Package 'postshock'

July 27, 2026

Type Package

Title Donor-Adjusted Post-Shock Forecasting

Version 0.2.0

Description Implements donor-adjusted methods for forecasting conditional means and variances after structural shocks. Historical donor episodes are weighted using covariates observed before each shock, and their estimated post-shock effects are combined with forecasts from a target-series model. The methods build on Lin and Eck (2021) <doi:10.1016/j.ijforecast.2021.03.010>. The package supports donor balancing weights, structured donor pools, autoregressive integrated moving average models, and generalized autoregressive conditional heteroscedasticity models with external regressors.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Depends R (>= 4.1.0)

Imports Rsolnp, garchx, forecast, lmtest, xts, zoo

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

LazyData true

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Qiyang Wang [aut, cre] (ORCID: <https://orcid.org/0009-0002-6373-440X>), Daniel J. Eck [aut]

Maintainer Qiyang Wang <wangqiyang497@gmail.com>

Repository CRAN

Date/Publication 2026-07-27 14:40:02 UTC

Contents

`aapl_control_input` *Apple control dataset*

Description

A processed dataset used in the Apple (AAPL) validation study. It contains aligned target and donor shock windows together with macro-financial covariates and associated shock indices used by the synthetic forecasting procedure.

Usage

```
aapl_control_input
```

Format

A list with components:

covar_names Character vector of covariate names.

Y List of outcome series (target + donors).

X List of covariate matrices aligned with each series in `Y`.

shock_time_vec Integer vector of local shock indices.

shock_length_vec Integer vector of post-shock lengths.

Details

The raw market data were originally retrieved from Yahoo Finance via the `quantmod` package. CPI data (series CPIAUCSL) were obtained from the Federal Reserve Economic Data (FRED).

The distributed dataset contains trading-day aligned panels and localized inflation-adjusted shock windows converted to 2020-dollar units.

Source

Yahoo Finance (via `quantmod`); Federal Reserve Economic Data (FRED), CPIAUCSL.

`aapl_onestep_input` *Apple one-step control-shock experiment input*

Description

Prepared input object for the Apple one-step post-shock forecasting experiment used in the technical report.

Usage

```
data(aapl_onestep_input)
```

Format

A named list with components:

Y_series_list List of target and donor response series.

X_for_dbw List of target and donor covariate series used for DBW.

shock_time_vec Local shock-time indices for the target and donor windows.

shock_length_vec Shock lengths for the target and donor windows.

Z_tgt_base Target baseline window used for one-step regression forecasting.

control_shock_day Date of the control shock included in the baseline model.

target_day Forecast target date.

last_obs_day Last observed date used for one-step prediction.

donor_effect_days Donor shock-effect dates used in the experiment.

release_day MacBook release date associated with the target event.

Details

The object supports reproduction of the experiment combining:

- a one-step local regression baseline for the target series,
- donor-side shock effect estimation,
- donor balancing weights (DBW),
- control-shock adjustment via `build_control_xreg()`.

`auto_garchx` *Auto-select a GARCHX specification by BIC*

Description

Grid-searches over GARCHX order specifications and selects the model with the lowest BIC. The function optionally refits the selected model using a backcast value initialized at the estimated unconditional variance, clamped to a reasonable range relative to the sample variance.

Usage

```
auto_garchx(
  y,
  xreg = NULL,
  max.p = 3,
  max.q = 3,
  search_o = c(0L, 1L),
  final_refit = TRUE,
  clamp_factor = c(0.1, 10),
  vcov.type = c("ordinary", "robust"),
  verbose = TRUE
)

auto_garch(
  y,
  xreg = NULL,
  max.p = 3,
  max.q = 3,
  search_o = c(0L, 1L),
  final_refit = TRUE,
  clamp_factor = c(0.1, 10),
  vcov.type = c("ordinary", "robust"),
  verbose = TRUE
)
```

Arguments

`y`	Numeric vector of returns or residuals. Missing, infinite, and NaN values are not allowed.
`xreg`	Optional matrix or data frame of exogenous regressors. If provided, it must have the same number of rows as `length(y)`.
`max.p`	Nonnegative integer. Maximum GARCH lag order to search.
`max.q`	Nonnegative integer. Maximum ARCH lag order to search.
`search_o`	Integer vector specifying asymmetry orders to search. Supported values are `0` and `1`, where `0` denotes a symmetric GARCH model and `1` denotes a GJR-type asymmetric model.

final_refit	Logical. If TRUE, the BIC-selected model is optionally refit using a backcast value based on the estimated unconditional variance.
clamp_factor	Length-two numeric vector c(low, high). If the estimated unconditional variance falls outside [low, high] * var(y), the sample variance is used as the backcast.
vcov.type	Character string. Either "ordinary" or "robust", passed to garchx::garchx().
verbose	Logical. If TRUE, warnings and refit messages are printed.

