Package 'SChangeBlock'

May 7, 2026

Title Spatial Structural Change Detection by an Analysis of Variability Between Blocks of Observations

Version 0.1.0

Description Provides methods to detect structural changes in time series or random fields (spatial data). Focus is on the detection of abrupt changes or trends in independent data, but the package also provides a function to de-correlate data with dependence. The functions are based on the test suggested in Schmidt (2024) <DOI:10.3150/23-BEJ1686> and the work in Görz and Fried (2025) <DOI:10.48550/arXiv.2512.11599>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports Rcpp, ggplot2, nortest, expm, robcp

Depends R (>= 3.3.1)

LinkingTo Rcpp

Author Sheila Goerz [aut, cre], Roland Fried [ctb, ths]

Maintainer Sheila Goerz <sheila.goerz@tu-dortmund.de>

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-01-09 16:30:08 UTC

Contents

`autocov` *Autocovariance matrix*

Description

Estimates the autocovariance matrix for a given data matrix X. Via the parameter `direction`, it is possible to estimate only row- or columnwise autocovariance matrices, which is useful if the autocovariance function is separable.

Usage

```
autocov(X, b, M = as.integer(c(1, 1)), direction = 0L, type = 0L)
```

Arguments

`X`	numeric matrix,
`b`	numeric vector containing two integer values: the bandwidths for the row- resp. column-wise estimation. (Up to which lag should the autocovariances be estimated?) If `direction` > 0: only one integer must be supplied. Bandwidths must be smaller than the dimensions of X.
`M`	numeric vector containing two integer values, only needed for `type` = 1, see Details.
`direction`	0: all directions, 1: only row-wise autocovariances, 2: only column-wise autocovariances.
`type`	0: ordinary autocovariance estimation, 1: difference-based autocovariance estimation. See Details.

Details

In this function, the autocovariance matrix of `X` is interpreted as the autocovariance matrix of `x = as.vector(X)`, i.e. where `X` is ordered into a vector column-wise. If `type = 0`, the autocovariance to lags h_1, h_2 is estimated using the regular estimator

$$\hat{\gamma}_{\text{reg}}(h_1, h_2) = \frac{1}{(n-h_1)(m-h_2)} \sum_{i=1}^{n-h_1} \sum_{j=1}^{m-h_2} (Y_{i,j} - \bar{Y})(Y_{i+h_1,j+h_2} - \bar{Y}).$$

If `type = 1`, the autocovariance to lags h_1, h_2 is estimated by a difference-based version, inspired by the estimator of Tecuapetla-Gómez and Munk (2017) for time series:

$$\hat{\gamma}_{\text{diff}}(h_1, h_2) = \hat{\sigma}^2_{\text{diff}} - \frac{1}{2(n-h_1)(m-h_2)} \sum_{i=1}^{n-h_1} \sum_{j=1}^{m-h_2} (Y_{i,j} - Y_{i+h_1,j+h_2})^2$$

with

$$\hat{\sigma}^2_{\text{diff}} = \frac{1}{4} \left(\frac{1}{n(m-M_2)} \sum_{i=1}^{n} \sum_{j=1}^{m-M_2} (Y_{i,j} - Y_{i,j+M_2})^2 + \frac{1}{(n-M_1)m} \sum_{i=1}^{n-M_1} \sum_{j=1}^{m} (Y_{i,j} - Y_{i+M_1,j})^2 \right),$$

where $M_1 =$ `M[1]`, $M_2 =$ `M[2]`.

Value

A numeric matrix of size $N \times N$. If `direction = 0` then $N =$ `prod(dim(X))`. If `direction = 1` then $N =$ `ncol(X)`, if `direction = 2` then $N =$ `nrow(X)`.

References

Tecuapetla-Gómez, I., & Munk, A. (2017). Autocovariance estimation in regression with a discontinuous signal and m-dependent errors: A difference-based approach. Scandinavian Journal of Statistics, 44(2), 346-368.

