Package: coresynth (via r-universe)

June 13, 2026

Title Fast and Unified Synthetic Control Methods

Version 0.2.1

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>, Arkhangelsky, Athey, Hirshberg, Imbens and Wager (2021) <doi:10.1257/aer.20190159>, Xu (2017) <doi:10.1017/pan.2016.2>, Athey, Bayati, Doudchenko, Imbens and Khosravi (2021) <doi:10.1080/01621459.2021.1891924>, and Agarwal, Shah and Shen (2025) <doi: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

Config/pak/sysreqs libicu-dev

Repository https://yo5uke.r-universe.dev

Date/Publication 2026-06-11 11:38:26 UTC

RemoteUrl https://github.com/yo5uke/coresynth

RemoteRef HEAD

RemoteSha 79aed6d8fd2c75f5a7954d16ffa3c5b04db686ce

Contents

augment_scm *Augmented Synthetic Control Method (Ridge ASCM)*

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

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

Details

tau_aug = tau_SCM + (m_tr_post - sum_j W_j * m_j_post)

where m_i_post = Y_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 (m_tr_post - sum_j W_j 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 *Conformal Inference for Synthetic Control Estimators*

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

fit	A coresynth object from scm_fit().
tau0	Null value of the ATT for the reported p-value (default 0).
q	Exponent of the S_q test statistic (S_q = (T_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.
alternative	"two.sided" (default), "greater", or "less".
ci	Logical; construct a confidence interval by test inversion (default TRUE).
level	Confidence level for the interval (default 0.95).
grid	Optional numeric vector of candidate τ_0 values for test inversion. When NULL (default), a grid of n_grid points is centred on the point estimate with half-width grid_mult times the pre-treatment residual standard deviation.
n_grid	Number of grid points when grid = NULL (default 200).
grid_mult	Half-width multiplier when grid = NULL (default 4).
...	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` *Export coresynth Results to JSON*

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 `scm_fit()`.
`file`	Output file path. Default `"coresynth_results.json"`. Pass `NULL` to skip writing and return the R list invisibly.
`inference`	Optional list from `mspe_ratio_pval()` or `gsc_boot()`. When provided, populates the `inference` section and updates `estimate` with `p_value`, `se`, `ci_lower`, `ci_upper`.
`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` *Glance at an inference result*

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 scm_fit() 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: mean(|ATT*| >= |ATT_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

`gsc_ife_cpp` *Fast Interactive Fixed Effects (IFE) for Generalized Synthetic Control*

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 beta 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

`Y_co`	Control units outcome matrix (T x N_co)
`Y_tr_pre`	Treated units pre-treatment outcomes (T_pre x N_tr)
`r`	Number of latent factors (must be <= min(T, N_co))
`X_co`	Time-varying covariate cube (T x N_co x p). Pass an empty cube (0 slices) for the covariate-free estimator.
`X_tr_pre`	Time-varying covariate cube for treated units in the pre-treatment window (T_pre x N_tr x p). Required for correct Step 2 loading estimation per Xu (2017): lambda_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 != 0).
`max_iter`	Maximum EM iterations (default 50)
`tol`	Convergence tolerance on relative beta change (default 1e-6)

Value

A list with components:

- `F`: estimated time factors (T x r).
- `L_co`: control-unit factor loadings (N_co x r).
- `L_tr`: treated-unit factor loadings (N_tr x r).
- `Y_tr_hat`: estimated treated-unit counterfactual outcomes (T x N_tr).
- `singular_values`: singular values from the final truncated SVD.
- `beta`: estimated covariate coefficients (p x 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 scm_fit() 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 H0 (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) ~ N(0, Q) Observation: y_t = W z_t + eps_t, eps_t ~ 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_obs) P(t|t-1) (I - K W_obs)^T + K R_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 *Permutation Inference via MSPE Ratio for SCM*

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

fit	A coresynth object from scm_fit() with method = "scm".
mspe_threshold	Minimum pre-treatment MSPE for including a control unit in the two-sided test. Ignored for one-sided tests. Default: 0 (no filtering).
max_iter	Passed to scm_placebo_cpp(). Default 100L.
tol	Passed to scm_placebo_cpp(). Default 1e-4.
use_covariates	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).
alternative	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

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

`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

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

Details

The refit uses the outcomes of periods `1..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.

---

plot.coresynth *Plot a coresynth model*

Description

Plot a coresynth model

Usage

```
## S3 method for class 'coresynth'
plot(x, type = c("trend", "gap", "weights"), ...)
```

Arguments

x	A coresynth object.
type	One of "trend" (observed vs synthetic), "gap" (ATT over time), or "weights" (donor unit weight bar chart).
...	Ignored.

Value

A ggplot2 plot object.

---

plot.scm_design *Plot an scm_design object*

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.

---

pred *Predictor Specification for SCM*

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

vars	Character vector of variable names. All variables share the same times window and op operator. Use separate pred() calls for variables with different time windows.
times	Numeric/integer vector of time values to aggregate over.
op	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

data	Long-format data frame (one row per unit–time).
outcome	Name of the outcome column.
unit	Name of the unit identifier column.

