Package: VARcpDetectOnline (via r-universe)

January 10, 2025

Title Sequential Change Point Detection for High-Dimensional VAR Models

Version 0.1.0

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 Vector Auto-Regressive (VAR) models, effectively identifying shifts in temporal and cross-correlations within high-dimensional time series data. The package includes functions to generate synthetic VAR data, detect change points in high-dimensional time series, and analyze real-world data. It also demonstrates an application to financial data: the daily log returns of 186 S&P 500 stocks from 2004-02-06 to 2016-03-02.

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports MASS, sparsevar

License MIT + file LICENSE

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

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

Config/pak/sysreqs libicu-dev

Repository https://helloworld9293.r-universe.dev

RemoteUrl https://github.com/helloworld9293/varcpdetectonline

RemoteRef HEAD

RemoteSha db0f21dc0bfbcc0db8d8a150f180c812412afe1c

Contents

`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.
## Not run:
# 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

## End(Not run)
```

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(sparsevar)
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