sub-family). This `se = 0` convention is the simultaneous-CI analog of the `NA` standard error `eventStudy()` reports for the same structurally-degenerate event times; both assign the effect an estimate of 0. On every band built from the selected support, which is every band but the desparsified one a high-dimensional `fetwfe()` fit gets under `method = "bootstrap"`, a signed `"custom"` contrast whose weights cancel on that support is degenerate too: its estimate and standard error are exactly 0 or rounding error, so which side of zero its point-sized interval falls on carries no information, and its `NA` adjusted p-value is the signal.

High-dimensional (`p >= NT`) bands. One route out of the four is uniformly valid: a `fetwfe()` fit under `method = "bootstrap"`. That is the full-design desparsified band of Theorem `debiased.highdim.joint.thm` (Faletto 2025), whose family-wise coverage is validated near-nominally in simulation.

Every other `p >= NT` combination returns the fixed-p **post-selection** band on the selected support instead, and that band is not valid there: a #308 coverage study measures it at roughly 0.09 against 0.95 nominal, **even when the selected support is low-dimensional**, because the bridge shrinks the band center toward zero and the post-selection standard error understates variability. Sparsity does

not rescue it; only debiasing does. Concretely this covers a non-`fetwfe()` fit (e.g. `betwfe()`) on **either** method – the desparsified construction is `fetwfe()`-only, so no valid high-dimensional band exists for those estimators – and a `fetwfe()` fit on the `method = "analytic"` **default**, where the remedy is simply to pass `method = "bootstrap"`.

The band is still returned rather than replaced by an error, but every such call **emits a warning** naming the applicable remedy. The warning is a classed condition, `"fetwfe_highdim_postselection_band"`, so it can be caught or muffled precisely:

```
withCallingHandlers(
  simultaneousCIs(fit),
  fetwfe_highdim_postselection_band = function(w) invokeRestart("muffleWarning")
)
```

`eventStudy()` signals the same condition, for the same reason. It does **not** fire at fit time (a `ci_type = "simultaneous"` fit is silent), nor inside `print()`, `summary()`, or `plot()`, all three of which muffle it; of those, `print()` and `summary()` render a two-line caveat under their CATT preview instead, and `plot()` renders none. It also does not fire on the `p >= NT` all-zero-support path, which returns a degenerate band and carries its own (#304) condition already.

A `gls = FALSE` (`calc_ses = FALSE`) high-dimensional `fetwfe()` fit has no analytic standard errors at all, so `method = "analytic"` errors there rather than warning; `method = "bootstrap"` still works, because the desparsified band is built from the empirical per-unit influence function.

Paper grounding. Theorem (c') tight Gaussianity (Faletto 2025, `paper_arxiv.tex:1233`) guarantees the joint asymptotic normality; Assumption (Psi-IF) (assumption equation `paper_arxiv.tex:2013`; in-prose discussion at paper line 1268) is the influence-function condition the package's default cohort-sample-proportions estimator satisfies; the fixed-dim framing follows the paper's AE point 1(d).

Conservative fallback. When the fit was made with `se_type = "conservative"`, the function falls back to Bonferroni-corrected pointwise CIs (the Cauchy-Schwarz upper bound used for the conservative scalar SE does not generalize to a `K x K` covariance matrix) and emits a brief `message()`. The `$critical_value` field is set to the Bonferroni value in this branch.

Numerical integration. The critical value is computed via `mvtnorm::qmvnorm(..., algorithm = mvtnorm::GenzBretz())` (mvtnorm's default quasi-Monte Carlo integrator; sub-second through `K` up to about 100, so no `K` cap is of practical concern for FETWFE families). `mvtnorm` is an `Imports` dependency (as of version 1.16.0, when simultaneous bands became the default reported confidence interval; see the `ci_type` argument of `fetwfe()`). The function uses it only when `K > 1` and `se_type != "conservative"` (the `K = 1` and conservative paths bypass the dependency), and retains a defensive `stop()` with an actionable message if it is somehow unavailable (e.g., a corrupted install).

Determinism contract. The function is deterministic in its inputs: the same fit plus the same `family`, `alpha`, and `contrasts` always produces the same critical value across calls. This is achieved by wrapping the internal `mvtnorm::qmvnorm()` call with a save/restore of the caller's `.Random.seed` and a fixed internal `set.seed(1L)` immediately before the call. The function does NOT mutate the caller's `.Random.seed` (the save/restore via `on.exit()` leaves the caller's RNG state identical pre- and post-call), matching the convention adopted by `R/fetwfe_core.R::getBetaCV()` in PR #181 / v1.13.5. Users do not need to call `set.seed()` before `simultaneousCIs()` to get reproducible results, and downstream RNG-using code observes no perturbation.

Value

An object of S3 class "simultaneous_cis": a list with

ci A data frame with columns effect, estimate, simultaneous_ci_low, simultaneous_ci_high, pointwise_ci_low, pointwise_ci_high (one row per effect in the family).

adjusted_p_values Numeric vector of length K: the single-step max-T multiplicity-adjusted (family-wise) p-value for each effect, the exact dual of the simultaneous band (a coefficient lies outside the (1 - alpha) band iff its adjusted p-value is < alpha). Computed via mvtnorm::pmvnorm() over the same correlation matrix the band uses (or, under se_type = "conservative", the Bonferroni adjustment min(1, K * pointwise_p)). NA for degenerate effects. (#200)

critical_value The simultaneous critical value c_{1 - alpha} (or, when the fit used se_type = "conservative", the Bonferroni critical value qnorm(1 - alpha/(2K)) – see Details).

pointwise_critical_value qnorm(1 - alpha/2), for reference.

bonferroni_critical_value qnorm(1 - alpha/(2K)), for reference.

family The requested family (character).

alpha The significance level used.

K The number of effects in the family (integer).

For method = "bootstrap" the list additionally carries method, B, seed, multiplier, and regime ("fixed-p" or "high-dimensional"); the critical_value is the bootstrap sup-t quantile and the standard errors backing ci are the cluster-robust per-unit ones (so for a non-cluster fit the bootstrap band may differ from the analytic homoskedastic band). A "high-dimensional" fit additionally carries per-effect feasibility, converged, and lambda_node (the nodewise-direction diagnostics), plus lambda_c (the leading constant used) and lambda_c_selection ("fixed" or "cv"). When lambda_c = "cv" it also carries lambda_cv, the cross-validation diagnostics (the grid, the per-grid feasibility flags and CV losses, and whether the theory-scale fallback fired), matching debiasedATT()'s lambda_cv.

