Package ‘starvars’

September 17, 2020

Type Package
Title Vector Logistic Smooth Transition Models / Realized Covariances
Construction
Version 1.1.1
Description Allows the user to estimate a vector logistic smooth transition autoregres-
sive model via maximum log-likelihood or nonlinear least squares. It further per-
mits to test for linearity in the multivariate framework against a vector logistic smooth transi-
tion autoregressive model with a single transition variable. The estimation method is dis-
ized covariances can be constructed from stock market prices or returns, as explained in Ander-
License GPL
Encoding UTF-8
LazyData true
Depends R (>= 3.5.0)
Imports MASS, ks, zoo, data.table, methods, matrixcalc, vars, maxLik,
 rlist, fGarch, lubridate, xts, lessR, quantmod
URL https://github.com/andbucci/starvars
NeedsCompilation no
Author Andrea Bucci [aut, cre, cph],
 Giulio Palomba [aut],
 Eduardo Rossi [aut],
 Andrea Faragalli [ctb]
Maintainer Andrea Bucci <andrea.bucci@unich.it>
Repository CRAN
Date/Publication 2020-09-17 11:50:02 UTC

R topics documented:

coef ......................................................... 2
logLik ..................................................... 3
Coefficient method for objects of class `VLSTAR`

Description

Returns the coefficients of a VLSTAR model for objects generated by `VLSTAR()`.

Usage

```r
## S3 method for class 'VLSTAR'
coef(object, ...)
```

Arguments

- `object` An object of class ‘VLSTAR’; generated by `VLSTAR()`
- `...` Currently not used.

Author(s)

Andrea Bucci

References


See Also

`VLSTAR`

Examples

```r
##
##See 'VLSTAR' examples
##
##
```
**logLik**

*Log-Likelihood method*

---

**Description**

Returns the log-Likelihood of a VLSTAR object.

**Usage**

```r
## S3 method for class 'VLSTAR'
logLik(object, type = c(‘Univariate’, ‘Multivariate’), ...)
```

**Arguments**

- `object`: An object of class ‘VLSTAR’ obtained through `VLSTAR()`.
- `type`: Type of Log-Likelihood to be showed (univariate or multivariate).
- `...`: further arguments to be passed to and from other methods

**Details**

The log-likelihood of a VLSTAR model is defined as:

\[
\log l(y_t | I_t; \theta) = -\frac{T \tilde{n}}{2} \ln(2\pi) - \frac{T}{2} \ln |\Omega| - \frac{1}{2} \sum_{t=1}^{T} (y_t - \tilde{G}_t B z_t)' \Omega^{-1} (y_t - \tilde{G}_t B z_t)
\]

**Value**

An object with class attribute `logLik`.

**Author(s)**

The code was written by Andrea Bucci

**References**


**See Also**

`VLSTAR`

**Examples**

```r
##
##See 'VLSTAR' examples
##
##```
lrvarbart  

*Long-run variance using Bartlett kernel*

**Description**

Function returns the long-run variance of a time series, relying on the Bartlett kernel. The window size of the kernel is the cube root of the sample size.

**Usage**

`lrvarbart(x)`

**Arguments**

- `x`  
  a `(T x 1)` vector containing the time series over period `T`

**Value**

- `lrv`  
  long-run variance

- `bandwidth`  
  size of the window

**Author(s)**

The code was written by Andrea Bucci.

**References**


**Examples**

```r
data(Realized)

lrvarbart(Realized[,1])
```
**multiCUMSUM**

*Multivariate CUMSUM test*

**Description**
Function returns the test statistics for the presence of co-breaks in a set of multivariate time series.

**Usage**
```r
multiCUMSUM(data, conf.level = 0.95, max.breaks = 7)
```

