Package: lagprotest (via r-universe)

November 2, 2026

Title Lagged Procrustean Randomization Test for Multivariate Time Series

Version 0.4.0

Description Extends the Procrustean randomization test (PROTEST) to ordered multivariate series. Concordance between two configurations is scanned across candidate temporal lags. Significance is assessed with phase-randomized surrogates (default), cyclic shifts of the full series or free permutation of rows, and a global max-statistic p-value accounts for scanning several lags and selecting the best one. Returns the lag profile, the lag of maximal concordance and per-observation Procrustes residuals.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports graphics, stats, utils, vegan

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/diogoprov/lagprotest, https://provetelab.org/lagprotest/

BugReports https://github.com/diogoprov/lagprotest/issues

Config/roxygen2/version 8.1.1

Language en-US

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

Date/Publication 2026-11-02 05:08:31 UTC

RemoteUrl https://github.com/diogoprov/lagprotest

RemoteRef main

RemoteSha 6658ce29783b7e95b10674a9e6bb1793388e2f52

Contents

lagged_protest 2
lagged_protest_pairs 5
plot.lagged_protest 6
residuals.lagged_protest 7
simulate_lagged_series 8
summary.lagged_protest 9

Index **10**

`lagged_protest` *Lagged Procrustean randomization test*

Description

Scans Procrustes concordance between two multivariate series across candidate temporal lags and tests it by permutation at each lag.

Usage

```
lagged_protest(
  X,
  Y,
  lags = 0:6,
  permutation = c("phase", "cyclic", "free"),
  nperm = 999,
  mirror = TRUE,
  bidirectional = FALSE,
  scale = TRUE,
  p.adjust.method = "none",
  period = NULL,
  seed = NULL
)
```

Arguments

`X`, `Y`	Numeric matrices or data frames with rows as ordered time points and columns as variables, or `dist` objects, which are converted to principal coordinates. Both must have the same number of rows.
`lags`	Integer vector of non-negative candidate lags.
`permutation`	Permutation scheme, `"phase"`, `"cyclic"` or `"free"`.
`nperm`	Number of surrogates (`permutation = "phase"`) or random permutations (`permutation = "free"`). Ignored for the cyclic scheme, whose permutations are enumerated.
`mirror`	Logical; with `permutation = "cyclic"`, also use the time-reversed shifts of `Y`, doubling the permutation set.
`bidirectional`	Logical; if `TRUE`, also test the negative lags `-lags` (Y leads X).

`scale` Logical; z-score the columns of `X` and `Y` (computed on the full series) before analysis. Ignored for `dist` inputs.

`p.adjust.method` Correction of the per-lag p-values for multiple lags: `"maxT"` for the single-step max-statistic adjustment, or any method of `stats::p.adjust()`. Defaults to `"none"`.

`period` Optional length of the seasonal cycle, in rows. If given, `X` and `Y` are truncated to their first `floor(n / period) * period` rows (whole cycles) before analysis; a message reports the rows dropped. `NULL` (default) uses all rows.

`seed` Optional integer seed for the `"phase"` and `"free"` schemes. The global random number stream is left untouched.

Details

Rows of `X` and `Y` are ordered time points (e.g., monthly centroids) and must correspond one to one. At lag `k > 0`, `X` at time `t` is paired with `Y` at time `t + k` (X leads Y), and the series are truncated to the overlapping window of `n - k` rows. Negative lags (Y leads X) are tested when `bidirectional = TRUE`. The statistic at each lag is the symmetric Procrustes sum of squares m^2 from `vegan::procrustes()`; the lag of maximal concordance is the one with the smallest m^2.

Three permutation schemes are available:

- `"phase"` (default): `nperm` phase-randomized surrogates of `Y` are drawn. Each surrogate keeps the amplitude spectrum of every column of `Y` (hence its autocorrelation) and the cross-spectra among its columns, but receives new Fourier phases, shared by all columns. Surrogates are applied jointly to all lags and do not reproduce the observed alignment, which makes this the recommended scheme for the global p-value. It assumes a stationary `Y` and treats the series as periodic with period `n`. In simulations, it was liberal for short, strongly autocorrelated series (AR(1) of 0.7 or more with `n` of 48 or less).
- `"cyclic"`: the full `Y` series is cyclically shifted by `1, ..., n - 1` positions before lagging, and optionally also time-reversed (`mirror = TRUE`). This keeps the temporal autocorrelation of `Y` under the null. All `n - 1` (or `2(n - 1)`) shifts are enumerated, so `nperm` and `seed` are ignored, and the smallest attainable p-value is `1 / (2(n - 1) + 1)` (0.043 for `n = 12` with `mirror = TRUE`). The scheme assumes the series spans complete cycles, i.e., that its period matches the number of rows (e.g., 12 monthly samples of an annual cycle). Because a shift of `Y` combined with lag `k` is equivalent to another lag, some shifts reproduce the observed alignment at other lags; the global p-value (see below) is then valid but has low power.
- `"free"`: rows of `Y` are permuted at random within the overlapping window, as in `vegan::protest()`. This assumes exchangeable rows and can strongly inflate Type I error when both series are autocorrelated.