References

Faletto, G. (2025). Fused Extended Two-Way Fixed Effects for Difference-in- Differences with Staggered Adoptions. arXiv:2312.05985.

Hothorn, T., Bretz, F., & Westfall, P. (2008). Simultaneous Inference in General Parametric Models. *Biometrical Journal* 50(3), 346-363.

See Also

eventStudy() for the per-point event-study estimates and Wald intervals that family = "event_study" provides simultaneous bands over.

Examples

```
coefs <- genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2)
sim <- simulateData(coefs, N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
fit <- fetwfeWithSimulatedData(sim)
sci <- simultaneousCIs(fit, family = "event_study", alpha = 0.05)
print(sci)
```

`tidy.betwfe` *Tidy a* `betwfe` *fitted object*

Description

Like `tidy.fetwfe()` but for a BETWFE fit. Includes the `selected` column reflecting BETWFE's bridge-penalized selection.

Usage

```
## S3 method for class 'betwfe'
tidy(x, conf.int = TRUE, conf.level = 1 - x$alpha, ...)
```

Arguments

`x`	An object of class `"betwfe"` returned by `betwfe()`.
`conf.int`	Logical; include CI columns.
`conf.level`	Numeric in (0, 1); defaults to `1 - x$alpha`. Applies only to the overall-ATT row; the cohort rows pass through the fit-time `catt_df` bounds (reflecting the fit's `ci_type`) and are not recomputed at `conf.level` (#197). See `tidy.fetwfe()`.
`...`	Unused.

Value

A data frame with `G + 1` rows.

Examples

```
## Not run:
  res <- betwfeWithSimulatedData(
    simulateData(genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  )
  broom::tidy(res)

## End(Not run)
```

`tidy.cohortStudy` *Tidy a* `cohortStudy` *object*

Description

Returns a broom-style tidy data frame for the output of `cohortStudy()`. Renames the snake_case columns of `catt_df` to broom conventions (`se` -> `std.error`, `p_value` -> `p.value`, `ci_low` / `ci_high` -> `conf.low` / `conf.high`) and adds a `term` column (`"cohort_<cohort label>"`) plus a `statistic` column (`estimate` / `std.error`) so the schema matches `tidy.eventStudy()` for downstream `bind_rows()` consumers. When the input carries a `selected` column (`fetwfe` / `betwfe`), it is passed through as the final column.

Usage

```
## S3 method for class 'cohortStudy'
tidy(x, ...)
```

Arguments

`x`	A `cohortStudy` object returned by `cohortStudy()`.
`...`	Unused; present for S3 compatibility.

Details

Confidence intervals come from the cohort fit's stored bounds (which encode the alpha passed at fit time); unlike `tidy.eventStudy()`, this method does not recompute the CIs at a custom `conf.level` because the standard errors in `catt_df` are already paired with the fit-time bounds (`ci_low` / `ci_high`), so re-emitting those is the minimum-surprise behavior.

Value

A data frame with one row per treated cohort and columns `term`, `estimate`, `std.error`, `statistic`, `p.value`, `conf.low`, `conf.high`, and (if present in the input) `selected`.

Examples

```
## Not run:
  res <- fetwfeWithSimulatedData(
    simulateData(genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  )
  broom::tidy(cohortStudy(res))

## End(Not run)
```

`tidy.cohortTimeATTs` *Tidy a* `cohortTimeATTs` *object*

Description

Returns a broom-style tidy data frame for the output of `cohortTimeATTs()`. Renames the snake_case columns to broom conventions (`se` -> `std.error`, `p_value` -> `p.value`, `ci_low` / `ci_high` -> `conf.low` / `conf.high`), keeps the `time` column, and adds a `term` column (`"cohort_<cohort label>_time_<time>"`) plus a `statistic` column (`estimate` / `std.error`) so the schema parallels `tidy.cohortStudy()` and `tidy.eventStudy()` for downstream `bind_rows()` consumers. When the input carries a `selected` column (`fetwfe` / `betwfe`), it is passed through as the final column.

Usage

```
## S3 method for class 'cohortTimeATTs'
tidy(x, ...)
```

Arguments

`x`	A `cohortTimeATTs` object returned by `cohortTimeATTs()`.
`...`	Unused; present for S3 compatibility.

Details

Confidence intervals are the pointwise `1 - alpha` Wald bounds `cohortTimeATTs()` computed (encoding the `alpha` passed there); like `tidy.cohortStudy()` this method passes them through rather than recomputing at a custom `conf.level`.

Value

A data frame with one row per `(cohort, time)` cell and columns `term`, `time`, `estimate`, `std.error`, `statistic`, `p.value`, `conf.low`, `conf.high`, and (if present in the input) `selected`.

Examples

```
## Not run:
  res <- fetwfeWithSimulatedData(
    simulateData(genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  )
  broom::tidy(cohortTimeATTs(res))

## End(Not run)
```

tidy.debiased_att *Tidy a debiased overall-ATT estimate*

Description

One-row `broom::tidy()` data frame for a `debiasedATT()` result, following the package's `broom` column convention (`term`, `estimate`, `std.error`, `conf.low`, `conf.high`) – the same columns `tidy.fetwfe()` / `tidy.eventStudy()` return.

Usage

```
## S3 method for class 'debiased_att'
tidy(x, ...)
```

Arguments

`x`	A `"debiased_att"` object from `debiasedATT()`.
`...`	Ignored.

Value

A one-row data frame with columns `term`, `estimate`, `std.error`, `conf.low`, `conf.high`.

tidy.etwfe *Tidy an* `etwfe` *fitted object*

Description

Like `tidy.fetwfe()` but for an ETWFE fit. Has no `selected` column (ETWFE does no regularized selection).

