

Package ‘coresynth’

July 12, 2026

Title Fast and Unified Synthetic Control Methods

Version 0.3.0

Description A unified 'Formula' interface to the Synthetic Control Method (SCM) and related panel-data causal inference estimators: Synthetic Difference-in-Differences (SDID), Generalized Synthetic Control (GSC), Matrix Completion (MC), Time-Aware Synthetic Control (TASC), and Synthetic Interventions (SI), together with an experimental-design variant. Computational bottlenecks (quadratic programming, singular value decomposition, and Kalman filtering) are implemented in 'C++' via 'RcppArmadillo'. Methods are described in Abadie, Diamond and Hainmueller (2010) <[doi:10.1198/jasa.2009.ap08746](https://doi.org/10.1198/jasa.2009.ap08746)>, Arkhangelsky, Athey, Hirshberg, Imbens and Wager (2021) <[doi:10.1257/aer.20190159](https://doi.org/10.1257/aer.20190159)>, Xu (2017) <[doi:10.1017/pan.2016.2](https://doi.org/10.1017/pan.2016.2)>, Athey, Bayati, Doudchenko, Imbens and Khosravi (2021) <[doi:10.1080/01621459.2021.1891924](https://doi.org/10.1080/01621459.2021.1891924)>, and Agarwal, Shah and Shen (2025) <[doi:10.1287/opre.2025.1590](https://doi.org/10.1287/opre.2025.1590)>.

License MIT + file LICENSE

URL <https://github.com/yo5uke/coresynth>, <https://yo5uke.com/coresynth/>

BugReports <https://github.com/yo5uke/coresynth/issues>

Encoding UTF-8

Config/roxygen2/markdown TRUE

RoxygenNote 8.0.0

Depends R (>= 4.1.0)

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, Formula, ggplot2, broom, jsonlite

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Yosuke Abe [aut, cre]

Maintainer Yosuke Abe <yosuke.abe0507@gmail.com>

Repository CRAN
Date/Publication 2026-07-12 15:50:08 UTC

Contents

augment_scm	2
conformal_inference	3
export_json	5
glance.coresynth_inference	5
gsc_boot	6
gsc_ife_cpp	7
gsc_inference	8
kalman_smoother_cpp	9
loo_donors	10
mspe_ratio_pval	11
placebo_in_time	12
plot.coresynth	13
plot.scm_design	15
plot.scm_placebo	15
pred	17
scm_design	18
scm_fit	20
scm_inner_weights_cpp	22
scm_placebo_cpp	23
scm_weights_cpp	24
sdid_estimate_cpp	25
sdid_inference	25
sdid_placebo_cpp	26
sdid_time_weights_cpp	27
sdid_unit_weights_cpp	28
si_inference	28
si_pcr_cpp	29
soft_impute_cpp	30
tensor_unfold_cpp	30
tidy.coresynth_inference	31
treated_outcomes	31
Index	34

augment_scm	<i>Augmented Synthetic Control Method (Ridge ASCM)</i>
-------------	--

Description

Applies a ridge-regression-based bias correction to a fitted SCM object, following Ben-Michael, Feller & Rothstein (2021, JASA). The corrected estimator is:

Usage

```
augment_scm(fit, lambda_ridge = NULL)
```

Arguments

<code>fit</code>	A coreshynth object from <code>scm_fit()</code> with method = "scm".
<code>lambda_ridge</code>	Ridge penalty (non-negative). NULL (default) selects the penalty by leave-one-out cross-validation on the control units.

Details

$$\text{tau_aug} = \text{tau_SCM} + (\text{m_tr_post} - \sum_j W_j * \text{m_j_post})$$

where $\text{m_i_post} = Y_{\text{pre_i}}$ beta_hat is the ridge outcome model prediction for unit i 's mean post-treatment outcome, and beta_hat is estimated by ridge regression across control units.

Value

A list with:

- `att_aug`: Augmented ATT estimate
- `delta`: Bias correction term $(\text{m_tr_post} - \sum_j W_j \text{m_j_post})$
- `att_scm`: Original SCM ATT for comparison
- `lambda_ridge`: Ridge penalty used
- `beta_hat`: Ridge regression coefficients (length T_{pre})

conformal_inference	<i>Conformal Inference for Synthetic Control Estimators</i>
---------------------	---

Description

Implements the permutation-based conformal inference procedure of Chernozhukov, Wuthrich & Zhu (2021, JASA). The test inverts a sharp null $H_0 : \tau = \tau_0$ by imputing the treated post-treatment counterfactual as $Y_{1t} - \tau_0$, re-estimating the counterfactual proxy on **all** T periods (imposing the null), and computing a moving-block permutation p-value from the estimated residuals. A confidence interval is obtained by test inversion over a grid of candidate τ_0 .

Usage

```
conformal_inference(
  fit,
  tau0 = 0,
  q = 1,
  alternative = c("two.sided", "greater", "less"),
  ci = TRUE,
  level = 0.95,
```

```

    grid = NULL,
    n_grid = 200L,
    grid_mult = 4,
    ...
)

```

Arguments

<code>fit</code>	A coresynth object from <code>scm_fit()</code> .
<code>tau0</code>	Null value of the ATT for the reported p-value (default 0).
<code>q</code>	Exponent of the S_q test statistic ($S_q = (T_{\text{post}}^{-1} \sum u_t ^q)^{1/q}$). Default 1, robust to heavy-tailed data (CWZ 2021). Used only for alternative = "two.sided"; one-sided tests use the signed mean post-treatment residual.
<code>alternative</code>	"two.sided" (default), "greater", or "less".
<code>ci</code>	Logical; construct a confidence interval by test inversion (default TRUE).
<code>level</code>	Confidence level for the interval (default 0.95).
<code>grid</code>	Optional numeric vector of candidate τ_0 values for test inversion. When NULL (default), a grid of <code>n_grid</code> points is centred on the point estimate with half-width <code>grid_mult</code> times the pre-treatment residual standard deviation.
<code>n_grid</code>	Number of grid points when <code>grid = NULL</code> (default 200).
<code>grid_mult</code>	Half-width multiplier when <code>grid = NULL</code> (default 4).
<code>...</code>	Unused.

Details

Supported for **sharp** (single-cohort) fits with method in `c("scm", "sdid", "gsc", "mc", "si")`. Staggered, multi-arm, and tasc fits are not supported (use `sdid_inference()`, `gsc_inference()`, or `si_inference()` instead).

Value

A list of class `c("conformal_inference", "coresynth_inference")` with `estimate`, `se` (NA; conformal has no SE), `p_value` (at `tau0`), `ci_lower`, `ci_upper`, `method` ("conformal"), `n_controls`, `alternative`, `staggered` (FALSE), plus `tau0`, `q`, `grid`, and `p_grid` (p-values along the grid). Compatible with `tidy()` / `glance()`.

