

Package ‘rhosa’

July 23, 2025

Title Higher-Order Spectral Analysis
Date 2024-05-17
Version 0.3.0
Description Higher-order spectra or polyspectra of time series, such as bispectrum and bicoherence, have been investigated in abundant literature and applied to problems of signal detection in a wide range of fields. This package aims to provide a simple API to estimate and analyze them. The current implementation is based on Brillinger and Irizarry (1998) <[doi:10.1016/S0165-1684\(97\)00217-X](https://doi.org/10.1016/S0165-1684(97)00217-X)> for estimating bispectrum or bicoherence, Lii and Heland (1981) <[doi:10.1145/355958.355961](https://doi.org/10.1145/355958.355961)> for cross-bispectrum, and Kim and Powers (1979) <[doi:10.1109/TPS.1979.4317207](https://doi.org/10.1109/TPS.1979.4317207)> for cross-bicoherence.
License GPL-3
Encoding UTF-8
URL <https://tabe.github.io/rhosa/>
BugReports <https://github.com/tabe/rhosa/issues>
RoxygenNote 7.3.1
Imports parallel
Suggests ggplot2, knitr, rmarkdown, testthat (>= 2.1.0)
VignetteBuilder knitr
NeedsCompilation no
Author Takeshi Abe [aut, cre] (ORCID: <<https://orcid.org/0000-0002-7074-4561>>)
Maintainer Takeshi Abe <tabe@fixedpoint.jp>
Repository CRAN
Date/Publication 2024-05-17 09:30:02 UTC

Contents

bicoherence	2
biperiodogram	3
bispectrum	4

cross_bicoherence	6
cross_bispectrum	7
kim_and_powers_model	8
mode_matching	9
three_channel_model	10
Index	12

bicoherence	<i>Estimate bicoherence from given time series data.</i>
-------------	--

Description

Estimate magnitude-squared bicoherence from given real- or complex-valued time series data.

Usage

```
bicoherence(  
  data,  
  window_function = NULL,  
  mc = FALSE,  
  mc_cores = getOption("mc.cores", 2L),  
  alpha = 0.05,  
  p_adjust_method = "BH"  
)
```

Arguments

data	Given time series, as a data frame or matrix with which columns correspond to sampled stretches.
window_function	A window function's name for tapering. Defaults to NULL ("no tapering"). Currently the following window functions are available: Hamming window ("hamming"), Hann window ("hann"), and Blackman window ("blackman").
mc	If TRUE, calculation is done in parallel computation. Defaults to FALSE.
mc_cores	The number of cores in use for parallel computation, passed <code>parallel::mcmapply()</code> etc. as <code>mc.cores</code> .
alpha	The alpha level of the hypothesis test. Defaults to 0.05.
p_adjust_method	The correction method for p-values, given to <code>p.adjust()</code> . Defaults to "BH" (Benjamini and Hochberg). No correction if a non-character is given.

Value

A data frame including the following columns:

f1: The first elements of frequency pairs.

f2: The second elements of frequency pairs.

value: The estimate of magnitude-squared bicoherence at the respective frequency pair.

p_value: The (corrected, if requested) p-value for hypothesis testing under null hypothesis that bicoherence is 0.

significance: TRUE if the null hypothesis of the above hypothesis test is rejected with given alpha level.

References

Brillinger, D.R. and Irizarry, R.A. "An investigation of the second- and higher-order spectra of music." Signal Processing, Volume 65, Issue 2, 30 March 1998, Pages 161-179.