Usage

```
## S3 method for class 'etwfe'
tidy(x, conf.int = TRUE, conf.level = 1 - x$alpha, ...)
```

Arguments

`x`	An object of class `"etwfe"` returned by `etwfe()`.
`conf.int`	Logical; include CI columns.
`conf.level`	Numeric in (0, 1); defaults to `1 - x$alpha`. Applies only to the overall-ATT row; the cohort rows pass through the fit-time `catt_df` bounds (reflecting the fit's `ci_type`) and are not recomputed at `conf.level` (#197). See `tidy.fetwfe()`.
`...`	Unused.

Value

A data frame with G + 1 rows.

Examples

```
## Not run:
  res <- etwfeWithSimulatedData(
    simulateData(genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  )
  broom::tidy(res)

## End(Not run)
```

---

tidy.eventStudy *Tidy an* eventStudy *object*

Description

Returns a broom-style tidy data frame for the output of eventStudy(). Renames existing columns to broom conventions (se → std.error, p_value → p.value) and adds a term column ("e<event_time>") plus a statistic column (estimate / std.error) so the schema matches tidy.<estimator>() for downstream bind_rows() consumers.

Usage

```
## S3 method for class 'eventStudy'
tidy(x, conf.int = TRUE, conf.level = 0.95, ...)
```

Arguments

x	An object of class "eventStudy" returned by eventStudy().
conf.int	Logical; include conf.low / conf.high columns.
conf.level	Numeric in (0, 1). Retained for broom-convention parity (default 0.95) but no longer recomputes the event-study CIs: as of version 1.16.0 (#197) the conf.low / conf.high columns pass through the eventStudy object's stored ci_low / ci_high (reflecting the fit's ci_type). To change the confidence level, recompute eventStudy(fit, alpha = ...) at the desired alpha first.
...	Unused.

Details

The eventStudy() output stores its confidence-interval bounds (ci_low / ci_high), which reflect the fit's ci_type (#197): simultaneous (family-wise, uniform) by default, or pointwise when the fit used ci_type = "pointwise". When conf.int = TRUE (the default), conf.low / conf.high PASS THROUGH those stored bounds rather than recomputing from estimate +/- z * se — so the tidied event-study CIs agree with print / summary / plot and with simultaneousCIs() under the default. When conf.int = FALSE, the CI columns are omitted. (Degenerate event times carry NA bounds under both ci_type settings.)

Value

A data frame with one row per event-time and columns `term`, `event_time`, `n_cohorts`, `estimate`, `std.error`, `statistic`, `p.value`, and (when `conf.int = TRUE`) `conf.low` / `conf.high`.

Examples

```
## Not run:
  res <- fetwfeWithSimulatedData(
    simulateData(genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  )
  broom::tidy(eventStudy(res))

## End(Not run)
```

`tidy.fetwfe` *Tidy an* `fetwfe` *fitted object*

Description

Returns a broom-style tidy data frame for an object of class `"fetwfe"`. Row 1 is the overall ATT (`term = "ATT"`); subsequent rows are the cohort-specific ATTs (`term = "Cohort <adoption-time>"`), one per treated cohort, sorted by ascending cohort label. Standard error, z-statistic, and p-value reflect the value of `se_type` used at fit time (model-based by default, cluster-robust under `se_type = "cluster"`). Cohorts that the bridge penalty zeroed out (`selected = FALSE`) carry `NA` for `std.error` / `statistic` / `p.value`.

Usage

```
## S3 method for class 'fetwfe'
tidy(x, conf.int = TRUE, conf.level = 1 - x$alpha, ...)
```

Arguments

`x`	An object of class `"fetwfe"` returned by `fetwfe()`.
`conf.int`	Logical. If `TRUE` (default), `conf.low` and `conf.high` columns are included.
`conf.level`	Numeric in (0, 1). Applies only to the overall-ATT row (row 1), whose CI is recomputed at this level; defaults to `1 - x$alpha` (faithful to the alpha used at fit time; deviates from `broom::tidy.lm`'s `0.95` default by design). The cohort rows pass through the fit-time `catt_df` bounds (reflecting the fit's `ci_type` and the fit-time alpha) and are NOT recomputed at `conf.level` (#197) — mirroring `tidy.cohortStudy()`'s stored-bounds behavior.
`...`	Unused; present for S3 compatibility.

Details

The cohort-row `conf.low` / `conf.high` columns pass through the fit's stored `catt_df` bounds, so they reflect the fit's `ci_type` (#197): simultaneous (family-wise) by default, or pointwise when the fit used `ci_type = "pointwise"`. They are NOT recomputed from `conf.level` (see the `conf.level` note). The overall-ATT row (row 1) is a scalar, so its CI is the pointwise Wald interval at `conf.level` (pointwise == simultaneous for a single effect).

Value

A data frame with `G + 1` rows and columns `term`, `estimate`, `std.error`, `statistic`, `p.value`, optionally `conf.low` / `conf.high`, and `selected` (logical).

Examples

```
## Not run:
  res <- fetwfeWithSimulatedData(
    simulateData(genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  )
  broom::tidy(res)

## End(Not run)
```

`tidy.FETWFE_tes` *Tidy a* `FETWFE_tes` *simulation truth object*

Description

Returns a `broom`-style tidy data frame for the population-truth object returned by `getTes()`. Row 1 is the overall true ATT (`term = "ATT_true"`); subsequent rows are the true cohort ATTs (`term = "Cohort <adoption-time>"`, using the simulator's convention that cohort `g` adopts at calendar time `g + 1`, so the labels match what `tidy.<estimator>` uses on a fitted panel generated from the same `FETWFE_coefs`). Standard error / statistic / p-value columns are always `NA_real_` — there is no sampling distribution for a population truth. When `conf.int = TRUE` (default, matching the sibling tidy methods), `conf.low` / `conf.high` columns are included and also set to `NA_real_`. When `conf.int = FALSE`, those columns are omitted.

Usage

```
## S3 method for class 'FETWFE_tes'
tidy(x, conf.int = TRUE, conf.level = 0.95, ...)
```

Arguments

`x`	An object of class `"FETWFE_tes"` returned by `getTes()`.

conf.int	Logical; include conf.low / conf.high columns. Defaults to TRUE to match the sibling tidy methods and preserve pre-#84 backward compatibility. Population-truth objects have no sampling distribution, so the CI columns are always filled with NA_real_ when included.
conf.level	Numeric in (0, 1). Accepted for broom-convention parity but unused (no CIs to compute for a population truth); validated regardless. Defaults to 0.95 (FETWFE_tes objects do not carry an alpha slot, so there is no fitted-object value to default to).
...	Unused.

