Package 'GrangerRKHS'

September 12, 2026

Type Package

Title RKHS-Based Nonlinear Granger Causality Testing via Conditional Centering

Version 0.1.0

Description Provides methods for nonlinear Granger causality testing in reproducing kernel Hilbert space (RKHS), using kernel ridge regression for conditional mean estimation and conditional centering for construction of the test statistic.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports stats

NeedsCompilation no

Author Yuhan Tian [aut, cre],
Adam Waterbury [aut],
Marie-Christine Dueker [aut]

Maintainer Yuhan Tian <tyh9293@gmail.com>

Repository CRAN

Date/Publication 2026-09-12 07:20:02 UTC

Contents

`gc_sigma2_hat_null_exact` *Estimate Innovation Variance Under the Null Model*

Description

Estimates the innovation variance under the null Granger causality model using kernel ridge regression with an exact degrees-of-freedom correction.

Usage

```
gc_sigma2_hat_null_exact(
  X,
  p = 1L,
  target = 1L,
  source_set,
  lambda = 1/(nrow(X) - p),
  kernel = kernel_gaussian_tau,
  kernel_args = list()
)
```

Arguments

`X`	Numeric data matrix with rows representing time points and columns representing variables.
`p`	Integer. Lag order.
`target`	Integer index of the response variable.
`source_set`	Integer vector specifying the candidate source variables.
`lambda`	Nonnegative ridge regularization parameter. Default is `1 / n`, where `n = nrow(X) - p`.
`kernel`	Kernel function used in kernel ridge regression. Default is `kernel_gaussian_tau`.
`kernel_args`	List of additional arguments passed to `kernel`.

Details

The estimator is

$$\hat{\sigma}^2 = \frac{\|(I - S)y\|^2}{\operatorname{tr}\left[(I - S)^\top (I - S)\right]},$$

where S is the kernel ridge regression smoother matrix.

The null model regresses the target variable on the lagged conditioning variables, where the conditioning set is defined as the complement of `source_set`.

If `X` is a `Tn x d` matrix, then the effective sample size is

$$n = Tn - p.$$

The smoother matrix is

$$S = K(K + n\lambda I)^{-1},$$

where K is the empirical kernel matrix.

Value

A list containing:

- sigma2_hat: estimated innovation variance.
- rss: residual sum of squares.
- df_exact: exact effective degrees of freedom correction $\mathrm{tr}((I - S)^\top (I - S))$.

Examples

```
set.seed(1)

sim <- nlVAR_sim(
  Tn = 200,
  d = 2,
  p = 1,
  g = function(z) c(0.5 * tanh(z[1]), 0.3 * z[2])
)

gc_sigma2_hat_null_exact(
  X = sim$X,
  p = 1,
  target = 1,
  source_set = 2
)
```

---

granger_rkhs *Nonlinear Granger Causality Test via Conditional Centering in RKHS*

Description

Performs a nonlinear Granger causality test in reproducing kernel Hilbert spaces (RKHSs) using conditional centering.

Usage

```
granger_rkhs(
  X,
  p = 1L,
  target = 1L,
  source_set,
  lambda_f,
  lambda_K,
  kernel1 = kernel_gaussian_tau,
  kernel2 = kernel_gaussian_tau,
  kernel1_args = list(),
  kernel2_args = list(),
  sigma2hat,
  pval_method = c("satterthwaite", "simulation", "both"),
  B = 5000,
  eig_tol = 1e-10,
  return_null_sim = FALSE
)
```

Arguments

X	Numeric data matrix with rows representing time points and columns representing variables.
p	Integer. Lag order.
target	Integer index of the target variable.
source_set	Integer vector specifying the candidate Granger-causal source variables.
lambda_f	Ridge regularization parameter for the first-stage kernel ridge regression used to estimate the null conditional mean.
lambda_K	Ridge regularization parameter used to construct the conditional-centering operator for the second-stage kernel.
kernel1	Kernel function for the first-stage kernel ridge regression.
kernel2	Kernel function for the second-stage RKHS statistic.
kernel1_args	List of additional arguments passed to kernel1.
kernel2_args	List of additional arguments passed to kernel2.
sigma2hat	Estimated innovation variance.
pval_method	Character string specifying the p-value approximation method. Choices are "satterthwaite", "simulation", or "both".
B	Integer. Number of Monte Carlo simulations used when pval_method involves simulation.
eig_tol	Nonnegative threshold for filtering small eigenvalues.
return_null_sim	Logical indicating whether to return the simulated null distribution.