```
## S3 method for class 'multiCUMSUM'
print(x, ...)
```

**Arguments**
- `data` a \((T \times N)\) matrix or \(data.frame\) containing the \(N\) time series over period \(T\)
- `conf.level` Confidence level. By default set to 0.95
- `max.breaks` Integer, determines the highest number of common breaks from 1 to 7.
- `x` object of class `multiCUMSUM`
- `...` further arguments to be passed to and from other methods

**Value**
- **Lambda Test statistics**
  a matrix of test statistics on the presence of a number of co-break equal to `max.breaks` in the conditional mean
- **Omega Test statistics**
  a matrix of test statistics on the presence of a number of co-break equal to `max.breaks` in the conditional variance
- **Break location**
  the index and the Date where the common breaks are located

**Author(s)**
The code was written by Andrea Bucci and Giulio Palomba.

**References**
Examples

data(Realized)

testCS <- multiCUMSUM(Realized[,1:10], conf.level = 0.95)
testCS

plot

Plot methods for a VLSTAR object

Description

Plot method for objects with class attribute VLSTAR and vlstarpred.

Usage

## S3 method for class 'VLSTAR'
plot(x, names = NULL, main.fit = NULL, main.acf = NULL,
     main.pacf = NULL, main.logi = NULL,
     ylim.fit = NULL, ylim.resid = NULL, lty.fit = NULL,
     lty.resid = NULL, lty.logi = NULL,
     lwd.fit = NULL, lwd.resid = NULL, lwd.logi = NULL, lag.acf = NULL,
     lag.pacf = NULL, col.fit = NULL,
     col.resid = NULL, col.logi = NULL, ylab.fit = NULL, ylab.acf = NULL,
     ylab.pacf = NULL, ylab.logi = NULL, xlab.fit = NULL, xlab.resid = NULL,
     xlab.logi = NULL, mar = par("mar"), oma = par("oma"),
     adj.mtext = NA, padj.mtext = NA, col.mtext = NA,...)

## S3 method for class 'vlstarpred'
plot(x, type = c("single", "multiple"), names = NULL,
     main = NULL, xlab = NULL, ylab = NULL,
     lty.obs = 2, lty.pred = 1, lty.ci = 3, lty.vline = 1, lwd.obs = 1, lwd.pred = 1,
     lwd.ci = 1, lwd.vline = 1, col.obs = NULL, col.pred = NULL, col.ci = NULL,
     col.vline = NULL, ylim = NULL, mar = par("mar"), oma = par("oma"), ...)

Arguments

adj.mtext Adjustment for mtext().
col.ci Character vector, colors for the interval forecast when an object of class 'vlstarpred' is used.
col.fit Character vector, colors for diagram of fit.
col.logi Character vector, colors for logistic function plot.
col.mtext  Character, color for mtext(), only applicable.
col.obs  Character vector, colors for the observed values when an object of class ‘vlstarpred’ is used.
col.pred  Character vector, colors for the predicted values when an object of class ‘vlstarpred’ is used.
col.resid  Character vector, colors for residual plot.
col.vline  Character vector, colors for the vertical line when an object of class ‘vlstarpred’ is used.
lag.acf  Integer, lag.max for ACF of residuals.
lag.pacf  Integer, lag.max for PACF of residuals.
lty.ci  Vector, lty for the interval forecast when an object of class ‘vlstarpred’ is used.
lty.fit  Vector, lty for diagram of fit.
lty.resid  Vector, lty for residual plot.
lty.logi  Vector, lty for the plot of the logistic function.
lty.obs  Vector, lty for the plot of the observed values when an object of class ‘vlstarpred’ is used.
lty.pred  Vector, lty for the plot of the predicted values when an object of class ‘vlstarpred’ is used.
lty.vline  Vector, lty for the vertical line when an object of class ‘vlstarpred’ is used.
lwd.ci  Vector, lwd for the interval forecast when an object of class ‘vlstarpred’ is used.
lwd.fit  Vector, lwd for diagram of fit.
lwd.logi  Vector, lwd for the plot of the logistic function.
lwd.obs  Vector, lwd for the plot of the observed values when an object of class ‘vlstarpred’ is used.
lwd.pred  Vector, lwd for the plot of the predicted values when an object of class ‘vlstarpred’ is used.
lwd.resid  Vector, lwd for residual plot.
lwd.vline  Vector, lwd for the vertical line when an object of class ‘vlstarpred’ is used.
main  Character vector, the titles of the plot.
main.acf  Character vector, main for residuals’ ACF.
main.fit  Character vector, main for diagram of fit.
main.pacf  Character vector, main for residuals’ PACF.
main.logi  Character vector, main for the plot of the logistic function.
mar  Setting of margins.
names  Character vector, the variables names to be plotted. If left NULL, all variables are plotted.
oma  Setting of outer margins.
adj.mtext  Adjustment for mtext().
type       Character, if multiple all plots are drawn in a single device, otherwise the plots are shown consecutively.
x         An object of class ‘VLSTAR’ or ‘vlstarpred’.
xlab        Character vector signifying the labels for the x-axis.
xlab.fit     Character vector, xlab for diagram of fit.
xlab.resid   Character vector, xlab for residual plot.
xlab.logi    Character vector, xlab for the plot of the logistic function.
ylab        Character vector signifying the labels for the y-axis.
ylab.acf     Character, ylab for ACF.
ylab.fit     Character vector, ylab for diagram of fit.
ylab.pacf    Character, ylab for PACF.
ylab.resid   Character vector, ylab for residual plot.
ylab.logi    Character vector, ylab for the plot of the logistic function.
ylim        Vector, the limits of the y-axis.
ylim.fit     Vector, ylim for diagram of fit.
ylim.resid   Vector, ylim for residual plot.
...        Passed to internal plot function.

Details

When the plot function is applied to a VLSTAR object, the values of the logistic function, given the estimated values of gamma and c through VLSTAR, are reported.

Author(s)

Andrea Bucci

References


See Also

VLSTAR, predict.VLSTAR

Examples

##
##See 'VLSTAR' examples
##
**predict**

**VLSTAR Prediction**

**Description**

One-step or multi-step ahead forecasts, with interval forecast, of a VLSTAR object.

**Usage**