Details

The garchx package uses order c(q, p, o), where q is the ARCH order, p is the GARCH order, and o is the asymmetry order. The trivial specification c(0, 0, 0) is skipped. Among all converged models, the function selects the specification with the lowest BIC.

When final_refit = TRUE, the selected model is refit using a scalar backcast.values initialized from the estimated unconditional variance. For a GJR specification, the unconditional variance is approximated as

$$\omega/(1 - \alpha - \beta - 0.5\gamma).$$

If this value is invalid or outside the clamped range, the sample variance is used instead.

Value

A list containing:

- fit: the selected garchx fit object.
- q, p, o: selected GARCHX orders.
- bic, aic: information criteria for the selected fit.
- refit_used: whether the final backcast refit replaced the original grid-search fit.

Examples

```
set.seed(1)
y <- scale(rnorm(500))[, 1]

out <- auto_garchx(
  y = y,
  max.p = 2,
  max.q = 2,
  search_o = c(0, 1),
  final_refit = TRUE,
  vcov.type = "robust",
  verbose = FALSE
)

c(out$q, out$p, out$o)
out$bic
```

build_control_xreg *Helper Function: Build control-shock xreg matrix*

Description

Internal helper to convert shock specifications into a design matrix for ARIMA xreg.

Usage

```
build_control_xreg(last, control_spec = NULL)
```

Arguments

last	Integer; length of the training period.
control_spec	List or data.frame defining shock timing, length, and shape.

Value

A numeric matrix or NULL.

cop_oil_postshock *COP oil post-shock input data*

Description

A processed dataset used in the ConocoPhillips (COP) empirical study. It contains aligned target and donor shock windows together with macro-financial covariates and associated shock indices used by the synthetic forecasting procedure.

Usage

```
cop_oil_postshock
```

Format

A list with components:

covar_names Character vector of covariate names.

Y List of outcome series (target + donors).

X List of covariate matrices aligned with each series in `Y`.

shock_time_vec Integer vector of local shock indices.

shock_length_vec Integer vector of post-shock lengths.

Details

The raw market data were originally retrieved from Yahoo Finance via the quantmod package. CPI data (series CPIAUCSL) were obtained from the Federal Reserve Economic Data (FRED).

The distributed dataset contains trading-day aligned panels and localized inflation-adjusted shock windows converted to 2020-dollar units.

Source

Yahoo Finance (via quantmod); Federal Reserve Economic Data (FRED), CPIAUCSL.

---

dbw *Donor Balancing Weights (DBW)*

Description

Compute donor weights on pre-shock data so that a convex combination of donors matches the target under L1/L2 distance. Supports PCA, sum-to-one, box constraints, and L1/L2 regularization on weights.

Usage

```
dbw(
  X,
  dbw_indices,
  shock_time_vec,
  scale = FALSE,
  center = FALSE,
  sum_to_1 = TRUE,
  bounded_below_by = 0,
  bounded_above_by = 1,
  princ_comp_count = NULL,
  normchoice = "l2",
  penalty_normchoice = "l1",
  penalty_lambda = 0,
  Y = NULL,
  Y_lookback_indices = NULL,
  X_lookback_indices = NULL,
  inputted_transformation = base::identity,
  verbose = FALSE
)
```

Arguments

X	List of data.frame/matrix. First is target (X[[1L]]); the remaining are donors (X[[2L]] ... X[[n+1L]]). Rows = time, columns = features.
dbw_indices	Integer vector of feature columns to use.

shock_time_vec Integer vector, same length as X; each series is truncated to rows 1:shock_i before matching.

scale, center Logical; z-score/center features before matching.

sum_to_1 Logical; if TRUE, enforce sum(W)=1.

bounded_below_by, bounded_above_by
Numeric scalars; lower/upper bounds for each weight (broadcast to all weights).

princ_comp_count
NULL or positive integer; if not NULL, apply PCA and keep that many principal components.

normchoice Character; "l1" or "l2" distance.

penalty_normchoice
Character; "l1" (LASSO) or "l2" (Ridge) on weights.

penalty_lambda Numeric >= 0; regularization strength (0 = off).