Details

The procedure first estimates the conditional mean of the target series under the null model using kernel ridge regression. It then constructs a conditionally centered second-stage kernel and combines it with the null-model residuals to test for additional nonlinear predictive dependence attributable to the candidate source variables.

Let

$$y_t = X_{t,\text{target}}.$$

The argument `target` specifies the response variable being tested, while `source_set` specifies the candidate Granger-causal source variables.

The conditioning set is defined as the complement of `source_set`, meaning that all remaining variables are included in the null model.

The null hypothesis states that the lagged variables in `source_set` do not provide additional predictive information about the target variable beyond the lagged conditioning variables.

The first-stage null model estimates the conditional mean of the target series given the lagged conditioning variables using kernel ridge regression.

The resulting residuals are used together with a conditionally centered second-stage kernel to construct the RKHS statistic

$$\frac{1}{n} e^\top K_2^{(c)} e,$$

where $K_2^{(c)}$ denotes the conditionally centered second-stage kernel matrix.

Under the null hypothesis, the asymptotic distribution is approximated by a weighted sum of chi-square random variables.

Two methods are implemented for approximating the null distribution and calculating p-values:

- Satterthwaite approximation.
- Monte Carlo simulation using empirical eigenvalues.

Value

A list containing the fitted objects, RKHS statistic, eigenvalues, null weights, and p-values.

Main returned components include:

- `Z_RKHS_sq`: RKHS norm-based test statistic.
- `pval`: reported p-value.
- `pval_sat`: Satterthwaite approximation p-value.
- `pval_sim`: simulation-based p-value.
- `eigvals`: positive eigenvalues used in the null approximation.
- `w`: null-distribution weights.
- `f_hat`: fitted null-model conditional mean.
- `e`: residuals from the null-model fit.

Examples

```
set.seed(1)

# ------------------------------------------------------------
# Example 1: Null model
# X2 does NOT Granger-cause X1
# ------------------------------------------------------------

g_null <- function(z) {
  c(
    0.5 * tanh(z[1]),
    0.3 * z[2]
  )
}

sim_null <- nlVAR_sim(
  Tn = 200,
  d = 2,
  p = 1,
  g = g_null
)

sigma2hat_null <- gc_sigma2_hat_null_exact(
  X = sim_null$X,
  p = 1,
  target = 1,
  source_set = 2
)$sigma2_hat

n_eff_null <- nrow(sim_null$X) - 1

out_null <- granger_rkhs(
  X = sim_null$X,
  p = 1,
  target = 1,
  source_set = 2,
  lambda_f = log(n_eff_null) / sqrt(n_eff_null),
  lambda_K = 1 / (n_eff_null * log(n_eff_null)),
  sigma2hat = sigma2hat_null
)

out_null$pval

# ------------------------------------------------------------
# Example 2: Alternative model
# X2 DOES Granger-cause X1
# ------------------------------------------------------------

g_alt <- function(z) {
  c(
    0.5 * tanh(z[1]) + 0.4 * z[2],
    0.3 * z[2]
```

```
  )
}

sim_alt <- nlVAR_sim(
  Tn = 200,
  d = 2,
  p = 1,
  g = g_alt
)

sigma2hat_alt <- gc_sigma2_hat_null_exact(
  X = sim_alt$X,
  p = 1,
  target = 1,
  source_set = 2
)$sigma2_hat

n_eff_alt <- nrow(sim_alt$X) - 1

out_alt <- granger_rkhs(
  X = sim_alt$X,
  p = 1,
  target = 1,
  source_set = 2,
  lambda_f = log(n_eff_alt) / sqrt(n_eff_alt),
  lambda_K = 1 / (n_eff_alt * log(n_eff_alt)),
  sigma2hat = sigma2hat_alt
)

out_alt$pval
```

`kernel_gaussian_tau` *Gaussian Kernel with Bandwidth Parameter*

Description

Computes the Gaussian kernel

Usage

```
kernel_gaussian_tau(x, y, tau = sqrt(2))
```

Arguments

`x`	Numeric vector.
`y`	Numeric vector of the same length as `x`.
`tau`	Positive scalar bandwidth parameter.

Details

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{\tau^2}\right),$$

where $\|\cdot\|$ denotes the Euclidean norm.

The default bandwidth is `sqrt(2)`.

This kernel is symmetric, positive definite, and bounded by 1.

Value

A numeric scalar containing the kernel value $K(x, y)$.