```r
## S3 method for class 'VLSTAR'
predict(object, ..., n.ahead = 1, conf.lev = 0.95,
st.new = NULL, M = 5000, B = 1000, st.num = NULL, newdata = NULL,
method = c('naive', 'Monte Carlo', 'bootstrap'))
```

**Arguments**

- `object`: An object of class ‘VLSTAR’ obtained through `VLSTAR()`.
- `...`: Further arguments to be passed to and from other methods.
- `n.ahead`: An integer specifying the number of ahead predictions.
- `conf.lev`: Confidence level of the interval forecast.
- `st.new`: Vector of new data for the transition variable.
- `M`: An integer with the number of errors sampled for the Monte Carlo method.
- `B`: An integer with the number of errors sampled for the bootstrap method.
- `st.num`: An integer with the index of dependent variable if `st.new` is NULL and the transition variable is a lag of one of the dependent variables.
- `method`: A character identifying which multi-step ahead method should be used among naive, Monte Carlo and bootstrap.
- `newdata`: Data frame or matrix of new data for the exogenous variables.

**Value**

A list containing:

- `forecasts`: Data frame of predictions for each dependent variable and the \((1 - \alpha)\) prediction intervals.

**Author(s)**

The code was written by Andrea Bucci and Eduardo Rossi.
References


See Also

VLSTAR for log-likehood and nonlinear least squares estimation of the VLSTAR model.

Examples

```r
##
##See 'VLSTAR' examples
##
```

---

**rcov**

*Realized Covariance*

Description

Function returns the vectorization of the lowest triangular of the Realized Covariance matrices for different frequencies.

Usage