Y Optional list (same length as X) of auxiliary features to merge.

Y_lookback_indices
Optional integer vector/list; backward row indices for Y (e.g., c(1,2,5) means last, 2nd last, 5th last).

X_lookback_indices
Optional integer vector/list; backward row indices for X. If NULL, use only the last pre-shock row.

inputted_transformation
Function applied to each Y[[i]] before merging; defaults to base::identity.

verbose Logical. If TRUE, print optimization progress and diagnostic information.

Details

With sum_to_1=TRUE and nonnegative bounds, $||W||_1 = 1$; L1 regularization on $||W||_1$ is then a constant and does not change the optimum. Use L2 if you want an effective penalty under simplex constraints.

Value

A list with:

- opt_params: numeric vector of donor weights (length = #donors)
- convergence: "convergence" or "failed_convergence"
- loss: final matching loss (L1 or L2)

Examples

```
set.seed(1)
T <- 50; K <- 3
target <- data.frame(x1=rnorm(T), x2=rnorm(T, 0.2), x3=rnorm(T,-0.1))
donor1 <- data.frame(x1=rnorm(T, 0.1), x2=rnorm(T,-0.1), x3=rnorm(T, 0.3))
donor2 <- data.frame(x1=rnorm(T,-0.2), x2=rnorm(T, 0.4), x3=rnorm(T, 0.0))
X_list <- list(target, donor1, donor2)
```

```
shock  <- c(40, 40, 40)

out <- dbw(
  X = X_list, dbw_indices = 1:K, shock_time_vec = shock,
  center = TRUE, scale = TRUE,
  sum_to_1 = TRUE, bounded_below_by = 0, bounded_above_by = 1,
  normchoice = "l2",
  penalty_normchoice = "l2", penalty_lambda = 1e-2
)
out$opt_params; out$loss; out$convergence
```

`iyg_onestep_input` *One-step volatility forecasting input for IYG*

Description

A ready-to-use input object for the IYG one-step post-shock volatility forecasting experiment. The target episode is the 2016 U.S. election, and the candidate donor episodes include earlier U.S. elections and Brexit-related political uncertainty events.

The object stores three pre-assembled donor-pool specifications: `all`, `restricted`, and `elections_only`. Each specification contains the `Y_series_list`, `covariates_series_list`, `target_covariates`, `shock_time_vec`, and `shock_length_vec` arguments required by `SynthVolForecast()`.

Usage

```
iyg_onestep_input
```

Format

A named list with the following components:

description Short description of the data object.

target_symbol Character scalar giving the target asset symbol.

target_event Character scalar giving the target event name.

inputs Named list of ready-to-use `SynthVolForecast()` input objects. The current object includes `all`, `restricted`, and `elections_only`.

donor_pool_specs Named list describing the donor events included in each donor-pool specification.

evaluation List describing the realized-variance proxy, forecast horizon, and proxy index.

Each element of `inputs` is itself a named list with the following components:

target_event Character scalar giving the target event name.

donor_events Character vector of donor event names.

Y_series_list List of return series, with the target series first followed by the donor series.

covariates_series_list List of donor covariate data frames.

target_covariates Covariate data frame for the target episode.

shock_time_vec Integer vector of shock-time indices for the target and donor series.

shock_length_vec Integer vector of post-shock window lengths for the target and donor series.

Details

The outcome series are daily log returns for IYG. The covariates include aligned macro-financial variables used for donor balancing in the `SynthVolForecast()` workflow. Since the true conditional variance is latent in real data, empirical evaluation uses the next-day squared return as a realized variance proxy.

The `all` donor pool contains all candidate donor episodes. The `restricted` donor pool contains the 2012 Election, Brexit Poll Released, and Brexit episodes. The `elections_only` donor pool contains the 2004, 2008, and 2012 U.S. election episodes.