References

Chernozhukov, V., Wuthrich, K., & Zhu, Y. (2021). An Exact and Robust Conformal Inference Method for Counterfactual and Synthetic Controls. *Journal of the American Statistical Association*, 116(536), 1849-1864.

export_json	<i>Export coresynth Results to JSON</i>
-------------	---

Description

Generates a comprehensive, standardized JSON record covering all six estimators. Suitable for reproducibility workflows (Xu & Yang 2026) and downstream tooling. Pass the result of `mspe_ratio_pval()` or `gsc_boot()` via the `inference` argument to include inference results.

Usage

```
export_json(x, file = "coresynth_results.json", inference = NULL, digits = 6L)
```

Arguments

x	A coresynth object from <code>scm_fit()</code> .
file	Output file path. Default "coresynth_results.json". Pass NULL to skip writing and return the R list invisibly.
inference	Optional list from <code>mspe_ratio_pval()</code> or <code>gsc_boot()</code> . When provided, populates the inference section and updates estimate with <code>p_value</code> , <code>se</code> , <code>ci_lower</code> , <code>ci_upper</code> .
digits	Number of significant digits applied to numeric values (default 6L).

Value

Invisibly, the R list that was (or would be) serialized.

glance.coresynth_inference	<i>Glance at an inference result</i>
----------------------------	--------------------------------------

Description

One-row summary of a `coresynth_inference` (or `sdid_inference`) object.

Usage

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

Arguments

x	An inference object.
...	Unused.

Value

A one-row data.frame with columns method, n_controls, staggered, estimate, std.error, p.value, conf.low, conf.high, alternative, n_boot_valid.

gsc_boot

Parametric Bootstrap Inference for GSC (Xu 2017 S.3)

Description

Generates the null distribution of the ATT under H0 (no treatment effect) by parametric resampling from the estimated IFE factor model. Under H0, both the control panel and treated unit are generated from the fitted factor model with homoskedastic noise. When the fit includes covariate adjustment (beta), the covariate contribution is included in the simulated DGP and re-estimated in each bootstrap replicate.

Usage

```
gsc_boot(fit, B = 499L, alpha = 0.05, seed = NULL)
```

Arguments

fit	A coresynth object from <code>scm_fit()</code> with method = "gsc".
B	Bootstrap replications (default 499L).
alpha	Significance level for the confidence interval (default 0.05).
seed	RNG seed for reproducibility (default NULL).

Value

A list with:

- p_value: Two-sided p-value: $\text{mean}(|\text{ATT}^*| \geq |\text{ATT}_{\text{obs}}|)$
- ci_lower: Lower bound of $(1-\alpha)*100\%$ bootstrap CI
- ci_upper: Upper bound of $(1-\alpha)*100\%$ bootstrap CI
- se: Bootstrap standard error
- boot_dist: Numeric vector of length B (bootstrap ATT* values)
- att_obs: Observed ATT from the original fit

Description

Implements Xu (2017) IFE model with optional covariate adjustment. When X_{co} has $p > 0$ slices, runs an EM loop alternating between: E-step: truncated SVD of $Y_{tilde} = Y_{co} - X_{co} * \beta$. M-step: panel OLS to update β given current factors. When X_{co} has 0 slices (default), falls back to the plain 3-step estimator.

Usage

```
gsc_ife_cpp(Y_co, Y_tr_pre, r, X_co, X_tr_pre, max_iter = 50L, tol = 1e-06)
```

Arguments

<code>Y_co</code>	Control units outcome matrix ($T \times N_{co}$)
<code>Y_tr_pre</code>	Treated units pre-treatment outcomes ($T_{pre} \times N_{tr}$)
<code>r</code>	Number of latent factors (must be $\leq \min(T, N_{co})$)
<code>X_co</code>	Time-varying covariate cube ($T \times N_{co} \times p$). Pass an empty cube (0 slices) for the covariate-free estimator.
<code>X_tr_pre</code>	Time-varying covariate cube for treated units in the pre-treatment window ($T_{pre} \times N_{tr} \times p$). Required for correct Step 2 loading estimation per Xu (2017): λ_{hat} is estimated from $Y_{tr_pre} - X_{tr_pre} * \beta$ (covariate- demeaned). Pass an empty cube (0 slices) to skip demeaning (backward-compatible, but biased when $\beta \neq 0$).
<code>max_iter</code>	Maximum EM iterations (default 50)
<code>tol</code>	Convergence tolerance on relative beta change (default 1e-6)

Value

A list with components:

- `F`: estimated time factors ($T \times r$).
- `L_co`: control-unit factor loadings ($N_{co} \times r$).
- `L_tr`: treated-unit factor loadings ($N_{tr} \times r$).
- `Y_tr_hat`: estimated treated-unit counterfactual outcomes ($T \times N_{tr}$).
- `singular_values`: singular values from the final truncated SVD.
- `beta`: estimated covariate coefficients ($p \times 1$), empty when no covariates are supplied.

gsc_inference

*Non-parametric Inference for GSC (Xu 2017)***Description**

Estimates SE and confidence intervals for the ATT via non-parametric cluster bootstrap or jackknife over control units. Works for both sharp and staggered GSC fits. For staggered fits, bootstrap resamples each cohort's control pool independently, and jackknife uses a per-cohort LOO with delta-method variance aggregation.

Usage

```
gsc_inference(
  fit,
  method = c("bootstrap", "jackknife", "jackknife_global"),
  n_boot = 499L,
  level = 0.95,
  alternative = c("two.sided", "greater", "less"),
  seed = NULL
)
```

Arguments

fit	A coresynth object from <code>scm_fit()</code> with method = "gsc".
method	"bootstrap" (default) or "jackknife".
n_boot	Number of bootstrap replications (default 499L; ignored for jackknife).
level	Confidence level (default 0.95).
alternative	"two.sided" (default), "greater", or "less".
seed	RNG seed for reproducibility (default NULL).

Details

Note: `gsc_boot()` performs a *parametric* bootstrap under H_0 (hypothesis testing). `gsc_inference()` provides *non-parametric* SE and CIs suitable for inference about the ATT magnitude.

Value

A list of class `coresynth_inference`.

kalman_smoother_cpp *Kalman Filter and RTS Smoother (TASC)*

Description

Implements the Kalman filter (forward pass) and Rauch-Tung-Striebel smoother (backward pass) for the state-space model in Rho et al. (2026):

Usage

```
kalman_smoother_cpp(Y, W, A, C, Q, R, z0, P0)
```

Arguments

Y	Observed data matrix (N x T). Use NA for unobserved entries.
W	Observation / loading matrix (N x r)
A	State transition matrix (r x r). Pass diag(r) for random-walk dynamics.
C	State drift vector (r x 1)
Q	State noise covariance (r x r)
R	Observation noise covariance (N x N, diagonal in practice)
z0	Initial state mean (r x 1)
P0	Initial state covariance (r x r)

Details

State: $z(t+1) = A z(t) + C + \eta(t)$, $\eta(t) \sim N(0, Q)$ Observation: $y_t = W z_t + \epsilon_t$, $\epsilon_t \sim N(0, R)$

Observation rows with NA (treated post-intervention) are automatically dropped at each time step so only control-unit rows update the filter.

