Package: FluxPoint (via r-universe)

May 10, 2026

Type Package

Title Change Point Detection for Non-Stationary and Cross-Correlated Time Series

Version 0.1.2

Maintainer Yuhan Tian <yuhan.tian@fau.de>

Description Implements methods for multiple change point detection in multivariate time series with non-stationary dynamics and cross-correlations. The methodology is based on a model in which each component has a fluctuating mean represented by a random walk with occasional abrupt shifts, combined with a stationary vector autoregressive structure to capture temporal and cross-sectional dependence. The framework is broadly applicable to correlated multivariate sequences in which large, sudden shifts occur in all or subsets of components and are the primary targets of interest, whereas small, smooth fluctuations are not. Although random walks are used as a modeling device, they provide a flexible approximation for a wide class of slowly varying or locally smooth dynamics, enabling robust performance beyond the strict random walk setting.

License GPL-2

Encoding UTF-8

Imports blockmatrix, corpcor, doParallel, ggplot2, glmnet, MASS, Matrix, nnls, pracma, SimDesign

RoxygenNote 7.3.3

NeedsCompilation no

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

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

Date/Publication 2026-01-10 14:40:20 UTC

RemoteUrl https://github.com/cran/FluxPoint

RemoteRef HEAD

RemoteSha 87aa8657841188ab13fe8dc307ab8a64254a174f

Contents

`add_jumps` *Add mean shifts to multivariate time series data*

Description

Adds constant mean shifts to a multivariate time series by applying a fixed jump vector at evenly spaced change points. After each change point, all subsequent observations are shifted by the specified vector.

Usage

```
add_jumps(data, delta, num)
```

Arguments

`data`	Numeric matrix of dimension $n \times p$, representing the time series data.
`delta`	Numeric vector of length p, specifying the shift magnitudes added to each variable after each change point.
`num`	Integer; number of change points. The data are divided evenly into `num` + 1 segments, and `delta` is added cumulatively after each change point.

Details

The total length of the time series is denoted by n. Change points are placed at evenly spaced locations given by $k\lfloor n/(num+1)\rfloor$, for $k = 1, \ldots, num$. After each change point, a constant shift vector `delta` is added to all subsequent observations. This construction produces synthetic data with known and controlled mean shifts, making the function useful for simulation studies and benchmarking change point detection methods.

Value

A numeric matrix of the same dimension as ‘data‘, containing the adjusted series with added mean shifts.

`estimate_musseg` *Estimate fluctuating mean segmentwise given detected change points*

Description

Estimates the fluctuating mean sequence $\{\boldsymbol{\mu}_t\}_{t=1}^n$ segmentwise by applying the maximum likelihood estimation (MLE) procedure within each segment defined by detected change points.

Usage

```
estimate_musseg(data, cps, Sig_eta, Sig_nu, Phi, Sig_e1)
```

Arguments

`data`	Numeric matrix of dimension $n \times p$, representing the multivariate time series $\{\mathbf{y}_t\}_{t=1}^n$.
`cps`	Numeric vector of detected change point locations (sorted indices).
`Sig_eta`	Numeric $p \times p$ covariance matrix $\Sigma_{\boldsymbol{\eta}}$ of the random walk innovation.
`Sig_nu`	Numeric $p \times p$ covariance matrix $\Sigma_{\boldsymbol{\nu}}$ of the VAR(1) innovation.
`Phi`	Numeric $p \times p$ autoregressive coefficient matrix Φ.
`Sig_e1`	Numeric $p \times p$ initial-state covariance matrix $\Gamma_{\boldsymbol{\epsilon}}(0)$.

Details

The time series is partitioned into contiguous segments defined by the specified change points. Within each segment, `estimate_mus` is applied to obtain the maximum likelihood estimate of the fluctuating mean sequence for that interval. The resulting segment-wise estimates are then concatenated to form a complete piecewise estimate of $\boldsymbol{\mu}_t$ over the entire time series.

Value

A numeric matrix of dimension $n \times p$, containing the estimated fluctuating mean sequence across all segments.

