Package ‘fetwfe’

May 15, 2025

Title Fused Extended Two-Way Fixed Effects

Version 1.0.0

Maintainer Gregory Faletto <`gfaletto@gmail.com`>

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. Provides flexible tuning parameters (including user-specified or data-driven choices for penalty parameters), detailed output including overall and cohort-specific treatment effects with confidence intervals, and extensive diagnostic tools. Also provides functions for generating simulated panel data formatted for estimating FETWFE, and running and evaluating simulations. See details in Faletto (2025) (<doi:10.48550/arXiv.2312.05985>).

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

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

Suggests bacondecomp, knitr, rmarkdown, dplyr

VignetteBuilder knitr

NeedsCompilation no

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

Repository CRAN

Date/Publication 2025-05-15 05:10:02 UTC

Contents

`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
)
```

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 using the estimator from Pesaran (2015, Section 26.5.1) with ridge regression. 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 using the estimator from Pesaran (2015, Section 26.5.1) with ridge regression. 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.

Value

A named list with 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.
`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, and `1 - alpha` confidence interval bounds.
`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 `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.

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. Pesaran, M. H. . Time Series and Panel Data Econometrics. Number 9780198759980 in OUP Catalogue. Oxford University Press, 2015. URL https://ideas.repec.org/b/oxp/obooks/9780198759980.html.

Examples

```
set.seed(23451)

library(bacondecomp)

data(divorce)

# sig_eps_sq and sig_eps_c_sq, calculated in a separate run of `fetwfe(),
# are provided to speed up the computation of the example
res <- fetwfe(
    pdata = divorce[divorce$sex == 2, ],
    time_var = "year",
    unit_var = "st",
    treatment = "changed",
    covs = c("murderrate", "lnpersinc", "afdcrolls"),
    response = "suiciderate_elast_jag",
    sig_eps_sq = 0.1025361,
    sig_eps_c_sq = 4.227651e-35,
    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`

`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
)
```

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.

Value

A named list with 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.
`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, and `1 - alpha` confidence interval bounds.
`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 `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.

Examples

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

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

  result <- fetwfeWithSimulatedData(sim_data)
```

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

genCoefs *Generate Coefficient Vector for Data Generation*

Description

This function generates a coefficient vector `beta` for simulation studies of the fused extended two-way fixed effects estimator. It returns an S3 object of class `"FETWFE_coefs"` containing `beta` along with simulation parameters `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.

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 that is sparse, from which `beta` is derived.

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

A named list with two 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 containing the true cohort-specific treatment effects, calculated by averaging the coefficients corresponding to the treatment dummies for each cohort.

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)

## 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")
```

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.
`distribution`	Character. Distribution to generate covariates. Defaults to `"gaussian"`. If set to `"uniform"`, covariates are drawn uniformly from $[-\sqrt{3}, \sqrt{3}]$..

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"
)
```

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.
`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 $[-\sqrt{3}, \sqrt{3}]$.

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)
```

Index