Examples

```
data("iyg_onestep_input")

names(iyg_onestep_input)
names(iyg_onestep_input$inputs)

iyg_restricted <- iyg_onestep_input$inputs$restricted
names(iyg_restricted)
iyg_restricted$donor_events
```

`SynthPrediction` *SynthPrediction - Synthetic Prediction with Donor Balancing and Shock Adjustment*

Description

Build k-step-ahead forecasts for a **target** time series by: (1) estimating each **donor** series' post-shock fixed effect via (S)ARIMA with a post-shock indicator; (2) combining donor effects by weights (uniform or dbw); and (3) adjusting the target ARIMA forecast by the weighted donor effect. Indices across lists are assumed aligned.

Seasonality: `seasonal=TRUE` by default. For *each* series, seasonality is enabled only if `forecast::findfrequency()` detects a frequency > 1 (NA or <= 1 disables seasonality for that series).

Usage

```
SynthPrediction(
  Y_series_list,
  covariates_series_list,
  shock_time_vec,
  shock_length_vec,
```

```
  k = 1,
  dbw_scale = TRUE,
  dbw_center = TRUE,
  dbw_indices = NULL,
  use_dbw = TRUE,
  princ_comp_input = NULL,
  covariate_indices = NULL,
  geometric_sets = NULL,
  days_before_shocktime_vec = NULL,
  arima_order = NULL,
  seasonal = TRUE,
  seasonal_order = NULL,
  seasonal_period = NULL,
  user_ic_choice = c("aicc", "aic", "bic")[1],
  stepwise = TRUE,
  approximation = TRUE,
  penalty_lambda = 0,
  penalty_normchoice = "l1",
  control_shocks = NULL,
  pools = NULL,
  pool_agg = "sum",
  pool_weights = NULL,
  base_use_dbw = FALSE,
  pool_use_dbw = TRUE,
  ...
)
```

Arguments

`Y_series_list`	List of time series (e.g., `xts`). `[[1]]` is the target; `[[2]]..[[n+1]]` are donors.
`covariates_series_list`	List of time series aligned with `Y_series_list`. Used for DBW features and (optionally) as `xreg` in ARIMA. You may pass `Y_series_list` here as a simple baseline.
`shock_time_vec`	Vector of shock start times, each either a time index present in the corresponding series or a positive integer row number. Length must equal `length(Y_series_list)`.
`shock_length_vec`	Integer vector (same length) giving the post-shock window length (in rows) used to estimate donor fixed effects.
`k`	Integer forecast horizon for the target.
`dbw_scale, dbw_center`	Logical flags forwarded to `dbw` when computing donor weights.
`dbw_indices`	Integer vector of covariate columns used by `dbw`. If `NULL`, informative columns are auto-selected at pre-shock rows.
`use_dbw`	Logical; if `FALSE`, use equal donor weights; otherwise use `dbw`.
`princ_comp_input`	Optional PCA dimension for `dbw` (forwarded only).

`covariate_indices`
: Integer vector of columns from `covariates_series_list[[i]]` to include as `xreg` in ARIMA (lagged for the target; contemporaneous for donors). If `NULL`, donors use only the post-shock indicator and the target uses no `xreg`.

`geometric_sets, days_before_shocktime_vec`
: Reserved (ignored).

`arima_order`
: Optional fixed ARIMA order `c(p,d,q)` for all series; if `NULL`, `forecast::auto.arima()` is used.

`seasonal`
: Logical: whether seasonal models are allowed/requested. Default `TRUE`.

`seasonal_order`
: Optional seasonal order `c(P,D,Q)` used only when `arima_order` is supplied and seasonal detection is `TRUE`.

`seasonal_period`
: Optional seasonal period. If `NULL` and `seasonal=TRUE`, it is auto-detected per series via `findfrequency()`.

`user_ic_choice`
: Information criterion for `auto.arima` (`"aicc"`, `"aic"`, `"bic"`).

`stepwise, approximation`
: Passed through to `forecast::auto.arima()`.

`penalty_lambda`
: Numeric L1/L2 penalty forwarded to dbw.

`penalty_normchoice`
: Character `"l1"` or `"l2"`; forwarded to dbw.

`control_shocks`
: Optional list of shock specifications to control for (remove) in the target series model. Typically `list(list(time=..., length=..., shape=...))`.

`pools`
: Optional named list of donor pools. If provided, the call is dispatched to `SynthPredictionMultiPool()`.