Value

A data frame with G + 1 rows and columns term, estimate, std.error, statistic, p.value, and (when conf.int = TRUE) conf.low / conf.high.

Examples

```
## Not run:
  coefs <- genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2)
  broom::tidy(getTes(coefs))

## End(Not run)
```

---

tidy.simultaneous_cis *Tidy a simultaneous-confidence-interval object*

Description

One row per effect for a simultaneousCIs() result, in broom form: the simultaneous band as conf.low / conf.high, plus the pointwise band as pointwise.conf.low / pointwise.conf.high.

Usage

```
## S3 method for class 'simultaneous_cis'
tidy(x, ...)
```

Arguments

x	A "simultaneous_cis" object from simultaneousCIs().
...	Ignored.

Value

A data frame with one row per effect and columns term, estimate, conf.low, conf.high, pointwise.conf.low, and pointwise.conf.high.

`tidy.twfeCovs` *Tidy a* `twfeCovs` *fitted object*

Description

Like `tidy.etwfe()` but for a TWFE-with-covariates fit. Has no `selected` column (`twfeCovs` is pure OLS and does no regularized selection). `twfeCovs` estimates one pooled effect per cohort, so the returned frame has the same `G + 1` rows (overall ATT in row 1, then one row per cohort) as the sibling estimators.

Usage

```
## S3 method for class 'twfeCovs'
tidy(x, conf.int = TRUE, conf.level = 1 - x$alpha, ...)
```

Arguments

`x`	An object of class `"twfeCovs"` returned by `twfeCovs()`.
`conf.int`	Logical; include CI columns.
`conf.level`	Numeric in (0, 1); defaults to `1 - x$alpha`. Applies only to the overall-ATT row; the cohort rows pass through the fit-time `catt_df` bounds (reflecting the fit's `ci_type`) and are not recomputed at `conf.level` (#197). See `tidy.fetwfe()`.
`...`	Unused.

Value

A data frame with `G + 1` rows.

Examples

```
## Not run:
  res <- twfeCovsWithSimulatedData(
    simulateData(genCoefs(G = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5, seed = 123)
  )
  broom::tidy(res)

## End(Not run)
```

twfeCovs *Two-way fixed effects with covariates and a single pooled treatment effect per cohort*

Description

WARNING: This function should NOT be used for estimation. It is a biased estimator of treatment effects. Implementation of two-way fixed effects with covariates and a single pooled treatment effect per cohort. Estimates overall ATT as well as CATT (cohort average treatment effects on the treated units). It is implemented only for the sake of the simulation studies in Faletto (2025). This estimator is only unbiased under the assumptions that treatment effects are homogeneous across covariates and are identical within cohorts across all times since treatment.

Usage

```
twfeCovs(
  pdata,
  time_var,
  unit_var,
  treatment,
  response,
  covs = c(),
  indep_counts = NA,
  sig_eps_sq = NA,
  sig_eps_c_sq = NA,
  verbose = FALSE,
  alpha = 0.05,
  add_ridge = FALSE,
  allow_no_never_treated = TRUE,
  se_type = "default",
  ci_type = c("simultaneous", "pointwise")
)
```

Arguments

pdata	Dataframe; the panel data set. Each row should represent an observation of a unit at a time. Should contain columns as described below.
time_var	Character; the name of a single column containing a variable for the time period. This column is expected to contain integer values (for example, years). Recommended encodings for dates include format YYYY, YYYYMM, or YYYYMMDD, whichever is appropriate for your data.
unit_var	Character; the name of a single column containing a variable for each unit. This column is expected to contain character values (i.e. the "name" of each unit).
treatment	Character; the name of a single column containing a variable for the treatment dummy indicator. This column is expected to contain integer values, and in particular, should equal 0 if the unit was untreated at that time and 1 otherwise.

Treatment should be an absorbing state; that is, if unit `i` is treated at time `t`, then it must also be treated at all times `t` + 1, ..., `T`. Any units treated in the first time period will be removed automatically. Please make sure yourself that at least some units remain untreated at the final time period ("never-treated units").

`response` Character; the name of a single column containing the response for each unit at each time. The response must be an integer or numeric value.

`covs` (Optional.) Either a character vector containing the names of the columns for covariates (e.g., `covs = c("x1", "x2")`), or a one-sided formula (e.g., `covs = ~ x1 + x2`) – the formula form mirrors the convention used by `did::att_gt(xformla = ...)`. Only additive bare variable names are supported in the formula form; for derived variables, compute them in the data frame first and pass via the character-vector form. All of these columns are expected to contain integer, numeric, or factor values, and any categorical values will be automatically encoded as binary indicators. If no covariates are provided, the treatment effect estimation will proceed, but it will only be valid under unconditional versions of the parallel trends and no anticipation assumptions. Default is c().

`indep_counts` (Optional.) Integer; a vector. If you have a sufficiently large number of units, you can optionally randomly split your data set in half (with `N` units in each data set). The data for half of the units should go in the `pdata` argument provided above. For the other `N` units, simply provide the counts for how many units appear in the untreated cohort plus each of the other `G` cohorts in this argument `indep_counts`. The benefit of doing this is that the standard error for the average treatment effect will be (asymptotically) exact instead of conservative. The length of `indep_counts` must equal 1 plus the number of treated cohorts in `pdata`. All entries of `indep_counts` must be strictly positive (if you are concerned that this might not work out, maybe your data set is on the small side and it's best to just leave your full data set in `pdata`). The sum of all the counts in `indep_counts` must match the total number of units in `pdata`. Default is NA (in which case the conservative standard error formula will be used).

`sig_eps_sq` (Optional.) Numeric; the variance of the row-level IID noise assumed to apply to each observation. See Section 2 of Faletto (2025) for details. It is best to provide this variance if it is known (for example, if you are using simulated data). If this variance is unknown, this argument can be omitted, and the variance will be estimated by REML on the linear mixed-effects model `y ~ X + (1 | unit)` via `lme4::lmer` (Bates et al. 2015; Patterson & Thompson 1971). Default is NA.