Examples

```
set.seed(123)
p <- 3
mu0 <- rep(0, p)
deltas <- list(c(3, 0, -3), c(-2, 4, 0))
Sig_eta <- diag(0.01, p)
Sig_nu  <- random_Signu(p, 0)
```

```
Phi     <- random_Phi(p, p)
Sig_e1  <- get_Sig_e1_approx(Sig_nu, Phi)

# Generate data and estimate mean segmentwise after known CPs
Y <- generate_data(mu0, deltas, Sig_eta, Sig_nu, Phi, Sig_e1,
                   errortype = "n", number_cps = 2, lengthofeachpart = 100)
cps <- c(100, 200)
mu_seg <- estimate_musseg(Y, cps, Sig_eta, Sig_nu, Phi, Sig_e1)
dim(mu_seg)
```

`estimate_RWVAR_cp_heter` *Robust parameter estimation (RPE) for multivariate time series*

Description

Applies the robust parameter estimation (RPE) procedure componentwise to a multivariate time series in order to estimate the diagonal elements of $\Sigma_{\boldsymbol{\eta}}$, $\Sigma_{\boldsymbol{\nu}}$, and Φ.

Usage

```
estimate_RWVAR_cp_heter(
  data,
  L = 15,
  phiLower = -0.8,
  phiUpper = 0.8,
  sigetaLower = 0,
  sigetaUpper = Inf,
  signuLower = 1e-06,
  signuUpper = Inf,
  num_inis = 20,
  CPs = NULL
)
```

Arguments

`data`	Numeric matrix of dimension $n \times p$, representing the multivariate time series $\{\mathbf{y}_t\}_{t=1}^{n}$.
`L`	Integer; number of lag differences used in each univariate RPE estimation (default = 15).
`phiLower`, `phiUpper`	Numeric; lower and upper bounds for the autoregressive coefficient ϕ.
`sigetaLower`, `sigetaUpper`	Numeric; lower and upper bounds for σ_η^2, the random walk innovation variance.
`signuLower`, `signuUpper`	Numeric; lower and upper bounds for σ_ν^2, the VAR(1) innovation variance.

num_inis	Integer; number of initial values of ϕ used for grid search initialization (default = 20).
CPs	Optional numeric vector of change point locations (indices). If provided, differenced data overlapping these points are removed for more robust estimation.

Details

This function performs the RPE procedure for each variable (column) in 'data' independently, using estimate_RWVAR_cp as the univariate estimator. The resulting estimates are combined into diagonal matrices:

- $\Sigma_{\boldsymbol{\nu}}$ — estimated innovation covariance of the VAR(1) component.
- $\Sigma_{\boldsymbol{\eta}}$ — estimated innovation covariance of the random walk component.
- Φ — estimated autoregressive coefficient matrix.

Value

A list containing:

- 'Sig_nu' — Diagonal matrix of estimated $\sigma^2_{\nu,i}$.
- 'Sig_eta' — Diagonal matrix of estimated $\sigma^2_{\eta,i}$.
- 'Phi' — Diagonal matrix of estimated autoregressive coefficients ϕ_i.

Examples

```
set.seed(123)
p <- 3

# True (diagonal) parameters for simulation
mu0    <- rep(0, p)
Sig_eta <- diag(0.01, p)
Sig_nu  <- random_Signu(p, 0)   # diagonal here since num_nonzero = 0
Phi     <- random_Phi(p, 0)     # diagonal here since num_nonzero = 0
Sig_e1  <- get_Sig_e1_approx(Sig_nu, Phi)

# Two evenly spaced change points
deltas <- list(c(3, 0, -3), c(-2, 4, 0))
Y <- generate_data(mu0, deltas, Sig_eta, Sig_nu, Phi, Sig_e1,
                   errortype = "n", number_cps = 2, lengthofeachpart = 100)

# Provide CP locations to remove affected differences in RPE
CPs <- c(100, 200)

# Componentwise robust parameter estimation
fit <- estimate_RWVAR_cp_heter(Y, L = 15, CPs = CPs)

# Estimated diagonal matrices:
fit$Sig_eta
fit$Sig_nu
fit$Phi
```

`estimatePhinu_nondiag` *Estimate non-diagonal VAR(1) parameters after mean removal*

Description

Estimates the non-diagonal autoregressive coefficient matrix Φ and innovation covariance matrix $\Sigma_{\boldsymbol{\nu}}$ for the residual process obtained after removing the estimated fluctuating mean from the data. The estimation applies the Lasso to encourage sparsity in the cross-variable dependence structure.