Both the phase and the cyclic schemes treat the series as periodic with period `n`. When the series contains a seasonal cycle whose length does not divide `n` (e.g., 18 monthly samples of an annual cycle), no scheme is calibrated, because two series sharing a period are Procrustes-concordant whatever their phase. Supplying `period` truncates both series to their first `floor(n / period) * period` rows, so that the analysis spans whole cycles. In simulations, this restored the Type I error of series that already spanned complete cycles, whereas removing the seasonal component by harmonic regression did not.

P-values are computed as `(b + 1) / (N + 1)`, where `b` is the number of permuted m^2 values smaller than or equal to the observed one and `N` the number of permutations. Each permutation is applied jointly to all lags.

The per-lag p-values (`p.value`) are not corrected for testing several lags, nor for selecting the lag with the smallest m^2. Reporting the raw p-value at the optimal lag therefore inflates Type I error roughly to the rate at which any lag is significant. The global p-value (`p.global`) accounts for both: it compares the observed minimum m^2 across lags with the distribution of the minimum across lags under each permutation (a max-statistic test), so it is the p-value of the whole scan-and-select procedure and is the one to report for the optimal lag. With `p.adjust.method = "maxT"`, every lag receives the corresponding single-step max-statistic adjusted p-value, and the adjusted p-value at the optimal lag equals `p.global`. Any method of `stats::p.adjust()` can be requested instead; with few permutations, as in the cyclic scheme, corrections such as Holm may make significance unattainable. Note that m^2 values at different lags are based on different numbers of rows (`n - |k|`).

Value

An object of class `"lagged_protest"`, a list with:

`optimal_lag` Lag with the smallest m^2.

`m2` m^2 at the optimal lag.

`r` Procrustes correlation, $\sqrt{1 - m^2}$, at the optimal lag.

`p.value` Raw permutation p-value at the optimal lag.

`p.adjusted` P-value at the optimal lag adjusted with `p.adjust.method` (equal to `p.value` when `"none"`).

`p.global` Max-statistic p-value of the scan across lags, accounting for testing several lags and selecting the optimal one.

`n_perm` Number of permutations or surrogates at the optimal lag (the exact count for the cyclic scheme).

`profile` Data frame with `lag`, `n` (rows used), `m2`, `r`, `p.value`, `p.adjusted` and `nperm` for every lag.

`residuals` Data frame of per-observation Procrustes residuals at the optimal lag, with the paired row labels of `X` and `Y`.

`procrustes` The `vegan::procrustes()` fit at the optimal lag.

`null` List of permuted m^2 values, one element per lag; element `i` of every vector comes from the same permutation.

`permutation`, `p.adjust.method`, `n`, `period`, `call` Settings used; `n` is the number of rows after any truncation to whole cycles.

See Also

`vegan::protest()`, `vegan::procrustes()`

Examples

```
sim <- simulate_lagged_series(n_times = 12, true_lag = 2, noise_sd = 0.3,
                              seed = 1)
fit <- lagged_protest(sim$X, sim$Y, lags = 0:4, seed = 1)
fit
fit$p.global
fit_cyclic <- lagged_protest(sim$X, sim$Y, lags = 0:4,
                             permutation = "cyclic")
fit_cyclic$profile
plot(fit)
residuals(fit)
```

---

lagged_protest_pairs *Lagged PROTEST for several pairs of data sets*

Description

Runs `lagged_protest()` on pairs of data sets from a named list, such as the modules of a multi-trait study measured on the same time points, and collects the results in one table.

Usage

```
lagged_protest_pairs(
  data,
  pairs = NULL,
  lags = 0:6,
  bidirectional = TRUE,
  p.adjust.method = "BH",
  seed = NULL,
  ...
)
```

Arguments

`data`	A named list of numeric matrices, data frames or `dist` objects, all with the same number of rows (time points).
`pairs`	Pairs to analyze: `NULL` (default) for all unordered pairs, or a list of character vectors of length 2, or a two-column character matrix, giving names of `data`. The first element of each pair is passed as `X`.
`lags, bidirectional`	Passed to `lagged_protest()`. `bidirectional` defaults to `TRUE` here.
`p.adjust.method`	Correction of the global p-values across pairs, any method of `stats::p.adjust()`. Defaults to `"BH"`.
`seed`	Passed to `lagged_protest()` for every pair.
`...`	Further arguments passed to `lagged_protest()`, such as `permutation`, `nperm` or `period`.

Details

By default every unordered pair is analyzed once with `bidirectional = TRUE`, so both directions are covered by the sign of the lag: a positive optimal lag means that the first data set of the pair (`x`) leads the second (`y`), and a negative one that `y` leads `x`. The global p-values of the pairs are then adjusted for the number of pairs with `p.adjust.method`, because the max-statistic test accounts for the lags scanned within a pair but not for the pairs examined.