The P update uses the numerically stable Joseph form: $P(t|t) = (I - K W_{\text{obs}}) P(t|t-1) (I - K W_{\text{obs}})^T + K R_{\text{obs}} K^T$

Value

A list with z_smooth, P_smooth, P_cross, z_pred, z_upd. P_cross is an r x r x (T-1) cube. Slice t (C++ 0-indexed, t=0,...,T-2) stores P(t+1, t | T) (0-indexed), i.e. P(t+2, t+1 | T) in 1-indexed Shumway-Stoffer notation. Formula: $P(t+1|T) * J_t^T$ (eq. 6.68-6.69).

loo_donors

*Leave-One-Out Donor Robustness for SCM***Description**

Iteratively re-estimates the synthetic control excluding one contributing donor at a time, holding the predictor weights V fixed at their baseline values (Abadie, Diamond & Hainmueller 2015, footnote 20). The spread of the leave-one-out ATT estimates shows how much the result hinges on any single donor.

Usage

```
loo_donors(fit, weight_threshold = 1e-06)
```

Arguments

`fit` A sharp coresynth object from `scm_fit()` with method = "scm".

`weight_threshold` Only donors whose baseline weight exceeds this value are dropped (removing a zero-weight donor cannot change the fit). Default 1e-6.

Details

For penalised fits (`lambda_pen` used), the same penalty is re-applied in each leave-one-out QP.

Value

A list with:

- `att_original`: baseline ATT
- `results`: data.frame with one row per excluded donor (`donor`, `weight`, `att_loo`)
- `att_range`: range of the leave-one-out ATTs

See Also

`placebo_in_time()`, `mspe_ratio_pval()`

mspe_ratio_pval	<i>Permutation Inference via MSPE Ratio for SCM</i>
-----------------	---

Description

Computes the Abadie et al. (2010) / Abadie (2021) permutation p-value. For each control unit, a leave-one-out synthetic control is fitted.

Usage

```
mspe_ratio_pval(
  fit,
  mspe_threshold = 0,
  max_iter = 100L,
  tol = 1e-04,
  use_covariates = FALSE,
  alternative = c("two.sided", "greater", "less")
)
```

Arguments

<code>fit</code>	A coresynth object from <code>scm_fit()</code> with method = "scm".
<code>mspe_threshold</code>	Minimum pre-treatment MSPE for including a control unit in the two-sided test. Ignored for one-sided tests. Default: 0 (no filtering).
<code>max_iter</code>	Passed to <code>scm_placebo_cpp()</code> . Default 100L.
<code>tol</code>	Passed to <code>scm_placebo_cpp()</code> . Default 1e-4.
<code>use_covariates</code>	If TRUE and the fit used predictor covariates, applies the same covariate spec to each placebo unit (R-level loop). Default FALSE (faster C++ outcomes-only placebos).
<code>alternative</code>	Direction of the alternative hypothesis: "two.sided" (default) uses the MSPE ratio statistic; "greater" tests whether the treatment increased the outcome; "less" tests whether the treatment decreased the outcome. One-sided tests use the signed ATT as the test statistic.

Details

When `alternative = "two.sided"` (default), the test statistic is the post/pre MSPE ratio, following Abadie et al. (2010). When `alternative = "greater"` or `"less"`, the test statistic is the signed average post-treatment gap (ATT), giving a one-sided permutation test as recommended by Abadie (2021) S.3.5 for improved power when the direction of the treatment effect is known.

Value

An object of class `scm_placebo` (a list) with:

- `p_value`: Permutation p-value between 0 and 1

- `mspe_ratio_treated`: $MSPE_{post} / MSPE_{pre}$ for the treated unit (two.sided only)
- `mspe_ratios_all`: Named numeric vector (treated first, then controls); two.sided only
- `placebo_effects`: Named N_{co} -vector of placebo ATT estimates
- `treated_effect`: ATT estimate for the treated unit
- `n_placebo_used`: Number of control units used
- `gaps`: $T \times N_{co}$ matrix of placebo gap paths (unit minus its synthetic control over all periods), for the Abadie et al. (2010) Figure 4-7 plot
- `treated_gap`: T -vector of the treated unit's gap path
- `mspe_pre_treated`, `mspe_pre_placebo`: Pre-treatment MSPEs used for the relative pruning rule in `plot.scm_placebo()`
- `times`, `T_pre`: Time axis metadata for plotting

See Also

`plot.scm_placebo()` for the placebo gap and MSPE ratio plots.

placebo_in_time

In-Time Placebo (Backdating) Test for SCM

Description

Re-estimates the synthetic control after artificially backdating the treatment to a pre-treatment period, following Abadie, Diamond & Hainmueller (2015) and Abadie & Vives-i-Bastida (2022, principle 7: "out-of-sample validation is key"). Only pre-treatment data enter the exercise, so the placebo gap after `t0_placebo` is uncontaminated by the actual intervention. A credible design shows no sizable divergence at the backdated treatment time.

Usage

```
placebo_in_time(fit, t0_placebo = NULL)
```

Arguments

<code>fit</code>	A sharp <code>coresynth</code> object from <code>scm_fit()</code> with <code>method = "scm"</code> .
<code>t0_placebo</code>	Backdated treatment period as a 1-based position in <code>fit\$times</code> ; must satisfy $2 \leq t0_placebo < T_pre$. Default <code>floor(T_pre / 2)</code> .

Details

The refit uses the outcomes of periods $1 \dots t0_placebo$ as predictors (the `predictors = NULL` default), regardless of how the original fit was specified, because user-supplied `pred()` windows cannot be lagged automatically (ADH 2015 lag their predictors by hand).

Value

A list with:

- `t0_placebo`: the backdated treatment period used
- `times`: time values of the pre-treatment window
- `unit_weights`: placebo donor weights
- `Y_treat, Y_synth, gap`: series over the pre-treatment window
- `placebo_att`: mean placebo gap over $(t0_placebo, T_pre]$
- `fit_rmspe`: RMSPE over the placebo fitting window $1..t0_placebo$
- `eval_rmspe`: RMSPE over the placebo post window $(t0_placebo, T_pre]$

See Also

[`mspe\_ratio\_pval\(\)`](#) for in-space placebos, [`loo\_donors\(\)`](#) for donor-robustness checks.

<code>plot.coresynth</code>	<i>Plot a coresynth model</i>
-----------------------------	-------------------------------

Description

Plot a coresynth model

Usage

```
## S3 method for class 'coresynth'
plot(
  x,
  type = c("trend", "gap", "weights"),
  colors = NULL,
  labels = NULL,
  vline = list(),
  hline = list(),
  fill = NULL,
  top_n = Inf,
  ...
)
```

Arguments

<code>x</code>	A coresynth object.
<code>type</code>	One of "trend" (observed vs synthetic), "gap" (ATT over time), or "weights" (donor unit weight bar chart).
<code>colors</code>	For type = "trend": a named vector overriding series colors, e.g. <code>c(Treated = "black")</code> (valid names: "Treated", "Synthetic Control"). For type = "gap": a single color string for the gap line. Ignored for type = "weights" (use fill instead).

labels	For type = "trend": a named vector overriding the legend text of individual series, e.g. <code>c(Treated = "California")</code> (valid names: "Treated", "Synthetic Control"). Series not mentioned keep their default label, and colors keys always refer to the original series names regardless of relabeling. Ignored for other types (no legend).
vline	Aesthetic overrides for the vertical treatment-time line, as a list passed to <code>ggplot2::geom_vline()</code> (e.g. <code>list(color = "red")</code>). NULL or FALSE hides the line entirely. Applies to "trend" and "gap".
hline	Aesthetic overrides for the horizontal zero line in type = "gap", as a list passed to <code>ggplot2::geom_hline()</code> . NULL or FALSE hides the line. Ignored for other types.
fill	For type = "weights": a single color string overriding the bar fill. Ignored for other types.
top_n	For type = "weights": show only the top_n donors with the largest weights. The default Inf keeps every donor with a non-negligible weight. Ignored for other types.
...	Ignored.

