Package ‘VARcpDetectOnline’

July 2, 2026

Title Sequential Change Point Detection for High-Dimensional VAR Models

Version 0.2.1

Description Implements the algorithm introduced in Tian, Y., and Safikhani, A. (2024) <doi:10.5705/ss.202024.0182>, ``Sequential Change Point Detection in High-dimensional Vector Auto-regressive Models''. This package provides tools for detecting change points in the transition matrices of VAR models, effectively identifying shifts in temporal and cross-correlations within high-dimensional time series data.

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Imports MASS, corpcor, Matrix, glmnet, doParallel, stats

Suggests ggplot2

License GPL-2 | file LICENSE

URL https://github.com/Helloworld9293/VARcpDetectOnline

BugReports https://github.com/Helloworld9293/VARcpDetectOnline/issues

NeedsCompilation no

Author Yuhan Tian [aut, cre],
Abolfazl Safikhani [aut]

Maintainer Yuhan Tian <tyh9293@gmail.com>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2026-07-02 21:50:29 UTC

Contents

fitVAR 2
generateVAR 3
get_cps 4
sp500 5
VAR_cpDetect_Online 6

Index 9

`fitVAR` *Fit VAR Model with Elastic Net via Cross Validation*

Description

Estimates a (possibly high-dimensional) VAR model using penalized least squares with an elastic net penalty and cross validation. This function is adapted from the *sparsevar* package (https://github.com/svazzole/sparsevar/tree/master), which is distributed under the GNU General Public License v2. The code has been modified to specifically implement the elastic net penalty (penalty = "ENET") and cross validation (method = "cv").

Usage

```
fitVAR(data, p = 1, ...)
```

Arguments

`data` A numeric matrix or data frame with time series data (observations in rows, variables in columns).

`p` Integer. The order of the VAR model.

`...` Additional options for estimation. Global options include:

- `threshold`: Logical. If `TRUE`, all entries smaller than the oracle threshold are set to zero.
- `scale`: Logical. Whether to scale the data (default is `FALSE`).
- `nfolds`: Integer. The number of folds used for cross validation (default is 10).
- `parallel`: Logical. If `TRUE`, use multicore backend (default is `FALSE`).
- `ncores`: Integer. If `parallel = TRUE`, specify the number of cores to use.
- `alpha`: Numeric. The elastic net mixing parameter (default is 1, i.e. LASSO).
- `type.measure`: Character. The error measure for CV (e.g., `"mse"` or `"mae"`).
- `nlambda`: Integer. The number of lambda values to use in cross validation (default is 100).
- `leaveOut`: Integer. In time slice validation, leave out the last observations (default is 15).
- `horizon`: Integer. The forecast horizon to use for estimating error (default is 1).
- `lambda`: Either a numeric vector of lambda values or a string indicating which lambda to use (default is `"lambda.min"`).
- `return_fit`: Logical. If `TRUE`, return the complete fit object.

Value

A list with the following components:

mu	A vector of means for each variable.
A	A list (of length p) of the estimated coefficient matrices for the VAR process.
fit	(Optional) The complete results of the penalized least squares estimation.
lambda	The chosen lambda value (by cross validation).
mse	The minimum mean squared error from cross validation.
mse_sd	The standard deviation of the mean squared error.
time	Elapsed time for the estimation.
series	The (possibly transformed) input time series.
residuals	The residuals of the VAR model.
sigma	The estimated variance/covariance matrix of the residuals.

References

The original source code is adapted from the sparsevar package, which is distributed under the GNU General Public License v2.

---

generateVAR *Generate VAR Data*

Description

This function generates Vector Auto-Regressive (VAR) data based on the provided parameters.

Usage

```
generateVAR(n, As, Sig, h, isOldProvided = FALSE, oldxs = NULL)
```

Arguments

n	Integer. The length of the VAR data to be generated.
As	List. A list containing the transition matrices for the VAR process.
Sig	Matrix. The covariance matrix of errors.
h	Integer. The order of the VAR process.
isOldProvided	Logical. If TRUE, the VAR data will be generated based on the last observations from the previous segment of data. Defaults to FALSE.
oldxs	Matrix. A p by h matrix containing the last observations from the previous segment of data. Required if isOldProvided = TRUE.