Examples

```
f <- function(x) {
  sin(2 * x) + sin(3 * x + 1) + sin(2 * x) * sin(3 * x + 1)
}
v <- sapply(seq_len(1280), f) + rnorm(1280)
m <- matrix(v, nrow = 128)
bc1 <- bicoherence(m)
bc2 <- bicoherence(m, "hamming")
bc3 <- bicoherence(m, "hann", mc = TRUE, mc_cores = 1L)
```

biperiodogram

Calculate biperiodogram

Description

Calculate the biperiodogram of real-valued time series

Usage

```
biperiodogram(
  x,
  dft_given = FALSE,
  mc = FALSE,
  mc_cores = getOption("mc.cores", 2L)
)
```

Arguments

<code>x</code>	Given time series (or its DFT), as a data frame or matrix with which columns correspond to sampled stretches
<code>dft_given</code>	If TRUE, suppose that DFTs are given instead of time series data and skip the fast fourier transform. Default: FALSE.
<code>mc</code>	If TRUE, calculation is done in parallel computation. Defaults to FALSE.
<code>mc_cores</code>	The number of cores in use for parallel computation, passed <code>parallel::mcmapply()</code> etc. as <code>mc.cores</code> .

Value

A list with names

f1: The first elements of frequency pairs.

f2: The second elements of frequency pairs.

value: The biperiodogram as a matrix. Each of its rows is for a frequency pair; its columns correspond to stretches.

References

Hinich, M.J., 1994. Higher order cumulants and cumulant spectra. *Circuits Systems and Signal Process* 13, 391–402. doi:10.1007/BF01183737

Examples

```
f <- function(x) {
  sin(2 * x) + sin(3 * x + 1) + sin(2 * x) * sin(3 * x + 1)
}
v <- sapply(seq_len(1280), f) + rnorm(1280)
m <- matrix(v, nrow = 128)
bp <- biperiodogram(m)

m2 <- stats::mvfft(m)
bp2 <- biperiodogram(m2, dft_given = TRUE)
```

bispectrum

Estimate bispectrum from time series data.

Description

Estimate bispectrum from real- or complex-valued time series data.

Usage

```

bispectrum(
  data,
  window_function = NULL,
  mc = FALSE,
  mc_cores = getOption("mc.cores", 2L)
)

```

Arguments

<code>data</code>	Given time series, as a data frame or matrix with which columns correspond to sampled stretches.
<code>window_function</code>	A window function's name for tapering. Defaults to NULL ("no tapering"). Currently the following window functions are available: Hamming window ("hamming"), Hann window ("hann"), and Blackman window ("blackman").
<code>mc</code>	If TRUE, calculation is done in parallel computation. Defaults to FALSE.
<code>mc_cores</code>	The number of cores in use for parallel computation, passed <code>parallel::mcmapply()</code> etc. as <code>mc.cores</code> .

Value

A data frame including the following columns:

f1: The first elements of frequency pairs.

f2: The second elements of frequency pairs.

value: The estimated bispectrum at each frequency pair.

References

Brillinger, D.R. and Irizarry, R.A. "An investigation of the second- and higher-order spectra of music." *Signal Processing*, Volume 65, Issue 2, 30 March 1998, Pages 161-179.

Examples

```

f <- function(x) {
  sin(2 * x) + sin(3 * x + 1) + sin(2 * x) * sin(3 * x + 1)
}
v <- sapply(seq_len(1280), f) + rnorm(1280)
m <- matrix(v, nrow = 128)
bs1 <- bispectrum(m)
bs2 <- bispectrum(m, "hamming")
bs3 <- bispectrum(m, "blackman", mc = TRUE, mc_cores = 1L)

```

cross_bicoherence	<i>Estimate cross-bicoherence from time series data.</i>
-------------------	--

Description

Estimate cross-bicoherence from three real-valued time series data.

Usage

```
cross_bicoherence(
  x,
  y,
  z = y,
  dft_given = FALSE,
  mc = FALSE,
  mc_cores = getOption("mc.cores", 2L)
)
```

Arguments

x	Given 1st time series, as a data frame or matrix with which columns correspond to sampled stretches.
y	Given 2nd time series, with the same dimension as x.
z	Optional 3rd time series, with the same dimension as x (and thus as y). If omitted, y is used instead.
dft_given	If TRUE, suppose that DFTs are given instead of time series data and skip the fast fourier transform. Default: FALSE.
mc	If TRUE, calculation is done in parallel computation. Defaults to FALSE.
mc_cores	The number of cores in use for parallel computation, passed <code>parallel::mclapply()</code> etc. as <code>mc.cores</code> .