Value

A ggplot2 plot object.

Examples

```
set.seed(1)
panel <- expand.grid(unit = 1:10, year = 1:20)
panel$treated <- as.integer(panel$unit == 5 & panel$year > 15)
panel$gdp <- panel$unit + 0.5 * panel$year +
  rnorm(nrow(panel)) + 3 * panel$treated
fit <- scm_fit(gdp ~ treated | unit + year, data = panel, method = "scm")

plot(fit, type = "trend")
plot(fit, type = "gap")
plot(fit, type = "weights")
plot(fit, type = "weights", top_n = 5)

# Customize series colors, legend text, and reference lines
plot(fit, type = "trend",
     colors = c(Treated = "black"),
     labels = c(Treated = "Unit 5"),
     vline = list(color = "red", linetype = "dashed"))
```

plot.scm_design	<i>Plot an scm_design object</i>
-----------------	----------------------------------

Description

Plot an scm_design object

Usage

```
## S3 method for class 'scm_design'
plot(x, type = c("outcome", "gap"), ...)
```

Arguments

x	An scm_design object.
type	"outcome" (default): synthetic treated vs synthetic control outcome series over all periods. "gap": estimated treatment effect in the experimental periods, with split-conformal confidence intervals.
...	Currently ignored.

Value

A ggplot object: for type = "outcome", the synthetic treated and synthetic control outcome series; for type = "gap", the estimated treatment effect over the experimental periods with split-conformal confidence intervals. The object is returned for printing or further customisation.

plot.scm_placebo	<i>Plot SCM In-Space Placebo Results</i>
------------------	--

Description

Visualizes the placebo study returned by `mspe_ratio_pval()`, following Abadie, Diamond & Hainmueller (2010, Section 3.4).

Usage

```
## S3 method for class 'scm_placebo'
plot(
  x,
  type = c("gaps", "ratios"),
  mspe_prune = Inf,
  colors = NULL,
  labels = NULL,
  vline = list(),
  hline = list(),
  ...
)
```

Arguments

<code>x</code>	A <code>scm_placebo</code> object from <code>mspe_ratio_pval()</code> .
<code>type</code>	One of "gaps" (ADH 2010, Figures 4-7) or "ratios" (ADH 2010, Figure 8).
<code>mspe_prune</code>	Only for <code>type = "gaps"</code> : exclude placebo units whose pre-treatment MSPE exceeds <code>mspe_prune</code> times the treated unit's. Default <code>Inf</code> (no pruning). A rule stated on the RMSPE scale, such as tidysynth's "2 times the treated unit's pre-period RMSPE", corresponds to the squared multiple (<code>mspe_prune = 4</code>).
<code>colors</code>	A named vector overriding series colors, e.g. <code>c(Treated = "black")</code> . Valid names: "Treated", "Placebo (donor pool)".
<code>labels</code>	A named vector overriding the legend text of individual series, e.g. <code>c(Treated = "California")</code> . Valid names: "Treated", "Placebo (donor pool)". Series not mentioned keep their default label, and <code>colors</code> keys always refer to the original series names regardless of relabeling.
<code>vline</code>	Only for <code>type = "gaps"</code> : aesthetic overrides for the vertical treatment-time line, as a list passed to <code>ggplot2::geom_vline()</code> . <code>NULL</code> or <code>FALSE</code> hides the line entirely.
<code>hline</code>	Only for <code>type = "gaps"</code> : aesthetic overrides for the horizontal zero line, as a list passed to <code>ggplot2::geom_hline()</code> . <code>NULL</code> or <code>FALSE</code> hides the line entirely.
<code>...</code>	Ignored.

Details

`type = "gaps"` overlays the treated unit's gap path (treated minus synthetic control) on the placebo gap paths obtained by reassigning the intervention to each donor unit (ADH 2010, Figure 4). Placebo units whose synthetic control fits poorly before treatment carry no information about the rarity of a large post-treatment gap, so ADH exclude units whose pre-treatment MSPE exceeds a multiple of the treated unit's: 20, 5, and 2 in their Figures 5-7 (`mspe_prune`).

`type = "ratios"` shows the post/pre-treatment MSPE ratio of every unit (ADH 2010, Figure 8), the statistic behind the two-sided permutation p-value; it requires no pruning cutoff by construction.

Value

A `ggplot2` plot object.

See Also

`mspe_ratio_pval()`

Examples

```
set.seed(1)
panel <- expand.grid(unit = 1:10, year = 1:20)
panel$treated <- as.integer(panel$unit == 5 & panel$year > 15)
panel$gdp <- panel$unit + 0.5 * panel$year +
  rnorm(nrow(panel)) + 3 * panel$treated
fit <- scm_fit(gdp ~ treated | unit + year, data = panel, method = "scm")
placebo <- mspe_ratio_pval(fit)
```



```
# Treated gap overlaid on the donor-pool placebo gaps (ADH 2010, Fig. 4)
plot(placebo, type = "gaps")

# Prune poorly fitting placebos and relabel the legend
plot(placebo, type = "gaps", mspe_prune = 5,
      labels = c(Treated = "Unit 5"))

# Post/pre-treatment MSPE ratios (ADH 2010, Fig. 8)
plot(placebo, type = "ratios")
```

pred	<i>Predictor Specification for SCM</i>
------	--

Description

Creates a single predictor specification for use in `scm_fit()` with `method = "scm"`. Pass a `list()` of `pred()` calls as the `predictors` argument to define the full covariate matrix.

Usage

```
pred(vars, times, op = "mean")
```

Arguments

<code>vars</code>	Character vector of variable names. All variables share the same times window and <code>op</code> operator. Use separate <code>pred()</code> calls for variables with different time windows.
<code>times</code>	Numeric/integer vector of time values to aggregate over.
<code>op</code>	Aggregation operator applied to each variable over times. One of "mean" (default), "median", or "sum".

Value

A `pred_spec` object (a named list with class "pred_spec").

See Also

`scm_fit()` for the `predictors` argument that consumes a `list()` of `pred_spec` objects.