Examples

```
X <- genField(c(20, 20))
autocov(X, c(4, 4))[1:10, 1:100]

# if separable:
Sigma1 <- autocov(X, 4, direction = 1)
Sigma2 <- autocov(X, 4, direction = 2)
kronecker(Sigma1, Sigma2)[1:10, 1:100]
```

---

bandwidth *Bandwidth estimation*

Description

Calculate MSE-optimal bandwidths according to Andrews (1991).

Usage

```
bandwidth(X, p1 = 0.3, p2 = 0.3, lag = 1)
```

Arguments

X	numeric vector or matrix.
p1, p2	exponents for sample size n resp. estimated dependency, between 0 and 1.
lag	lag to which the autocorrelations are to be estimated. Integer > 0 but smaller than the length resp. number of rows and columns of X.

Details

Bandwidth $\boldsymbol{b}^{(n,m)} = (b_1^{(n)}, b_2^{(m)})$ is estimated via

$$b_i^{(k)} = \min\left(k-1, \max\left(1, k^{0.3}\left(\frac{2\rho_i}{1-\rho_i^2}\right)^{0.3}\right)\right),$$

where ρ_1 and ρ_2 are the mean row- resp. column-wise Spearman autocorrelations to lag 1.

Value

A numeric vector containing one or two elements, depending on if a vector or matrix is supplied. In case of a matrix: the first value is the bandwidth for the row-wise and the second one for the column-wise estimation.

References

Andrews, D. W. (1991). "Heteroskedasticity and autocorrelation consistent covariance matrix estimation". In: Econometrica: Journal of the Econometric Society, pp. 817–858.

Examples

```
X1 <- genField(c(50, 50), Theta = genTheta(1, 0.4))
bandwidth(X1, 0.3, 0.3)

Theta <- matrix(c(0.08, 0.1, 0.08, 0.8, 1, 0.8, 0.08, 0.1, 0.08), ncol = 3)
X2 <- genField(c(50, 50), Theta = Theta)
bandwidth(X2, 1/3, 2/3)
```

---

block_pValue *Block test p-value*

Description

Returns the p-value of a test statistic according to the block test for structural changes.

Usage

```
block_pValue(tn, fun = "gmd")
```

Arguments

tn	test statistic
fun	Character string; one of "gmd" (default), "var", "jb", "grubbs", "ANOVA", "ad", "sw" (see block_stat for details).

Value

A numeric value between 0 and 1.

---

block_stat *Block test statistic*

Description

Computes test statistics of the block test on structural changes.

Usage

```
block_stat(x, s, fun = "gmd", varEstim = var)
```

Arguments

x	times series or random field to be tested. Either a numeric vector or a numeric matrix.
s	parameter for the size of the blocks, 0.5 < s < 1, block length $l_n = [n^s]$. Default is sOpt(n, 0.6).
fun	Character string; one of "gmd" (default), "var", "jb", "grubbs", "ANOVA".
varEstim	variance estimator or variance estimation of the whole field or times series. Either a function to estimate the variance with, or a numeric value.

Details

First, the time series or random field is divided into blocks and the means of the blocks are computed. Then the function fun is applied to the block means:

- gmd: Gini's mean difference
- var: Ordinary variance estimator
- jb: Jarque-Bera test
- grubbs: Grubbs test for outliers
- ANOVA: simple ANOVA.

Value

A numeric value. For fun = "grubbs" it has the attribute n indicating the number of blocks, i.e. the number of observations used in the Grubbs test. For fun = "ANOVA" it has the attributes k (number of blocks) and N (total number of observations).

See Also

block_pValue

Examples

```
# time series with a shift
x <- arima.sim(model = list(ar = 0.5), n = 100)
x[1:50] <- x[1:50] + 1
block_stat(x, sOpt(100, 0.6))

# field without shift and ordinary variance
X <- genField(c(50, 50))
block_stat(X, sOpt(50, 0.6), "var")

# field with a shift and ordinary variance
X <- genField(c(50, 50), type = 2)
block_stat(X, sOpt(50, 0.6), "var")

# GMD test statistic, scaling variance estimated by the mad
block_stat(X, 0.6, fun = "var", varEstim = mad)
```

---

block_test *Block test for structural changes*

Description

A test to detect whether an underlying time series or random field is stationary (hypothesis) or if there is a location shift present in a region (alternative).