`pool_agg`
: Aggregation rule across pools ("sum" or "weighted").

`pool_weights`
: Optional named numeric vector of pool weights.

`base_use_dbw`
: Logical. Whether DBW is used in the baseline forecast.

`pool_use_dbw`
: Logical. Whether DBW is used within each pool.

`...`
: Additional arguments passed to `SynthPredictionMultiPool` when `pools` is supplied.

Value

A list with components:

- `linear_combinations`: numeric donor weights W.
- `predictions`: list with numeric vectors `unadjusted`, `adjusted`, and `arithmetic_mean`.
- `meta`: metadata including weights, shock timing, ARIMA orders, and the information criterion used.

If `pools` is provided, the function dispatches to `SynthPredictionMultiPool` and returns its output (see that function for the multi-pool return structure).

See Also

`dbw`, `auto.arima`, `Arima`

Examples

```
library(xts)
set.seed(1)
T  <- 80
ix <- seq.Date(as.Date("2022-01-01"), by = "day", length.out = T)
y1 <- xts(rnorm(T), ix)  # target
y2 <- xts(rnorm(T), ix)  # donor 1
y3 <- xts(rnorm(T), ix)  # donor 2
Y  <- list(y1, y2, y3)
X  <- list(y1, y2, y3)   # simple baseline for DBW/xreg
shock_time  <- rep(ix[60], length(Y))
shock_len   <- rep(5L,   length(Y))
out <- SynthPrediction(
   Y_series_list = Y,
   covariates_series_list = X,
   shock_time_vec = shock_time,
   shock_length_vec = shock_len,
   k = 2
)
str(out$predictions)
```

SynthPredictionMultiPool *Multi-pool shock aggregation for SynthPrediction*

Description

Extends `SynthPrediction()` to multiple donor pools representing distinct shock channels (e.g., supply-side, demand-side).

Usage

```
SynthPredictionMultiPool(
  Y_series_list,
  covariates_series_list,
  shock_time_vec,
  shock_length_vec,
  k = 1,
  pools,
  pool_agg = c("sum", "weighted"),
  pool_weights = NULL,
  base_use_dbw = FALSE,
  pool_use_dbw = TRUE,
  ...
)
```

Arguments

`Y_series_list`	List of time series (e.g., `xts`). `[[1]]` is the target; `[[2]]..[[n+1]]` are donors.
`covariates_series_list`	List of time series aligned with `Y_series_list`. Used for DBW features and (optionally) as `xreg` in ARIMA. You may pass `Y_series_list` here as a simple baseline.
`shock_time_vec`	Vector of shock start times, each either a time index present in the corresponding series or a positive integer row number. Length must equal `length(Y_series_list)`.
`shock_length_vec`	Integer vector (same length) giving the post-shock window length (in rows) used to estimate donor fixed effects.
`k`	Integer forecast horizon for the target.
`pools`	Named list. Each element is a character vector of donor series names (matching `names(Y_series_list)`). Pool names define channels.
`pool_agg`	Aggregation rule across pools. One of "sum" or "weighted".
`pool_weights`	Optional named numeric vector of pool weights.
`base_use_dbw`	Logical. Whether DBW is used in the baseline forecast.
`pool_use_dbw`	Logical. Whether DBW is used within each pool. #' @param ... Additional arguments forwarded to the underlying `SynthPrediction` calls.
`...`	Additional arguments passed to `SynthPredictionMultiPool` when `pools` is supplied.

Value

Same structure as `SynthPrediction()`, with extra: `meta$multi_pool` and `predictions$adjusted_multipool`.

`SynthVolForecast` *Synthetic Volatility Forecast with Donor Shock Adjustment*

Description

Produces k-step-ahead variance forecasts for a target series using a pre-shock GARCH model, then adjusts that forecast by a donor-based post-shock effect. Each donor is fitted with a GARCH-X model that includes a post-shock indicator. The donor-specific post-shock indicator coefficients are combined using donor balancing weights (DBW) to obtain the final volatility adjustment.