Usage

```
estimatePhinu_nondiag(
  epsilons,
  Sig_nu_diag,
  Phi_diag,
  replace_diag = FALSE,
  needReproduce = FALSE
)
```

Arguments

`epsilons`	Numeric matrix of dimension $n \times p$, representing the estimated residuals $\boldsymbol{\epsilon}_t = \mathbf{y}_t - \hat{\boldsymbol{\mu}}_t$.
`Sig_nu_diag`	Numeric $p \times p$ diagonal matrix providing initial (diagonal) estimates of $\Sigma_{\boldsymbol{\nu}}$.
`Phi_diag`	Numeric $p \times p$ diagonal matrix providing initial (diagonal) estimates of Φ.
`replace_diag`	Logical; if `TRUE`, replaces the diagonal entries of the estimated matrices with those from `Sig_nu_diag` and `Phi_diag` (default `FALSE`).
`needReproduce`	Logical; if `TRUE`, uses fixed fold assignments in cross-validation to ensure reproducibility (default `FALSE`).

Details

The function applies a Lasso-penalized VAR(1) fit to the residual process $\boldsymbol{\epsilon}_t$ to estimate cross-dependencies among variables. The fitting is performed using the function `fitVAR()`, which is adapted from the **sparsevar** package. When `replace_diag = TRUE`, the diagonal entries of Φ and $\Sigma_{\boldsymbol{\nu}}$ are replaced by their componentwise estimates obtained in Phase I for improved numerical stability.

Value

A list containing:

- 'Phi_hat' — Estimated non-diagonal autoregressive matrix Φ.
- 'Sig_nu_hat' — Estimated non-diagonal innovation covariance matrix $\Sigma_{\boldsymbol{\nu}}$.

FluxPoint *FluxPoint change point detection algorithm*

Description

Implements the full FluxPoint algorithm for detecting multiple change points in multivariate time series with non-stationary dynamics and cross-correlations. The procedure iteratively estimates model parameters and change point locations, alternating between parameter estimation and detection steps until convergence.

Usage

```
FluxPoint(
  data,
  w,
  tc,
  max_iter1,
  max_iter2,
  ignoreCross = FALSE,
  noeta = FALSE,
  nophi = FALSE,
  needReproduce = FALSE
)
```

Arguments

data	Numeric matrix of dimension $n \times p$ containing the observed multivariate time series.
w	Integer specifying the window size used by the detector.
tc	Numeric tuning constant used in the detection threshold $D = \texttt{tc} \cdot \min(4, \log(e^2 + p)) \cdot \log(n - w)$.
max_iter1	Integer specifying the maximum number of iterations for the first-stage loop, which alternates between diagonal robust parameter estimation and change point detection.
max_iter2	Integer specifying the maximum number of iterations for the second-stage refinement loop, which incorporates non-diagonal vector autoregressive updates.
ignoreCross	Logical; if TRUE, the algorithm terminates after the first stage and treats the components of the time series as independent.
noeta	Logical; if TRUE, forces $\Sigma_{\boldsymbol{\eta}} = 0$ and performs change point detection without accounting for random walk fluctuations in the mean.
nophi	Logical; if TRUE, forces $\Phi = 0$ and performs change point detection without accounting for temporal dependence. This option should only be used when ignoreCross = TRUE.
needReproduce	Logical; if TRUE, fixed folds are used in internal cross-validation steps to improve reproducibility.

Details

The algorithm proceeds through the following stages:

1. *Stage I (diagonal estimation):* Robust parameter estimation is performed to obtain diagonal estimates of $\Sigma_{\boldsymbol{\eta}}$, $\Sigma_{\boldsymbol{\nu}}$, and Φ. These estimates are used to construct the windowed covariance matrix Σ^*_{ALL} and its inverse. Change point detection is then carried out using the resulting detector statistic. The estimation and detection steps are iterated until the detected change points stabilize or `max_iter1` is reached.
2. *Stage II (refinement with cross-correlation):* If enabled, the fluctuating mean is estimated segmentwise and removed from the data. A sparse vector autoregressive model is then fitted to the residuals to obtain non-diagonal estimates of Φ and $\Sigma_{\boldsymbol{\nu}}$. The covariance matrix Σ^*_{ALL} is recomputed and change point detection is rerun. This refinement loop is repeated until convergence or until `max_iter2` is reached.