Usage

```
block_test(x, s, fun = "gmd", varEstim = var)
```

Arguments

x	times series or random field to be tested. Either a numeric vector or a numeric matrix.
s	parameter for the size of the blocks, 0.5 < s < 1, block length $l_n = \lceil n^s \rceil$. Default is sOpt(n, 0.6).
fun	Character string; one of "gmd" (default), "var", "jb", "grubbs", "ANOVA", "ad", "sw" (see block_stat for details).
varEstim	variance estimator or variance estimation of the whole field or times series. Either a function to estimate the variance with, or a numeric value.

Value

A list of the class "htest" containing the following components:

`statistic`	value of the test statistic (numeric).
`p.value`	p-value (numeric).
`alternative`	alternative hypothesis (character string).
`method`	name of the performed test (character string).
`data.name`	name of the data (character string).

References

Görz, S. and Fried, R. (2025). "Detecting changes in the mean of spatial random fields on a regular grid", *arXiv preprint* arXiv:2512.11599

Schmidt, S. K. (2024). Detecting changes in the trend function of heteroscedastic time series. Bernoulli, 30(4), 2598-2622.

See Also

block_stat, block_pValue

Examples

```
# time series with a shift
x <- arima.sim(model = list(ar = 0.5), n = 100)
x[1:50] <- x[1:50] + 1
block_test(x, sOpt(100, 0.6))

# field without a shift and ordinary variance
X <- genField(c(50, 50))
block_test(X, sOpt(50, 0.6), "var")

# field with a shift and ordinary variance
X <- genField(c(50, 50), type = 2)
block_test(X, sOpt(50, 0.6), "var")

#' # GMD test statistic, scaling variance estimated by the mad
block_stat(X, 0.6, fun = "var", varEstim = mad)
```

---

changeRegion *Change Region*

Description

Generates the indices for different types of change regions.

Usage

```
changeRegion(n, s, type = 1L, middle, delta = 0.15, distFun = dist)
```

Arguments

`n`	dimensions of the random field. Numeric vector.
`s`	parameter for the size of the blocks, 0.5 < s < 1, block length $l_n = [n^s]$.
`type`	change region type (integer or character). See "Details".
`middle`, `delta`, `distFun`	parameters for type 4L. See "Details".

Details

Change region types:

- 1L or "1a": exactly one block is shifted
- "1b": size of the shift region is one block, but the shift region lies in two blocks
- "1c": size of the shift region is one block, but the shift region lies in four blocks
- 2L: exactly half of the data is shifted
- 3L: there is a steady increase from left to right
- 4L:
- 5L: for demonstration purposes: the field is divided into 10 "columns". Every other column is shifted
- 6L:
- 7L: a "circle" including everything within `distFun` `delta` from `middle` is shifted

Value

Types `1a`, `1b`, `1c`, `2`, `4`, `5`	A vector of indices.
Type `3`	A numeric (n x n) matrix containing the heights of the shifts at the corresponding locations.

See Also

genField

Examples

```
changeRegion(c(50, 50), 0.6, "1a")
changeRegion(c(50, 50), type = 2)
changeRegion(c(50, 50), type = 3L)
changeRegion(c(50, 50), type = 7L, middle = c(10, 10))
```

`decorr` *De-correlation*

Description

De-correlates a random field or time series, so that the resulting values can be treated as independent.

Usage

```
decorr(X, lags, method = 1L, separable = FALSE, M = 1, type = 0)
```

Arguments

`X`	Random Field, numeric matrix, or time series
`lags`	numeric vector containing two integer values: the bandwidths for the row- reps. column-wise autocovariance estimation. (Up to which lag should the autocovariances be estimated?) Lags must be smaller than the dimensions of X.
`method`	1L: square root of the matrix via `robcp::modifChol()`, inversion via `solve()` 2L: square root and inversion via singular value decomposition 3L: square root via `expm::sqrtm()`, inversion via `solve()`
`separable`	if the autocovariance function is (assumed to be) separable in the two directions of X, those two autocovariances can be estimated separately and then combined (after square root and inversion) as a Kronecker product.
`M`	numeric vector containing two integer values, only needed for `type = 1`, see `autocov()`.
`type`	0: ordinary autocovariance estimation, 1: difference-based autocovariance estimation. See `autocov()`.