```r
rcov(data, freq = c('daily', 'monthly', 'quarterly', 'yearly'), make.ret = TRUE, cholesky = FALSE)
```

Arguments

data a (T x N) xts object containing the N price/return series over period T

freq a string defining the desired frequency for the Realized Covariance matrices between "daily", "monthly", "quarterly" or "yearly"

make.ret boolean, in case it is TRUE the data are converted in returns, FALSE otherwise

cholesky boolean, in case it is TRUE the Cholesky factors of the Realized Covariance matrices are calculated, FALSE by default
Value

Realized Covariances

A $M \times (N + 1)/2$ matrix of realized covariances, where $M$ is the number of lower frequency data

Cholesky Factors (optional)

A $M \times (N + 1)/2$ matrix of Cholesky factors of the realized covariance matrices, where $M$ is the number of lower frequency data

returns (optional)

A $M \times N$ matrix of returns, when make.ret = TRUE

Author(s)

The code was written by Andrea Bucci

References


Examples

```r
data(Sample5minutes)
rc <- rcov(Sample5minutes, freq = 'daily', cholesky = TRUE, make.ret = TRUE)
rc
```

Realized

*Monthly time series used to test VLSTAR models.*

Description

This data set contains the series of realized covariances in 4 stock market indices, i.e. SP-500, Nikkei, DAX, and FTSE, Dividend Yield and Earning Price growth rate, inflation growth rates for U.S., U.K., Japan and Germany, from August 1990 to June 2018.

Usage

```r
data(Realized)
```
Format

A zoo data frame with 334 monthly observations, ranging from 1990:M8 until 2018:M6.

- **SP**: Monthly realized variances of S&P 500 index.
- **SP-NIKKEI**: Monthly realized covariances between S&P 500 and Nikkei.
- **SP-FTSE**: Monthly realized covariances between S&P 500 and FTSE.
- **SP-DAX**: Monthly realized covariances between S&P 500 and DAX.
- **NIKKEI**: Monthly realized variances of Nikkei index.
- **NIKKEI-FTSE**: Monthly realized covariances between Nikkei and FTSE.
- **NIKKEI-DAX**: Monthly realized covariances between Nikkei and DAX.
- **FTSE**: Monthly realized variances of FTSE index.
- **FTSE-DAX**: Monthly realized covariances between FTSE and DAX.
- **DAX**: Monthly realized variances of DAX index.
- **DP**: Monthly Dividends growth rate over the past year relative to current market prices; S&P 500 index.
- **EP**: Monthly Earnings growth rate over the past year relative to current market prices; S&P 500 index.
- **Inf_US**: US monthly Industrial Production growth.
- **Inf_UK**: UK monthly Industrial Production growth.
- **Inf_JPN**: Japan monthly Industrial Production growth.
- **Inf_GER**: Germany monthly Industrial Production growth.

Author(s)

Andrea Bucci

See Also

- `rcov` to build realized covariances from stock prices or returns.

---

**Sample5minutes**

*Ten simulated prices series for 19 trading days in January 2010.*

Description

Ten hypothetical price series were simulated according to the factor diffusion process discussed in Barndorff-Nielsen et al.

Usage

```r
data("Sample5minutes")
```

Format

*xts object*

Author(s)

Andrea Bucci
Summary method for objects of class VLSTAR

Description

‘summary’ methods for class ‘VLSTAR’.

Usage

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

## S3 method for class 'VLSTAR'
print.summary(x, ...)
```

Arguments

- **object**: An object of class ‘VLSTAR’ obtained through `VLSTAR()`.
- **x**: A summary object of class ‘VLSTAR’ obtained through `summary()`.
- **...**: further arguments to be passed to and from other methods

Value

A data frame of predictions for each dependent variable and the \((1-\alpha)\) prediction intervals.

Author(s)

The code was written by Andrea Bucci

References


See Also

- `VLSTAR`

Examples

```r
##
##See 'VLSTAR' examples
##
```
Description

This data set contains the series of daily prices of Google, Microsoft and Amazon stocks from January 3, 2005 to June 16, 2020, gathered from Yahoo.