Examples

```
# Three variables averaged over the same window
pred(c("lnincome", "retprice", "age15to24"), 1980:1988)

# Single variable at a specific year
pred("cigsale", 1975)

# Single variable averaged over a range
pred("beer", 1984:1988)

# Abadie, Diamond & Hainmueller (2010) California Prop 99 style: combine
# several covariates aggregated over different windows plus three outcome
# lags at specific years. The resulting list is passed to
# scm_fit(..., predictors = predictors).
predictors <- list(
  pred(c("lnincome", "retprice", "age15to24"), 1980:1988),
  pred("beer", 1984:1988),
  pred("cigsale", 1988),
  pred("cigsale", 1980),
  pred("cigsale", 1975)
)
predictors
```

scm_design

Experimental Synthetic Control Design

Description

Selects which units to assign to the treatment arm (and which to the control arm) in a planned experiment, following Abadie and Zhao (2026). Both sets of units are chosen by minimising the distance between their weighted-average predictor vectors and the population-average predictor vector $\bar{X}$, so the resulting estimates are less susceptible to post-randomisation bias than pure random assignment.

Usage

```
scm_design(
  data,
  outcome,
  unit,
  time,
  T0,
  T_fit = NULL,
  m_min = 1L,
  m_max = 1L,
  f = NULL,
  predictors = NULL,
  design = c("base", "weakly_targeted", "unit_level"),
```

```

    beta = 1,
    xi = 1,
    alpha = 0.05,
    normalize = TRUE,
    max_subsets = 100000L
)

```

Arguments

<code>data</code>	Long-format data frame (one row per unit–time).
<code>outcome</code>	Name of the outcome column.
<code>unit</code>	Name of the unit identifier column.
<code>time</code>	Name of the time identifier column.
<code>T0</code>	Last pre-experimental period (a value present in the time column). Periods after <code>T0</code> are the experimental periods.
<code>T_fit</code>	Number of fitting periods, counted from the start of the pre-experimental phase. Defaults to <code>NULL</code> , which uses all pre-experimental periods for fitting (no blank periods; inference disabled). When <code>T_fit</code> is smaller than the total number of pre-experimental periods, the remaining periods become blank periods used for inference.
<code>m_min</code>	Minimum number of units assigned to treatment (default 1).
<code>m_max</code>	Maximum number of units assigned to treatment (default 1).
<code>f</code>	Named numeric vector of population weights f_j . Defaults to uniform weights $1/J$. Will be normalised to sum to 1.
<code>predictors</code>	A <code>list()</code> of <code>pred()</code> specifications that define the predictor matrix X_j . Defaults to <code>NULL</code> , which uses all fitting-period outcome values as predictors.
<code>design</code>	Design formulation: "base" (default), "weakly_targeted", or "unit_level".
<code>beta</code>	Trade-off parameter $\beta > 0$ for the Weakly targeted design (default 1).
<code>xi</code>	Trade-off parameter $\xi > 0$ for the Unit-level design (default 1).
<code>alpha</code>	Significance level for confidence intervals (default 0.05).
<code>normalize</code>	If <code>TRUE</code> (default), each row of the predictor matrix is divided by its cross-unit standard deviation before optimisation, so predictors measured on different scales contribute equally.
<code>max_subsets</code>	Maximum number of treatment-set candidates to evaluate before switching to random sampling (default 100 000).

Details

Three design formulations are available:

- "base" (eq. 7): both the synthetic treated and the synthetic control independently target the population average $\bar{X}$.
- "weakly_targeted" (eq. 9): the treated and control weights jointly minimise the distance of the synthetic treated to $\bar{X}$ plus beta times the distance between the synthetic control and the synthetic treated.

- "unit_level" (eq. 10): each treated unit gets its own synthetic control; the treated weights trade off matching $\bar{X}$ against ξ times each treated unit's own synthetic-control fit, and the aggregate control weight is the w -weighted combination of the per-unit controls (eq. 11).

Inference uses "blank periods" — pre-experimental periods whose outcomes were *not* used to estimate the weights. Set `T_fit` strictly smaller than the number of pre-experimental periods to enable the permutation test and split- conformal confidence intervals from Section 3 of Abadie and Zhao (2026).

Value

An object of class "scm_design" with components:

- `treated_units`: unit identifiers selected for treatment
- `control_units`: unit identifiers in the control pool
- `w`: J -length weight vector for the synthetic treated unit (sums to 1)
- `v`: J -length weight vector for the synthetic control unit (sums to 1)
- `tau_hat`: estimated treatment effects for each experimental period
- `p_value`: permutation p-value (NA when blank periods are unavailable)
- `ci_lower`, `ci_upper`: per-period split-conformal confidence interval
- `Y_synth_tr`, `Y_synth_co`: synthetic treated/control series (all periods)
- `estimate`: ATT (mean of `tau_hat`)

References

Abadie, A. and Zhao, J. (2026). "Synthetic Controls for Experimental Design." MIT Working Paper.

scm_fit

Fit a Synthetic Control Method Model

Description

Unified formula interface for Synthetic Control and related causal inference methods. The formula syntax is:

Usage

```
scm_fit(
  formula,
  data,
  method = c("scm", "sdid", "gsc", "mc", "tasc", "si"),
  predictors = NULL,
  covariates = NULL,
  v_selection = c("insample", "oos"),
  donor_mspe_threshold = Inf,
  lambda_pen = NULL,
```

```

v_optim = c("coord_descent", "auto", "bfgs"),
...
)

```

Arguments

formula	A Formula object, e.g. $y \sim D \mid \text{unit} + \text{time}$.
data	A data.frame in long format (one row per unit-time).
method	One of "scm", "sdid", "gsc", "mc", "tasc", "si".
predictors	A list() of <code>pred()</code> specifications that define the predictor matrix for SCM (see Abadie et al. 2010, S.2.3). Each <code>pred()</code> entry aggregates one or more variables over a time window. Pass NULL (default) to use all pre-treatment outcome periods as predictors. Applies to method = "scm" only. Predictor rows are scaled by their standard deviation across all units before optimisation, matching the Synth reference implementation (ADH 2011, JSS); pass <code>scale_predictors = FALSE</code> to disable.
covariates	An optional named list of additional time-varying covariates to partial out before estimation. Each element is a character string naming a column in data. Supported for method = "sdid", "scm", and "gsc".
v_selection	V matrix selection method for method = "scm". "insample" (default) follows Abadie et al. (2010): V is chosen by minimising in-sample pre-treatment MSPE. "oos" follows Abadie (2021) S.3.2 / ADH (2015): the pre-treatment window is split into a training half and a validation half. In the default outcomes-as-predictors case, candidate $W(V)$ are fitted on training-half outcomes only, V^* minimises the validation-half MSPE, and the final W^* is refit with V^* on the outcomes of the last $\text{floor}(T_{\text{pre}}/2)$ pre-treatment periods (so <code>v_weights</code> has $\text{floor}(T_{\text{pre}}/2)$ entries). With user-supplied predictors, the predictor matrix is fixed and only the MSPE evaluation window is restricted to the validation half; lag your <code>pred()</code> windows to the training period for a fully out-of-sample exercise.
donor_mspe_threshold	Donor pool filtering threshold (Abadie 2021 S.4). For method = "scm" only. Each donor's individual pre-treatment MSPE (using that donor alone as the counterfactual) is divided by the minimum such MSPE across all donors. Donors whose ratio exceeds this threshold are excluded from estimation. Inf (default) disables filtering.
lambda_pen	Penalised SCM parameter (Abadie & L'Hour 2021, JASA). For method = "scm" only. NULL (default) runs standard unpenalised SCM. "auto" selects the penalty via out-of-sample pre-treatment MSPE on the same validation window as <code>v_selection</code> = "oos". A non-negative number uses that value directly.
v_optim	Outer V-optimisation method for method = "scm". "coord_descent" (default) uses the existing C++ coordinate descent with 11-point grid search – fastest when $k = T_{\text{pre}}$ is large (outcomes-only). "bfgs" uses R's L-BFGS-B, which requires only $O(k^2)$ inner QP calls and is faster when k is small (e.g. external predictors with $k \leq 15$). "auto" selects "bfgs" when $k \leq 15$, otherwise "coord_descent".
...	Additional arguments forwarded to the specific method (e.g. <code>r</code> , <code>lambda</code> , <code>zeta2</code>).