Details

The contents of `X` are ordered into a vector x column-wise. The autocovariance matrix Σ of x is estimated by `autocov()`. Σ is taken the square root of and being inverted using the functions specified in `method`. Then

$$y = \Sigma^{-\frac{1}{2}}(x - \bar{x}).$$

Then y is ordered back into a matrix Y with the same dimension as X.

Value

De-correlated random field or time series; same data type and size as input `X`.

Examples

```
x <- arima.sim(list(ar = 0.4), 200)
y <- decorr(x, 3)

oldpar <- par(mfrow = c(2, 2))
acf(x)
pacf(x)
acf(y)
pacf(y)
par(oldpar)

X <- genField(c(20, 20), Theta = genTheta(1, 0.4))
Y <- decorr(X, c(2, 2))
```

`genField` *Generate random fields*

Description

Generate random fields on a regular grid. Dependency can be modeled to some extend.

Usage

```
genField(
  n,
  distr = rnorm,
  type = 0L,
  H = 100,
  Theta = NULL,
  q = NULL,
  param = NULL,
  ...
)
```

Arguments

`n`	dimensions of the requested random field. Numeric vector.
`distr`	function specifying the error distribution.
`type`	change region type (integer or character). See `changeRegion`.
`H`	height of the location shift (numeric value).
`Theta`	matrix specifying the dependency between observations of a 2-dim random field. Has to be a 2-dim. matrix where there is an odd number of entries in both dimensions. Explanations are given in genTheta.
`q`	dependency parameter for the model order of the MA field (integer > 0).

param	dependency parameters for the model parameters of the MA field (numeric vector.) Both q and param are ignored if Theta is supplied. A dependency matrix is generated using genTheta.
...	additional arguments for the generation of changeRegion.

Details

The dependent random field is generated as follows: Denote with (e_{ij}) the error matrix from which the random field (Y_{ij}) (under the hypothesis, i.e. without any location shift) is generated.

Value

The function returns a matrix of dimension n that has the class "RandomField".

See Also

changeRegion, genTheta

Examples

```
genField(c(50, 50))
genField(c(50, 50), type = 3, Theta = genTheta(1, 0.2))
```

---

genTheta *Dependency Matrix Theta*

Description

This function generates a symmetric dependency matrix Θ of a specific type of spatial MA(q) model.

Usage

```
genTheta(q, param, structure = "MA")
```

Arguments

q	model order (integer).
param	MA parameter (numeric value between 0 and 1).
structure	Character string, either "MA" or "AR" indicating the structure of the dependency matrix. Details below.

Details

Symmetric spatial MA(q) model (or an approximation to a spatial AR(1) model) for 2-dim. random fields:

$$Y_{ij} = \sum_{k=-q}^{q} \sum_{l=-q}^{q} \theta_{kl} \varepsilon_{kl}.$$

$(\theta_{kl}) = \Theta$.

For "MA":

$$\theta_{kl} = \texttt{param}^{|k-q-1|+|l-q-1|}.$$

For "AR":

$$\theta_{kl} = \tilde{\theta}_{kl} / \sqrt{\sum_{|k|\le q} \sum_{|l|\le q} \tilde{\theta}_{kl}^2} \quad \text{with} \quad \tilde{\theta}_{kl} = \texttt{param}^{\sqrt{k^2+l^2}}.$$

Value

A matrix of size (2q + 1) x (2q + 1).

Examples

```
genTheta(1, 0.2, "MA")

genTheta(40, 0.2, "AR")
```

GMD *Gini's mean difference*

Description

Calculates Gini's mean difference of a given vector `x`.

Usage

```
GMD(x)
```

Arguments

`x`	numeric vector.

Value

A numeric value.

Examples

```
x <- rnorm(100)
GMD(x)
```

grubbs *Grubbs outlier test*

Description

Computes the test statistic and the critical value of the outlier test according to Grubbs.