`sig_eps_c_sq` (Optional.) Numeric; the variance of the unit-level IID noise (random effects) assumed to apply to each observation. See Section 2 of Faletto (2025) for details. It is best to provide this variance if it is known (for example, if you are using simulated data). If this variance is unknown, this argument can be omitted, and the variance will be estimated by REML via `lme4::lmer` on the linear mixed-effects model `y ~ X + (1 | unit)` (Bates et al. 2015; Patterson & Thompson 1971). Default is NA.

`verbose` Logical; if TRUE, more details on the progress of the function will be printed as the function executes. Default is FALSE.

`alpha` Numeric; function will calculate (1 - `alpha`) confidence intervals for the cohort average treatment effects that will be returned in `catt_df`.

add_ridge (Optional.) Logical; if TRUE, adds a small amount of ridge regularization to the (untransformed) coefficients to stabilize estimation. Default is FALSE.

allow_no_never_treated (Optional.) Logical; if TRUE (default) and the input panel contains no never-treated units, the panel is auto-truncated by dropping time periods at and after the latest cohort's start time — the units in that latest cohort then serve as the never-treated comparison group in the retained sub-panel — with a warning naming the dropped periods. If FALSE, the estimator stops with an error in this case (the package's behavior prior to version 1.5.6). The argument has no effect when the input already contains never-treated units. Default is TRUE.

se_type Character; one of "default", "conservative", or "cluster". "default" returns the tight Gaussian variance sqrt(att_var_1 + att_var_2) from Theorem (c$'$) under Assumption (Psi-IF) (asymptotically exact for the package's default cohort sample-proportions estimator); "conservative" returns the Cauchy-Schwarz upper bound from Theorem (c) (use only when the propensity-score estimator violates (Psi-IF)); "cluster" is an *experimental* unit-clustered Liang-Zeger sandwich SE on the OLS-selected support (see the companion vignette inference_vignette for the formula, the assumptions, and the theory-pending caveat). Default is "default". v1.12.0 introduced the tight Gaussian default; versions <= 1.11.7 used the conservative Cauchy-Schwarz formula as the default.

ci_type Character; one of "simultaneous" (default) or "pointwise". Controls the confidence-interval bounds reported for the cohort-specific ATTs (in catt_df). "simultaneous" reports parametric simultaneous (family-wise, uniform) bands computed via simultaneousCIs(): the band covers all cohort effects jointly with probability 1 - alpha, matching the default presentation of did::aggte(cband = TRUE). "pointwise" reports per-effect Wald intervals (each covers its own effect with probability 1 - alpha, no joint guarantee — the behavior of versions <= 1.15.1). Both the interval bounds and the per-cohort p-values (p_value) follow ci_type (single-step max-T multiplicity-adjusted under "simultaneous", per-cohort Wald under "pointwise"; #200); the standard errors (se) are identical under both settings. twfeCovs estimates a single pooled effect per cohort, so only the cohort family is affected (it has no event-study surface). When standard errors are unavailable (e.g., a rank-deficient design) the bounds are NA under both settings. Default is "simultaneous".

Value

An object of class twfeCovs containing the following elements:

att_hat The estimated overall average treatment effect for a randomly selected treated unit.

att_se A standard error for the ATT. If the Gram matrix is not invertible, this will be NA.

att_p_value A two-sided p-value for the overall ATT against the null H_0: tau = 0, computed as 2 * pnorm(-|att_hat / att_se|). NA if att_se is zero or NA. Standard post-OLS interpretation; twfeCovs does not perform selection.

catt_hats	A named vector containing the estimated average treatment effects for each cohort.
catt_ses	A named vector containing the (asymptotically exact) standard errors for the estimated average treatment effects within each cohort.
cohort_probs	A vector of the estimated probabilities of being in each cohort conditional on being treated, which was used in calculating att_hat. If indep_counts was provided, cohort_probs was calculated from that; otherwise, it was calculated from the counts of units in each treated cohort in pdata.
catt_df	A data frame (with S3 class c("catt_df", "data.frame")) displaying the cohort names (cohort), average treatment effects (estimate), standard errors (se), 1 - alpha confidence interval bounds (ci_low, ci_high), and per-cohort p-values (p_value). No selected column; twfeCovs does not perform selection. The catt_df S3 class makes [[/ $ / [access on the pre-1.11.0 Title-Case column names (Cohort, Estimated TE, SE, ConfIntLow, ConfIntHigh, P_value) stop() with a migration message pointing to the new name. See NEWS.md for the rename table.
beta_hat	The full vector of estimated coefficients.
treat_inds	The indices of beta_hat corresponding to the treatment effects for each cohort.
treat_int_inds	The indices of beta_hat corresponding to the interactions between the treatment effects for each cohort and the covariates.
sig_eps_sq	Either the provided sig_eps_sq or the estimated one, if a value wasn't provided.
sig_eps_c_sq	Either the provided sig_eps_c_sq or the estimated one, if a value wasn't provided.
X_ints	The design matrix created containing all interactions, time and cohort dummies, etc.
y	The vector of responses, containing nrow(X_ints) entries.
X_final	The design matrix after applying the change in coordinates to fit the model and also multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.
y_final	The final response after multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.
N	The final number of units that were in the data set used for estimation (after any units may have been removed because they were treated in the first time period).
T	The number of time periods in the final data set.
G	The final number of treated cohorts that appear in the final data set.
R	Deprecated alias for G, retained for backward compatibility; populated with the same value. Use G. Will be removed in a future release.
d	The final number of covariates that appear in the final data set (after any covariates may have been removed because they contained missing values or all contained the same value for every unit).
p	The final number of columns in the full set of covariates used to estimate the model.

y_mean
: Numeric scalar; mean of the original (pre-centering) response. Stored so downstream methods (augment(), predict()) can return fitted values on the original-response scale.

response_col_name
: Character scalar; the response column name in the original pdata. Reserved for future augment() / predict() methods.

time_var, unit_var, treatment
: Character scalars; the corresponding arguments the user passed.

covs
: Character vector; the original covs argument (pre-factor- expansion).

calc_ses
: Logical indicating whether standard errors were calculated.