Value

A data frame including the following columns:

f1: The first elements of frequency pairs.

f2: The second elements of frequency pairs.

value: The estimated value of magnitude-squared cross-bicoherence at the respective frequency pair.

References

Kim, Y.C., Powers, E.J., 1979. Digital Bispectral Analysis and Its Applications to Nonlinear Wave Interactions. IEEE Trans. Plasma Sci. 7, 120–131. <https://doi.org/10.1109/TPS.1979.4317207>

Examples

```

x <- seq_len(1280)
v1 <- sapply(x, function(x) {sin(2 * x)}) + rnorm(1280)
v2 <- sapply(x, function(x) {sin(3 * x + 1)}) + rnorm(1280)
v3 <- sapply(x, function(x) {cos(2 * x) * cos(3 * x + 1)}) + rnorm(1280)
m1 <- matrix(v1, nrow = 128)
m2 <- matrix(v2, nrow = 128)
m3 <- matrix(v3, nrow = 128)
xbc1 <- cross_bicoherence(m1, m2, m3)

d1 <- stats::mvfft(m1)
d2 <- stats::mvfft(m2)
d3 <- stats::mvfft(m3)
xbc2 <- cross_bicoherence(d1, d2, d3, dft_given = TRUE)

xbc3 <- cross_bicoherence(d1, d2, d3, dft_given = TRUE, mc = TRUE, mc_cores = 1L)

```

cross_bispectrum	<i>Estimate cross-bispectrum from time series data.</i>
------------------	---

Description

Estimate cross-bispectrum from three real-valued time series data.

Usage

```

cross_bispectrum(
  x,
  y,
  z = y,
  dft_given = FALSE,
  mc = FALSE,
  mc_cores = getOption("mc.cores", 2L)
)

```

Arguments

x	Given 1st time series, as a data frame or matrix with which columns correspond to sampled stretches.
y	Given 2nd time series, with the same dimension as x.
z	Optional 3rd time series, with the same dimension as x (and thus as y). If omitted, y is used instead.
dft_given	If TRUE, suppose that DFTs are given instead of time series data and skip the fast fourier transform. Default: FALSE.
mc	If TRUE, calculation is done in parallel computation. Defaults to FALSE.
mc_cores	The number of cores in use for parallel computation, passed <code>parallel::mclapply()</code> etc. as <code>mc.cores</code> .

Value

A data frame including the following columns:

f1: The first elements of frequency pairs.

f2: The second elements of frequency pairs.

value: The estimated cross-bispectrum at each frequency pair.

References

K. S. Lii and K. N. Helland. 1981. Cross-Bispectrum Computation and Variance Estimation. *ACM Trans. Math. Softw.* 7, 3 (September 1981), 284–294. DOI:<https://doi.org/10.1145/355958.355961>

Examples

```
x <- seq_len(1280)
v1 <- sapply(x, function(x) {sin(2 * x)}) + rnorm(1280)
v2 <- sapply(x, function(x) {sin(3 * x + 1)}) + rnorm(1280)
v3 <- sapply(x, function(x) {cos(2 * x) * cos(3 * x + 1)}) + rnorm(1280)
m1 <- matrix(v1, nrow = 128)
m2 <- matrix(v2, nrow = 128)
m3 <- matrix(v3, nrow = 128)
xbs1 <- cross_bispectrum(m1, m2, m3)

d1 <- stats::mvfft(m1)
d2 <- stats::mvfft(m2)
d3 <- stats::mvfft(m3)
xbs2 <- cross_bispectrum(d1, d2, d3, dft_given = TRUE)

xbs3 <- cross_bispectrum(d1, d2, d3, dft_given = TRUE, mc = TRUE, mc_cores = 1L)
```

`kim_and_powers_model` *A test signal of the phase coherence between three oscillators*

Description

Generate test signals which involve three oscillators described in Kim and Powers (1979).