Details

```
outcome ~ treatment | unit_id + time_id
```

Value

An object of classes `c("coresynth_<method>", "coresynth")`. Fits with staggered adoption additionally inherit from `"coresynth_staggered"`, and multi-arm SI fits from `"coresynth_multiarm"`; S3 methods such as `tidy()` and `augment()` dispatch on these subclasses. All methods return at minimum:

- `method`: estimator name
- `estimate`: average treatment effect (ATT)
- `times`: time index vector
- `T_pre`: number of pre-treatment periods
- `Y_treat`: treated unit outcome series
- `gap`: treatment effect series (`Y_treat` - counterfactual)

Examples

```
# Synthetic balanced panel: 10 units over 20 periods, unit 1 treated
# after period 15.
set.seed(1)
panel <- expand.grid(unit = 1:10, year = 1:20)
panel$treated <- as.integer(panel$unit == 1 & panel$year > 15)
panel$gdp <- panel$unit + 0.5 * panel$year +
  rnorm(nrow(panel)) + 3 * panel$treated

fit <- scm_fit(gdp ~ treated | unit + year, data = panel, method = "sdid")
summary(fit)

# Visualise the estimated gap (requires ggplot2)
plot(fit, type = "gap")
```

scm_inner_weights_cpp *SCM Inner Weights (QP Given V)*

Description

Solves the inner-loop QP for SCM: given a fixed diagonal metric matrix V , finds donor weights W on the simplex minimising the V -weighted covariate loss.

Usage

```
scm_inner_weights_cpp(X0, X1, V_diag)
```

Arguments

X_0	Covariate matrix for control units ($k \times N_{co}$)
X_1	Covariate vector for the treated unit ($k \times 1$)
V_{diag}	Diagonal of the metric matrix V ($k \times 1$, non-negative, need not sum to 1)

Value

Donor weight vector W ($N_{co} \times 1$) on the unit simplex

scm_placebo_cpp	<i>Fast Leave-One-Out Placebo Test for SCM (Abadie et al. 2010)</i>
-----------------	---

Description

For each control unit, treats it as pseudo-treated and fits SCM weights from the remaining $N_{co}-1$ donors. Returns MSPE components for constructing MSPE-ratio permutation p-values in R.

Usage

```
scm_placebo_cpp(Y_pre, Y_post, max_iter = 100L, tol = 1e-04)
```

Arguments

Y_{pre}	Control pre-treatment outcomes ($T_{pre} \times N_{co}$)
Y_{post}	Control post-treatment outcomes ($T_{post} \times N_{co}$)
<code>max_iter</code>	Outer coordinate-descent iterations (default 100)
<code>tol</code>	Convergence tolerance for V updates (default $1e-4$)

Value

A list with:

- `mspe_pre`: N_{co} -vector of pre-treatment MSPE per placebo unit
- `mspe_post`: N_{co} -vector of post-treatment MSPE per placebo unit
- `effects`: N_{co} -vector of mean post-period gap per placebo unit
- `gaps`: $(T_{pre} + T_{post}) \times N_{co}$ matrix of placebo gap paths

scm_weights_cpp

*SCM Outer Weights (Joint Optimization of W and V)***Description**

Jointly optimises donor weights W (on the simplex) and the diagonal metric matrix V via coordinate descent on the pre-treatment prediction MSPE, following Abadie, Diamond & Hainmueller (2010).

Usage

```
scm_weights_cpp(X0, X1, Z0, Z1, max_iter = 100L, tol = 1e-04, t_train = -1L)
```

Arguments

<code>X0</code>	Covariate matrix for control units ($k \times N_{co}$, typically pre-treatment outcomes)
<code>X1</code>	Covariate vector for the treated unit ($k \times 1$)
<code>Z0</code>	Outcome matrix for control units in the pre-treatment window ($T_{pre} \times N_{co}$)
<code>Z1</code>	Outcome vector for the treated unit in the pre-treatment window ($T_{pre} \times 1$)
<code>max_iter</code>	Maximum coordinate-descent iterations (default 100)
<code>tol</code>	Convergence tolerance on MSPE improvement (default $1e-4$)
<code>t_train</code>	Validation-window split for V selection. -1 (default): V selected on the full Z window (in-sample). Positive: rows $t_train..(T_{pre}-1)$ of Z form the validation window used to select V (W is fitted on the full X throughout); after selecting V^* , W is refit and the reported loss uses the full Z window.

Details

When $t_train > 0$, V is selected by minimising MSPE on a validation window (rows $t_train..T_{pre}-1$ of Z) while W is fitted on the full predictor matrix X . This is appropriate when X is a fixed predictor matrix that contains no validation-period outcome information (the user-supplied predictors case). For the outcomes-only case the proper Abadie (2021) S.3.2 train/validation split is implemented in R (`.scm_oos_outcomes()`): candidate $W(V)$ are fitted on training-half outcomes only, by passing the training rows as X and the validation rows as Z to this function with $t_train = -1$.

Value

A list with:

- W : Donor weight vector ($N_{co} \times 1$) on the unit simplex
- V : Optimal metric diagonal ($k \times 1$, normalised to sum to 1)
- $loss$: Final pre-treatment prediction loss (full pre-treatment window)

sdid_estimate_cpp	<i>Calculate SDID Estimate (tau_sdid)</i>
-------------------	---

Description

Given unit weights omega and time weights lambda, computes the SDID estimator as a weighted two-way difference:

Usage

sdid_estimate_cpp(Y_pre_co, Y_post_co, Y_pre_tr, Y_post_tr, omega, lambda)

Arguments

Y_pre_co	Control pre-treatment outcomes (T_pre x N_co)
Y_post_co	Control post-treatment outcomes (T_post x N_co)
Y_pre_tr	Treated pre-treatment outcomes (T_pre x 1)
Y_post_tr	Treated post-treatment outcomes (T_post x 1)
omega	Unit weights (N_co x 1)
lambda	Time weights (T_pre x 1)

Details

$$\tau_{sdid} = (Y_{tr_post_mean} - Y_{tr_pre_wt}) - (Y_{co_post_wt} - Y_{co_pre_wt})$$

Value

A single numeric value: the SDID treatment-effect estimate tau_sdid.

sdid_inference	<i>Inference for Synthetic Difference-in-Differences</i>
----------------	--

Description

Computes standard errors and p-values for a SDID estimate using one of three methods: permutation placebo test (Algorithm 4), cluster bootstrap (Algorithm 2), or leave-one-out jackknife (Algorithm 3), following Clarke et al. (2023).