Examples

```
x <- c(1, 2)
y <- c(2, 3)

kernel_gaussian_tau(x, y)

kernel_gaussian_tau(x, y, tau = 1)
```

`kernel_gram` *Construct an Empirical Kernel (Gram) Matrix*

Description

Computes the empirical kernel matrix (Gram matrix)

Usage

```
kernel_gram(Z, kernel, ...)
```

Arguments

`Z`	Numeric matrix whose rows represent input vectors.
`kernel`	Kernel function taking arguments `(x, y, ...)` and returning a scalar kernel value.
`...`	Additional arguments passed to `kernel`.

Details

$$K_{st} = K(z_s, z_t),$$

where K is a user-specified kernel function and z_s, z_t are rows of the input matrix.

If Z is an N x q matrix, the returned Gram matrix has dimension N x N.

The implementation explicitly enforces symmetry by computing only the upper triangular part and copying values to the lower triangular part.

Value

A symmetric numeric matrix whose (s,t) entry is kernel(Z[s,], Z[t,], ...).

Examples

```
set.seed(1)

Z <- matrix(rnorm(20), ncol = 2)

K <- kernel_gram(
  Z,
  kernel = kernel_gaussian_tau,
  tau = 1
)

dim(K)
```

---

lagged_design *Construct Lagged Design Matrix for VAR Models*

Description

Constructs the lagged design matrix for a vector autoregressive process of order p.

Usage

```
lagged_design(X, p)
```

Arguments

X	Numeric matrix of observations with rows representing time and columns representing dimensions.
p	Integer. Lag order.

Details

Each row of the output matrix is of the form

$$(X_{t-1}^\top, X_{t-2}^\top, \ldots, X_{t-p}^\top)^\top,$$

ordered from newest lag to oldest lag.

If `X` is a `Tn x d` matrix, the returned matrix has dimension `(Tn - p) x (d * p)`.

Value

A numeric matrix with `nrow(X) - p` rows and `ncol(X) * p` columns. The `t`-th row contains the stacked lag vector $(X_{t-1}, \ldots, X_{t-p})$.

Examples

```
X <- matrix(
  1:12,
  nrow = 6,
  ncol = 2
)

lagged_design(X, p = 2)
```

`nlVAR_sim` *Simulate a Nonlinear Vector Autoregressive Process*

Description

Simulates observations from the nonlinear vector autoregressive model

Usage

```
nlVAR_sim(
  Tn,
  d,
  p,
  g,
  noise_sampler = function(m, d) matrix(stats::rnorm(m * d), nrow = m),
  x0 = NULL,
  burn = 200
)
```

Arguments

Tn	Integer. Number of observations returned after burn-in.
d	Integer. Dimension of the process.
p	Integer. Lag order.
g	Function mapping a numeric vector of length d * p to a numeric vector of length d.
noise_sampler	Function generating innovations. It should take arguments (m, d) and return an m x d matrix. The default generates independent standard Gaussian innovations.
x0	Optional p x d matrix of initial lag values. Row 1 corresponds to X_{t-1}, row 2 to X_{t-2}, and so on. If NULL, initialization starts from zero.
burn	Integer. Burn-in length discarded before returning the series.

Details

$$X_t = g(X_{t-1}, \ldots, X_{t-p}) + \varepsilon_t,$$

where $X_t \in \mathbb{R}^d$.

The lag vector passed to g is ordered from newest to oldest:

$$z = (X_{t-1}^\top, \ldots, X_{t-p}^\top)^\top.$$

Internally, the simulator stores lagged observations using a d x p buffer whose columns correspond to $X_{t-1}, \ldots, X_{t-p}$.

The function supports the special cases p = 1 and burn = 0.

Value

A list containing:

- X: a Tn x d matrix of simulated observations.
- eps: a Tn x d matrix of innovations.
- d: dimension of the process.
- p: lag order.
- Tn: number of returned observations.
- burn: burn-in length.
- call: matched function call.

Examples

```
d <- p <- 2

g_bounded <- function(z) {
  x1 <- z[1:d]
  x2 <- z[(d + 1):(2 * d)]

  out1 <- 1.5 * tanh(0.7 * x1[1] - 0.3 * x1[2] + 0.4 * x2[1])
  out2 <- sin(0.8 * x1[2]) + 0.3 * tanh(x2[1])

  c(out1, out2)
}

sim <- nlVAR_sim(
  Tn = 100,
  d = 2,
  p = 2,
  g = g_bounded
)

head(sim$X)
```

Index