Usage

```
kim_and_powers_model(
  fbfN = 0.22,
  fcfN = 0.375,
  fdfN = fbfN + fcfN,
  num_points = 128,
  num_records = 64,
  noise_sd = 0.1,
  phase_coherence = TRUE,
  product_term = FALSE
)
```

Arguments

fbfN	b's frequency divided by the Nyquist frequency; 0.220 by default.
fcfN	c's frequency divided by the Nyquist frequency; 0.375 by default.
fdfN	d's frequency divided by the Nyquist frequency; fbfN + fcfN by default.
num_points	The number of sampling points in a record; 128 by default.
num_records	The number of records; 64 by default.
noise_sd	The standard deviation of a Gaussian noise perturbing samples; 0.1 (-20dB) by default.
phase_coherence	If TRUE (default), the phase coherence in the signal d is on; otherwise off.
product_term	If TRUE, the product of b and c is included in the model; FALSE by default.

Details

This function produces a list of numeric vectors; its each element represents a test signal in which three oscillators b, c, and d are superimposed. The ratio of the frequency of b (f1) to the Nyquist frequency is 0.220 and the ratio of the frequency of c (f2) to the Nyquist frequency is 0.375, by default. The d's frequency f3 is equal to f1 + f2 unless specified otherwise. Optionally the product of b and c is also added to signals.

Value

A matrix of num_points rows x num_records columns.

Examples

```
data <- kim_and_powers_model()
```

mode_matching	<i>Estimate cross-bicoherence's empirical null distribution by a mode matching method</i>
---------------	---

Description

Estimate false discovery rate by fitting scaled chi-squared distribution as an empirical null of cross-bicoherence with Schwartzman's mode matching method.

Usage

```
mode_matching(xbc, t_max = NULL, d = 0.001)
```

Arguments

xbc	cross-bicoherence, returned from cross_bicoherence.
t_max	the upper limit of interval
	S_0
	, see the reference.
d	the bin width of the tuning parameter.

References

Schwartzman, Armin. “Empirical Null and False Discovery Rate Inference for Exponential Families.” *Annals of Applied Statistics* 2, no. 4 (December 2008): 1332–59. <https://doi.org/10.1214/08-AOAS184>.

three_channel_model	<i>A three-channel model of quadratic phase coupling</i>
---------------------	--

Description

Simulate observations by a three-channel model of quadratic phase coupling.

Usage

```
three_channel_model(  
  f1,  
  f2,  
  f3,  
  num_samples = 256,  
  num_observations = 100,  
  input_freq = c(1.2, 0.7, 0.8),  
  noise_sd = 1  
)
```

Arguments

f1	A function of period 2π for the first channel.
f2	A function of period 2π for the second channel.
f3	A function of period 2π for the third channel.
num_samples	The number of sampling points in an observation.
num_observations	The number of observations.
input_freq	The scaling factor for the frequencies of input periodic functions. It can be a scalar or a vector of length three. If a scalar is given, the same frequency is used for all of inputs.
noise_sd	The standard deviation of a Gaussian noise perturbing samples. It can be a scalar or a vector of length three. If a scalar is given, the same value is used for all of noises. Giving 0 is possible and specifies no noise.

Details

Given three periodic functions, this function generates a list of three data frames in which each column represents a simulated observation at a channel. The phase is chosen at random from $[0, 2\pi]$ for each observation and each channel.

Value

A list of six data frames: i1, i2, i3, o1, o2, and o3. Each element has num_observations columns and num_samples rows. i1, i2, and i3 are observations of input signals; o1, o2, and o3 are of output.

Examples

```
sawtooth <- function(r) {  
  x <- r/(2*pi)  
  x - floor(x) - 0.5  
}  
data <- three_channel_model(cos, sin, sawtooth,  
                           input_freq = c(0.2, 0.3, 0.4),  
                           noise_sd = 0.9)
```

Index

bicoherence, [2](#)
biperiodogram, [3](#)
bispectrum, [4](#)

cross_bicoherence, [6](#)
cross_bispectrum, [7](#)

kim_and_powers_model, [8](#)

mode_matching, [9](#)

p.adjust, [2](#)
parallel::mclapply, [6](#), [7](#)
parallel::mcmapply, [2](#), [4](#), [5](#)

three_channel_model, [10](#)