Usage

data("techprices")

Format

An xts object with 3890 daily observations, ranging from from January 3, 2005 to June 16, 2020.

- Google: daily closing prices of Google (GOOG) stock.
- Microsoft: daily closing prices of Microsoft (MSFT) stock.
- Amazon: daily closing stock prices of Amazon (AMZN) stock.

Author(s)

Andrea Bucci

Description

This function allows the user to estimate the coefficients of a VLSTAR model with \( m \) regimes through maximum likelihood or nonlinear least squares. The set of starting values of Gamma and C for the convergence algorithm can be either passed or obtained via searching grid.

Usage

```
VLSTAR(y, exo = NULL, p = 1, m = 2, st = NULL, constant = TRUE, 
   starting = NULL, n.combi = NULL, 
   method = c('ML', 'NLS'), n.iter = 500, epsilon = 10^(-3))
## S3 method for class 'VLSTAR'
print(x, ...)
```
Arguments

- **y**: data.frame or matrix of dependent variables of dimension \((Txn)\)
- **exo**: (optional) data.frame or matrix of exogenous variables of dimension \((Txk)\)
- **p**: lag order
- **m**: number of regimes
- **st**: single transition variable for all the equation of dimension \((Tx1)\)
- **constant**: TRUE or FALSE to include or not the constant
- **starting**: (optional) set of initial values for Gamma and C, inserted as a list of length \(m-1\). Each element of the list should contain a data.frame with 2 columns (one for Gamma and one for c), and \(n\) rows.
- **n.combi**: Number of combination for the searching grid of Gamma and C
- **method**: Fitting method: maximum likelihood or nonlinear least squares.
- **n.iter**: number of iteration of the algorithm until forced convergence
- **epsilon**: convergence check measure
- **x**: An object of class ‘VLSTAR’ obtained through `VLSTAR()` to be printed.
- **...**: further arguments to be passed to and from other methods

Details

The multivariate smooth transition model is an extension of the smooth transition regression model introduced by Bacon and Watts (1971) (see also Anderson and Vahid, 1998). The general model is

\[
y_t = \mu_0 + \sum_{j=1}^{p} \Phi_{0,j} y_{t-j} + A_0 x_t \cdot G_t(s_t; \gamma, c) [\mu_1 + \sum_{j=1}^{p} \Phi_{1,j} y_{t-j} + A_1 x_t] + \varepsilon_t
\]

where \(\mu_0\) and \(\mu_1\) are the \(n \times 1\) vectors of intercepts, \(\Phi_{0,j}\) and \(\Phi_{1,j}\) are square \(n \times n\) matrices of parameters for lags \(j = 1, 2, \ldots, p\), \(A_0\) and \(A_1\) are \(n \times k\) matrices of parameters, \(x_t\) is the \(k \times 1\) vector of exogenous variables and \(\varepsilon_t\) is the innovation. Finally, \(G_t(s_t; \gamma, c)\) is a \(n \times n\) diagonal matrix of transition function at time \(t\), such that

\[
G_t(s_t; \gamma, c) = \{G_{1,t}(s_{1,t}; \gamma_1, c_1), G_{2,t}(s_{2,t}; \gamma_2, c_2), \ldots, G_{\tilde{n},t}(s_{\tilde{n},t}; \gamma_{\tilde{n}}, c_{\tilde{n}})\}.
\]

Each diagonal element \(G_{r,t}\) is specified as a logistic cumulative density functions, i.e.