Usage

```
SynthVolForecast(
  Y_series_list,
  covariates_series_list,
  target_covariates,
  shock_time_vec,
  shock_length_vec,
```

```
  k = 1,
  dbw_scale = TRUE,
  dbw_center = TRUE,
  dbw_indices = NULL,
  princ_comp_input = NULL,
  covariate_indices = NULL,
  penalty_lambda = 0,
  penalty_normchoice = "l1",
  return_fits = FALSE,
  garch_order = NULL,
  max.p = 3,
  max.q = 3,
  backcast.initial = NULL,
  vcov.type = "robust"
)
```

Arguments

`Y_series_list`	List of numeric series. The first element is the target series, and the remaining elements are donor series.
`covariates_series_list`	List of data frames or matrices containing donor covariates. Its length must equal the number of donors.
`target_covariates`	Data frame or matrix containing target covariates. These covariates are used for DBW matching. By default, the target-side GARCH model is fitted without exogenous regressors.
`shock_time_vec`	Integer vector giving the shock index for each series. Its length must equal `length(Y_series_list)`.
`shock_length_vec`	Integer vector giving the number of post-shock rows for which the donor post-shock indicator equals one.
`k`	Integer forecast horizon.
`dbw_scale`	Logical. Whether to scale DBW features.
`dbw_center`	Logical. Whether to center DBW features.
`dbw_indices`	Integer vector specifying which covariate columns are used by dbw. If `NULL`, all columns of `target_covariates` are used.
`princ_comp_input`	Optional integer specifying the number of principal components used inside dbw. If `NULL`, PCA is not used.
`covariate_indices`	Integer vector specifying which donor covariate columns are included in donor-side GARCH-X models. If `NULL`, donors use only the post-shock indicator.
`penalty_lambda`	Numeric penalty weight forwarded to dbw.
`penalty_normchoice`	Character string, either `"l1"` or `"l2"`, forwarded to dbw.

return_fits	Logical. If TRUE, the fitted donor and target GARCH objects are returned.
garch_order	Optional fixed GARCH-X order c(q, p, o). If NULL, the order is selected by auto_garchx.
max.p	Nonnegative integer. Maximum GARCH lag order passed to auto_garchx.
max.q	Nonnegative integer. Maximum ARCH lag order passed to auto_garchx.
backcast.initial	Optional positive numeric scalar used as backcast.values when garch_order is fixed.
vcov.type	Character string. Either "ordinary" or "robust", passed to garchx() and lmtest::coeftest().

Details

The target-side model is fitted only on the pre-shock target series and does not use exogenous regressors. This avoids requiring future target-side covariates at forecast time. Donor-side models are GARCH-X specifications that include a post-shock indicator and, optionally, selected donor covariates. The coefficient on the post-shock indicator is interpreted as the donor-specific short-horizon volatility effect.

The final adjusted forecast is computed as

$$\hat{\sigma}^{2,adj}_{0,T+h} = \max\{\hat{\sigma}^{2,raw}_{0,T+h} + \hat{\omega}, \epsilon\},$$

where $\hat{\omega}$ is the DBW-weighted donor shock effect and $\epsilon = 10^{-8}$ enforces a strictly positive variance forecast.

Forecast accuracy is evaluated outside this function using realized variance proxies, such as next-period squared returns.

Value

A list containing:

- linear_combinations: numeric vector of donor weights.
- predictions: list with unadjusted, adjusted, and arithmetic_mean variance forecasts.
- meta: diagnostic information, including donor weights, donor-specific post-shock effects, standard errors, selected orders, and the combined shock effect.
- donor_fits: returned only if return_fits = TRUE.
- target_fit: returned only if return_fits = TRUE, or in the no-donor fast path.

Examples

```
set.seed(1)
y_tgt <- rnorm(200)
y_d1  <- rnorm(200)
y_d2  <- rnorm(200)

Y <- list(y_tgt, y_d1, y_d2)
```

```
Xt  <- cbind(x1 = rnorm(200), x2 = rnorm(200))
Xd1 <- cbind(x1 = rnorm(200), x2 = rnorm(200))
Xd2 <- cbind(x1 = rnorm(200), x2 = rnorm(200))

out <- SynthVolForecast(
  Y_series_list = Y,
  covariates_series_list = list(Xd1, Xd2),
  target_covariates = Xt,
  shock_time_vec = c(150L, 150L, 150L),
  shock_length_vec = c(10L, 10L, 10L),
  k = 2,
  covariate_indices = 1:2,
  dbw_indices = 1:2
)

str(out$predictions)
```

Index