Package 'postshock'

November 2, 2026

Type Package

Title Donor-Adjusted Post-Shock Forecasting

Version 0.2.3

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 2026-11-02 14:00:08 UTC

Contents

aapl_control_input . . . 2
aapl_onestep_input . . . 3
aapl_s1_input . . . 4
auto_garchx . . . 5
build_control_xreg . . . 7
cop_oil_postshock . . . 8
dbw . . . 9
iyg_onestep_input . . . 11
SynthPrediction . . . 12
SynthPredictionMultiPool . . . 16
SynthVolForecast . . . 17
technical_report . . . 20

Index **22**

`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()`.

`aapl_s1_input` *Boundary-aligned Apple S1 article inputs*

Description

Frozen inputs for the retrospective one-step Apple S1 illustration. Derived without downloading or changing observations from `aapl_onestep_input` and `aapl_control_input`. This object contains neither fitted models nor precomputed forecast values.

Usage

```
data(aapl_s1_input)
```

Format

A named list:

Y_series_list Target AAPL close observations from 7 October through 10 November 2020 (25 trading rows), followed by three unchanged 25-row donor series. Prices retain the source objects' 2020-dollar processing.

covariates_series_list Aligned GSPC and IRX matching panels. GSPC retains the source's 2020-base index scaling; IRX retains its quoted units.

shock_time_vec Last pre-effect boundaries: target row 25 and donor rows obtained by locating each effect date and subtracting one (row 12).

shock_length_vec One observation for each effect window. Donor response models therefore use rows 1 through 13, not all supplied rows.

control_shocks Target-only nested point specification for the 9 October 2020 response. Forecast-time controls are zero.

settings Explicit S1 parameters: one-step automatic ARIMA with AICc, stepwise and approximation enabled, no seasonality, no response-model market regressors, ten-row centered/scaled DBW, no PCA and zero penalty.

dates Supplied and fitted window sizes, start/end dates, boundaries, effect dates and matching starts for every series. Donor effect dates are 7 June 2017, 31 October 2018 and 4 June 2019.

control_date Historical control response date.

evaluation 11 November 2020 and its realized AAPL close; separate from all target fitting and matching inputs and not passed to the model.

provenance Source names, schema and retrospective-data limitations.

Details

Overlapping dated rows of the two old objects are checked for exact equality. Their union supplies the target window and separate evaluation observation. Donor panels are unchanged; API boundaries immediately precede the located donor effect observations. The old objects are retained unchanged.

Market prices originate from Yahoo Finance, with annual CPI processing based on FRED CPIAUCSL as documented in the article supplement. Retrieval timestamps, raw provider snapshots and some original object builders are unavailable. This is not a real-time-vintage backtest or an exact reproduction of the original delayed-response experiment. Provider revisions need not reconstruct the same historical values.

The development-checkout builder is 'data-raw/aapl_s1_input.R' (excluded from the built tarball). Its implementation is installed as 'replication/apple_data_builder.R'. The shared Apple diagnostic script is installed as 'replication/apple_diagnostic.R' and retains the article's numerical assertions. Donor-effect and weight reference checks retain their original absolute tolerances; their error bounds are propagated to the combined adjustment, with a separate weighted-sum identity check. Its implementation manifest distinguishes same-version builds.

Examples

```
data(aapl_s1_input)
input_fields <- c("Y_series_list", "covariates_series_list", "shock_time_vec",
                  "shock_length_vec", "control_shocks")
arguments <- c(aapl_s1_input[input_fields], aapl_s1_input$settings)
fit <- do.call(SynthPrediction, arguments)
fit$predictions
aapl_s1_input$evaluation
```

---

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 ARCH lag order to search.
max.q	Nonnegative integer. Maximum GARCH 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 GARCH order, p is the ARCH 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. A candidate is eligible only if its top-level convergence is a finite numeric scalar equal to zero, its coefficient vector is nonempty and finite, and its BIC is a finite scalar. Missing or unknown convergence flags are rejected. Standard errors need not all be finite. Exact BIC ties retain the first eligible model in the original grid order. If no candidate qualifies, an error reports counts of fit, convergence, coefficient and BIC rejections. This uses the numerical garchx status, not DBW's string status.

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. The original backcast skip conditions retain the eligible grid winner. A refit uses the same eligibility

checks and replaces the grid fit only for strictly lower BIC; errors, invalid status/parameters/BIC, and ties retain the original grid object.

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 a single finite positive integer. If not `NULL`, apply PCA and retain that many right-singular-vector directions from the processed matching matrix. Invalid or unavailable dimensions cause an error; requests are not rounded or truncated.

`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.

PCA is applied after feature selection, pre-shock truncation, lookback expansion, stacking the target and donors, removal of constant columns, and the existing `center`/`scale` operations. The available direction count is `min(nrow(dat), ncol(dat))` for the processed matching matrix `dat`, as supplied by the default `svd(dat)`. This is a dimension bound, not a numerical-rank threshold: zero singular values are not filtered and no automatic component selection is performed. The existing restriction against PCA with only one remaining feature is retained. `NULL` leaves the unprojected matching path unchanged; PCA does not force centering.

Donor weights are optimized in the common PC score space, not taken from explained-variance proportions. The return structure and optimizer-status reporting are unchanged, including `"failed_convergence"` when reported by the solver. Explicit PCA prediction calls to `SynthPrediction()` additionally reject unsuccessful status or unusable weights.

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 uses an explicit period or `forecast::findfrequency()` on that series' fitting window. Periods above one enable seasonal search; otherwise it is off.