\[
G_{r,t}(s_{r,t}; \gamma_r, c_r) = \left[1 + \exp\left\{ -\gamma_r (s_{r,t} - c_r) \right\} \right]^{-1}
\]

for latex and \(r = 0, 1, \ldots, m - 1\), so that the first model is a Vector Logistic Smooth Transition AutoRegressive (VLSTAR) model. The ML estimator of \(\theta\) is obtained by solving the optimization problem

\[
\hat{\theta}_{ML} = \arg \max_\theta logL(\theta)
\]

where \(logL(\theta)\) is the log-likelihood function of VLSTAR model, given by

\[
ll(y_t | I_t; \theta) = -\frac{T \tilde{n}}{2} \ln(2\pi) - \frac{T}{2} \ln |\Omega| - \frac{1}{2} \sum_{t=1}^{T} (y_t - \tilde{G}_t B z_t)^\prime \Omega^{-1} (y_t - \tilde{G}_t B z_t)
\]
The NLS estimators of the VLSTAR model are obtained by solving the optimization problem

$$\hat{\theta}_{NLS} = \arg \min_{\theta} \sum_{t=1}^{T} (y_t - \Psi_t' B' x_t)'(y_t - \Psi_t' B' x_t).$$

Generally, the optimization algorithm may converge to some local minimum. For this reason, providing valid starting values of $\theta$ is crucial. If there is no clear indication on the initial set of parameters, $\theta$, this can be done by implementing a grid search. Thus, a discrete grid in the parameter space of $\Gamma$ and $C$ is create to obtain the estimates of $B$ conditionally on each point in the grid. The initial pair of $\Gamma$ and $C$ producing the smallest sum of squared residuals is chosen as initial values, then the model is linear in parameters. The algorithm is the following:

1. Construction of the grid for $\Gamma$ and $C$, computing $\Psi$ for each point in the grid
2. Estimation of $\hat{B}$ in each equation, calculating the residual sum of squares, $Q_t$
3. Finding the pair of $\Gamma$ and $C$ providing the smallest $Q_t$
4. Once obtained the starting-values, estimation of parameters, $B$, via nonlinear least squares (NLS)
5. Estimation of $\Gamma$ and $C$ given the parameters found in step 4
6. Repeat step 4 and 5 until convergence.

Value

An object of class VLSTAR, with standard methods.

Author(s)

The code was written by Andrea Bucci

References


See Also

VLSTARjoint to test the presence of a unique transition variable among equations and predict.VLSTAR for details on predictions produced for this model; summary, coef, plot, loglik

Examples

data(Realized)
y <- Realized[-1,1:10]
y <- y[-nrow(y),]
st <- Realized[-nrow(Realized),1]
st <- st[-length(st)]
fit.VLSTAR <- VLSTAR(y, p = 1, n.combi = 3,
n.iter = 3, st = st, method = 'NLS')

# a few methods for VLSTAR
summary(fit.VLSTAR)
plot(fit.VLSTAR)
predict(fit.VLSTAR, newdata = Realized[nrow(Realized),1:10])
logLik(fit.VLSTAR)
coef(fit.VLSTAR)

---

**VLSTARjoint**  
*Joint linearity test*

**Description**

This function allows the user to test linearity against a Vector Smooth Transition Autoregressive Model with a single transition variable.

**Usage**

```r
VLSTARjoint(y, exo, st, st.choice = FALSE, alpha = 0.05)
## S3 method for class 'VLSTARjoint'
print(x, ...)
```

**Arguments**

- `y` data.frame or matrix of dependent variables of dimension (Txn)
- `exo` (optional) data.frame or matrix of exogenous variables of dimension (Txk)
- `st` a vector with single transition variable for all the equation of dimension (Tx1) or a matrix with R potential variables of dimension (TxR)
- `st.choice` boolean identifying whether the transition variable should be selected from a matrix of R potential variables of dimension (TxR)
- `alpha` Confidence level
- `x` `VLSTARjoint` object
- `...` further arguments to be passed to and from other methods

**Details**

Given a VLSTAR model with a unique transition variable, $s_{1t} = s_{2t} = \ldots = s_{nt} = s_t$, a generalization of the linearity test presented in Luukkonen, Saikkonen and Terasvirta (1988) may be implemented.