Value

A data matrix of dimensions p by n.

Examples

```
# Example usage
As <- list(matrix(c(0.5, 0.2, 0.1, 0.4), 2, 2))
Sig <- diag(2)
data <- generateVAR(n = 100, As = As, Sig = Sig, h = 1)
```

---

get_cps *Identify the Beginning of the Alarm Clusters*

Description

This function clusters alarms into groups and identifies the starting points of the alarm clusters. If the next alarm occurs within a specified window size (w) from the current alarm, it will be considered part of the current cluster. Otherwise, a new cluster will be formed.

Usage

```
get_cps(alarms, w)
```

Arguments

alarms	A numeric vector. The alarms raised during the monitoring process.
w	An integer. The window size used to group alarms into clusters.

Value

A numeric vector containing the starting points of the alarm clusters. If the next alarm is within w observations of the current alarm, the next alarm will be considered part of the current alarm cluster. Otherwise, a new cluster is formed and the next alarm is considered the beginning of a new alarm cluster.

Examples

```
# Example usage:
alarms <- c(10, 15, 30, 35, 60)
change_points <- get_cps(alarms, w = 10)
```

sp500 *S&P 500 Daily Log Returns and Corresponding Dates*

Description

This dataset contains daily log returns for 186 stocks in the S&P 500 index from February 6, 2004, to March 2, 2016. The daily log returns are calculated using the adjusted daily closing prices. The dataset also contains the corresponding dates for each log return.

Usage

```
data(sp500)
```

Format

A list with two elements:

`sp500$log_daily_return` A matrix with dimensions 3037 (rows, trading days) by 186 (columns, stocks).

`sp500$date` A vector of length 3037, containing the dates for each trading day.

Details

The dataset is provided as an `.RData` file containing:

- `sp500$log_daily_return`: A matrix of daily log returns with 3037 rows (trading days) and 186 columns (stocks).
- `sp500$date`: A vector of length 3037 containing the dates for each daily log return.

Source

Data from the S&P 500 stock index (2004-2016).

See Also

`VAR_cpDetect_Online`, `get_cps`

Examples

```
# Example Usage: Applying Change Point Detection to S&P 500 Data
# This is an example of how to apply the change point detection method
# (using the VAR_cpDetect_Online function) on the daily log return
# dataset from the S&P 500 (stored in the sp500 dataset). The code
# below calculates the average return volatility for all stocks, applies
# the change point detection algorithm, and plots the results with detected
# change points shown as vertical red and black lines.

# Load the dataset
data(sp500)
```

```
# Set parameters
library(ggplot2)
set.seed(2024)
n_sp <- nrow(sp500$log_daily_return)
p_sp <- ncol(sp500$log_daily_return)

# Calculate average return volatility for all data points
volatility_sum <- rep(0, (n_sp - 21))
for(col in 1:p_sp){
  temp <- as.numeric(sp500$log_daily_return[, col])
  temp1 <- rep(0, (n_sp - 21))
  for(row in 1:(n_sp - 21)){
    temp1[row] <- sd(temp[(row):(row + 21)])
  }
  volatility_sum <- volatility_sum + temp1
}
volatility_ave <- volatility_sum / p_sp

# Apply change point detection method
n0 <- 200
w <- 22
alpha <- 1 / 5000

res <- VAR_cpDetect_Online(t(sp500$log_daily_return), n0, w, alpha, 1, FALSE, TRUE, 5 / w, TRUE)
res_sp <- res$alarm_locations + n0
res_sp_cps <- res$cp_locations + n0
# Get the estimated starting points of each alarm cluster
cps_est_sp <- unique(res_sp_cps[which(res_sp %in% get_cps(res_sp, w))])

# Prepare data for plotting
y_values <- c(volatility_ave)
x_values <- sp500$date[1:(n_sp - 21)]
df <- data.frame(y_values, x_values)
plot_sp <- ggplot(df, aes(y = y_values, x = x_values)) +
  geom_line() +
  theme(legend.position = "none") +
  labs(title = "", x = "", y = "") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
 geom_vline(xintercept = sp500$date[res_sp], linetype = "solid", color = "red", alpha = .1) +
  geom_vline(xintercept = sp500$date[cps_est_sp], linetype = "solid", color = "black")

# Print the detected change points
sp500$date[cps_est_sp] # The dates for the starting of the alarm clusters
plot_sp
```