Value

A list containing:

- `cps`: Sorted indices of the detected change points.
- `Sig_eta_hat`: Final estimate of $\Sigma_{\boldsymbol{\eta}}$.
- `Sig_nu_hat`: Final estimate of $\Sigma_{\boldsymbol{\nu}}$, which may be non-diagonal if the second-stage refinement is performed.
- `Phi_hat`: Final estimate of Φ, which may be non-diagonal if the second-stage refinement is performed.
- `muhats`: Estimated fluctuating mean sequence.
- `detectorStats`: Detector statistic evaluated over time.
- `cps_at`: A list mapping each detected change point to the indices of components selected as contributing to that change.

References

Tian, Y. and Safikhani, A. (2025). Multiple change point detection in time series with non-stationary dynamics. Manuscript under review.

Examples

```
## Minimal runnable example (fast)
set.seed(123)
p <- 1
mu0 <- rep(0, p)
deltas <- list(c(3), c(-3))
Sig_eta <- diag(0.01, p)
Sig_nu  <- random_Signu(p, 0)
Phi     <- random_Phi(p, 0)
Sig_e1  <- get_Sig_e1_approx(Sig_nu, Phi)

# Simulate data with two evenly spaced change points
Y <- generate_data(mu0, deltas, Sig_eta, Sig_nu, Phi, Sig_e1,
```

```
                    errortype = "n", number_cps = 2, lengthofeachpart = 100)

# Run the algorithm
out <- FluxPoint(Y, w = 20, tc = 1, max_iter1 = 5, max_iter2 = 5)
out$cps
# Visualization
p1 <- plot_FluxPoint(Y, out$muhats, out$cps, titlename = "", xaxis = "Time")
print(p1)


## More realistic example (may take longer)
set.seed(123)
p <- 3
mu0 <- rep(0, p)
deltas <- list(c(3, 0, -3), c(0, -2, 4))
Sig_eta <- diag(0.01, p)
Sig_nu  <- random_Signu(p, 0)
Phi     <- random_Phi(p, p)
Sig_e1  <- get_Sig_e1_approx(Sig_nu, Phi)

Y <- generate_data(mu0, deltas, Sig_eta, Sig_nu, Phi, Sig_e1,
                    errortype = "n", number_cps = 2, lengthofeachpart = 100)

out <- FluxPoint(Y, w = 20, tc = 1, max_iter1 = 5, max_iter2 = 5)
out$cps

# Visualization
p1 <- plot_FluxPoint(Y, out$muhats, out$cps, titlename = "", xaxis = "Time")
print(p1)
```

`FluxPoint_raw` *Core change point detection algorithm (given known parameters)*

Description

Implements the core step of the proposed change point detection (CPD) algorithm to estimate the locations of change points, given the inverse windowed covariance $\Sigma^{*-1}_{\mathrm{ALL}}$. The method computes detector statistics over a moving window using a projection-based quadratic form and identifies candidate change points via peak detection.

Usage

```
FluxPoint_raw(data, Sig_all_inv, w, D, needReproduce = FALSE)
```

Arguments

`data`	Numeric matrix of dimension $n \times p$, the multivariate time series.

`Sig_all_inv`	Numeric matrix of dimension $(pw) \times (pw)$, the inverse of the combined covariance Σ^*_{ALL} (accounts for random walk and VAR(1) effects within a window of size w).
`w`	Integer; window size used in the moving-window detection step.
`D`	Numeric; detection threshold used in the peak-finding step.
`needReproduce`	Logical; if `TRUE`, a fixed fold assignment is used in cross-validation to ensure reproducibility (default `FALSE`).

Details

For each center index k, a window of width w is formed and contrast vectors are constructed to compare the first and second halves of the window. Before computing the detector statistic, a component-selection step is performed using an ℓ_1-penalized regression (lasso, via **glmnet**) with weights $\Sigma^{*-1}_{\mathrm{ALL}}$ to identify variables that exhibit a shift. The resulting active set determines the projection used in the statistic. Sparse projection matrices indexed by the active-set pattern are cached and reused for computational efficiency. The detector statistic is defined as a weighted quadratic form measuring deviation from the baseline (no-change) projection, and locations at which the statistic exceeds the threshold `D` are declared as estimated change points.

