Package 'fetwfe'

May 24, 2026

Title Fused Extended Two-Way Fixed Effects

Version 1.10.0

Maintainer Gregory Faletto <gfaletto@gmail.com>

Depends R (>= 4.1.0)

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://github.com/gregfaletto/fetwfePackage

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

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

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

Suggests bacondecomp, knitr, rmarkdown, dplyr, did, expm, ggplot2, broom, lme4, testthat (>= 3.0.0), tibble

VignetteBuilder knitr

NeedsCompilation no

Author Gregory Faletto [aut, cre] (ORCID: <https://orcid.org/0000-0001-8298-1401>)

Repository CRAN

Date/Publication 2026-05-24 00:20:02 UTC

Contents

`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 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(R = 3, T = 6, d = 2, density = 0.5,
                               eff_size = 2),
                      N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5)
  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(R = 3, T = 6, d = 2, density = 0.5,
                               eff_size = 2),
                      N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5)
  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(R = 3, T = 6, d = 2, density = 0.5,
                               eff_size = 2),
                      N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5)
  res <- fetwfeWithSimulatedData(sim)
  broom::augment(res, data = sim$pdata)

## End(Not run)
```

---

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"
)
```

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.) Character; a vector containing the names of the columns for covariates. 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 `R` 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.

lambda.max (Optional.) Numeric. A penalty parameter lambda will be selected over a grid search by BIC in order to select a single model. 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. The smallest lambda penalty parameter that will be considered. See the description of lambda.max for details. Default is NA.

nlambda (Optional.) Integer. The total number of lambda penalty parameters that will be 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" (the package's Assumption-F1-based standard error from the paper) or "cluster" (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). "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. Default is "default".

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 dataframe displaying the cohort names, average treatment effects, standard errors, 1 - alpha confidence interval bounds, 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.

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 — Either the provided lambda.max or the one that was used, if a value wasn't provided. (This is returned to help with getting a reasonable range of lambda values for grid search.)

lambda.max_model_size — The size of the selected model corresponding 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 size of the selected model corresponding to 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 BIC. 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 size of the model that was selected. 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.

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.

R — The final number of treated cohorts that appear in the final data set.

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" or "cluster").

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

```
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[["Estimated TE"]] <- 100 * catt_df_pct[["Estimated TE"]]
catt_df_pct[["SE"]] <- 100 * catt_df_pct[["SE"]]
catt_df_pct[["ConfIntLow"]] <- 100 * catt_df_pct[["ConfIntLow"]]
catt_df_pct[["ConfIntHigh"]] <- 100 * catt_df_pct[["ConfIntHigh"]]

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"
)
```

Arguments

simulated_obj	An object of class "FETWFE_simulated" containing the simulated panel data and design matrix.
lambda.max	(Optional.) Numeric. A penalty parameter lambda will be selected over a grid search by BIC in order to select a single model. The largest lambda in the grid will be lambda.max. If no lambda.max is provided, one will be selected automatically. For lambda <= 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. The smallest lambda penalty parameter that will be considered. See the description of lambda.max for details. Default is NA.

nlambda (Optional.) Integer. The total number of lambda penalty parameters that will be 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" (the package's Assumption-F1-based standard error from the paper) or "cluster" (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). "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. Default is "default".

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 dataframe displaying the cohort names, average treatment effects, standard errors, `1 - alpha` confidence interval bounds, 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`.

`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` Either the provided `lambda.max` or the one that was used, if a value wasn't provided. (This is returned to help with getting a reasonable range of `lambda` values for grid search.)

`lambda.max_model_size`
The size of the selected model corresponding `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 size of the selected model corresponding to `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 BIC. 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 size of the model that was selected. 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.

`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.
R	The final number of treated cohorts that appear in the final data set.
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" 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).