Usage

```
sdid_inference(
  fit,
  method = c("placebo", "bootstrap", "jackknife", "jackknife_global"),
  n_boot = 200L,
  level = 0.95,
  alternative = c("two.sided", "greater", "less"),
  seed = NULL
)
```

Arguments

<code>fit</code>	A coresynth object with <code>method = "sdid"</code> (sharp adoption only).
<code>method</code>	Inference method: "placebo" (permutation), "bootstrap", or "jackknife".
<code>n_boot</code>	Number of bootstrap replications (only for <code>method = "bootstrap"</code>).
<code>level</code>	Confidence level for the interval (only for <code>method = "bootstrap"</code> or <code>"jackknife"</code>).
<code>alternative</code>	Direction of the alternative hypothesis: "two.sided", "greater", or "less".
<code>seed</code>	Integer seed for reproducibility (only for <code>method = "bootstrap"</code>).

Value

A list with:

- `estimate`: The SDID point estimate.
- `se`: Standard error (bootstrap / jackknife only).
- `p_value`: Permutation or normal-approximation p-value.
- `ci_lower`, `ci_upper`: Confidence interval bounds (bootstrap / jackknife).
- `method`: The inference method used.
- `n_controls`: Number of control units.
- `alternative`: The alternative hypothesis direction.
- `placebo_effects`: Named vector of LOO placebo effects (placebo only).
- `boot_ests`: Bootstrap estimate distribution (bootstrap only).

sdid_placebo_cpp

Fast Placebo Test for SDID

Description

For each control unit, treats it as the "pseudo-treated" unit and estimates the leave-one-out SDID effect. The distribution of these placebo effects provides a permutation-based null distribution for inference.

Usage

```
sdid_placebo_cpp(Y_pre, Y_post, time_weights, zeta2)
```

Arguments

<code>Y_pre</code>	Control units pre-treatment outcomes ($T_{pre} \times N_{co}$)
<code>Y_post</code>	Control units post-treatment outcomes ($T_{post} \times N_{co}$)
<code>time_weights</code>	Lambda weights for pre-treatment periods ($T_{pre} \times 1$)
<code>zeta2</code>	Ridge penalty (same as used in the main estimate)

Value

A numeric vector of length N_{co} . Each element is the leave-one-out placebo SDID effect obtained by treating that control unit as the pseudo-treated unit; the vector serves as a permutation-based null distribution for inference.

`sdid_time_weights_cpp` *Calculate SDID Time Weights (lambda)*

Description

Solves the time-weight QP (with implicit intercept λ_0 concentrated out):

Usage

```
sdid_time_weights_cpp(Y_pre_co, Y_post_target, zeta_t)
```

Arguments

<code>Y_pre_co</code>	Pre-treatment outcomes for control units, row-demeaned ($T_{pre} \times N_{co}$)
<code>Y_post_target</code>	Post-treatment mean per control unit, demeaned ($N_{co} \times 1$)
<code>zeta_t</code>	Ridge penalty for time weights (paper: $1e-6 * \sigma_{hat}$)

Details

min over λ in Δ_{pre} : $\|Y_{post_target} - Y_{pre_co}^T \lambda\|^2 + zeta_t^2 * N_{co} * \|\lambda\|^2$

The caller is responsible for pre-demeaning `Y_pre_co` (row-wise) and `Y_post_target` (subtract the cross-unit mean) to concentrate out λ_0 , as described in Arkhangelsky et al. (2021) Algorithm 1, Eq. (2.3).

Value

A numeric vector of length T_{pre} holding the SDID time weights λ (non-negative and summing to one).

sdid_unit_weights_cpp *Calculate SDID Unit Weights (omega)*

Description

Solves the regularized QP: min over omega in Delta: $\sum_t (\sum_i \omega_i Y_{it} - Y_{tr_t})^2 + \zeta^2 * T_{pre} * \|\omega\|^2$

Usage

```
sdid_unit_weights_cpp(Y_pre, Y_tr_pre, zeta2)
```

Arguments

Y_pre	Pre-treatment outcome matrix for control units ($T_{pre} \times N_{co}$)
Y_tr_pre	Pre-treatment outcome vector for treated unit ($T_{pre} \times 1$), averaged if multiple
zeta2	Ridge penalty parameter (ζ^2). The code internally multiplies by T_{pre} per the paper.

Details

This corresponds to equation (5) in Arkhangelsky et al. (2021).

Value

A numeric vector of length N_{co} holding the SDID unit weights omega (non-negative and summing to one).

si_inference *Non-parametric Inference for SI (Agarwal et al. 2025)*

Description

Estimates SE and confidence intervals for the ATT via non-parametric cluster bootstrap or jackknife over control units. Works for both sharp and staggered SI fits. For staggered fits, bootstrap resamples each cohort's control pool independently, and jackknife uses a per-cohort LOO with delta-method variance aggregation.

Usage

```
si_inference(
  fit,
  method = c("bootstrap", "jackknife", "jackknife_global"),
  n_boot = 499L,
  level = 0.95,
  alternative = c("two.sided", "greater", "less"),
  seed = NULL
)
```

Arguments

fit	A coresynth object from <code>scm_fit()</code> with method = "si".
method	"bootstrap" (default) or "jackknife".
n_boot	Number of bootstrap replications (default 499L; ignored for jackknife).
level	Confidence level (default 0.95).
alternative	"two.sided" (default), "greater", or "less".
seed	RNG seed for reproducibility (default NULL).

Value

A list of class `coresynth_inference`.

si_pcr_cpp	<i>SI-PCR: Synthetic Interventions via Principal Component Regression</i>
------------	---

Description

Implements the SI-PCR estimator of Agarwal et al. (2025). Uses the top-k SVD of pre-treatment control outcomes to find donor weights that predict each treated unit's pre-treatment trajectory, then applies those weights to post-treatment control outcomes.

Usage

```
si_pcr_cpp(Y_pre_co, Y_post_co, Y_pre_tr, k)
```

Arguments

Y_pre_co	Pre-treatment control outcomes ($T_{pre} \times N_{co}$)
Y_post_co	Post-treatment control outcomes ($T_{post} \times N_{co}$)
Y_pre_tr	Pre-treatment treated outcomes ($T_{pre} \times N_{tr}$)
k	Number of SVD components to retain

Value

A list with:

- W: Donor weight matrix ($N_{co} \times N_{tr}$)
- Y_hat: Counterfactual post-treatment outcomes ($T_{post} \times N_{tr}$)

soft_impute_cpp

Fast Matrix Completion using Soft-Impute Algorithm

Description

Solves: $\min_L (1/2) \|O - (Y - L)\|_F^2 + \lambda \|L\|_*$ via iterative SVD soft-thresholding (Mazumder, Hastie, Tibshirani 2010). Note: λ is NOT normalized by $|O|$. Default $\lambda = 0.01 * \sigma_{\max}(Y)$.