Value

A list with:

- 'shiftIndices' — Binary matrix ($n \times p$) indicating selected components at each index.
- 'detectorStats' — Numeric vector of detector values over time.
- 'Projection_list' — Cache of projection matrices by active-set pattern.
- 'cps' — Indices of detected change points.

Examples

```
## Minimal runnable example (fast)
set.seed(123)
p <- 1
mu0 <- rep(0, p)
deltas <- list(c(3), c(4))
Sig_eta <- diag(0.01, p)
Sig_nu  <- random_Signu(p, 0)
Phi     <- random_Phi(p, 0)
Sig_e1  <- get_Sig_e1_approx(Sig_nu, Phi)

# Simulate data with two evenly spaced change points
Y <- generate_data(mu0, deltas, Sig_eta, Sig_nu, Phi, Sig_e1,
                   errortype = "n", number_cps = 2,
                   lengthofeachpart = 100)

# Windowed covariance and its inverse
w <- 20
Sigs <- get_Sigs(w, p, Sig_eta, Sig_nu, Phi, Sig_e1)
Sig_all_inv <- Sigs$Sig_all_inv
```

```
# Run detector with a common threshold choice
n <- nrow(Y)
D <- min(4, log(exp(2) + p)) * log(n - w)
res <- FluxPoint_raw(Y, Sig_all_inv, w, D)
res$cps


## More realistic example (may take longer)
set.seed(123)
p <- 3
mu0 <- rep(0, p)
deltas <- list(c(3, 0, -3), c(0, -2, 4))
Sig_eta <- diag(0.01, p)
Sig_nu  <- random_Signu(p, 0)
Phi     <- random_Phi(p, p)
Sig_e1  <- get_Sig_e1_approx(Sig_nu, Phi)

Y <- generate_data(mu0, deltas, Sig_eta, Sig_nu, Phi, Sig_e1,
                   errortype = "n", number_cps = 2,
                   lengthofeachpart = 100)

w <- 20
Sigs <- get_Sigs(w, p, Sig_eta, Sig_nu, Phi, Sig_e1)
Sig_all_inv <- Sigs$Sig_all_inv

n <- nrow(Y)
D <- min(4, log(exp(2) + p)) * log(n - w)
res <- FluxPoint_raw(Y, Sig_all_inv, w, D)
res$cps
```

`generate_data` *Generate multivariate time series from the proposed model*

Description

Simulates a multivariate time series following the proposed model structure, where the mean component evolves as a random walk with abrupt shifts, overlaid by a stationary VAR(1) process to account for temporal and cross-sectional correlations.

Specifically, at each time point $t = 1, \ldots, n$, the data are generated as

$$\mathbf{y}_t = \boldsymbol{\mu}_t + \boldsymbol{\epsilon}_t,$$

where, for $t = 2, \ldots, n$,

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \boldsymbol{\eta}_t + \boldsymbol{\delta}_t,$$

and

$$\boldsymbol{\epsilon}_t = \Phi \boldsymbol{\epsilon}_{t-1} + \boldsymbol{\nu}_t.$$

Here, $\boldsymbol{\eta}_t$ denotes the random walk innovation with covariance $\Sigma_{\boldsymbol{\eta}}$, and $\boldsymbol{\nu}_t$ is the VAR(1) innovation with covariance $\Sigma_{\boldsymbol{\nu}}$. The vector $\boldsymbol{\delta}_t$ is nonzero only at change points.

Usage

```
generate_data(
  mu0,
  deltas,
  Sig_eta,
  Sig_nu,
  Phi,
  Sig_e1,
  errortype,
  df = 10,
  number_cps,
  lengthofeachpart
)
```

Arguments

`mu0`	Numeric vector of length p. The initial mean vector $\boldsymbol{\mu}_0$.
`deltas`	A list of numeric vectors, each representing the jump magnitude $\boldsymbol{\delta}_t$ at a change point.
`Sig_eta`	Numeric $p \times p$ covariance matrix $\Sigma_{\boldsymbol{\eta}}$ of the random walk innovation.
`Sig_nu`	Numeric $p \times p$ covariance matrix $\Sigma_{\boldsymbol{\nu}}$ of the VAR(1) innovation.
`Phi`	Numeric $p \times p$ autoregressive coefficient matrix Φ.
`Sig_e1`	Numeric $p \times p$ initial-state covariance matrix of the VAR(1) process.
`errortype`	Character; either "n" (Gaussian) or "t" (Student's t) specifying the distribution of the innovations.
`df`	Degrees of freedom for the t-distribution (used only when 'errortype = "t"'). Default is 10.
`number_cps`	Integer; number of change points (m).
`lengthofeachpart`	Integer; number of observations between consecutive change points ($\tau_{k+1} - \tau_k$).