cohort_probs_overall
: A vector of the estimated cohort probabilities on the overall sample (treated and untreated), used in computing the variance of the overall ATT.

indep_counts_used
: Logical scalar; TRUE if a valid indep_counts argument was provided and used for asymptotically-exact ATT inference, FALSE otherwise.

se_type
: Character scalar; the se_type argument the user passed ("default", "conservative", or "cluster").

alpha
: The alpha level used for confidence intervals.

ci_type
: Character scalar; the ci_type argument the user passed ("simultaneous" or "pointwise"), controlling whether the reported catt_df confidence-interval bounds are simultaneous (family-wise) or pointwise.

catt_band_applied
: Logical scalar; TRUE when the fit-time cohort-family simultaneous band was computed and written into catt_df, and FALSE otherwise — including when ci_type = "pointwise", when standard errors were unavailable, and when the band construction degraded. It records only whether the band was applied: it does not imply the band is wider than the pointwise interval (they coincide when fewer than two effects have positive variance) and does not imply the band is informative (a degenerate fit can carry an all-zero band with TRUE).

internal
: A list containing internal outputs that are typically not needed for interpretation, packaged here for parity with fetwfe() so downstream consumers can use a single canonical access path across all four estimator classes (#144). The first five sub-slots (X_ints, y, X_final, y_final, calc_ses) are also duplicated at top level for backward compat; variance_components and first_year live only under $internal:

  X_ints The design matrix containing all interactions, time and cohort dummies, etc. Same value as top-level X_ints.

  y The vector of responses. Same as top-level y.

  X_final The design matrix after the change-of-coordinates step. Same as top-level X_final.

  y_final The transformed response vector. Same as top-level y_final.

  calc_ses Logical indicating whether standard errors were calculated. Same as top-level calc_ses.

variance_components A list exposing the two variance pieces (`att_var_1`, `att_var_2`) plus paper-notation counterparts (`V_1`, `V_2`) and unit-scaled variance estimators (`tilde_v_N`, `hat_v_N`, `tilde_v_N_C`, `tilde_v_N_C_pi_hat`, `tilde_v_N_C_pi_hat_cons`, `tilde_v_N_cons`). The Wald CI is `[hat_T_N +- qnorm(1-alpha/2)` (paper Eq. `conf.int.form`). New in v1.12.0 (issue #141 + #146).

first_year Integer or numeric scalar; the first (earliest) `time_var` value in the panel after `idCohorts()` processing. Consumed by `eventStudy()` to map `cohort_probs`' cohort labels (treatment-start years) to 1-based panel-time-index offsets when the labels are integer-coercible. New in v1.13.3 (issue #174).

Author(s)

Gregory Faletto

References

Faletto, G (2025). Fused Extended Two-Way Fixed Effects for Difference-in-Differences with Staggered Adoptions. *arXiv preprint arXiv:2312.05985*. https://arxiv.org/abs/2312.05985.

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2015). Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software*, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Patterson, H. D., & Thompson, R. (1971). Recovery of inter-block information when block sizes are unequal. *Biometrika*, 58(3), 545-554.

Pinheiro, J. C., & Bates, D. M. (2000). *Mixed-Effects Models in S and S-PLUS*. Springer.

Examples

```
## Not run:
library(bacondecomp)

data(castle)

# Response: the log homicide rate. Treatment: `cdl` records the share of
# the year the castle-doctrine law was in effect, so `cdl > 0` gives the
# absorbing 0/1 treatment indicator.
castle$l_homicide <- log(castle$homicide)
castle$treated <- as.integer(castle$cdl > 0)

# No `covs` here: twfeCovs is pure OLS (no bridge penalty), and castle's
# smallest adoption cohorts contain a single state, so the design is
# rank-deficient once any covariate is added.
res <- twfeCovs(
    pdata = castle,
    time_var = "year",
    unit_var = "state",
    treatment = "treated",
    response = "l_homicide",
    verbose = TRUE)

# Print results
```

```
print(res, max_cohorts = Inf)

## End(Not run)
```

`twfeCovs-class` *TWFE-With-Covariates Output Class*

Description

S3 class for the output of `twfeCovs()`. Carries the same styled `print` / `summary` / `coef` surface as the three sibling estimators, plus `tidy` / `glance` (broom) and `simultaneousCIs`. `plot` and `augment` are intentionally not provided: `twfeCovs()` estimates one pooled effect per cohort, so it has no per-(cohort, time) / event-study basis to plot, and its coefficient vector is in a reduced basis that `augment()`'s fitted-value path does not match (#58). Both raise an informative error.

`twfeCovsWithSimulatedData`
Run twfeCovs on Simulated Data

Description

This function runs the bridge-penalized extended two-way fixed effects estimator (`twfeCovs()`) on simulated data. It is simply a wrapper for `twfeCovs()`: it accepts an object of class `"FETWFE_simulated"` (produced by `simulateData()`) and unpacks the necessary components to pass to `twfeCovs()`. So the outputs match `twfeCovs()`, and the needed inputs match their counterparts in `twfeCovs()`.

Usage

```
twfeCovsWithSimulatedData(
  simulated_obj,
  verbose = FALSE,
  alpha = 0.05,
  add_ridge = FALSE,
  allow_no_never_treated = TRUE,
  se_type = "default",
  ci_type = c("simultaneous", "pointwise")
)
```

Arguments

`simulated_obj`	An object of class `"FETWFE_simulated"` containing the simulated panel data and design matrix.
`verbose`	Logical; if TRUE, more details on the progress of the function will be printed as the function executes. Default is FALSE.

alpha — Numeric; function will calculate (1 - alpha) confidence intervals for the cohort average treatment effects that will be returned in catt_df.

add_ridge — (Optional.) Logical; if TRUE, adds a small amount of ridge regularization to the (untransformed) coefficients to stabilize estimation. Default is FALSE.

allow_no_never_treated — (Optional.) Logical; if TRUE (default) and the input panel contains no never-treated units, the panel is auto-truncated by dropping time periods at and after the latest cohort's start time — the units in that latest cohort then serve as the never-treated comparison group in the retained sub-panel — with a warning naming the dropped periods. If FALSE, the estimator stops with an error in this case (the package's behavior prior to version 1.5.6). The argument has no effect when the input already contains never-treated units. Default is TRUE.