`VAR_cpDetect_Online` *VAR_cpDetect_Online: Sequential change point Detection for Vector Auto-Regressive Models*

Description

This function performs sequential change point detection in high-dimensional time series data modeled as a Vector Auto-Regressive (VAR) process, targeting changes in the transition matrices that encode temporal and cross-correlations.

Usage

```
VAR_cpDetect_Online(
  data,
  n0,
  w,
  alpha,
  h,
  RLmode = TRUE,
  needRefine = TRUE,
  refineSize = 1/5,
  needRefineCorrection = TRUE
)
```

Arguments

`data`	A matrix where rows represent different dimensions (features) and columns represent observations. The first `n0` columns are treated as historical data.
`n0`	Integer. The size of the historical data (number of columns in `data` treated as historical).
`w`	Integer. The size of the sliding window used for calculating test statistics; referred to as the pre-specified detection delay.
`alpha`	Numeric. The desired false alarm rate, where 1/alpha represents the targeted average run length (ARL), which should exceed the length of the data to be monitored.
`h`	Integer. The order of the VAR process.
`RLmode`	Logical. If `TRUE`, the algorithm terminates when the first alarm is issued.
`needRefine`	Logical. If `TRUE`, a refinement process is conducted to pinpoint the change point location.
`refineSize`	Numeric. The proportion of the new window size to the original window size, used during refinement.
`needRefineCorrection`	Logical. If `TRUE`, a confirmation step is performed during the refinement process to verify the detected change point.

Details

This function fits a VAR model to the historical data using the l1 penalty and calculates test statistics for the sliding window to detect change points. If refinement is enabled, a second step narrows down the change point location. Optionally, a correction step can verify the detected change points.

Value

A list containing:

`RL` The index (ignoring historical data) of the last observation read by the algorithm when the first alarm is issued. This is returned only if `RLmode = TRUE`.

`cp_refined` The refined estimate for the location (ignoring historical data) of the change point. This is returned only if `RLmode = TRUE` and `needRefine = TRUE`.

`alarm_locations` A vector of indices (ignoring historical data) where alarms were raised. This is returned only if `RLmode = FALSE`.

`cp_locations` A vector of refined change point locations (ignoring historical data), corresponding 1-to-1 with the `alarm_locations`. This is returned only if `RLmode = FALSE` and `needRefine = TRUE`.

Examples

```
library(MASS)
set.seed(2024)
As <- list(matrix(c(0.5, 0.2, 0.1, 0.4), 2, 2))
As_new <- list(matrix(c(-0.5, 0.2, 0.1, -0.4), 2, 2))
Sig <- diag(2)
data_IC <- generateVAR(n = 400, As = As, Sig = Sig, h = 1)
data_OC <- generateVAR(n = 100, As = As_new, Sig = Sig, h = 1,
                       isOldProvided = TRUE, oldxs = data_IC[, ncol(data_IC)])
data <- cbind(data_IC, data_OC)
result <- VAR_cpDetect_Online(data, n0 = 300, w = 20, alpha = 1/200, h = 1,
                              RLmode = TRUE, needRefine = TRUE, refineSize = 1/5,
                              needRefineCorrection = TRUE)
print(result)
```

Index

fitVAR, 2

generateVAR, 3
get_cps, 4, *5*

sp500, 5

VAR_cpDetect_Online, *5*, 6