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 last pre-shock boundaries, 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`
NULL or a single finite positive integer, forwarded as `dbw(princ_comp_count = princ_comp_input)` when DBW is enabled. Invalid or unavailable dimensions cause an error. The direction limit follows the processed matching matrix described in dbw. Single-pool `use_dbw=FALSE` ignores this unused setting.

`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 only any supplied controls. With extra donor covariates, selected X and Y must be finite, complete and row-aligned throughout the original donor fitting window. Invalid rows, columns or indexed dates cause an error before fitting or fallback.

`geometric_sets` Reserved and ignored; retained for compatibility.

`days_before_shocktime_vec`
Single lookback length for DBW, not a per-series vector. NULL uses the last pre-shock row; a positive length uses that many observations backwards from each boundary.

`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 the chosen period exceeds one. Ignored during automatic order selection.

`seasonal_period`
Optional single finite positive integer in observation steps, not calendar units. With `seasonal=TRUE`, an explicit period is used or NULL detects it per fitting window via `findfrequency()`. Period one disables seasonal search. The parameter is ignored when `seasonal=FALSE`; user periods are not rounded.

`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 series-level list; only its first element is used for historical target controls. For one point control, use `list(list(list(time=3, length=1, shape="point")))`. The first element can also be a specification or data frame accepted by `build_control_xreg`. Future control values are zero; donors do not receive these controls.

`pools` Optional named list of donor pools. If provided, the call is dispatched to `SynthPredictionMultiPool()`. The PCA setting `princ_comp_input` is forwarded; each internal call uses PCA only when its existing `base_use_dbw` or `pool_use_dbw` switch enables DBW. If both switches are false, PCA is not evaluated. Common matching, ARIMA, seasonal, penalty and control settings are forwarded unchanged. Single-pool `use_dbw` does not override the MultiPool switches.

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.

Details

With an explicit PCA request and DBW enabled, preprocessing, SVD and DBW errors stop prediction instead of falling back to equal weights. An empty matching feature set also causes an error. A DBW result must report the string "convergence" and supply one finite real numeric weight per donor. Its weights must satisfy the existing simplex constraints, allowing absolute tolerances of 1e-7 on each bound and 1e-6 on the sum-to-one constraint. Weights are neither clipped nor renormalized. These tolerances match the existing DBW regression checks and accommodate numerical residuals from the unchanged solver tolerance of 1e-8. The DBW weights, not PC explained-variance proportions, aggregate donor effects. With princ_comp_input=NULL, the existing warning, fallback and returned-status behavior is preserved.

Extra donor ARIMAX regressors are checked on rows 1 through boundary plus shock length. No rows are omitted or imputed. Errors identify the donor, affected variable, original row and available date/time; only selected columns and the fitting window are checked. Indexed Y and X must agree within a donor; calendars across historical donors need not agree. Unindexed inputs use positional alignment. Target lagged-regressor handling is unchanged.

For automatic ARIMA, the selected or detected period is carried by the univariate response's ts frequency. Numerical observations, order and window length are unchanged. Existing input frequency is not a separate priority: this interface uses its explicit-period or per-window detection rule. No calendar gaps are filled. The target's forecast-period response is not used for detection. A donor automatic-fit error retains the original non-seasonal ARIMA(1,1,1) fallback, with a warning; input validation errors are not caught by this fallback.

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 last pre-shock boundaries, 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.
`...`	Common matching, ARIMA, seasonal, penalty and target-control arguments forwarded unchanged to the underlying `SynthPrediction` calls. Duplicate named settings and `use_dbw` in `...` are errors: use `base_use_dbw` and `pool_use_dbw` for the two paths. `geometric_sets` remains reserved; there are no pool-specific overrides.

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)`, where `q` is the GARCH order, `p` is the ARCH order, and `o` is the asymmetry order. If `NULL`, the order is selected by `auto_garchx`.

`max.p`
: Nonnegative integer. Maximum ARCH lag order passed to `auto_garchx`.

`max.q`
: Nonnegative integer. Maximum GARCH 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)
```

---

technical_report *Read the full Technical Report*

Description

Locate the bundled PDF and sources for the full postshock Technical Report, covering methods, simulations, and the COP, IYG, and Apple applications.

Details

The report is installed in `technical_report/` and can be located with `system.file()`. It is a prebuilt document, distinct from the shorter `postshock-workflow` vignette. Reading it requires a PDF viewer, not R Markdown, LaTeX, an internet connection, or rerunning any analysis.

The verified PDF is retained unchanged from the 0.2.1 documentation build. See the package NEWS for subsequent maintenance changes, including the 0.2.2 Apple diagnostic checks. The R Markdown report and two exploratory simulation notebooks are included for reference; the notebooks are not required to read or regenerate the main report. Regeneration requires **rmarkdown**, **knitr**, **ggplot2**, Pandoc, and XeLaTeX in addition to the installed package and its dependencies.

See Also

`utils::vignette("postshock-workflow", package = "postshock")`

Examples

```
report <- system.file("technical_report", "Technical_report.pdf",
                      package = "postshock", mustWork = TRUE)
file.exists(report)
## Not run:
utils::browseURL(report)

## End(Not run)
```

Index

∗ **datasets**
aapl_control_input, 2
aapl_onestep_input, 3
aapl_s1_input, 4
cop_oil_postshock, 8
iyg_onestep_input, 11

aapl_control_input, 2
aapl_onestep_input, 3
aapl_s1_input, 4
Arima, *15*
auto.arima, *15*
auto_garch *(*auto_garchx*)*, 5
auto_garchx, 5, *19*

build_control_xreg, 7

cop_oil_postshock, 8

dbw, 9, *12*, *14, 15*, *18, 19*

iyg_onestep_input, 11

SynthPrediction, 12, *17*
SynthPredictionMultiPool, *15*, 16
SynthVolForecast, 17

technical_report, 20