Examples

```
## Not run:
  # Generate coefficients
  coefs <- genCoefs(R = 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)

  result <- betwfeWithSimulatedData(sim_data)

## 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"
)
```

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.) Character; a vector containing the names of the columns for covariates. 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 R 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" (the package's Assumption-F1-based standard error from the paper) or "cluster" (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".

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 dataframe displaying the cohort names, average treatment effects, standard errors, 1 - alpha confidence interval bounds, and per-cohort p-values (P_value). No selected column; ETWFE does not perform selection.

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.

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

`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"` or `"cluster"`).

`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>()`.

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` 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"
)
```

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"` (the package's Assumption-F1-based standard error from the paper) or `"cluster"` (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"`.

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 dataframe displaying the cohort names, average treatment effects, standard errors, 1 - alpha confidence interval bounds, and per-cohort p-values (P_value). No selected column; ETWFE does not perform selection.
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.
R	The final number of treated cohorts that appear in the final data set.
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"` or `"cluster"`).

`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).

Examples

```
## Not run:
  # Generate coefficients
  coefs <- genCoefs(R = 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)

  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 - r` (where `t` is calendar time and `r` 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)` when `indep_counts` was supplied to the fit (asymptotically exact), else the conservative Cauchy-Schwarz bound `sqrt(var_1 + var_2 + 2 sqrt(var_1 * var_2))`.

Usage

```
eventStudy(x, alpha = 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).

Value

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

event_time Integer; event time `e = t - r`, 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; two-sided Wald p-value (`2 * pnorm(-|estimate / se|)`), `NA` when `se` is `0` or `NA`.

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.

Examples

```
## Not run:
  coefs <- genCoefs(R = 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)
  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).

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"
)
```

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.) Character; a vector containing the names of the columns for covariates. 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 `R` 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.

`lambda.max` (Optional.) Numeric. A penalty parameter `lambda` will be selected over a grid search by BIC in order to select a single model. 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. The smallest lambda penalty parameter that will be considered. See the description of lambda.max for details. Default is NA.

nlambda (Optional.) Integer. The total number of lambda penalty parameters that will be 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" (the package's Assumption-F1-based standard error from the paper) or "cluster" (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). "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. Default is "default".

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. 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). 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 dataframe displaying the cohort names, average treatment effects, standard errors, `1 - alpha` confidence interval bounds, 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`.

`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` Either the provided `lambda.max` or the one that was used, if a value wasn't provided. (This is returned to help with getting a reasonable range of `lambda` values for grid search.)

`lambda.max_model_size` The size of the selected model corresponding to `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 size of the selected model corresponding to `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 BIC. 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 size of the model that was selected. 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.

`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.

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

`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.

`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"` or `"cluster"`).

`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.

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.

Examples

```
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 `fetwfe()` requires.
castle$l_homicide <- log(castle$homicide)
castle$treated <- as.integer(castle$cdl > 0)

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

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

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

Description

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

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"
)
```

Arguments

`simulated_obj`	An object of class `"FETWFE_simulated"` containing the simulated panel data and design matrix.
`lambda.max`	(Optional.) Numeric. A penalty parameter `lambda` will be selected over a grid search by BIC in order to select a single model. The largest `lambda` in the grid will be `lambda.max`. If no `lambda.max` is provided, one will be selected automatically. For `lambda <= 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. The smallest `lambda` penalty parameter that will be considered. See the description of `lambda.max` for details. Default is NA.

`nlambda` (Optional.) Integer. The total number of `lambda` penalty parameters that will be 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"` (the package's Assumption-F1-based standard error from the paper) or `"cluster"` (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). `"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`. Default is `"default"`.

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. 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). 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 dataframe displaying the cohort names, average treatment effects, standard errors, `1 - alpha` confidence interval bounds, 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`.

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
: Either the provided `lambda.max` or the one that was used, if a value wasn't provided. (This is returned to help with getting a reasonable range of `lambda` values for grid search.)

lambda.max_model_size
: The size of the selected model corresponding to `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 size of the selected model corresponding to `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 BIC. 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 size of the model that was selected. 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.

`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.

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

`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.

`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"` or `"cluster"`).

`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.

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(R = 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)

  result <- fetwfeWithSimulatedData(sim_data)

## End(Not run)
```

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()`.