Details

The total length of the time series is given by $n = (number\_cps + 1) \times lengthofeachpart$, so that the specified change points partition the data into equally sized segments. When $\Sigma_{\boldsymbol{\eta}} = 0$, the model reduces to a piecewise constant mean process with no random walk component. When $\Phi = 0$, the process reduces to a random walk model without vector autoregressive dependence. If both $\Sigma_{\boldsymbol{\eta}} = 0$ and $\Phi = 0$, the model simplifies to the classical piecewise constant setting commonly used in multiple change point analysis. The two innovation components are generated independently.

The innovations $\boldsymbol{\eta}_t$ and $\boldsymbol{\nu}_t$ are drawn either from a multivariate normal distribution (when `errortype = "n"`) using `mvrnorm`, or from a multivariate Student's t distribution (when `errortype = "t"`) using `rmvt`.

Value

A numeric matrix of dimension $n \times p$, with $n = (number\_cps + 1)\, lengthofeachpart$, containing the simulated observations $\{\mathbf{y}_t\}_{t=1}^{n}$.

Examples

```
set.seed(123)
p <- 3
mu0 <- rep(0, p)
deltas <- list(c(3, 0, -3), c(-2, 4, 0))
Sig_eta <- diag(0.01, p)
Sig_nu  <- random_Signu(p, 0)
Phi <- random_Phi(p, p)
Sig_e1 <- get_Sig_e1_approx(Sig_nu, Phi)

Y <- generate_data(mu0, deltas, Sig_eta, Sig_nu, Phi, Sig_e1,
                   errortype = "n", number_cps = 2, lengthofeachpart = 100)
dim(Y)
```

`get_metrics` *Evaluate change point detection accuracy metrics*

Description

Computes standard evaluation metrics — bias, precision, recall, and F1-score — for change point detection results by comparing estimated change points against true ones within a specified tolerance (acceptance radius).

Usage

```
get_metrics(n, num_cps, est_cps, accept_radius)
```

Arguments

`n`	Integer; total series length.
`num_cps`	Integer; true number of change points.
`est_cps`	Numeric vector of estimated change point locations.
`accept_radius`	Numeric; tolerance radius within which an estimated change point is considered correctly detected (a true positive).

Details

True change points are assumed to occur at evenly spaced positions. An estimated change point is counted as a true positive if it lies within `accept_radius` of any true change point location. Estimated points that do not match any true change point are classified as false positives, while true change points that are not detected are counted as false negatives. Bias is defined as the absolute difference between the true and estimated numbers of change points.

The metrics are defined as:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Value

A list containing:

- 'bias' — Absolute difference between true and estimated number of change points.
- 'precision' — Proportion of correctly detected change points among all detections.
- 'recall' — Proportion of true change points correctly detected.
- 'F1' — Harmonic mean of precision and recall.

`get_Sig_e1_approx` *Approximate the long-run covariance matrix* $\Gamma\_\boldsymbol{\epsilon}(0)$

Description

Computes an approximate long-run covariance matrix $\Gamma_{\boldsymbol{\epsilon}}(0)$ for the stationary VAR(1) process

$$\boldsymbol{\epsilon}_t = \Phi\boldsymbol{\epsilon}_{t-1} + \boldsymbol{\nu}_t,$$

where $\boldsymbol{\nu}_t$ has innovation covariance $\Sigma_{\boldsymbol{\nu}}$. The approximation is obtained via a truncated series expansion up to order 'm'.

Usage

```
get_Sig_e1_approx(Sig_nu, Phi, m = 6)
```

Arguments

`Sig_nu`	Numeric $p \times p$ matrix representing the innovation covariance $\Sigma_{\boldsymbol{\nu}}$.
`Phi`	Numeric $p \times p$ autoregressive coefficient matrix Φ.
`m`	Integer (default = 6). Number of lag terms used in the truncated series expansion. Larger values yield higher accuracy but increase computation time.