se_type — Character; one of "default", "conservative", or "cluster". "default" returns the tight Gaussian variance sqrt(att_var_1 + att_var_2) from Theorem (c$'$) under Assumption (Psi-IF) (asymptotically exact for the package's default cohort sample-proportions estimator); "conservative" returns the Cauchy-Schwarz upper bound from Theorem (c) (use only when the propensity-score estimator violates (Psi-IF)); "cluster" is an *experimental* unit-clustered Liang-Zeger sandwich SE on the OLS-selected support (see the companion vignette inference_vignette for the formula, the assumptions, and the theory-pending caveat). Default is "default". v1.12.0 introduced the tight Gaussian default; versions <= 1.11.7 used the conservative Cauchy-Schwarz formula as the default.

ci_type — Character; one of "simultaneous" (default) or "pointwise". Controls the confidence-interval bounds reported for the cohort-specific ATTs (in catt_df). "simultaneous" reports parametric simultaneous (family-wise, uniform) bands computed via simultaneousCIs(): the band covers all cohort effects jointly with probability 1 - alpha, matching the default presentation of did::aggte(cband = TRUE). "pointwise" reports per-effect Wald intervals (each covers its own effect with probability 1 - alpha, no joint guarantee — the behavior of versions <= 1.15.1). Both the interval bounds and the per-cohort p-values (p_value) follow ci_type (single-step max-T multiplicity-adjusted under "simultaneous", per-cohort Wald under "pointwise"; #200); the standard errors (se) are identical under both settings. twfeCovs estimates a single pooled effect per cohort, so only the cohort family is affected (it has no event-study surface). When standard errors are unavailable (e.g., a rank-deficient design) the bounds are NA under both settings. Default is "simultaneous".

Value

An object of class twfeCovs containing the following elements:

att_hat — The estimated overall average treatment effect for a randomly selected treated unit.

att_se — A standard error for the ATT. If indep_counts was provided, this standard error is asymptotically exact; otherwise, it is asymptotically conservative. If the Gram matrix is not invertible, this will be NA.

att_p_value	A two-sided p-value for the overall ATT against the null H_0: tau = 0, computed as 2 * pnorm(-|att_hat / att_se|). NA if att_se is zero or NA. Standard post-OLS interpretation; twfeCovs does not perform selection.
catt_hats	A named vector containing the estimated average treatment effects for each cohort.
catt_ses	A named vector containing the (asymptotically exact, non-conservative) standard errors for the estimated average treatment effects within each cohort. If the Gram matrix is not invertible, the entries are NA.
cohort_probs	A vector of the estimated probabilities of being in each cohort conditional on being treated, which was used in calculating att_hat. If indep_counts was provided, cohort_probs was calculated from that; otherwise, it was calculated from the counts of units in each treated cohort in pdata.
catt_df	A data frame (with S3 class c("catt_df", "data.frame")) displaying the cohort names (cohort), average treatment effects (estimate), standard errors (se), 1 - alpha confidence interval bounds (ci_low, ci_high), and per-cohort p-values (p_value). No selected column; twfeCovs does not perform selection. The catt_df S3 class makes [[/ $ / [access on the pre-1.11.0 Title-Case column names (Cohort, Estimated TE, SE, ConfIntLow, ConfIntHigh, P_value) stop() with a migration message pointing to the new name. See NEWS.md for the rename table.
beta_hat	The full vector of estimated coefficients.
treat_inds	The indices of beta_hat corresponding to the treatment effects for each cohort.
treat_int_inds	The indices of beta_hat corresponding to the interactions between the treatment effects for each cohort and the covariates.
sig_eps_sq	Either the provided sig_eps_sq or the estimated one, if a value wasn't provided.
sig_eps_c_sq	Either the provided sig_eps_c_sq or the estimated one, if a value wasn't provided.
X_ints	The design matrix created containing all interactions, time and cohort dummies, etc.
y	The vector of responses, containing nrow(X_ints) entries.
X_final	The design matrix after applying the change in coordinates to fit the model and also multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.
y_final	The final response after multiplying on the left by the square root inverse of the estimated covariance matrix for each unit.
N	The final number of units that were in the data set used for estimation (after any units may have been removed because they were treated in the first time period).
T	The number of time periods in the final data set.
G	The final number of treated cohorts that appear in the final data set.
R	Deprecated alias for G, retained for backward compatibility; populated with the same value. Use G. Will be removed in a future release.
d	The final number of covariates that appear in the final data set (after any covariates may have been removed because they contained missing values or all contained the same value for every unit).

`p` The final number of columns in the full set of covariates used to estimate the model.

`calc_ses` Logical indicating whether standard errors were calculated.

`cohort_probs_overall`
A vector of the estimated cohort probabilities on the overall sample (treated and untreated), used in computing the variance of the overall ATT.

`indep_counts_used`
Logical scalar; `TRUE` if a valid `indep_counts` argument was provided and used for asymptotically-exact ATT inference, `FALSE` otherwise.

`se_type` Character scalar; the `se_type` argument the user passed (`"default"`, `"conservative"`, or `"cluster"`).

`alpha` The alpha level used for confidence intervals.

`ci_type` Character scalar; the `ci_type` argument the user passed (`"simultaneous"` or `"pointwise"`), controlling whether the reported `catt_df` confidence-interval bounds are simultaneous (family-wise) or pointwise.

`catt_band_applied`
Logical scalar; `TRUE` when the fit-time cohort-family simultaneous band was computed and written into `catt_df`, and `FALSE` otherwise — including when `ci_type = "pointwise"`, when standard errors were unavailable, and when the band construction degraded. It records only whether the band was applied: it does not imply the band is wider than the pointwise interval (they coincide when fewer than two effects have positive variance) and does not imply the band is informative (a degenerate fit can carry an all-zero band with `TRUE`).

`y_mean` Numeric scalar; mean of the original (pre-centering) response. Stored so downstream methods (`augment()`, `predict()`) can return fitted values on the original-response scale.

`response_col_name`
Character scalar; the response column name in the original `pdata`.

`time_var`, `unit_var`, `treatment`
Character scalars; the corresponding arguments the user passed.