Every pair uses the same `seed`, so results do not depend on the order or number of pairs.

Value

A data frame with one row per pair and columns `x`, `y`, `optimal_lag`, `n` (rows at the optimal lag), `m2`, `r`, `p.value` (raw p at the optimal lag), `p.global` and `p.global.adjusted` (adjusted across pairs). The `lagged_protest()` fits are stored, named `"x -> y"`, in the attribute `"fits"`.

See Also

`lagged_protest()`

Examples

```
sim <- simulate_lagged_series(n_times = 12, true_lag = 2, noise_sd = 0.3,
                              seed = 1)
blocks <- list(A = sim$X, B = sim$Y,
               C = simulate_lagged_series(n_times = 12, seed = 2)$X)
res <- lagged_protest_pairs(blocks, lags = 0:3, nperm = 199, seed = 1)
res
plot(attr(res, "fits")[["A -> B"]])
```

---

plot.lagged_protest *Plot the lag profile of a lagged PROTEST*

Description

Plots m^2 against lag. The dashed line is the `alpha` quantile of the permutation null distribution at each lag (unadjusted); observed values below it are significant at that level before correction. Filled points mark lags whose p-value (adjusted, if `p.adjust.method` was not `"none"`) is below `alpha`. With `global = TRUE`, the optimal lag is circled, filled when the global p-value is below `alpha`, and the global p-value is printed above the plot; this is the p-value to report for the optimal lag.

Usage

```
## S3 method for class 'lagged_protest'
plot(x, alpha = 0.05, global = TRUE, ...)
```

Arguments

x	A "lagged_protest" object.
alpha	Significance level.
global	Logical; mark the optimal lag with its global p-value.
...	Further arguments passed to graphics::plot().

Value

x, invisibly.

---

residuals.lagged_protest *Per-observation Procrustes residuals at the optimal lag*

Description

Per-observation Procrustes residuals at the optimal lag

Usage

```
## S3 method for class 'lagged_protest'
residuals(object, ...)
```

Arguments

object	A "lagged_protest" object.
...	Unused.

Value

A data frame with the paired row labels of X and Y and the residual of each pair after superimposition.

`simulate_lagged_series` *Simulate a pair of multivariate series with a known lag*

Description

Generates `X` as a multivariate seasonal series (sinusoids with a random common phase and staggered offsets among variables) plus AR(1) noise. Under the null (`true_lag = NULL`), `Y` is an independent draw from the same generator. Otherwise `Y` at time `t` is a linear map of `X` at time `t - true_lag` plus AR(1) noise; `X` is generated over `n_times + true_lag` points so that every row of `Y` has a source in `X`.

Usage

```
simulate_lagged_series(
  n_times = 24,
  n_vars_x = 5,
  n_vars_y = 5,
  true_lag = NULL,
  period = 12,
  amplitude = 1,
  autocorr = 0.5,
  noise_sd = 1,
  seed = NULL
)
```

Arguments

`n_times`	Number of time points.
`n_vars_x, n_vars_y`	Number of variables in `X` and `Y`.
`true_lag`	`NULL` for independent series, or a non-negative integer lag by which `X` leads `Y`. `0` means a contemporaneous association, not the absence of one.
`period`	Length of the seasonal cycle, in time steps.
`amplitude`	Amplitude of the seasonal sinusoids.
`autocorr`	AR(1) coefficient of the noise, in `(-1, 1)`.
`noise_sd`	Marginal standard deviation of the AR(1) noise.
`seed`	Optional integer seed. The global random number stream is left untouched.

Details

Setting `amplitude = 0` removes the seasonal component and leaves pure AR(1) series; setting `autocorr = 0` gives white noise around the seasonal signal (the design of the original simulation, Appendix S1).

Value

A list with matrices X (n_times x n_vars_x) and Y (n_times x n_vars_y), and true_lag.

Examples

```
null_pair <- simulate_lagged_series(n_times = 12, seed = 42)
lagged_pair <- simulate_lagged_series(n_times = 24, true_lag = 3, seed = 42)
str(lagged_pair)
```

---

summary.lagged_protest

Summarize a lagged PROTEST

Description

Summarize a lagged PROTEST

Usage

```
## S3 method for class 'lagged_protest'
summary(object, ...)
```

Arguments

object	A "lagged_protest" object.
...	Unused.

Value

The lag profile (a data frame), with the optimal lag flagged.

Index

graphics::plot(), *7*

lagged_protest, 2
lagged_protest(), *5*, *6*
lagged_protest_pairs, 5

plot.lagged_protest, 6

residuals.lagged_protest, 7

simulate_lagged_series, 8
stats::p.adjust(), *3–5*
summary.lagged_protest, 9

vegan::procrustes(), *3*, *4*
vegan::protest(), *3*, *4*