Details

For a stable VAR(1) process, the theoretical long-run covariance satisfies

$$\mathrm{vec}(\Gamma_{\boldsymbol{\epsilon}}(0)) = (I_{p^2} - \Phi \otimes \Phi)^{-1}\mathrm{vec}(\Sigma_{\boldsymbol{\nu}}).$$

To avoid matrix inversion, this function approximates the inverse by the truncated Neumann series:

$$(I_{p^2} - \Phi \otimes \Phi)^{-1} = I_{p^2} + \sum_{i=1}^{m}(\Phi^{\otimes i}),$$

where $\Phi^{\otimes i}$ denotes the Kronecker product of Φ^i with itself. The truncation level 'm' controls the approximation accuracy.

Value

A numeric $p \times p$ matrix giving the approximate $\Gamma_{\boldsymbol{\epsilon}}(0)$.

get_Sigs *Compute the covariance matrix* Σ_ALL$\hat{}*$ *for observations within a moving window*

Description

Calculates the covariance matrix Σ^*_{ALL} for all observations within a moving window of length w, given the random walk innovation covariance $\Sigma_{\boldsymbol{\eta}}$, the VAR(1) innovation covariance $\Sigma_{\boldsymbol{\nu}}$, the autoregressive coefficient matrix Φ, and the initial-state covariance matrix $\Gamma_{\boldsymbol{\epsilon}}(0)$ (denoted here by 'Sig_e1'). The resulting matrix accounts for both the random walk component and the temporal dependence introduced by the VAR(1) structure.

Usage

```
get_Sigs(w, p, Sig_eta, Sig_nu, Phi, Sig_e1)
```

Arguments

w	Integer; window size.
p	Integer; data dimension.
Sig_eta	Numeric $p \times p$ matrix representing the covariance of the random walk innovation $\Sigma_{\boldsymbol{\eta}}$.
Sig_nu	Numeric $p \times p$ matrix representing the covariance of the VAR(1) innovation $\Sigma_{\boldsymbol{\nu}}$.
Phi	Numeric $p \times p$ autoregressive coefficient matrix Φ.
Sig_e1	Numeric $p \times p$ matrix representing the covariance of the initial state $\Gamma_{\boldsymbol{\epsilon}}(0)$.

Details

The function decomposes the overall covariance matrix Σ^*_{ALL} into two additive components corresponding to the random walk contribution Σ_{RW} and the autoregressive contribution Σ_{AR}, so that

$$\Sigma^*_{\mathrm{ALL}} = \Sigma_{\mathrm{RW}} + \Sigma_{\mathrm{AR}}.$$

When $p = 1$, the construction reduces to the scalar random walk and AR(1) case, for which closed-form covariance expressions are available. For higher-dimensional settings, block-matrix structures are constructed using functions from the **blockmatrix** package to capture both cross-sectional and temporal dependence. The returned inverse matrix $(\Sigma^*_{\mathrm{ALL}})^{-1}$ is used in the main change point detection algorithm to adjust for the effects of random walk drift and vector autoregressive correlation.

Value

A list with the following components:

- 'Sig_AR' — Covariance contribution from the VAR(1) component (Σ_{AR}).
- 'Sig_RW' — Covariance contribution from the random walk component (Σ_{RW}).
- 'Sig_all' — Combined covariance matrix ($\Sigma^*_{\mathrm{ALL}} = \Sigma_{\mathrm{AR}} + \Sigma_{\mathrm{RW}}$).
- 'Sig_all_inv' — Inverse of the combined covariance matrix $(\Sigma^*_{\mathrm{ALL}})^{-1}$.

Examples

```
set.seed(1)
p <- 3
w <- 20
Sig_eta <- diag(0.01, p)
Sig_nu  <- random_Signu(p, 0)
Phi <- random_Phi(p, p)
Sig_e1 <- get_Sig_e1_approx(Sig_nu, Phi)
res <- get_Sigs(w, p, Sig_eta, Sig_nu, Phi, Sig_e1)
```

`plot_FluxPoint` *Plot multivariate time series with detected change points and estimated means*

Description

Visualizes multivariate time series data together with the estimated fluctuating mean sequence and detected change points obtained from the proposed change point detection (CPD) algorithm. Each variable is plotted in a separate panel (facet), with vertical dashed lines marking detected change points and solid curves showing the estimated means when provided.