`covs` Character vector; the original `covs` argument (pre-factor- expansion).

`internal` A list containing internal outputs that are typically not needed for interpretation, packaged here for parity with `fetwfe()` so downstream consumers can use a single canonical access path across all four estimator classes (#144). The first five sub-slots (`X_ints`, `y`, `X_final`, `y_final`, `calc_ses`) are also duplicated at top level for backward compat; `variance_components` and `first_year` live only under `$internal`:

- **X_ints** The design matrix containing all interactions, time and cohort dummies, etc. Same value as top-level `X_ints`.
- **y** The vector of responses. Same as top-level `y`.
- **X_final** The design matrix after the change-of-coordinates step. Same as top-level `X_final`.
- **y_final** The transformed response vector. Same as top-level `y_final`.
- **calc_ses** Logical indicating whether standard errors were calculated. Same as top-level `calc_ses`.

variance_components A list exposing the two variance pieces (`att_var_1`, `att_var_2`) plus paper-notation counterparts (`V_1`, `V_2`) and unit-scaled variance estimators (`tilde_v_N`, `hat_v_N`, `tilde_v_N_C`, `tilde_v_N_C_pi_hat`, `tilde_v_N_C_pi_hat_cons`, `tilde_v_N_cons`). The Wald CI is `[hat_T_N +- qnorm(1-alpha/2)` (paper Eq. `conf.int.form`). New in v1.12.0 (issue #141 + #146).

first_year Integer or numeric scalar; the first (earliest) `time_var` value in the panel after `idCohorts()` processing. Consumed by `eventStudy()` to map `cohort_probs`' cohort labels (treatment-start years) to 1-based panel-time-index offsets when the labels are integer-coercible. New in v1.13.3 (issue #174).

The returned object is an S3-classed `"twfeCovs"` list with `print()`, `summary()`, `coef()`, `tidy()`, `glance()`, and `simultaneousCIs()` methods, matching the three sibling estimators. `plot()` is intentionally not defined — `twfeCovs()` estimates one pooled effect per cohort, so there is no per-(cohort, time) / event-study structure to plot. `augment()` is intentionally not defined — the coefficient vector lives in a reduced cohort-level basis that `augment()`'s fitted-value path does not match. Both raise an informative error (#58).

Examples

```
## Not run:
  # Generate coefficients
  coefs <- genCoefs(G = 5, T = 30, d = 12, density = 0.1, eff_size = 2, seed = 123)

  # Simulate data using the coefficients
  sim_data <- simulateData(coefs, N = 120, sig_eps_sq = 5, sig_eps_c_sq = 5, seed = 123)

  result <- twfeCovsWithSimulatedData(sim_data)

## End(Not run)
```

Index

[.catt_df, 3
[<-.catt_df, 5
[[.catt_df, 4
[[<-.catt_df, 4
$.catt_df, 6
$<-.catt_df, 6

attgtToFetwfeDf, 7
augment.betwfe, 9
augment.etwfe, 10
augment.fetwfe, 10
augment.fetwfe(), *9, 10*
augment.twfeCovs, 11

betwfe, 12
betwfe(), *27*, *29*, *31*, *83, 84*, *86*, *101*
betwfe-class, 20
betwfeWithSimulatedData, 20
broom::tidy(), *15*, *23*, *38*, *45*, *55*, *64*

cohortStudy, 27
cohortStudy(), *15*, *23*, *28–30*, *34*, *38*, *45*, *51*, *55*, *65*, *84–86*, *102*
cohortTimeATTs, 28
cohortTimeATTs(), *28*, *34*, *51*, *103*

debiasedATT, 30
debiasedATT(), *35*, *56*, *65, 66*, *88*, *104*
debiasedATTWithSimulatedData, 35

etwfe, 36
etwfe(), *27*, *29*, *31*, *83, 84*, *86*, *104*
etwfe-class, 42
etwfeToFetwfeDf, 42
etwfeWithSimulatedData, 44
eventStudy, 48, *79*
eventStudy(), *15*, *23*, *27–30*, *34*, *38*, *45*, *55*, *64*, *83–86*, *99, 100*, *105*

fetwfe, 51, *74*
fetwfe(), *27*, *29–31*, *34, 35*, *83, 84*, *86*, *99*, *106*
fetwfe-class, 61
FETWFE_coefs-class, 61
FETWFE_simulated-class, 62
FETWFE_tes-class, 62
fetwfeWithSimulatedData, 62
fetwfeWithSimulatedData(), *35*

genCoefs, 70
genCoefsCore, 75
getTes, 78, *90*, *93*
getTes(), *107*
glance.betwfe, 80
glance.etwfe, 81
glance.etwfe(), *82*
glance.fetwfe, 81
glance.fetwfe(), *80, 81*
glance.twfeCovs, 82
glance.twfeCovs(), *11*

plot.betwfe, 83
plot.betwfe(), *86*
plot.etwfe, 84
plot.etwfe(), *86*
plot.fetwfe, 85
plot.fetwfe(), *83–85*
plot.simultaneous_cis, 87
plot.twfeCovs, 87
print.debiased_att, 88
print.simultaneous_cis, 88

simulateData, *78*, 89
simulateData(), *35*
simulateDataCore, 92
simultaneousCIs, 95
simultaneousCIs(), *15*, *23*, *30*, *34*, *38*, *45*, *49, 50*, *55*, *65*, *83*, *85–89*, *105*, *108*, *112*, *117*

tidy.betwfe, 101

tidy.cohortStudy, 102
tidy.cohortStudy(), *27*, *28*, *103*, *106*
tidy.cohortTimeATTs, 103
tidy.cohortTimeATTs(), *29*, *30*
tidy.debiased_att, 104
tidy.etwfe, 104
tidy.etwfe(), *109*
tidy.eventStudy, 105
tidy.eventStudy(), *102*, *103*
tidy.fetwfe, 106
tidy.fetwfe(), *101*, *104*, *109*
tidy.FETWFE_tes, 107
tidy.simultaneous_cis, 108
tidy.twfeCovs, 109
tidy.twfeCovs(), *11*, *87*
twfeCovs, 110
twfeCovs(), *27–29*, *31*, *109*
twfeCovs-class, 116
twfeCovsWithSimulatedData, 116