Usage

```
soft_impute_cpp(Y, O, lambda, max_iter = 1000L, tol = 1e-05)
```

Arguments

Y	Observed outcome matrix (N x T). Unobserved entries should be 0.
O	Binary mask matrix (N x T): 1 = observed, 0 = missing (treated post).
lambda	Nuclear norm penalty (soft-threshold on singular values).
max_iter	Maximum iterations.
tol	Convergence tolerance (relative Frobenius norm change).

Value

A numeric matrix of the same dimension as Y (N x T): the completed low-rank matrix L that minimises the soft-thresholded nuclear-norm objective.

tensor_unfold_cpp

Tensor Unfolding (Matricization) for Synthetic Interventions

Description

Tensor Unfolding (Matricization) for Synthetic Interventions

Usage

```
tensor_unfold_cpp(T_cube, mode)
```

Arguments

T_cube	A 3D array (cube) of dimensions (n1, n2, n3)
mode	The mode to unfold along (1, 2, or 3)

Value

A numeric matrix: the mode-mode unfolding (matricization) of T_cube, with dimensions $n_1 \times (n_2 * n_3)$, $n_2 \times (n_1 * n_3)$, or $n_3 \times (n_1 * n_2)$ for mode 1, 2, or 3 respectively.

```
tidy.coresynth_inference
```

Tidy an inference result

Description

Coerces a `coresynth_inference` or `sdid_inference` object to a one-row tidy data.frame with broom-style column names so it can be combined with regression output for paper tables.

Usage

```
## S3 method for class 'coresynth_inference'
tidy(x, conf.int = TRUE, ...)
```

Arguments

<code>x</code>	A <code>coresynth_inference</code> (or <code>sdid_inference</code>) object returned by <code>sdid_inference()</code> , <code>gsc_inference()</code> , or <code>si_inference()</code> .
<code>conf.int</code>	Logical. Include <code>conf.low</code> / <code>conf.high</code> columns when CI is available (default TRUE). Permutation placebo SE/CI are NA.
<code>...</code>	Unused.

Value

A one-row data.frame with columns `term`, `estimate`, `std.error`, `statistic`, `p.value`, `conf.low`, `conf.high`, `method`, `alternative`, `n_controls`, `staggered`.

```
treated_outcomes
```

Extract Outcome Series from a coresynth Fit

Description

Accessor generics that return the outcome series stored in a fitted `coresynth` object under a uniform interface, regardless of the estimation method:

Usage

```
treated_outcomes(x, ...)

## S3 method for class 'coresynth'
treated_outcomes(x, na.rm = FALSE, ...)

synthetic_outcomes(x, ...)

## S3 method for class 'coresynth'
```

```

synthetic_outcomes(x, na.rm = FALSE, ...)

## S3 method for class 'coresynth_tasc'
synthetic_outcomes(x, na.rm = FALSE, ...)

donor_outcomes(x, ...)

## S3 method for class 'coresynth_scm'
donor_outcomes(x, ...)

## S3 method for class 'coresynth_sdid'
donor_outcomes(x, ...)

## S3 method for class 'coresynth_si'
donor_outcomes(x, ...)

## S3 method for class 'coresynth_gsc'
donor_outcomes(x, ...)

## S3 method for class 'coresynth_mc'
donor_outcomes(x, ...)

## S3 method for class 'coresynth_tasc'
donor_outcomes(x, ...)

## S3 method for class 'coresynth'
donor_outcomes(x, ...)

```

Arguments

<code>x</code>	A coresynth object from <code>scm_fit()</code> .
<code>...</code>	Passed to methods.
<code>na.rm</code>	Logical; passed to the per-period averaging over multiple treated units (default FALSE).

Details

- `treated_outcomes()`: the treated unit's observed outcome series (length T). When several units are treated, their per-period mean.
- `synthetic_outcomes()`: the estimated counterfactual series (length T), i.e. the synthetic control or model-fitted outcome.
- `donor_outcomes()`: the $T \times N_{co}$ matrix of observed donor (control unit) outcomes over all periods.

Each accessor returns NULL when the requested series is not stored in the fit. In particular, staggered-adoption fits keep their data per cohort (in `fit$cohort_fits`), so the sharp-fit accessors return NULL for them.

Value

For `treated_outcomes()` and `synthetic_outcomes()`, a numeric vector of length T , or NULL.
For `donor_outcomes()`, a $T \times N_{co}$ numeric matrix (donors in columns, named when unit names are available), or NULL.

Examples

```
set.seed(1)
panel <- expand.grid(unit = 1:10, year = 1:20)
panel$treated <- as.integer(panel$unit == 1 & panel$year > 15)
panel$gdp <- panel$unit + 0.5 * panel$year +
  rnorm(nrow(panel)) + 3 * panel$treated
fit <- scm_fit(gdp ~ treated | unit + year, data = panel, method = "scm")

y1 <- treated_outcomes(fit) # observed treated series
y1_0 <- synthetic_outcomes(fit) # synthetic counterfactual
Yco <- donor_outcomes(fit) # donor outcome matrix
all.equal(y1 - y1_0, unname(fit$gap))
```

Index

`augment()`, 22
`augment_scm`, 2

`conformal_inference`, 3

`donor_outcomes (treated_outcomes)`, 31

`export_json`, 5

`ggplot2::geom_hline()`, 14, 16
`ggplot2::geom_vline()`, 14, 16
`glance()`, 4
`glance.coresynth_inference`, 5
`gsc_boot`, 6
`gsc_boot()`, 5
`gsc_ife_cpp`, 7
`gsc_inference`, 8
`gsc_inference()`, 31

`kalman_smoother_cpp`, 9

`loo_donors`, 10
`loo_donors()`, 13

`mspe_ratio_pval`, 11
`mspe_ratio_pval()`, 5, 10, 13, 15, 16

`placebo_in_time`, 12
`placebo_in_time()`, 10
`plot.coresynth`, 13
`plot.scm_design`, 15
`plot.scm_placebo`, 15
`plot.scm_placebo()`, 12
`pred`, 17
`pred()`, 19, 21

`scm_design`, 18
`scm_fit`, 20
`scm_fit()`, 3–6, 8, 10–12, 17, 29, 32
`scm_inner_weights_cpp`, 22
`scm_placebo_cpp`, 23

`scm_placebo_cpp()`, 11
`scm_weights_cpp`, 24
`sdid_estimate_cpp`, 25
`sdid_inference`, 25
`sdid_inference()`, 31
`sdid_placebo_cpp`, 26
`sdid_time_weights_cpp`, 27
`sdid_unit_weights_cpp`, 28
`si_inference`, 28
`si_inference()`, 31
`si_pcr_cpp`, 29
`soft_impute_cpp`, 30
`synthetic_outcomes (treated_outcomes)`, 31

`tensor_unfold_cpp`, 30
`tidy()`, 4, 22
`tidy.coresynth_inference`, 31
`treated_outcomes`, 31