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 `R`, `T`, and `d`. See the simulation studies section of Faletto (2025) for details.

Usage

```
genCoefs(R, T, d, density, eff_size, seed = NULL)
```

Arguments

`R`	Integer. The number of treated cohorts (treatment is assumed to start in periods 2 to `R + 1`).
`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 sparse coefficient vector `theta` is nonzero.
`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).
`seed`	(Optional) Integer. Seed for reproducibility.

Details

The length of `beta` is given by

$$p = R + (T-1) + d + dR + d(T-1) + num\_treats + (num\_treats \times d)$$

, where the number of treatment parameters is defined as

$$num\_treats = T \times R - \frac{R(R+1)}{2}$$

.

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 (e.g., `genBackwardsInvFusionTransformMat()` and `genInvTwoWayFusionTransformM`

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`.

R The provided number of treated cohorts.

T The provided number of time periods.

d The provided number of covariates.

seed The provided 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.

Examples

```
## Not run:
  # Generate coefficients
  coefs <- genCoefs(R = 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)
```

---

genCoefsCore *Generate Coefficient Vector for Data Generation*

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 `genRandomData()`, 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(R, T, d, density, eff_size, seed = NULL)
```

Arguments

R	Integer. The number of treated cohorts (treatment is assumed to start in periods 2 to R + 1).
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 sparse coefficient vector theta is nonzero.
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).
seed	(Optional) Integer. Seed for reproducibility.

Details

The length of beta is given by

$$p = R + (T - 1) + d + dR + d(T - 1) + num\_treats + (num\_treats \times d)$$

, where the number of treatment parameters is defined as

$$num\_treats = T \times R - \frac{R(R+1)}{2}$$

.

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 (e.g., genBackwardsInvFusionTransformMat() and genInvTwoWayFusionTransformM

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
  R <- 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(R = R, 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 = R + (T - 1) + d + d*R + d*(T - 1) + num_treats + num_treats*d,
  # where num_treats = T * R - (R*(R+1))/2.
  num_treats <- T * R - (R * (R + 1)) / 2
  p_expected <- R + (T - 1) + d + d * R + 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 calculates the overall average treatment effect on the treated (ATT) as the equal-weighted average of the cohort-specific treatment effects, and also returns the individual treatment effects for each treated cohort.

Usage

```
getTes(coefs_obj)
```

Arguments

`coefs_obj`	An object of class `"FETWFE_coefs"` containing the coefficient vector and simulation parameters.

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 the simple average of these cohort-specific effects.

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. It is computed as the (equal-weighted) mean of the cohort-specific treatment effects.

actual_cohort_tes A numeric vector of length `R` containing the true cohort-specific treatment effects, calculated by averaging the coefficients corresponding to the treatment dummies for each cohort.

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

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

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

Examples

```
## Not run:
# Generate coefficients
coefs <- genCoefs(R = 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)

# Or use the new print method for a self-describing display:
print(te_results)

## 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 13 columns.

Examples

```
## Not run:
  res <- betwfeWithSimulatedData(
    simulateData(genCoefs(R = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5)
  )
  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(R = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5)
  )
  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 13 columns.

Examples

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

## End(Not run)
```

`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 R 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
)
```

Arguments

`coefs_obj`	An object of class `"FETWFE_coefs"` containing the coefficient vector and simulation parameters.
`N`	Integer. Number of units in the panel.
`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 $[-\sqrt{3}, \sqrt{3}]$.

`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 no such condition is enforced.

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 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 $R + 1$) indicating how many units fall into the never-treated group and each of the R 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.

R Number of treated cohorts.

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(R = 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*

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 R 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,
  R,
  d,
  sig_eps_sq,
  sig_eps_c_sq,
  beta,
  seed = NULL,
  gen_ints = FALSE,
  distribution = "gaussian",
  guarantee_rank_condition = FALSE
)
```