Usage

```
plot_FluxPoint(data, muhats, cps, titlename = "", xaxis = "")
```

Arguments

`data`	Numeric matrix of dimension $n \times p$, representing the observed multivariate time series $\{\mathbf{y}_t\}_{t=1}^n$.
`muhats`	Optional numeric matrix of the same dimension as 'data', giving the estimated fluctuating mean sequence $\{\hat{\boldsymbol{\mu}}_t\}_{t=1}^n$. If `NULL`, only raw data and detected change points are shown.
`cps`	Numeric vector of detected change point locations.
`titlename`	Character string for the plot title.
`xaxis`	Character string for the x-axis label (e.g., "Time").

Details

When $p = 1$, the function produces a single plot displaying the observed time series, the estimated mean curve, and vertical dashed lines indicating the detected change points. When $p > 1$, each variable is shown in a separate facet with independently scaled y-axes for improved readability. If `muhats` is provided, the estimated mean is overlaid using `geom_line()`; otherwise, only the observed data and detected change points are displayed.

Value

A **ggplot2** object displaying the time series, estimated means (if provided), and detected change points.

`random_Phi` *Randomly generate an autoregressive coefficient matrix* Φ

Description

Generates a $p \times p$ autoregressive coefficient matrix Φ for the VAR(1) component in the proposed model. The diagonal entries are randomly chosen from {0.5, -0.5}, and a specified number of off-diagonal elements are randomly assigned nonzero values to introduce cross-dependence among variables.

Usage

```
random_Phi(p, num_nonzero)
```

Arguments

`p`	Integer. Dimension of the square matrix (p variables).
`num_nonzero`	Integer. Target number of nonzero off-diagonal entries in Φ.

Details

The diagonal elements are sampled independently from the set $\{0.5, -0.5\}$. Nonzero off-diagonal entries are then placed at random positions until the total number of nonzero off-diagonal elements reaches at least `num_nonzero`. Each nonzero off-diagonal element has magnitude 0.1 or 0.2 with equal probability and a randomly assigned sign. The resulting matrix Φ governs the temporal dependence of the stationary VAR(1) process

$$\boldsymbol{\epsilon}_t = \Phi \boldsymbol{\epsilon}_{t-1} + \boldsymbol{\nu}_t.$$

Value

A numeric $p \times p$ matrix representing the autoregressive coefficient matrix Φ with random diagonal entries in {0.5, -0.5} and approximately ‘num_nonzero‘ nonzero off-diagonal elements.

`random_Signu` *Randomly generate an innovation covariance matrix* $\Sigma_{\boldsymbol{\nu}}$

Description

Generates a symmetric $p \times p$ innovation covariance matrix $\Sigma_{\boldsymbol{\nu}}$ for the VAR(1) component in the proposed model. The diagonal elements are fixed at 0.5, and a specified number of off-diagonal elements are randomly assigned nonzero values to introduce cross-correlation between variables.

Usage

```
random_Signu(p, num_nonzero)
```

Arguments

`p`	Integer. Dimension of the covariance matrix (p variables).
`num_nonzero`	Integer. Target number of nonzero off-diagonal entries (counted individually; both upper and lower triangles are included). Since nonzero values are inserted in symmetric pairs, an even value is recommended. The maximum meaningful value is $p(p-1)$.

Details

Each nonzero off-diagonal entry is placed in symmetric pairs (i, j) and (j, i) to ensure symmetry of the matrix. The magnitudes of the nonzero entries are randomly drawn from the set $\{0.1, 0.2\}$ with randomly assigned signs. The diagonal entries are fixed at 0.5 to maintain a moderate level of innovation variance.

In the full model, $\Sigma_{\boldsymbol{\nu}}$ governs the variability of the VAR(1) innovation term $\boldsymbol{\nu}_t$ in $\boldsymbol{\epsilon}_t = \Phi\boldsymbol{\epsilon}_{t-1} + \boldsymbol{\nu}_t$.

Value

A numeric symmetric matrix of dimension $p \times p$ representing $\Sigma_{\boldsymbol{\nu}}$ with diagonal 0.5 and approximately ‘num_nonzero‘ nonzero off-diagonal entries.

Index