Assuming a 2-state VLSTAR model, such that

$$y_t = B_1 z_t + G_1 B_2 z_t + \varepsilon_t.$$
Where the null \( H_0 : \gamma_j = 0, j = 1, \ldots, \bar{n} \), is such that \( G_t \equiv (1/2)/\bar{n} \) and the previous Equation is linear. When the null cannot be rejected, an identification problem of the parameter \( \gamma_j \) in the transition function emerges, that can be solved through a first-order Taylor expansion around \( \gamma_j = 0 \).

The approximation of the logistic function with a first-order Taylor expansion is given by

\[
G(s_t; \gamma_j, c_j) = (1/2) + (1/4)\gamma_j(s_t - c_j) + r_{jt}
\]

where \( a_j = \gamma_j/4, b_j = 1/2 - a_jc_j \) and \( r_j \) is the error of the approximation. If \( G_t \) is specified as follows

\[
G_t = diag\{a_1s_t + b_1 + r_{1t}, \ldots, a_n s_t + b_n + r_{nt}\}
\]

where \( A = diag(a_1, \ldots, a_n), B = diag(b_1, \ldots, b_n) \) and \( R_t = diag(r_{1t}, \ldots, r_{nt}) \), \( y_t \) can be written as

\[
y_t = B_1z_t + (As_t + B + R_t)B_2z_t + \varepsilon_t
\]

Under the null, \( \Theta_0 = B_1 + B_2A \) and \( \varepsilon_t^* = R_tB_2 + \varepsilon_t \). It follows that the Lagrange multiplier test, under the null, is derived from the score

\[
\frac{\partial \log L(\bar{\theta})}{\partial \Theta_1} = \sum_{t=1}^{T} z_t s_t(y_t - \bar{B}_1z_t)^\Omega^{-1} = S(Y - Z\bar{B}_1)^\Omega^{-1},
\]

where

\[
S = z_t^t s_t z_t^t s_t^t
\]

and where \( \bar{B}_1 \) and \( \bar{\Omega} \) are estimated from the model in \( H_0 \). If \( P_Z = Z(Z'Z)^{-1}Z' \) is the projection matrix of \( Z \), the LM test is specified as follows

\[
LM = tr\{\bar{\Omega}^{-1}(Y - Z\bar{B}_1)'S'[I_t - P_Z]S^{-1}S'(Y - Z\bar{B}_1)\}.
\]

Under the null, the test statistics is distributed as a \( \chi^2 \) with \( \bar{n}(p \cdot \bar{n} + k) \) degrees of freedom.

**Value**

An object of class VLSTARjoint.

**Author(s)**

The code was written by Andrea Bucci

**References**


See Also

VLSTAR for log-likelihood and for NLS estimation of the VLSTAR model.

Examples

data(Realized)
VLSTARjoint(Realized[-1,1:10], Realized[-1,11:16], st=Realized[1:(nrow(Realized)-1),1])
Index

* **VLSTAR**
  - coef, 2
  - logLik, 3
  - plot, 6
  - predict, 9
  - summary, 13

* **Vector Autoregressive Smooth Transition**
  - logLik, 3
  - predict, 9
  - summary, 13

* **datasets**
  - Realized, 11
  - Sample5minutes, 12
  - techprices, 14

* **logLik**
  - logLik, 3
  - coef, 2, 16
  - coefficients (coef), 2
  - lrvarbart, 4
  - multiCUMSUM, 5
  - plot, 6, 16
  - predict, 9
  - predict.VLSTAR, 8, 16
  - print.multiCUMSUM (multiCUMSUM), 5
  - print.summary.VLSTAR (summary), 13
  - print.VLSTAR (VLSTAR), 14
  - print.VLSTARjoint (VLSTARjoint), 17
  - print.vlstarpred (predict), 9
  - rcov, 10, 12
  - Realized, 11
    - Sample5minutes, 12
    - summary, 13, 16
    - techprices, 14