Usage

```
grubbs(x, varEstim = var)

crit.grubbs(n, alpha = 0.05)
```

Arguments

x	numeric vector.
varEstim	Variance estimation or estimation function. Either a numeric value or a function taking one argument.
n	sample size; positive numeric value.
alpha	significance level; between 0 and 1.

Value

numeric value.

Examples

```
x <- rnorm(100)
grubbs(x) > crit.grubbs(100, 0.05)

# add outlier
x[1] <- x[1] + 100
grubbs(x) > crit.grubbs(100, 0.05)
```

Mu *Mu*

Description

This function returns a vector of the block means for a given random field X.

Usage

```
Mu(x, group = NULL, l = NULL)
```

Arguments

x	Numeric vector or matrix.
group	strictly positive integer vector or matrix indicating the group (or block) of the corresponding observation in X. Overwrites l if specified.
l	block length. Integer vector of length 1 or 2, depending on the number of dimensions of X, with strictly positive entries.

Value

A numeric vector of length `floor(n[1] / l[1]) * floor(n[2] / l[2])`.

Examples

```
X <- genField(c(50, 100), H = 100, type = 2)
M <- Mu(X, l = c(10, 20))

plot(X)
image(matrix(M, ncol = 5))
```

---

plot.RandomField *Plot random field*

Description

Plot random field

Usage

```
## S3 method for class 'RandomField'
plot(x, main = "", colors, name = "value", alpha = 1, p = NULL, ...)
```

Arguments

x	RandomField object.
main	title to the plot.
colors	vector of colors over which a gradient is created.
name	title of the legend, character string.
alpha	opacity of the raster tiles. Numeric value between 0 and 1.
p	additional ggplot2 component added BEFORE this function's components.
...	other parameters to be passed through to plotting functions.

Value

a ggplot2 object. Function is mostly called for its side effect.

Examples

```
plot(genField(c(50, 50)))
plot(genField(c(50, 50), type = 2))
```

`rmix` *Mixture distribution*

Description

Generates a random sample from a mixture normal distribution.

Usage

```
rmix(n, q = 0.01, h = 10, sigma = 1)
```

Arguments

`n`	sample size, numeric.
`q`	probability that observation is drawn from the contamination distribution, numeric.
`h`	mean of the contamination distribution, numeric.
`sigma`	standard deviation of the contamination distribution, numeric.

Details

The resulting sample is drawn from the distribution

$$(1-q)\mathcal{N}(0,1) \; + \; q\mathcal{N}(h,\sigma^2).$$

Value

Numeric vector of length n containing the random sample.

Examples

```
# random sample with 0.01 chance of contamination distribution with mean 10
rmix(100)

# random sample with 0.01 chance of contamination distribution with standard deviation 10
# IMPORTANT: h needs to be set to 0!
rmix(100, h = 0, sigma = 1)
```

skewness *Skewness and kurtosis*

Description

Compute the skewness and the kurtosis of the data vector `x`.

Usage

```
skewness(x)

kurtosis(x)
```

Arguments

`x`	numeric vector.

Value

A numeric value.

Examples

```
skewness(rnorm(100))
skewness(rexp(100, 2))

kurtosis(rnorm(100))
kurtosis(rt(100, 5))
```

sSizes *Optimal parameter s*

Description

Calculates the best parameter $\tilde{s}$ for a given approximation s, such that $n \ \% \ [n^s] = 0$.

Usage

```
sSizes(n, lower = 0.5, upper = 1, step = 0.1)

sOpt(n, s = 0.6)
```

Arguments

`n`	Sample size(s), numeric (vector).
`lower`, `upper`	lower and upper search border, between 0 and 1.
`step`	size of the step for the search, between 0 and 1.
`s`	Desired exponent, $0.5 \leq s \leq 1$.

Value

`sSizes` returns a data frame containing

`n`	the given sample size
`s`	the exponent in question
`ln`	the resulting block length
`bn`	the corresponding number of block
`ln.bn`	block length times number of blocks
`diff`	difference between the given sample size and the number of observations covered by the blocks

`sOpt` returns a numeric vector of the optimal exponent(s).

Examples

```
sSizes(50)
sSizes(50, 0.6, 0.8, 0.01)

sOpt(50, 0.6)
sOpt(100, 0.6)
```

Index