`time`	Name of the time identifier column.
`T0`	Last pre-experimental period (a value present in the time column). Periods after `T0` are the experimental periods.
`T_fit`	Number of fitting periods, counted from the **start** of the pre-experimental phase. Defaults to `NULL`, which uses all pre-experimental periods for fitting (no blank periods; inference disabled). When `T_fit` is smaller than the total number of pre-experimental periods, the remaining periods become blank periods used for inference.
`m_min`	Minimum number of units assigned to treatment (default 1).
`m_max`	Maximum number of units assigned to treatment (default 1).
`f`	Named numeric vector of population weights f_j. Defaults to uniform weights $1/J$. Will be normalised to sum to 1.
`predictors`	A `list()` of `pred()` specifications that define the predictor matrix X_j. Defaults to `NULL`, which uses all fitting-period outcome values as predictors.
`design`	Design formulation: `"base"` (default), `"weakly_targeted"`, or `"unit_level"`.
`beta`	Trade-off parameter $\beta > 0$ for the Weakly targeted design (default 1).
`xi`	Trade-off parameter $\xi > 0$ for the Unit-level design (default 1).
`alpha`	Significance level for confidence intervals (default 0.05).
`normalize`	If `TRUE` (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.
`max_subsets`	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 synthetic treated targets $\bar{X}$; the synthetic control targets the synthetic treated predictor vector (controlled by `beta`).
- `"unit_level"` (eq. 10): each treated unit gets its own synthetic control; the aggregate control weight is a convex combination (controlled by `xi`).

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 ~ D | unit + 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 pred() specifications that define the predictor matrix for SCM (see Abadie et al. 2010, S.2.3). Each pred() 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 scale_predictors = FALSE 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 floor(T_pre/2) pre-treatment periods (so v_weights has floor(T_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 pred() 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 v_selection = "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_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 <= 15). "auto" selects "bfgs" when k <= 15, otherwise "coord_descent".

... — Additional arguments forwarded to the specific method (e.g. r, lambda, zeta2).

Details

outcome ~ treatment | unit_id + time_id

Value

An object of classes c("coresynth_<method>", "coresynth"). 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

X0	Covariate matrix for control units (k x N_co)
X1	Covariate vector for the treated unit (k x 1)
V_diag	Diagonal of the metric matrix V (k x 1, non-negative, need not sum to 1)

Value

Donor weight vector W (N_co x 1) on the unit simplex

scm_placebo_cpp *Fast Leave-One-Out Placebo Test for SCM (Abadie et al. 2010)*

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 x N_co)
`Y_post`	Control post-treatment outcomes (T_post x N_co)
`max_iter`	Outer coordinate-descent iterations (default 100)
`tol`	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

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

X0	Covariate matrix for control units (k x N_co, typically pre-treatment outcomes)
X1	Covariate vector for the treated unit (k x 1)
Z0	Outcome matrix for control units in the pre-treatment window (T_pre x N_co)
Z1	Outcome vector for the treated unit in the pre-treatment window (T_pre x 1)
max_iter	Maximum coordinate-descent iterations (default 100)
tol	Convergence tolerance on MSPE improvement (default 1e-4)
t_train	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 x 1) on the unit simplex
- `V`: Optimal metric diagonal (k x 1, normalised to sum to 1)
- `loss`: Final pre-treatment prediction loss (full pre-treatment window)

---

sdid_estimate_cpp *Calculate SDID Estimate (tau_sdid)*

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 *Inference for Synthetic Difference-in-Differences*

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

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

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

`Y_pre`	Control units pre-treatment outcomes (T_pre x N_co)
`Y_post`	Control units post-treatment outcomes (T_post x N_co)
`time_weights`	Lambda weights for pre-treatment periods (T_pre x 1)
`zeta2`	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 lambda_0 concentrated out):

Usage

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

Arguments

Y_pre_co	Pre-treatment outcomes for control units, row-demeaned (T_pre x N_co)
Y_post_target	Post-treatment mean per control unit, demeaned (N_co x 1)
zeta_t	Ridge penalty for time weights (paper: 1e-6 * sigma_hat)

Details

min over lambda in Delta_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 lambda_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 lambda (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 x N_co)
Y_tr_pre	Pre-treatment outcome vector for treated unit (T_pre x 1), averaged if multiple
zeta2	Ridge penalty parameter (zeta^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 scm_fit() 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` *SI-PCR: Synthetic Interventions via Principal Component Regression*

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 x N_co)
`Y_post_co`	Post-treatment control outcomes (T_post x N_co)
`Y_pre_tr`	Pre-treatment treated outcomes (T_pre x N_tr)
`k`	Number of SVD components to retain

Value

A list with:

- `W`: Donor weight matrix (N_co x N_tr)
- `Y_hat`: Counterfactual post-treatment outcomes (T_post x N_tr)

`soft_impute_cpp` *Fast Matrix Completion using Soft-Impute Algorithm*

Description

Solves: min_L (1/2) ||O o (Y - L)||*F*^2 + *lambda* * ||*L*||* via iterative SVD soft-thresholding (Mazumder, Hastie, Tibshirani 2010). Note: lambda 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 n1 x (n2 * n3), n2 x (n1 * n3), or n3 x (n1 * n2) 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

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

Index