Arguments

N	Integer. Number of units in the panel.
T	Integer. Number of time periods.
R	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 $p = R + (T-1) + d + dR + d(T-1) + num\_treats + num\_treats \times d$, where $num\_treats = T \times R - \frac{R(R+1)}{2}$. • If gen_ints = TRUE and d = 0, the expected length is $p = R + (T-1) + num\_treats$. • If gen_ints = FALSE, the expected length is $p = R + (T-1) + d + num\_treats$.
seed	(Optional) Integer. Seed for reproducibility.
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 $[-\sqrt{3}, \sqrt{3}]$.
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 no such condition is enforced.

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 $R + 1$) indicating how many units fall into the never-treated group and each of the R 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.

R Number of treated cohorts.

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
  R <- 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(R = R, 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,
    R = R,
    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)
```

`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`.
`...`	Unused.

Value

A data frame with `R + 1` rows.

Examples

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

## End(Not run)
```

`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`.
`...`	Unused.

Value

A data frame with `R + 1` rows.

Examples

```
## Not run:
  res <- etwfeWithSimulatedData(
    simulateData(genCoefs(R = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5)
  )
  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). Confidence level for the CI columns; defaults to `0.95` (`eventStudy()` does not store the alpha it was called with, so there is no fitted-object value to default to).
`...`	Unused.

Details

The `eventStudy()` output stores Wald CIs at the alpha passed at computation time. When `conf.int = TRUE` (the default), `conf.low` / `conf.high` are recomputed from `estimate` and `std.error` at the supplied `conf.level`, which can therefore differ from the computation-time alpha. When `conf.int = FALSE`, the CI columns are omitted.

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(R = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5)
  )
  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). Confidence level for the CI columns. Defaults to `1 - x$alpha` (faithful to the alpha used at fit time; deviates from `broom::tidy.lm`'s `0.95` default by design).
`...`	Unused; present for S3 compatibility.

Value

A data frame with `R + 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(R = 3, T = 6, d = 2, density = 0.5, eff_size = 2),
                 N = 120, sig_eps_sq = 1, sig_eps_c_sq = 0.5)
  )
  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 `r` adopts at calendar time 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 `R + 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(R = 3, T = 6, d = 2, density = 0.5, eff_size = 2)
  broom::tidy(getTes(coefs))

## End(Not run)
```

`twfeCovs` *Two-way fixed effects with covariates and separate treatment effects for each 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 separate treatment effects for each 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"
)
```

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.) Character; a vector containing the names of the columns for covariates. 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 R 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" (the package's Assumption-F1-based standard error from the paper) or "cluster" (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".

Value

A named list with 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 dataframe displaying the cohort names, average treatment effects, standard errors, 1 - alpha confidence interval bounds, and per-cohort p-values (P_value). No selected column; twfeCovs does not perform selection.

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.
R	The final number of treated cohorts that appear in the final data set.
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" or "cluster").

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()`. Minimal surface (coef + a bare print that preserves the pre-#76 behavior of just dumping the list); a full styled `print` / `summary` like the three sibling estimators is a separate follow-up.

`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"
)
```

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" (the package's Assumption-F1-based standard error from the paper) or "cluster" (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".

Value

A named list with 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 dataframe displaying the cohort names, average treatment effects, standard errors, `1 - alpha` confidence interval bounds, and per-cohort p-values (`P_value`). No `selected` column; `twfeCovs` does not perform selection.

`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.

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

`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"` 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`.

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

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

Examples

```
## Not run:
  # Generate coefficients
  coefs <- genCoefs(R = 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)

  result <- twfeCovsWithSimulatedData(sim_data)

## End(Not run)
```

Index